

School of Computer Science

Interval-based Qualitative Spatial Reasoning

Anthony John Travers

This thesis is presented as part of
the requirements for the award of the
Degree of Doctor of Philosophy
of
Curtin University of Technology

March 4, 1998

“A dance run pointing straight up the comb signals that food is in the same direction as the sun. Straight down signals food is the exact opposite direction. All intermediate angles signal what you would expect. Fifty degrees to the left of the vertical signifies 50 degrees to the left of the sun’s direction in the horizontal plane. The accuracy of the dance is not to the nearest degree, however. Why should it be, for it is our arbitrary convention to divide the compass into 360 degrees? Bees divide the compass into about “8 bee degrees”. Actually, this is approximately what we do when we are not professional navigators. We divide our informal compass into eight quadrants: N, NE, E, SE, S, SW, W, NW.”

Richard Dawkins

The difference between the reason of man and the instinct of the beast is this, that the beast does but know, but the man knows that he knows.

John Donne

Abstract

The role of spatial reasoning in the development of systems in the domain of Artificial Intelligence is increasing. One particular approach, qualitative spatial reasoning, investigates the usage of abstract representation to facilitate the representation of and the reasoning with spatial information.

This thesis investigates the usage of intervals along global axes as the underlying representational and reasoning mechanism for a spatial reasoning system. Aspects that are unique to representing spatial information (flow and multi-dimensionality) are used to provide a method for classifying relations between objects at multiple levels of granularity. The combination of these two mechanisms (intervals and classification) provide the basis for the development of a querying system that allows qualitative queries about object relations in multi-dimensional space to be performed upon the representation.

The second issue examined by this thesis is the problem of representing intervals when all the interval relations may not be known precisely. A three part solution is proposed. The first shows how the simplest situation, where all relations are explicit and primitive, can be represented and integrated with the above mentioned querying system. The second situation demonstrates how, for interval relations that are primitive but are not all explicitly known, an effective point based representation may be constructed. Finally, when relations between intervals are disjunctions of possible primitive interval relations, a representation is presented which allows solutions to queries to be constructed from consistent data.

Our contribution is two-fold:

1. a method of classifying the spatial relations and the means of querying these relations;
2. a process of efficiently representing incomplete interval information and the means of efficiently querying this information.

The work presented in this thesis demonstrates the utility of a multi-dimensional qualitative spatial reasoning system based upon intervals. It also demonstrates

how an interval representation may be constructed for datasets that have variable levels of information about relationships between intervals represented in the dataset.

Acknowledgements

I would like to thank Svetha Venkatesh and Dorota Kieronska for supervising me and putting up with me for the last five years. They have kept me on track and this thesis would not have been possible without their guidance.

Thanks to Keven and Kim for helping to get me off track occasionally: it has kept me sane.

A special thanks to Lydia for listening and being there for me.

Contents

1	Introduction	1
1.1	Multi-dimensional Qualitative Spatial Reasoning	2
1.1.1	Qualitative Spatial Reasoning using Intervals	4
1.2	Contribution	6
1.3	Outline	6
2	Background	9
2.1	Time	10
2.1.1	Closure	10
2.1.2	Non-Closure representation	18
2.1.3	Discussion	21
2.2	Space	21
2.2.1	Relative vs global axes	22
2.2.2	n-D techniques	23
2.2.3	Qualitative spatial reasoning systems	25
2.2.4	Discussion	36
2.3	Conclusion	37
3	Multi-dimensional Spatial Reasoning	40
3.1	A-Space	41
3.2	NS-Relation	42
3.2.1	A-Relations	44
3.2.2	Orientation	44
3.2.3	New Spatial Relations	46
3.3	n-D Spatial Reasoning	48

3.3.1	n-D Classifications	49
3.3.2	Examples	52
3.4	Queries	58
3.4.1	Atomic queries	60
3.4.2	Composite n-D queries	64
3.4.3	Specification	69
3.4.4	Classifications and Queries	70
3.5	Conclusion	71
4	Interval Representation	72
4.1	Intervals and end points	74
4.1.1	Interval relations to point relations	75
4.1.2	Point relations to interval relations	75
4.2	Interval Representation: The Simple Case	78
4.2.1	Closure	78
4.2.2	Linear Ordering	78
4.3	Results	81
4.3.1	Simple queries	82
4.3.2	Composite Queries	89
4.4	An Application	94
4.4.1	The dataset	95
4.4.2	GIS Queries	96
4.5	Conclusion	102
5	Primitive Interval Representations where some relations are un-	
	known	103
5.1	Graph-based Interval Representation	104
5.1.1	Graph construction	104
5.2	Search	111
5.2.1	Point search	112
5.2.2	Costs	113
5.2.3	Search improvements	114

5.2.4	Interval search	115
5.3	Test Results	117
5.3.1	Data	118
5.3.2	Analysis	119
5.4	Discussion	130
5.5	Conclusion	131
6	Disjunctive Interval Representations	135
6.1	Disjunctive Relations	136
6.1.1	Weak links	136
6.1.2	Disjunction resolutions	140
6.1.3	Disjunction Table	143
6.2	Search	144
6.2.1	Inconsistency via disjunctions	145
6.2.2	Restricting Search	146
6.2.3	Solid Solutions	150
6.2.4	Weak Solutions	151
6.2.5	Inconsistency	158
6.2.6	Using Weak Solutions	159
6.3	Analysis	161
6.4	Results	163
6.4.1	Analysis	164
6.5	Summary	172
6.6	Conclusion	174
7	Conclusion	176
7.1	Discussion	176
7.2	Summary	177
7.3	Future Directions	180
A	Propagating order	181
B	End points	183

List of Figures

2.1	An example of i) SAT, ii) MLP and iii) ACSP	12
2.2	The interval relation $A \{b, bi\} B$ represented as two conjunctions of point relations and using PA.	17
2.3	An example of Gervini's Timegraph representation.	20
2.4	Examples of relative frames of reference: intrinsic, extrinsic and deictic.	23
2.5	A complex simplex	24
2.6	Two different occurrences of the <i>meets</i> relations for 2-D relations .	24
2.7	2-D objects represented as bounding boxes.	25
2.8	Objects may be represented as points and these points represented as 2D-Strings.	26
2.9	Complex relations are reduced to simpler relations by subdividing objects.	27
2.10	A C-String's example without the relation overlaps	28
2.11	Illustrating problems with overlaps	28
2.12	A single cut is needed to remove the overlaps relation	29
2.13	A B-String representation of the objects	29
2.14	Compass directions used to describe 2-D relations	32
2.15	Relations based on four extreme points	33
2.16	The zoning choices in i) led a relative orientation node (ii).	35
2.17	A structure that incorporates topological and orientation information	35
2.18	The point and interval-based spatial divisions.	36
2.19	The collision parallelogram is the intersection of each objects divi- sion lines.	36

3.1	Flows and represent different orderings along the same axis. . . .	43
3.2	The ordering of the limit points of the interval as designated by the flow in A-Space.	43
3.3	Each 2-D object is represented in terms of its bounding box. . . .	48
3.4	Examples of the {disjoint, ends} relation.	54
3.5	Classification 1 examples of five different C2 3-D Relations. . . .	56
3.6	The eight C1 3-D NS-Relations for the C2 3-D NS-Relation equal- ends-overlaps.	57
3.7	Two interpretations of between: non-intersecting “between” : A nibetween B C and intersecting between: A ibetween B C.	67
4.1	The three levels of representation: raw data represents the quan- titative information; the interval representation is the interval ap- proximation of the object; and the point representation is the point representation of intervals.	73
4.2	The complete system entails the representation of raw spatial data as intervals, the representation of these intervals as points, the specification of n-D queries, the processing of n-D queries into sin- gle dimensional interval queries, the conversion of interval queries to point queries and the search of point queries in a point repre- sentation.	82
4.3	A 2-D example environment.	83
4.4	An illustration of possible positions of the <i>objects</i> with relation to the <i>field</i> and the <i>road3</i>	90
4.5	The possible position of the <i>object</i> and the <i>object2</i> as described in the composite query (Query 13).	91
4.6	The possible position of <i>object</i> given with relation to <i>field</i> and <i>road3</i> (Query 14).	92
4.7	The possible position of <i>object</i> given the two constraints described by Query 15.	93
4.8	The possible positions of <i>object</i> as described in the query (Query 16).	94

4.9	GIRAS dataset from the US Department of the Interior: Land usage in area of WILMINGTON, DE NJ PA.	96
4.10	Pictorial representations of sample queries	97
4.11	A relational depiction of the objects <i>avx</i> and <i>axt</i> to the object <i>axs</i> . The 2D relation is disjoint-during $1 < 2-1 > < 2$	98
4.12	A relational depiction of the objects that satisfy the 2D query disjoint-during $?or-1 > < 2$	99
4.13	All the objects that are <i>during-disjoint</i> $1 > < 2-1 < 2$ to the objects to which <i>axs</i> is <i>disjoint-during</i> $?a-1 > < 2$	100
4.14	All the information about the objects that has been acquired by five different queries.	102
5.1	Three different situations where the three different local minimi- sation rules are used when adding the edge $\langle a, b \rangle$	107
5.2	Addition of extra information using local minimisation.	108
5.3	A search for the relationship <i>af</i> on a locally minimised graph results in a search that follows the path <i>abf</i> whereas a fully minimised graph (no <i>bf</i>) would require the search to use the path <i>abcdef</i> . . .	110
5.4	All the triples in which two of the components are a distance of two or less links from “a”.	110
5.5	A fully minimised graph where the number of edges is a maximum.	114
5.6	The final graph from Figure 5.2.	116
5.7	The relationship between intervals A and B.	117
5.8	The relationship between the construction times of the control and the local minimisation are given for the four datasizes (intervals) at different densities.	122
5.9	The relationship between the number of edges in the local minimi- sation and control algorithms for four datasizes.	123
5.10	The relationship between the number of edges in the local minimi- sation and control algorithms focusing on size 100 and 200.	123

5.11	The relationship between the search times for local minimisation and the control are projected against the initial density for a graph of size 100.	127
5.12	The relationship between the search times for local minimisation and the control are projected against the initial density for a graph of size 200.	127
5.13	The relationship between the search times for local minimisation and the control are projected against the initial density for a graph of size 300.	128
5.14	The relationship between the search times for local minimisation and the control are projected against the initial density for a graph of size 400.	128
5.15	An indication of the relationship between the number of edges (x-axis) and the search time (y-axis) for the control and locally minimised graphs of sizes 200, 400, 600 and 800.	129
6.1	The relationship between the disjunction and the entries in the graph.	138
6.2	The representation of weak links in a graph.	139
6.3	Adding links to a network.	141
6.4	The two alternative interval relations A disjoint $1 < 2$ B and A disjoint $1 > 2$ B	146
6.5	Restrictions on the search process.	150
6.6	An example graph containing points A, B, C and the search from point C for the point A.	154
6.7	A search generates a global history list that helps identify repetitive branches.	155
6.8	The growth of the local search lists as the search progresses. . . .	157
6.9	The two types of loops that are derived from consistent links (i.e. consistent with respect to each other).	159
6.10	A section of a graph and the corresponding dtable that refer to the intervals A and B	162

6.11	The performance of the primitive IR system vs the disjunctive IR system with respect to construction time.	167
6.12	The performance of the primitive IR system vs the disjunctive IR system with respect to the number of solid links.	170
6.13	The performance of the primitive IR system vs the disjunctive IR system with respect to the number of weak links.	171
6.14	The performance of the primitive IR system vs the disjunctive IR system with respect to search time.	173
A.1	Four different situations where point a precedes point b , indicated by the interval X	181
B.1	i) An interval Y in the A-Space A; ii) the interval Y with a specific ordering; and iii) and incorrect ordering with respect to the A-Space.	183
B.2	Labelling an interval	184

List of Tables

1.1	Allen's Interval Relations [2].	5
2.1	Allen's Interval Relations [2].	13
2.2	Allen's Transitivity Table (omitting '=') [2]	14
2.3	The composition table for the primitive point relations.	15
2.4	Topological relationships between two objects [18].	24
2.5	Guesgen's spatial interval algebra [35].	30
2.6	Composition rules for Guesgen's spatial interval algebra [35]. . . .	31
2.7	The composition table for the orientation relations.	34
3.1	A-Relations.	45
3.2	A-Relation instances and their Orientation discriminators.	47
3.3	Pictorial Representation of the C3 A-Relations.	53
3.4	Permutations of the equal, overlaps and ends relations in 3-D. . . .	55
4.1	The interval relations described in terms of the relationship between the end points of the intervals.	76
4.2	Interval relations defined in terms of their significant point relations.	77
5.1	The results of the four point queries and the deductive result between intervals B and A.	116
5.2	Point relation results for intervals C and G. Shows the intervals that match.	117
5.3	Results for data size 200 (100 intervals).	119
5.4	Results for data size 400 (200 intervals).	120
5.5	Results for data size 600 (300 intervals).	120

5.6	Results for data size 800 (400 intervals).	121
5.7	Twenty-nine valid relations	132
6.1	A comparison of the primitive IR system against three different initial densities of disjunctions using disjunctive IR system for a dataset of size 200 (100 intervals).	165
6.2	A comparison of the primitive IR system against three different initial densities of disjunctions using the disjunctive IR system for a dataset of size 400 (200 intervals).	166
6.3	A comparison of a primitive IR system against three different ini- tial densities of disjunctions using the disjunctive IR system for a dataset of size 600 (300 intervals).	168

Chapter 1

Introduction

We're drowning in information and starving for knowledge.

Rutherford D. Rogers

Since its conception, over 30 years ago, the field of Artificial Intelligence has investigated a range of problems from disembodied intelligence through to autonomous mobile agents [51]. Whilst investigating these problems, many new areas of interest have emerged as the field matures and grows.

A significant new area of research is Spatial Reasoning. No matter what the domain of investigation, spatial reasoning will play some role. For some problems this role is minor. Yet for many, spatial reasoning plays a significant part in developing solutions to a variety of problems (i.e. autonomous agents).

Spatial reasoning is not restricted to the field of Artificial Intelligence. It crosses many fields of computing from Geographic Information Systems [11, 58] through to the automated placement of components in VLSI Design [4, 14]. Indeed, spatial reasoning is not bound just to the field of computing. Cognitive scientists have investigated biological systems in an attempt to understand the spatial reasoning process observed in organic agents [38, 53].

The traditional approach to spatial reasoning is to utilise the computational power of computers to provide brute force solutions to spatial reasoning problems. This has lead to difficulties with the amount of information (and computation) required and the types of solutions provided by these systems.

These quantitative methods focus upon gathering precise information and returning equally precise information about spatial relationships or spatial actions. An alternative method is to classify and represent information according to a set of qualitative values. This simplifies the problem and provides accurate and efficient (though not precise) spatial information.

These types of qualitative solutions have arisen due to the demands for a higher speed of response and the production of results that have greater significance. Quantitative systems tend to become ensnared in large amounts of computation and produce, what is for some problems, overly precise information.

The types of problems for which qualitative spatial reasoning systems are most appropriate are those that require meaningful responses but do not require a high level of precision. Upon these types of systems queries may take the form of:

Query: Where is object A wrt object B?

Satisfactory response: To the left.

The type of information to be queried from the system determines what information needs to be represented and how reasoning about this information is performed. If a system requires qualitative information and is provided with quantitative data then much of this quantitative data can be ignored (via classification) as it is not needed for producing the qualitative responses to queries.

Qualitative systems may be used in conjunction with quantitative systems to provide a level of preprocessing to aid in the search for precise information. In our work we shall focus upon the problem of using qualitative information for multi-dimensional spatial reasoning.

1.1 Multi-dimensional Qualitative Spatial Reasoning

Space is vast, both in the possible positions of objects in space and the possible shapes of the objects. Yet queries about object relations and shape tend to take, in many situations, a distinctly qualitative nature.

Thus the domain of qualitative representation and reasoning with multi-dimensional space takes the form of large amounts of precise information upon which imprecise queries are performed. The choice of representation of the quantitative information in terms of the qualitative classifications plays an important role in determining what type of information is represented and thus what type of information can be extracted from the representation.

There are two significant choices for the qualitative representation of spatial information:

1. the method of representing the shape of an object and
2. the method of representing the position of the object in space.

These are not mutually exclusive choices. The shape of an object will dictate the representation of the object's position and vice versa. Further considerations include the types of queries to be performed on the representation, the efficiency of available methods of representing this information and the completeness of the information about the object's position and shape.

Once the quantitative information has been abstracted into qualitative information according to the above qualitative choices, there are significant restrictions upon the queries that can be made upon this representation. Much of the precise information about objects and their relations will be lost in this abstraction process thus preventing precise queries. Certain types of qualitative representation will prevent the querying of other types of qualitative information from that representation.

One of the most significant aspects of representing spatial information is the varying amounts of detail that are available about the shapes of objects and their relationships. The incompleteness of this information requires that the known details be used to infer some information about those unknown details of shapes and relationships. The incomplete information may be considered to be a form of uncertainty about relations, where the known information is not exact but falls within a set of values.

We now begin the process of restricting the domain in order to begin making choices about the information to be represented (and extracted) and the method

of representing (and extracting) that information.

1.1.1 Qualitative Spatial Reasoning using Intervals

The types of queries of the spatial information that we consider are:

- purely relational (no distance information);
- not precise (quantitative values are not required);
- multi-dimensional;
- limited shape information.

A qualitative representation of spatial information provides a method of classifying objects and the basis for constructing a query language for extracting information based on these classifications.

The method of qualitative representation that will be used in this thesis is based upon one dimensional intervals. By dividing space into three global perpendicular axes objects can be represented as intervals along these axes. The intervals will not be represented using quantitative information but based upon relations between intervals of the same dimension.

Much of the inspiration for this interval based representation of objects in space has been generated from the work in the domain of temporal reasoning. In particular, Allen [2] provides a powerful method of qualitative classification of interval relations (see Table 1.1). Allen also provided a method of representing these interval relations [2].

The effect of this qualitative interval representation on spatial objects and their relations is to reduce objects to simple approximations along each axis. Essentially, objects are represented as intervals in one dimension, bounding boxes in two dimensions and cuboids in three dimensions. The qualitative representation of the objects' relations is purely relational (defined in terms of their relationships to other objects).

Thus given our criteria that queries will be about the relationships between objects, that these relations need not be precise, that no distance information is

Table 1.1: Allen’s Interval Relations [2].

Relation	Symbol	Inverse	Pictorial Example
X before Y	b	bi	XXX YYY
X equal Y	=	=	XXX YYY
X meets Y	m	mi	XXXYYY
X overlaps Y	o	oi	XXX YYY
X during Y	d	di	XXX YYYYYYY
X starts Y	s	si	XXX YYYYY
X finishes Y	f	fi	XXX YYYYY

required and that limited shape information is queried: a qualitative representation using the interval based approximations is an adequate representation.

Objects are given a qualitative shape (interval) and position (relative to the other objects). We shall investigate the domain of temporal reasoning to provide methods for efficiently representing the interval information and techniques for dealing with the problems of incomplete information.

The representation of interval information that is incomplete provides the greatest obstacle for our proposed system. Allen’s idea of constructing closure is expensive (NP-Complete) and alternative systems of representation fail to capture the expressiveness of Allen’s system. It will be our task to develop a new method of efficiently representing incomplete interval information that allows for efficient querying of the interval information.

The proposed type of system is not intended as a complete solution to the spatial reasoning problem. It provides a solution for dealing with qualitative spatial information.

1.2 Contribution

Using intervals to represent space is not a unique approach. Our contribution is two-fold:

1. the method of classifying the spatial relations and the means of querying these relations;
2. the process of efficiently representing incomplete interval information and the means of efficiently querying this information.

Our system of classification of objects in multi-dimensional space using intervals makes use of the characteristics of space that are not present in temporal interval reasoning. This provides a logical method for performing multiple levels of queries about the information represented by the multi-dimensional intervals.

The efficient representation of qualitative interval information has been the focus of considerable effort in the field of temporal reasoning. Our approach is to represent intervals in terms of end points and interval relations in terms of conjunctions of point relations. For systems where information is incomplete, traditional methods either perform closure upon the representation or complete minimisation of the information [41].

Our method is to perform a type of minimisation that reduces the amount of explicit representation without requiring expensive construction costs. This is accomplished by limiting minimising in the graph to the area local to the addition of new information. We shall demonstrate how our system is able to provide information about individual interval relations without performing closure upon the representation.

1.3 Outline

In Chapter 2 we present the methods that are used for qualitative spatial reasoning and the representation of interval information. We examine a variety of different spatial representations looking at the benefits and costs of each. In the

domains of spatial and temporal reasoning the issues of point and interval representations are heavily debated. We shall look at this debate and study the significant contributions. When addressing the problem of representing interval information, where the information about these intervals is incomplete a balance must be struck between the construction time of the representation, the search time for interval relations and the space required to represent that information. We shall examine two key techniques that provide alternative methods of representing incomplete interval information.

Our work in multi-dimensional qualitative spatial reasoning is the focus of Chapter 3. We introduce our method of representing objects in space in terms of intervals and look at the significance of the lack of temporal flow in providing the ordering of intervals in space. Subsequently, we develop a querying system that allows for multi-level queries of the spatial relations.

In Chapter 4, we begin our analysis of the problem of representing interval information. For a simplistic type of data, a set of intervals where information about every interval relation is explicitly known and these relations are unambiguous, we show how the spatial representation system of Chapter 3 may be effectively represented using the end points of intervals. Artificial and real datasets are used to demonstrate the effectiveness of this system.

Beyond the trivial interval representation described in Chapter 4, Chapter 5 investigates the problem of representing a set of intervals where information about every interval relation is not explicitly known though those relations that are known are unambiguous. We demonstrate the performance of a point based representation that uses a variant of minimisation to represent the information. The method offers efficient construction time and space usage, as well as an efficient querying mechanism.

Chapter 6 investigates the problem of representing sets of intervals where information about every interval relation is not explicitly known and those that are known may be uncertain (one of a limited set of possibilities). We extend the system of Chapter 5 to provide a method of representing this type of interval information. Our results indicate that this system is able to extract information

about individual interval relations and provide information about other interval relations on which these solutions are dependent.

The concluding remarks are presented in Chapter 7.

Chapter 2

Background

Deep in the human unconsciousness is a pervasive need for a logical universe that makes sense. But the real universe is always one step beyond logic.

from 'The Sayings of Muad'Dib' by the Princess Irulan

The problem of representing and reasoning with space is complex and multifaceted. There is no single method of representation or reasoning that is superior to all others. Preferable methods for representation are dependent upon the tasks that are to be performed and the type of information that is represented. In certain situations precise quantitative information is necessary. In other situations, this precision inhibits the ability to access the data or complicates the representation and hence the reasoning that is to be performed.

Spatial representation and reasoning systems for 2-D and 3-D are often founded upon 1-D temporal representation and reasoning systems. We shall examine techniques of representing time to provide an insight into spatial representation.

In this chapter, we focus upon a selection of work that examines the problem of qualitative reasoning in time and space. We look at two approaches to representing temporal information, closure-based (Section 2.1.1) and methods that do not use closure (Section 2.1.2), that use either intervals or points as temporal atomic units. Qualitative spatial reasoning techniques – which have drawn upon the fields of temporal reasoning, geographic information systems and robotic navigation – are examined in Section 2.2. The conclusions are presented in Section 2.3.

2.1 Time

An analysis of the work in spatial reasoning would be incomplete without an analysis of the work in time. The techniques used in temporal reasoning are in many cases directly applicable to space. This application is limited due to the difference in the number of dimensions required to represent time and space, with space being 2 or 3 dimensional whereas time, arguably, can be modeled as a single dimension. Thus the use of temporal techniques in space must accommodate the problems introduced by multi-dimensionality and other properties unique to space.

2.1.1 Closure

There are two main methods for representing qualitative temporal information: as intervals or as points. An *interval* may be represented as a time duration that has a begin point and an end point. Similarly, a *point* represents a single instant in time. A quantitative representation of this information would involve precise values indicating the position of the points. Qualitative representations use relations between points or intervals that exclude quantitative information such as the precise positions of the points. The qualitative representations take advantage of the ordering of the points or the relationship between the intervals to represent the relations between points and intervals, respectively.

In this chapter we will focus on qualitative representations. Before proceeding into a discussion of various reasoning techniques, we define the basic representation for temporal information.

We define an abstract network of binary relations, $\langle TAU, TR \rangle$, where each node of the network represents a *temporal atomic unit*, TAU , and the edges between the nodes represent *temporal relations*, TR . A set of temporal relations is defined by a set of *primitive relations*, PR , and a set of *disjunctions of these primitive relations*, DPR , where each disjunct is of the form $PR_1 \vee PR_2 \vee \dots PR_m$ where $m \leq |PR|$. The properties of the temporal atomic units and their temporal relations, such as transitivity and asymmetry, allow for deductions. The intersection operator “+” denotes the combination of two alternative TRs for a

given pair of TAUs. The primitive relations that are members of both TRs form the result of the intersection.

$$(PR_1 \vee PR_2 \vee PR_3) + (PR_1 \vee PR_4 \vee PR_3) = (PR_1 \vee PR_3)$$

The composition operator “ \times ” uses a set of rules derived from the properties of the temporal atomic units and their primitive relations to propagate relations between pairs of temporal atomic units. Given two pairs of TAUs (A, B) and (B, C) and their relations TR_{AB} and TR_{BC} the relation TR_{AC} may be determined from a set of rules ($TR_{AB} \times TR_{BC} = TR_{AC}$).

The network and the operations that may be performed on the network amount to a relational algebra [66].

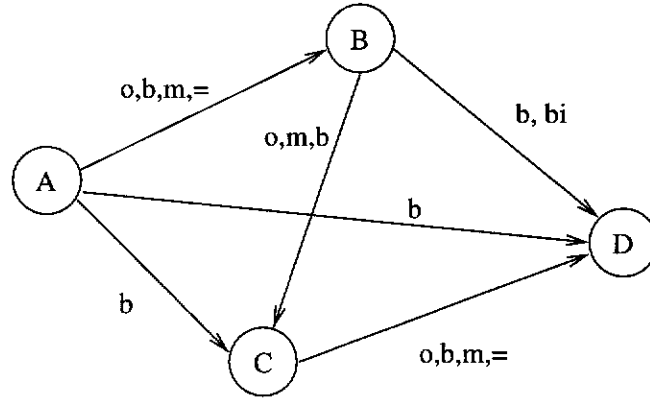
Given this algebra, we are interested in establishing all possible deductions from the initial network of binary relations (*closure*). There are three properties of closure on this algebra that are of interest [32].

- *Satisfiability problem (SAT)* is the problem of determining if for every pair of TAUs (A, B), a primitive relation from the TR_{AB} can be selected such that those relations are consistent. That is, there exists an assignment of values to all TAUs which satisfies all of those selected relations (see Figure 2.1i).
- *Minimal labeling problem (MLP)* is the problem of determining the TR' for every TR, $TR' \subset TR$ such that every element of the TR' is used in a consistent solution (see Figure 2.1ii).
- *All consistent solutions problem (ACSP)* is the problem of determining all consistent combinations from each TR (see Figure 2.1iii).

The SAT, MLP and ACSP are polynomially equivalent [32].

2.1.1.1 Interval Algebra

Interval algebra (IA) [2] is an algebra with 13 primitive relations (Table 2.1), 2^{13} disjunctions and composition rules as described in Allen’s transitivity table (Table 2.1.1.1). The 13 primitive interval relations are based upon seven basic relations and their inverses (except *equal* which is its own inverse).



i) A SAT: $A \prec B, B \prec D, A \prec C, C \prec D, B \prec C, A \prec D$

ii) MLP: $A \{o,b,m,=\} B, B \prec D, A \prec C, B \{o,m,b\} C, A \prec D, B \{o,m,b\} C$

iii) Part of ASCP:

$A \circ B, B \circ C, C \circ D, A \prec C, B \prec D, A \prec D$
 $A \prec B, B \prec C, C \prec D, A \prec C, B \prec D, A \prec D$
 $A \prec B, B \prec C, C \prec D, A \prec C, B \prec D, A \prec D$
 $A = B, B \prec C, C = D, A \prec C, B \prec D, A \prec D$

Figure 2.1: An example of i) SAT, ii) MLP and iii) ACSP

The following is an example of a disjunction of interval relations that indicates that **A** is either before (b) or after (bi) **B**:

$$A \{b, bi\} B.$$

The transitivity table provides the means of propagating interval relations. Relations that are disjunctions of relations between a pair of intervals may also be propagated by propagating each of the interval relations in the disjunction. For example, the transitivity table can be used to propagate disjunctive interval relations [2] as follows:

$$A \prec B \times B \{d \ di\} C \Rightarrow A \{b \ o \ d \ s\} C$$

This is derived from the transitivity table using the following deductions:

$$A \prec B \times B \prec C \Rightarrow A \{o \ d \ s\} C$$

$$A \prec B \times B \ di \ C \Rightarrow A \prec C$$

Table 2.1: Allen's Interval Relations [2].

Relation	Symbol	Inverse	Pictorial Example
X before Y	b	bi	XXX YYY
X equal Y	=	=	XXX YYY
X meets Y	m	mi	XXXYYY
X overlaps Y	o	oi	XXX YYY
X during Y	d	di	XXX YYYYYYY
X starts Y	s	si	XXX YYYYY
X finishes Y	f	fi	XXX YYYYYY

SAT, MLP and ACSP are NP-Complete for Interval Algebra [32, 75].

A simpler representation of Interval Algebra groups the 13 primitive interval relations into 3 different classes. These three classes become the primitive relations for a new algebra. Consider an algebra A_3 which is defined as follows:

$$A_3 : \{ b, bi, \cap \}$$

$$\text{where } \cap = \{ m, mi, o, oi, s, si, f, fi, d, di, = \}$$

SAT for A_3 is NP-Complete [32]. Yet we are able to define restricted subsets for this algebra for which SAT is polynomial. The full set of relations for A_3 is:

$$\delta_0 = \{ b, bi, \cap, b \cap, \cap bi, b bi, b \cap bi \}$$

For the following three restricted subsets of A_3 , the SAT is polynomial:

$$\delta_1 = \{ b, bi, \cap, b \cap, \cap bi, b \cap bi \}$$

$$\delta_2 = \{ b, bi, \cap, b bi \}$$

$$\delta_3 = \{ b bi, \cap, b \cap bi \}$$

**Table 2.2: Allen’s Transitivity Table (omitting ‘=’) [2] where
 $(A \text{ r1 } B) \times (B \text{ r2 } C) = (A \text{ r3 } C)$, $\text{dur} = (d \wedge s \wedge f)$ and $\text{con} =$
 $(di \wedge si \wedge fi)$**

B r2 C	b	bi	d	di	o	oi	m	mi	s	si	f	fi
A r1 B												
“before” b	b	no info	b o m d s	b	b	b o m d s	b	b o m d s	b	b	b o m d s	b
“after” bi	no info	bi	bi oi mi d f	bi	bi oi mi d f	bi	bi oi mi d f	bi	bi oi mi d f	bi	bi	bi
“during” d	b	bi	d	no info	b o m d s	bi oi mi d f	b	bi	d	bi oi mi d f	d	b o m d s
“contains” di	b fi o m di	bi oi di mi si	oi = dur o con	di	o di fi	oi di si	o di fi	oi di si	di fi o	di	di si oi	di
“overlaps” o	b	bi oi di mi si	o d s	b fi o m di	b o m	oi = dur o con	b	oi di si	o	di fi o	d s o	b o m
“overlapped- by” oi	b fi o m di	bi	oi d f	bi oi mi di si	oi = dur o con	bi oi mi	o di fi	bi	oi d f	oi bi mi	oi	oi di si
“meets” m	b	bi oi mi di si	o d s	b	b	o d s	b	f fi =	m	m	d s o	b
“met-by” mi	b fi o m di	bi	oi d f	bi	oi d f	bi	s si =	bi	d f oi	bi	mi	mi
“starts” s	b	bi	d	b fi o m di	b o m	oi d f	b	mi	s	s si =	d	b m o
“started by” si	b fi o m di	bi	oi d f	di	o di fi	oi	o di fi	mi	s si =	si	oi	di
“finishes” f	b	bi	d	bi oi mi di si	o d s	bi oi mi	m	bi	d	bi oi mi	f	f fi =
“finished-by” fi	b	bi oi mi di si	o d s	di	o	oi di si	m	si oi di	o	di	f fi =	fi

For a fourth restricted subset, the SAT is NP-Complete.

$$\delta_4 = \{ b \text{ bi}, b \cap, \cap \text{ bi}, b \cap \text{ bi} \}$$

See Golumbic and Shamir [32] for proofs.

Since it has been shown that for restricted subsets of A_3 there are polynomial

solutions, we can expect that there will be restricted subsets of Interval Algebra for which SAT is polynomial.

2.1.1.2 Point Algebra

Point Algebra is an algebra with 3 primitive relations ($\{<, >, =\}$), 2^3 disjunctions ($\{<, >, =, <=, =>, <=>\}$) and composition rules as defined in Table 2.3.

Table 2.3: The composition table for the primitive point relations.

	B r2 C		
	<	>	=
A r1 B			
<	<	<=>	<
>	<=>	>	>
=	<	>	=

Similar to Interval Algebra (IA), the constraint table of Point Algebra (PA) is used to propagate disjunctions of point relations by propagating each of the primitive point relations individually and finding the union of the resulting relations. For example, the composition of the following relations:

$$(A \{<= \} B) \times (B \{=> \} C) \Rightarrow A \{<=> \} C$$

is derived from the following primitive point compositions:

$$\begin{aligned} (A < B) \times (B = C) &\Rightarrow A < C \\ (A = B) \times (B = C) &\Rightarrow A = C \\ (A < B) \times (B > C) &\Rightarrow A <=> C \\ (A = B) \times (B > C) &\Rightarrow A > C \end{aligned}$$

A path consistency [52, 54] algorithm (first introduced by Allen [2] for IA) can be used to determine if every triple (set of three nodes) in the network is consistent with respect to each other (3-consistency) [75, 67]. This does not determine the consistency of the entire PA network. For a restricted subset of relations

$PA_c = \{<, >, =, <=, =>, <=>\}$ the path consistency algorithm computes SAT in $O(n^2)$ [75]. An extended consistency algorithm (4-consistency) [70] computes SAT for PA in $O(n^4)$ time. New techniques have improved this, such that SAT and MLP can be computed in $O(n^2)$ and $O(mn^2, n^3)$ respectively, where n is the number of points and m is the number of pairs of points whose relation is *not equal* (the disjunction $\{<>\}$) [29, 69].

An interval may be represented in terms of its extreme points (end points). Thus an interval \mathbf{X} may be represented as an ordered pair of points $x < X$ where x is the first extreme point (begin point) and X is the second extreme point (end point). For the rest of this thesis \mathbf{X} denotes the interval and x, X denote the interval's end points.

It is possible to represent an interval in terms of its end points and an interval relation in terms of a conjunction of end point relations. Not all interval relations in Interval Algebra can be correctly represented by Point Algebra using this method. The interval relations that cannot be represented by PA are those relations that represent “disjointness” between intervals. A complete enumeration of all relations from IA that can be correctly represented in PA is given in [70]. The subset of Interval Algebra that can be represented using Point Algebra is denoted SA.

The following example indicates why certain interval relations cannot be represented using PA. The interval relation $\mathbf{A} \{ b, bi \} \mathbf{B}$, depicted as two conjunctions of point relations in Figure 2.2i (bold line depicts the relation $\mathbf{A} bi \mathbf{B}$ and dashed line depicts the relation $\mathbf{A} b \mathbf{B}$), is an IA relation that represents “disjointness” of end points.

The interval relation is represented as a disjunction of two conjunctions

$$((a < b) \wedge (a < B) \wedge (A < b) \wedge (A < B)) \vee ((a > b) \wedge (a > B) \wedge (A > b) \wedge (A > B))$$

is represented in Point Algebra as:

$$(a <> b) \wedge (a <> B) \wedge (A <> b) \wedge (A <> B)$$

When the PA representation (Figure 2.2ii) is queried about the relationship between \mathbf{A} and \mathbf{B} , the following interval relation is returned $\{b, bi, o, oi, d, di\}$, indicating that the PA representation has not accurately represented the

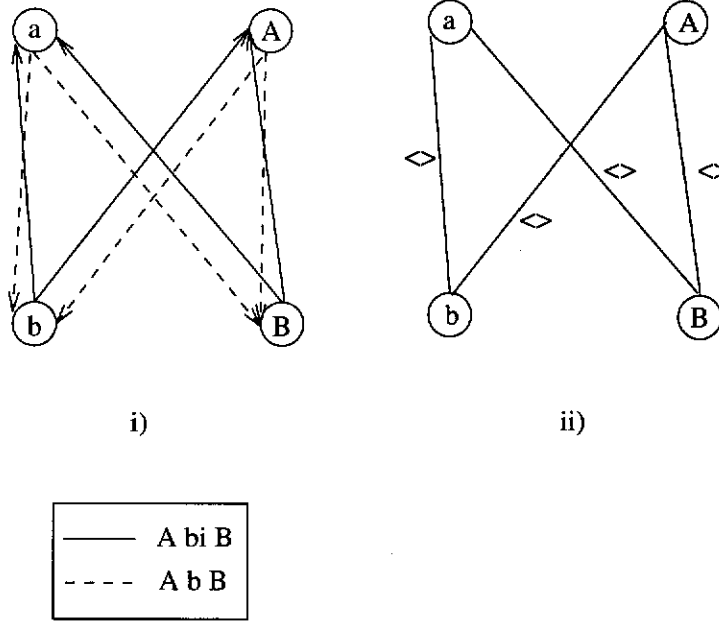


Figure 2.2: The interval relation $A \{b, bi\} B$ represented as i) two conjunctions of point relations and ii) using PA

intended information. Thus for certain interval relations, PA is unable to express the relational information.

The work in interval constraints and an analysis of Point Algebra [43, 44, 68, 74] have provided much of the basis for this work. Heuristics for Interval Algebra [71], non-convex intervals [42] and constraint propagation [45] examine other issues related to IA and PA that is outside the scope of this thesis.

2.1.1.3 Summary

The tasks of computing SAT, MLP and ACSP for Interval Algebra are NP-Complete. The SAT, MLP and ACSP for Point Algebra are polynomial but PA is not as expressive as Interval Algebra (there are relations in IA that cannot be represented by PA). For restricted subsets of Interval Algebra, the tasks of computing SAT, MLP and ACSP are polynomial.

The tractable restricted subsets of Interval Algebra do provide valuable domains but are limited to problems that rely on these restricted subsets of relations. It is our intention to focus upon the more general problem of representing Interval Algebra via alternative methods.

2.1.2 Non-Closure representation

The ability to compute SAT, MLP, ACSP for an algebra provides information about: whether or not there is a consistent interpretation, all the possible consistent labelings for a given relation and all the consistent combinations, respectively. The effectiveness of these solutions is mitigated by the fact that these problems are NP-Complete for Interval Algebra.

The issues of concern for any representation are the construction time, the space requirements, the query time and the ability to determine the consistency of the representation. An alternative method to computing closure on a representation, in order to determine the SAT, MLP or ACSP of the representation, is to provide a representation such that its consistency can be established via other means. The purpose of such a system is to provide efficient means of representing the information, querying and determining consistency.

These types of systems can be broadly classified as being non-closure systems. Non-closure systems are divided into:

- tractable subsets of Interval Algebra (like SA),
- full Interval Algebra.

For each of these types of systems, methods are examined for determining the consistency of the representation and provide efficient construction and query times.

We examine two systems that use non-closure methods for representing tractable subsets of algebras. A Time-Map Manager called Indexed Time Table (IxTeT) [31] and Timegraphs-II [28] represent only explicit relations about points. Timegraphs-II represents a larger algebra than SA.

Only explicit information about point relations is represented in IxTeT. The system has the same expressive power as PA. For every relation in PA, IxTeT represents the relationship between two points using between 0 and 2 links from the following set $\{<=, <>\}$. All $<=$ loops (cycles) in the network, where no pair of nodes used to construct the loop is connected by the $<>$ link, may be reduced to a single node. The consistency of the network is established by determining if,

when all such loops have been collapsed and the $<>$ links removed, there are no $<=$ loops in the network [31]. Otherwise, the representation contains inconsistent information.

The problem of searching for the relationships between two points is streamlined by translating the network into a maximum spanning tree. Ranks are assigned to the nodes according to the tree structure. The ranking system is used to help determine the relationship between points during the search process. The system allows for the addition and the removal of facts. Experimental results indicate a linear time and space for construction of the representation, for the retrieval, the addition and the removal of information.

IxTeT does not provide a complete solution to the problem of representing $<>$ relations as it “cannot derive some strict orderings induced by $<>$ relations” [28, p.4]. Timegraphs-II is a system that represents the explicit PA relations in a graph. Edges between nodes in the graph are restricted to the following: $\{<, <=, <>\}$. A $<$ -path is a path where every edge (link) in the path is labeled with the relation $<$. A $<=$ -path is a path that includes only labels $<$ or $<=$.

A timegraph is consistent if, similarly to IxTeT, there exists no $<=$ -cycle or $<$ -cycle where any pair of the nodes in the cycle is connected by a $<>$ edge. In non-closure systems, such as Timegraphs, querying about relations is a non-trivial problem. Timegraphs-II uses two mechanisms for efficient queries. The system has been designed for data that consists of groups of well-ordered points called *chains*, which are $<=$ paths, where no node from one chain belongs to another chain. A second method, similar to IxTeT, ranks elements of the nodes in the chains and the nodes are assigned *pseudo-times* to indicate their ranks with respect to the entire graph.

Links between chains, called *cross links* (or *meta-edges*), indicate the relationship between nodes in different chains and *meta-vertices* indicate vertices that are used by the cross links. *Transitive edges* are edges that connect nodes in the same $<=$ -chain. To help the query process, all implicit $<$ links are made explicit. An example of the timegraph’s representation is given in Figure 2.3 (from [28]).

The consistency can be determined in $O(e)$ time; ranking of the vertices in

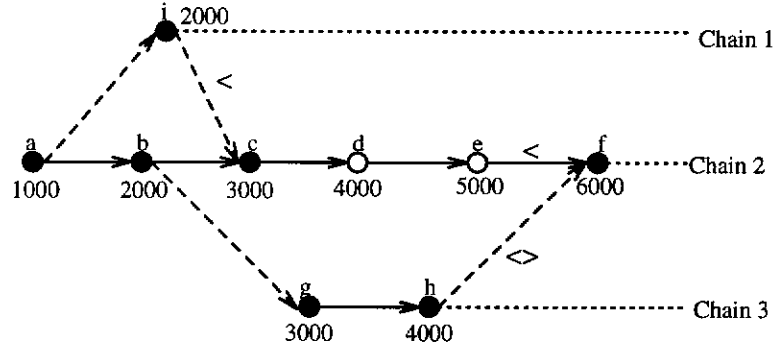


Figure 2.3: An example of Gervini's Timegraph representation where links between nodes in chains are solid edges, cross-edges are dashed, solid nodes are meta-vertices and unlabeled edges are \leq links.

$O(n + e)$ time and minimising cross links in $O(n + e)$ where n is the number of vertices and e is the number of edges.

This organisation of Timegraphs-II allows four types of queries to be computed in constant time, where:

- two points are alternative names of the same node (i.e. two points are equal);
- two points are members of the same chain;
- the points have the same pseudo-time and there is no $\langle \rangle$ link between them;
- there is a $\langle \rangle$ link connecting them.

For the remaining situations, where there maybe a \leq -path or a \leq -path between the nodes, queries are computed in $O(k + \hat{e} + \hat{n})$ where \hat{e} is the number of meta-edges, \hat{n} is the number of meta-vertices and k is a constant. The performance of this system is best when data is organised into chains of ordered points and thus the numbers of meta-edges and meta-vertices are low.

An extension to Timegraphs-II involves the addition of disjunctions of PA relations.

For example,

$$A \leq B \vee A \langle \rangle B$$

This allows for the expression of at least 2^{11} relations from Interval Algebra [28]. This type of representation allows for non-binary relations i.e. **I** before **J** or **I** after **K**.

Disjunctions of PA relations are maintained using a separate table and the disjunctions in the table are maintained using a set of pruning rules (based on derivability, tautology and resolution) to keep a minimum set of disjunctions. The consistency checking for this disjunctive timegraph is NP-Complete.

Other non-closure systems include Tachyon [65] (a point-based representation that integrates quantitative information), Sequence Graphs [15] (that exploit the ordering of intervals) and Timegraphs [30] (the predecessor to Timegraphs-II).

2.1.3 Discussion

The SAT, MLP and ACSP closures are NP-Complete for Interval Algebra and polynomial for Point Algebra. SAT for Point Algebra is $O(n^2)$ and MLP for Point Algebra is $O(mn^2, n^3)$ where n is the number of points and m is the number of $<>$ relations. Non-closure systems (for PA) are able to determine consistency, construct representation and search in linear time for restricted types of datasets. For more general problems, the complexity becomes $O(n^2)$.

The intractability of IA prevents any practical approach for representing large datasets of interval relations. Timegraphs-II extension indicates that there are partial solutions to specific problems.

2.2 Space

We shall examine the use of qualitative spatial reasoning in multi-dimensional space by examining key aspects of this domain. Using these aspects we shall describe several systems for representing and reasoning with multi-dimensional qualitative spatial information.

2.2.1 Relative vs global axes

There are two alternative frames of reference that may be used. A global reference selects a particular set of directions, i.e. compass directions or global axes, and relates all objects with respect to this reference. Relative frames of reference have been used to model human cognitive processes such that references are made with respect to the observer or the object [37, 60].

The use of relative frames of reference introduces problems related to the understanding of the objects. There are three different types of relative frames of reference.

- An *intrinsic* reference indicates that the relationship between the two objects is determined by one of the objects having a distinct surface that is assigned to be the *front* of that object and the relationship between the two objects is expressed with reference to this surface.

For example, “It was terrible, the boy was standing right *in front* of the car” (Figure 2.4i).

- An *extrinsic* reference indicates that the relationship between the two objects is expressed with reference to an external influence (an object or a given action).

For example, “The boy was standing *in front* of the fire hydrant”. The relation *front* was attributed to this relationship because the fire hydrant is assigned a “front” as it was *in front* of the large building (Figure 2.4ii).

- A *deictic* reference is dependent upon the perspective of the viewer. The relationship between the two objects is dependent upon the position of the objects relative to the position of the viewer.

For example, “The ambulance stopped *in front* of the boy” (Figure 2.4iii). From the observer’s point of view the ambulance is *in front* of the boy.

The use of these types of relative frames of reference requires information about: which objects have significant distinct surfaces, the frame of reference to be used and how relations can be translated between different relative references [37].

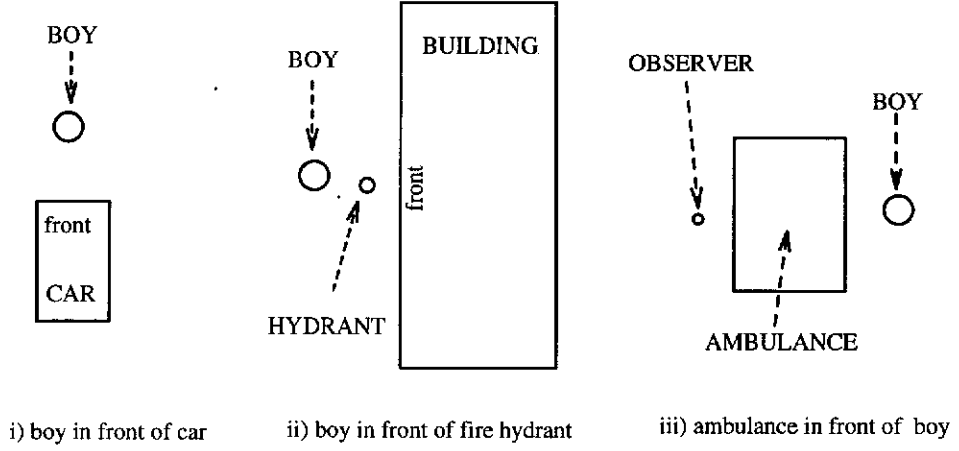


Figure 2.4: Examples of relative frames of reference: i) intrinsic, ii) extrinsic and iii) deictic.

2.2.2 n-D techniques

The qualitative representation of objects and their relations in multi-dimensional space has been based upon extending 1-D techniques to n-D. We shall look at how three different methods of representing 1-D information can be augmented to represent n-D space. The three 1-D representations are: point-based, interval-based and topology-based.

Topology considers the properties of objects with respect to how they intersect [18, 22]. Objects are represented as datasets and the relationship between the objects is represented in terms of their interiors ($^{\circ}$) and boundaries (∂).

The possible relationships between two objects expressed in terms of the intersection of the interiors and boundaries is represented in Table 2.4.

This topological information can be represented as an algebra with 8 primitive relations, 2^8 disjunctions and a composition table [19, 37].

The simplex concept is used to construct object descriptions. A 0-simplex is a node; a 1-simplex is an edge (a connection between two 0-simplexes); a 2-simplex is a triangle (three 1-simplexes), etc [21]. Complex simplexes can be constructed from a set of simplexes of the same dimension. In an object made up of simplexes, the boundary is the external skeleton of the simplex (see Figure 2.5).

An n-dimensional object has n different ways of meeting another object (see Figure 2.6) [23]. Extensions to the topological system include topologies with

Table 2.4: Topological relationships between two objects [18].

	$\partial \cap \partial$	$^{\circ} \cap ^{\circ}$	$\partial \cap ^{\circ}$	$^{\circ} \cap \partial$
disjoint	\emptyset	\emptyset	\emptyset	\emptyset
meet	$\neq \emptyset$	\emptyset	\emptyset	\emptyset
overlap	\emptyset	$\neq \emptyset$	$\neq \emptyset$	$\neq \emptyset$
inside	\emptyset	$\neq \emptyset$	$\neq \emptyset$	\emptyset
contains	\emptyset	$\neq \emptyset$	\emptyset	$\neq \emptyset$
covers	$\neq \emptyset$	$\neq \emptyset$	\emptyset	$\neq \emptyset$
coveredBy	$\neq \emptyset$	$\neq \emptyset$	$\neq \emptyset$	\emptyset
equal	$\neq \emptyset$	$\neq \emptyset$	\emptyset	\emptyset

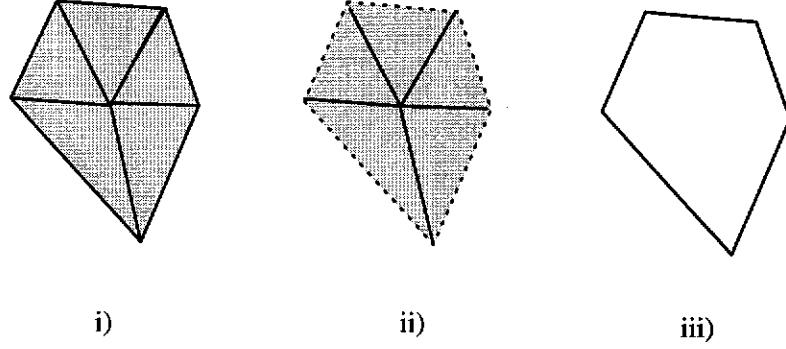


Figure 2.5: i) A complex simplex: ii) the interior and iii) the boundary

holes [20] and integrating topology with order [39].

Interval representations in 1-D time have been discussed previously in Sec-

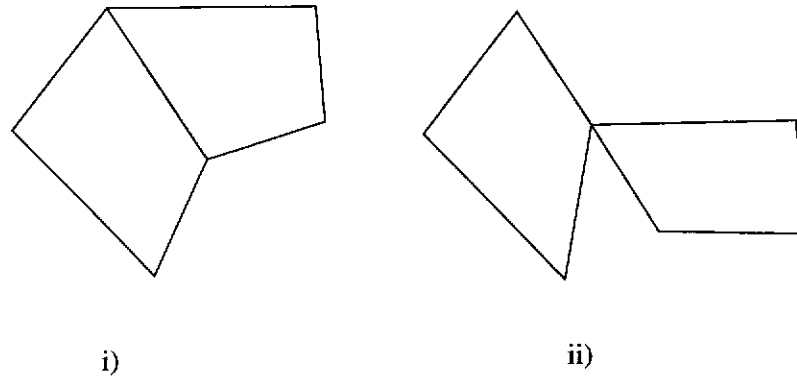


Figure 2.6: Two different *meets* relations for 2-D relations i) 1-meets and ii) 0-meets

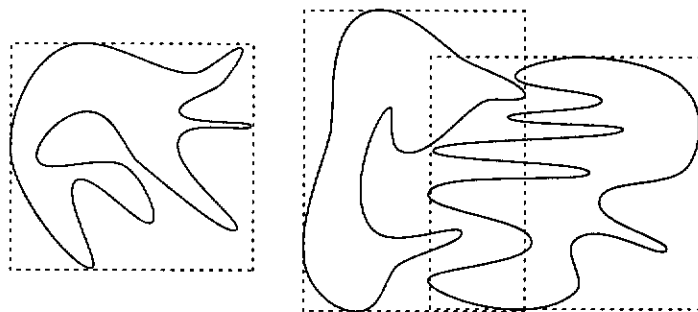


Figure 2.7: 2-D objects represented as bounding boxes.

tion 2.1.1.1. This system may be directly applied to the problem of representing 1-D space and extended for representing n -D space. An object in n -D space may be represented as n intervals along n orthogonal 1-D axes, and thus, 2-D objects would be represented as two 1-D intervals and 3-D objects as three 1-D intervals.

This representation introduces approximations to the shape of the n -D objects. For example, a 2-D object is represented by its bounding box, and relations between objects are expressed in terms of the relationships between these bounding box approximations (see Figure 2.7).

The representation of time as points has been discussed in Section 2.1.1.2. Points may be used to represent objects in n -D space. This results in a complete loss of information about the shape of the objects. Point representations may be used in conjunction with other methods to augment the representation such that information about an object's shape and the relationship between objects can be represented.

2.2.3 Qualitative spatial reasoning systems

An object can be represented as a single point. The virtue of this type of a system is that the points may be ordered linearly and thus storage and access to information about the point relations is efficient. The problem with representing objects as points is that much of the information about object shape and, consequently, the relations is lost.

Chang et. al. [7, 11] and Lee et. al. [48] present a system of representing 2-D objects as two 1-D strings of points, called *2-D Strings*. Global axes are used so

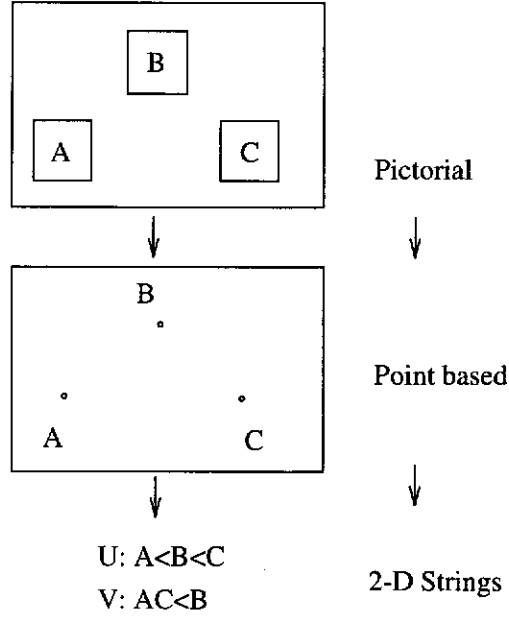


Figure 2.8: Objects may be represented as points and these points represented as 2D-Strings.

that all objects in the representation are related with respect to two orthogonal axes (see Figure 2.8). Using the relations $\{<, =\}$, a string of points is a linearly ordered set of points of the format:

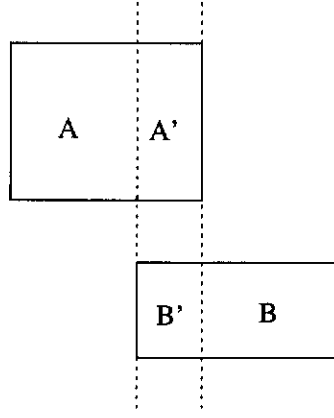
$$\{p_1 \ r_a \ p_2 \ r_b \ p_3 \ \dots \ r_\alpha \ p_\beta\}$$

where $p_\kappa \in$ the set of all points and $r_\mu \in \{<, =\}$. The notation AB may be used to represent equality between A and B (i.e. $A=B$).

Part of the work in strings has focused upon the use of similarity retrieval for matching strings [6, 12]. We shall not be looking at this work, as it is outside the scope of this thesis.

Methods to augment the point representation have focused upon increasing the expressive ability of the representation whilst maintaining the original efficiency of representing and accessing the information. By introducing an extra operator “meets”, Chang et. al. [8, 9, 10] changed the basis of representation and increased the expressiveness of the string based representation. The relation “meets” is a binary interval relation.

A vertical and horizontal line is projected from every object’s extreme points along the horizontal and vertical axes. When one of these lines crosses another

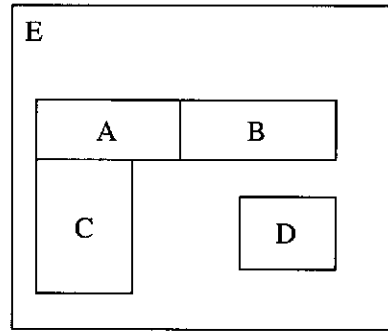


U: A|A'B'|B

Figure 2.9: Complex relations are reduced to simpler relations by subdividing objects.

object then that object is divided into two sub-objects. This process is performed for every object in the representation (see Figure 2.9). A string representation with an increased relational vocabulary $\{<, =, |\}$ (where $|$ is the relation “meets”) may be used to represent the resulting objects and their relations. This hybrid point-interval representation, called *2D G-Strings*, is able to accurately represent relations between objects (objects are treated as intervals) such that efficient representation and querying similar to the original 2D-Strings applies. The cutting reduces all thirteen interval relations to one of the three relations $\{<, =, |\}$, sets restricted to these three relations may be linearly ordered (String).

The main criticism of this type of representation is that when representing a large number of objects the numbers of sub-objects created via cutting is very large. Lee and Hsu [46, 47, 49] proposed increasing the number of interval relations used to represent the objects so that the need for cutting would consequently be reduced. The new set of relations are $\{<, =, |, \%, [,], /\}$ where the new operators are $\%$ - *during*⁻¹, $[$ - *starts*⁻¹, $]$ - *finishes*⁻¹ and $/$ - *overlaps*. Strings that do not use the relation *overlaps* are unambiguous (see Figure 2.10). The use of the *overlaps* relation generates ambiguous strings (see Figure 2.11). Thus the cut operator is used to eliminate the *overlaps* operator resulting in an unambiguous string that uses fewer cuts than G-Strings (see Figure 2.12).



U: $E\%((A[C])|(B)D))$

V: $E\%((C\%D)|(AB))$

Figure 2.10: A C-Strings example without the relation overlaps (note: extra round brackets are used to emphasise the operator precedence).

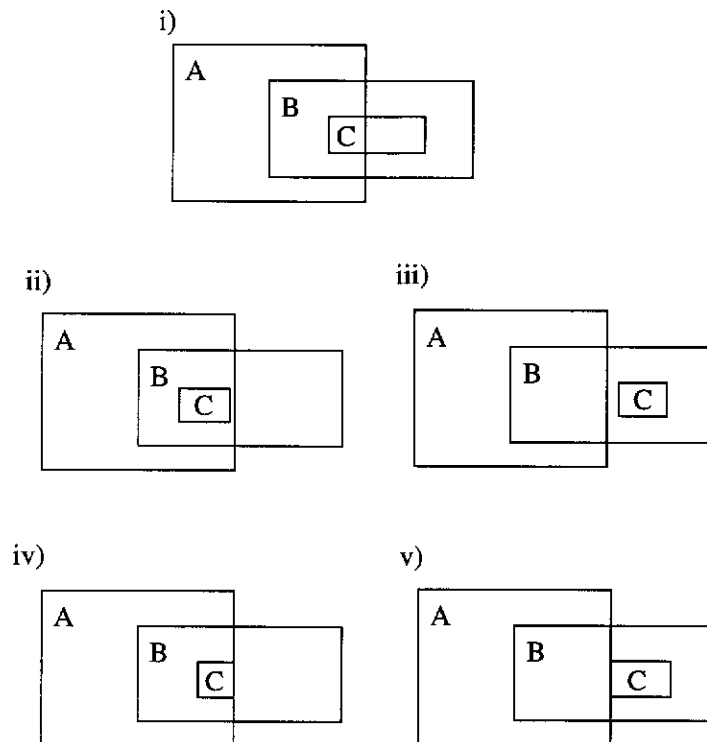


Figure 2.11: Illustrating problems with overlaps. The C-String Representation of i) $A / B \% C$ can be reconstructed as one of the relations in ii, iii, iv or v [49].

The resulting representation, called *2D C-Strings*, is able to represent all types of relations between objects efficiently (objects are treated as intervals).

The representation of intervals in terms of extreme points was proposed with

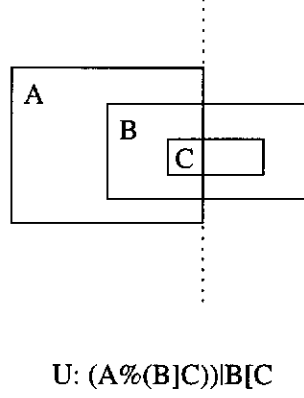


Figure 2.12: A single cut is needed to remove the overlaps relation

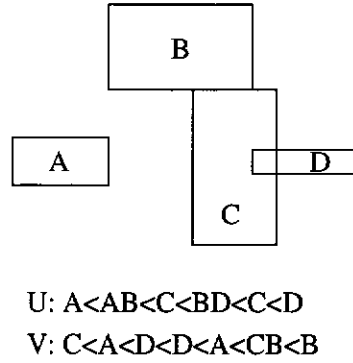


Figure 2.13: A B-String representation of the objects

respect to Point Algebra (in Section 2.1.1.2). Yang [50, 77] introduced the 2D B-Strings (see Figure 2.13), which is a linear ordering of the extreme points of objects. This representation is able to represent all interval relations and provide an efficient method for representation and access. 2D B-Strings capture all the simplicity of the original 2-D Strings, whilst being able to represent all primitive interval relations between intervals.

Interval representations of space have been defined using global and relative frames of reference. Guesgen [34, 35] defines an interval algebra with four primitive relations (Table 2.5), 2^4 disjunctions and a composition table as given in Table 2.6, where O_n represents the 1-D intervals used to represent the objects and S_n is the space the interval occupies. The algebra has the same properties of all types of interval algebra, and thus SAT, MLP and ACSP are NP-Complete in this case. The multi-dimensionality of the representation is a simple process of $n \times$ 1-D interval representations.

Table 2.5: Guesgen’s spatial interval algebra [35].

Relationship	Sym	Conv	Definition
O_1 left of O_2	\prec	\succ	$\forall x \in S_1, y \in S_2 : x < y$
O_1 attached to O_2	\preceq	\succeq	$\forall x \in S_1, y \in S_2 : x \leq y \wedge \exists x \in S_1, y \in S_2 : x = y$
O_1 overlapping O_2	\Leftarrow	\Rightarrow	$\exists x \in S_1 \forall y \in S_2 : x < y \wedge \exists y \in S_2 \forall x \in S_1 : x < y$ $\wedge \exists x \in S_1, y \in S_2 : x > y^1$
O_1 inside O_2	\sqsubset	\sqsupset	$\exists y \in S_2 \forall x \in S_1 : x < y \wedge \exists y \in S_2 \forall x \in S_1 : x > y$

A system of representing spatial information using intervals with a relative frame of reference has been developed for the domain of image database querying [13]. Objects in 3-D are represented in terms of their *minimum enclosing parallelepiped* (*mep*) and are given a directional component that is defined as the plane that passes through the centroid of the object and is orthogonal to the minimum axis of the *mep*.

Three different frames of reference are presented.

- A global referencing system, where *meps* of objects are translated as extreme points along the given axes.
- Object-centered referencing system, where the objects are given individual reference axes (similar to intrinsic and extrinsic).
- Observer-centered referencing, where the axes are relative to an observer’s point of view (similar to deictic).

Regardless of the referencing system, an indication is given about the directional component of the relations. There are three alternative criteria for classifying the directional component of a relation: collinear, parallel and non-parallel. Position relations of the 3-D objects are expressed in terms of three 1-D interval relations, similar to [35].

Papadias and Sellis [57] examine topological systems to augment both point and interval systems. A point-based representation of objects using a global set of

¹Definition modified due to error detected by [56]

Table 2.6: Composition rules for Guesgen's spatial interval algebra [35].

Relation between O_1 and O_2	Relation between O_2 and O_3							
	\swarrow	\searrow	\llcorner	\lrcorner	\Uparrow	\Downarrow	\sqsubset	\sqsupset
\swarrow	\swarrow	$?$	\swarrow	\swarrow, \llcorner \Uparrow, \sqsubset	\swarrow	\swarrow, \llcorner \Uparrow, \sqsubset	\swarrow, \llcorner \Uparrow, \sqsubset	\swarrow
\searrow	$?$	\searrow	\searrow, \lrcorner \Downarrow, \sqsupset	\searrow	\searrow, \lrcorner \Downarrow, \sqsupset	\searrow	\searrow, \lrcorner \Downarrow, \sqsupset	\searrow
\llcorner	\swarrow	\searrow, \lrcorner \Downarrow, \sqsupset	\swarrow	\sqsubset, \sqsupset	\swarrow	\sqsubset, \Uparrow	\Uparrow, \sqsupset^2	\swarrow, \llcorner
\lrcorner	\swarrow, \llcorner \Uparrow, \sqsubset	\searrow	\sqsubset, \sqsupset	\searrow	\Downarrow, \sqsupset	\searrow	\lrcorner, \Downarrow \sqsubset	\searrow, \lrcorner
\Uparrow	\swarrow	\searrow, \lrcorner \Downarrow, \sqsupset	\swarrow	\Downarrow, \sqsubset	\swarrow, \llcorner \Uparrow	\Uparrow, \Downarrow \sqsubset, \sqsupset	\Uparrow, \sqsubset	\swarrow, \llcorner \Uparrow, \sqsubset
\Downarrow	\swarrow, \llcorner \Uparrow, \sqsubset	\searrow	\Uparrow, \sqsubset	\searrow	\Uparrow, \Downarrow \sqsubset, \sqsupset	\searrow, \lrcorner \Downarrow	\Downarrow, \sqsupset	\searrow, \lrcorner \Downarrow, \sqsupset
\sqsubset	\swarrow	\searrow	\swarrow, \llcorner	\searrow, \lrcorner	\swarrow, \llcorner \Uparrow, \sqsubset	\searrow, \lrcorner \Downarrow, \sqsupset	\sqsubset	$?$
\sqsupset	\swarrow, \llcorner \Uparrow, \sqsubset	\searrow, \lrcorner \Downarrow, \sqsupset	\llcorner, \Uparrow \sqsupset	\lrcorner, \Downarrow \sqsupset	\Uparrow, \sqsubset	\Downarrow, \sqsupset	\Uparrow, \Downarrow \sqsubset, \sqsupset	\sqsupset

axes allows for the expression of directional information about object relations. The relational direction between two objects is described in terms of the relationships between the centroids of the objects along the two global axes. The following definitions are given for the eight compass directions (see Figure 2.14).

Given that $X(\text{object})$ and $Y(\text{object})$ are the positions of the object's centroid along the x and y axes respectively:

$$p \text{ north-west } q \Leftrightarrow X(p) < X(q) \wedge Y(p) > Y(q)$$

$$p \text{ north-of } q \Leftrightarrow X(p) = X(q) \wedge Y(p) > Y(q)$$

$$p \text{ north-east } q \Leftrightarrow X(p) > X(q) \wedge Y(p) > Y(q)$$

²Value modified to due to error detected by [56]

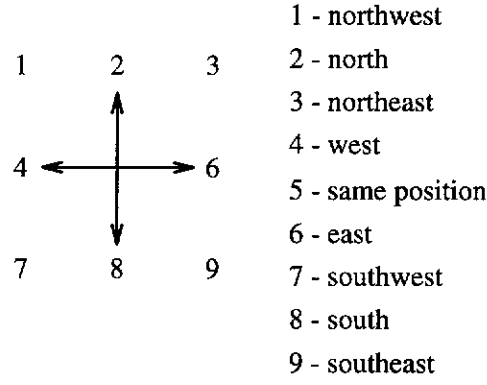


Figure 2.14: Compass directions used to describe 2-D relations

$$p \text{ west-of } q \Leftrightarrow X(p) < X(q) \wedge Y(p) = Y(q)$$

$$p \text{ same-pos } q \Leftrightarrow X(p) = X(q) \wedge Y(p) = Y(q)$$

$$p \text{ east-of } q \Leftrightarrow X(p) > X(q) \wedge Y(p) = Y(q)$$

$$p \text{ south-west } q \Leftrightarrow X(p) < X(q) \wedge Y(p) < Y(q)$$

$$p \text{ south-of } q \Leftrightarrow X(p) = X(q) \wedge Y(p) < Y(q)$$

$$p \text{ south-east } q \Leftrightarrow X(p) > X(q) \wedge Y(p) < Y(q)$$

By representing the object as four extreme points (equivalent to two intervals), the compass directions may be redefined. For example, the compass direction *north-of* may be redefined as follows:

$$p \text{ strong-north } q \Leftrightarrow Y(B(p)) > Y(U(q)) \text{ (Figure 2.15i)}$$

$$p \text{ weak-north } q \Leftrightarrow Y(U(p)) > Y(U(q)) \wedge Y(B(p)) \geq Y(B(q)) \\ \wedge Y(B(p)) < Y(U(q)) \text{ (Figure 2.15ii)}$$

$$p \text{ north-south } q \Leftrightarrow Y(U(p)) > Y(U(q)) \wedge Y(B(p)) < Y(B(q)) \text{ (Figure 2.15iii)}$$

where $U(\text{object})$ and $B(\text{object})$ are the upper and bottom extremes of the object.

These relations, using either 1-point or 4-points to define the object, may be augmented with topological relations. The relations are expressed as two values: directional and topological i.e. *p west-covers q*.

In a comprehensive work that integrates a point-based system, topology and relative frames of reference, Hernandez [36, 37] represents objects as points and augments this representation by an orientation system based upon subdividing

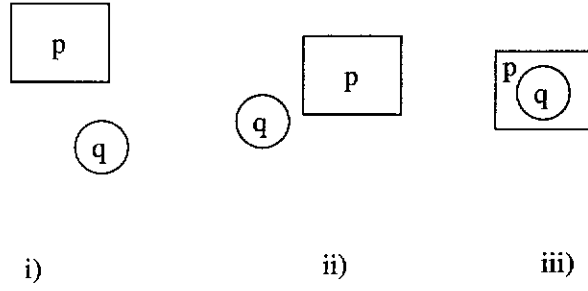


Figure 2.15: Relations based on four extreme points i) p strong-north q ii) p weak-north q and iii) p north-south q

space relative to a single point. Alternative choices of subdivision are depicted in Figure 2.16i.

A relative orientation node (*ron*) is used to indicate the relationship between two objects (see Figure 2.16ii). A composition table can be generated for propagating the relationships between *rons* (see Table 2.7).

The relations expressed in terms of intrinsic, extrinsic and deictic frames of reference are converted into a canonical representation by rotating the relations represented in the *ron* of the object. The topological and *ron* information about relations between objects may be used in concert to describe a relation in terms of both the topology and orientation. For example, if A *front* B and A *disjoint* B then the relation may be expressed as A [*front, disjoint*] B.

There are 64 (8×8) primitive relations, 2^{64} disjunctions and a composition table that may be derived from the composition tables for topology and orientation. To help visualise the disjunctions of relations, the topological relative orientation node (*tron*) may be used to represent the new composite topology and orientation relations (see Figure 2.17).

The structure of the *tron* is derived from the concepts introduced by analysing the results of deductions. For Interval Algebra, topology, orientation and the orientation/topology pair there is an observable property of the propagation. The result of the propagation of relations may be a disjunction of relations. Every relation in the disjunction is conceptual neighbour.

Definition 2.1 *Two relations between pairs of events are (conceptual) neighbours, if they can be directly transformed into one another by continuously de-*

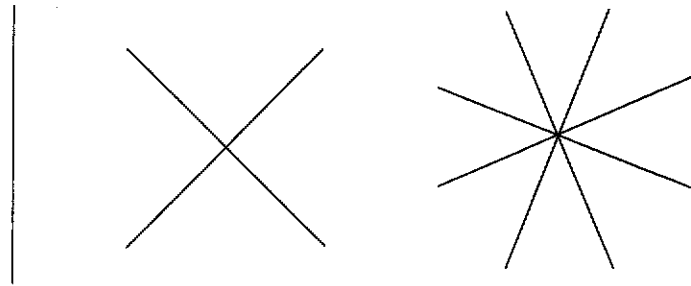
Table 2.7: The composition table for the orientation relations. The relation are back (b), left-back (lb), left (l), left-front (lf), front (f), right-front (rf), right (r) and right-back (rb).

A r l B	B r 2 C							
	b	lb	l	lf	f	rf	r	rb
b	b	{b, lb}	{b, lb, l}	{b, lb, l, lf}	?	{b, rb, r, rf}	{b, rb, r}	{b, rb}
lb	{lb, l}	lb	{lb, l}	{lb, l, lf}	{lb, l, lf, f}	?	{lb, b, rb, r}	{lb, b, rb}
l	{l, lb, b}	{l, lb}	l	{l, lf}	{l, lf, f}	{l, lf, f, rf}	?	{l, lb, b, rb}
lf	{lf, l, lb, b}	{lf, l, lb}	{lf, l}	lf	{lf, f}	{lf, f, rf}	{lf, f, rf, r}	?
f	?	{f, lf, l, lb}	{f, lf, l}	{f, lf}	f	{f, rf}	{f, rf, r}	{f, rf, r, rb}
rf	{rf, r, rb, b}	?	{rf, f, lf, l}	{rf, f, lf}	{rf, f}	rf	{rf, f}	{rf, f, rb}
r	{r, rb, b}	{r, rb, b, lb}	?	{r, rf, f, lf}	{r, rf, f}	{r, rf}	r	{r, rb}
rb	{rb, b}	{rb, b, lb}	{rb, b, lb, l}	?	{rb, r, rf, f}	{rb, r, rf}	{rb, r}	r

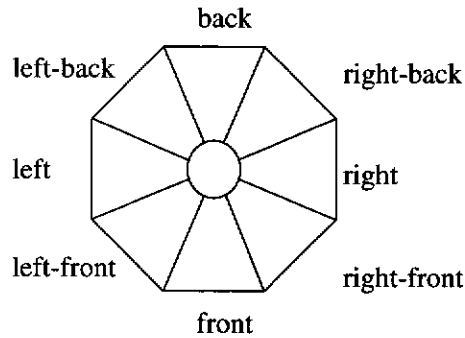
forming the events. [25, p. 204]

We shall not be using this propagation in our work, the details of the properties of conceptual neighbours may be found in [24, 25].

Mukerjee and Joe [55] present a system of representing and propagating relative frames of reference without translating relations to a common frame of reference. Directional information about object relations is determined by assigning each object a relative orientation. Two systems are used, one based upon points and the other upon intervals (see Figure 2.18).



i) Alternative choices for zoning



ii) Relative orientation node

Figure 2.16: The zoning choices in i) led a relative orientation node (ii).

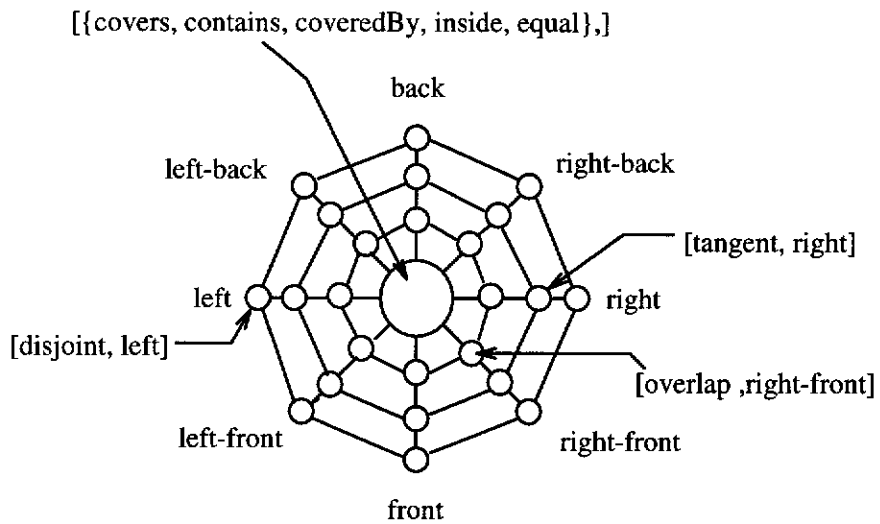
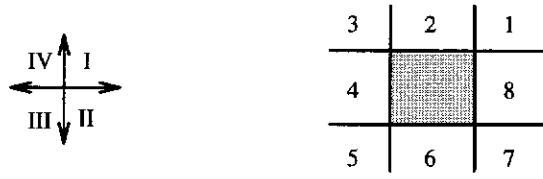


Figure 2.17: A structure that incorporates topological and orientation information

Relations between objects are propagated by the use of collision parallelograms to determine how the objects relate positionally to each other. For 2-D objects the lines used to divide the space (with respect to the object) are ex-



i) Point based division of space

ii) Interval based division of space

Figure 2.18: The point and interval-based spatial divisions.

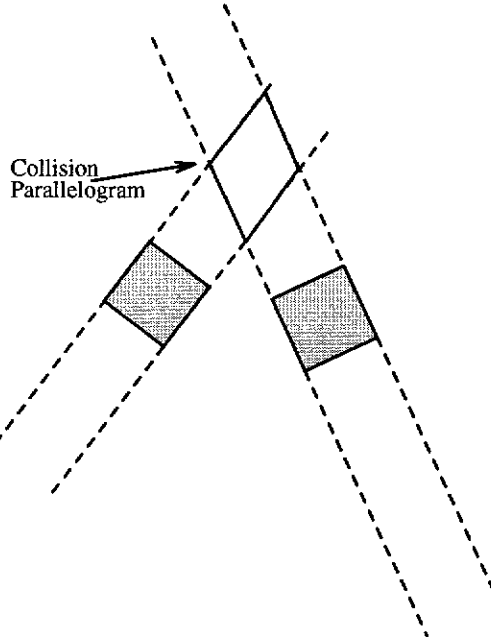


Figure 2.19: The collision parallelogram is the intersection of each objects division lines.

tended ad infinitum. If these lines between two different objects intersect then a relationship between the two objects may be deduced (see Figure 2.19). Propagation of these relations is inaccurate and can only be used in a limited way (see [26, 27, 78]).

A unifying theory for points and intervals expressed in terms of cyclic orders is given in [61]. Spatial extensions to traditional logics that will not be discussed include modal logics [72, 73] and fuzzy logic [16, 17].

2.2.4 Discussion

There are two key issues for the representation of qualitative spatial information:

- The use of global and relative frames of reference.
- The representation of point, interval and topological information in n-D space.

Systems have been developed for spatial reasoning that use global reference frames for interval systems, relative reference frames for interval systems, global reference frames for point systems augmented with topological information, relative reference frames for point systems augmented with topological information, and global references for intervals systems augmented with topological information.

2.3 Conclusion

There are many different methods for the representation of and reasoning with spatial and temporal information. We have examined a variety of methods for representing and reasoning with time and with space. We have concentrated on qualitative methods as they provide a means of approximating information that simplifies both the representation and the reasoning process.

Qualitative representations of temporal information have focused upon the use of point and interval primitives for representation. An algebra for both interval and point-based representations may be defined as a set of primitive relations, the set of disjunctions of these primitive relations and the rules for propagating information about these relations. Three properties that may be derived from a closure are of interest when representing these algebras. The *satisfiability problem*, *minimal labeling problem* and the *all consistent solutions problem* provide representations such that the existence of a consistent labeling, the minimum relations needed to represent all consistent labelings and a list of all the consistent labelings may be determined respectively.

Closure construction is an NP-Complete problem for Interval Algebra but for restricted subsets of Interval Algebra there are polynomial solutions to these problems. Polynomial solutions also exist for Point Algebra. Interval Algebra cannot be accurately represented using Point Algebra (i.e. in terms of end points

of intervals). For a restricted subset of Interval Algebra, Point Algebra may be used to provide a polynomial solution to the closure problem.

The intractability of Interval Algebra has led to the development of non-closure based systems for representing qualitative temporal information. These systems have focused upon providing efficient methods of representing interval information. They represent interval relations in terms of their end points, determining the consistency of the representation and provide systems that are able to efficiently access point relations. As expected, non-closure systems cannot be used to determine the consistency of Interval Algebra in polynomial time.

The problem of representing and reasoning with qualitative spatial information can be considered with respect to two key aspects: 1) The use of global or relative frames of reference and 2) the representation of spatial information in terms of topology, intervals and/or points.

Spatial information may be represented and reasoned with in terms of a fixed global frame of reference (i.e. cartesian axes). Alternatively, relations between objects may be expressed relative to a characteristic of the object (front) or an observer's point of view. These relative frames of reference may be transferred to a common (global) frame of reference or reasoned with as is.

Topological information about spatial relations involves determining how two objects intersect in space. Position and other shape-based information may be represented in terms of intervals and points. The representation of multi-dimensional objects using either points or intervals simplifies the shape of the objects and thus the positional relationships between the objects. Combinations of topological, point and interval representations have been used to develop a variety of spatial reasoning systems for a variety of problems.

It is our aim to develop a multi-dimensional spatial reasoning system using $n \times 1$ -D intervals. We shall take advantage of the multi-dimensional spatial domain to provide a method of reasoning (at multiple levels) about the relationships between objects. A querying system shall be developed from this representation to provide a means of performing complex queries. We shall also provide a non-closure based system for representing and accessing interval information.

By taking advantage of the characteristics of space, our system goes beyond the straightforward extrapolation of temporal intervals to the spatial domain proposed by Guesgen [34, 35]. The ability to perform queries at multiple levels of granularity enables sophisticated querying of the spatial information. Our non-closure based representation of intervals allows for efficient representation and querying of the interval information that has varying states of certainty. Considering the only other approaches are either incomplete or NP-Complete, this method offers a significant alternative for representing and reasoning with incomplete interval information.

Chapter 3

Multi-dimensional Spatial Reasoning

‘Space,’ it says ‘is big. Really big. You just won’t believe how vastly hugely mindbogglingly big it is. I mean you may think it’s a long way down the road to the chemist, but that’s just peanuts to space. Listen . . .’

The Hitch Hikers Guide to the Galaxy

Interval-based representations are more expressive than point-based ones. We begin our work by describing a multi-dimensional spatial reasoning system based on intervals. Our representation allows for the grouping of relations into a hierarchic abstraction, which is used as a foundation of the spatial reasoning system. Like Guesgen [35], we shall treat an n -D system as being equivalent to n 1-D systems. There are thirteen interval relations in 1-D and thus in n -D there are 13^n relations (i.e. 169 for 2-D and 2197 for 3-D).

The layout of this chapter is as follows. The basic definitions are introduced in Section 3.1. The new spatial relation, NS-Relation, is defined in Section 3.2. This relation is used to develop a multi-dimensional spatial reasoning system described in Section 3.3. The querying system is discussed in Section 3.4 and conclusions follow in Section 3.5.

3.1 A-Space

In this section we define the basic concepts necessary to introduce our work.

Definition 3.1 *A topological space is a pair (\mathbf{A}, τ) consisting of a set \mathbf{A} and τ a collection of subsets of \mathbf{A} , called open sets, satisfying the following axioms: i) the union of open sets is an open set, ii) the finite intersection of open sets is an open set, iii) \mathbf{A} and the empty set \emptyset are open sets.*

[64, p3]

Definition 3.2 *The dual notion of a closed set is related to the notion of open set: it is a set whose complement is open. Thus, if $U \in \tau$ then \mathbf{A}/U is closed, and, conversely, if F is closed then \mathbf{A}/F is open.*

As a result of the duality of set-theoretic operations, the collection of all closed sets of a space \mathbf{A} satisfy the following properties:

- i) The sets \mathbf{A}, \emptyset are closed.*
- ii) The intersection of any collection of closed sets is closed.*
- iii) The union of any finite number of closed sets is closed.*

[5, p41]

Definition 3.3 *A point \mathbf{p} is a limit point of a set \mathbf{A} if every open set containing \mathbf{p} contains at least one point of \mathbf{A} distinct from \mathbf{p} . A limit point of a set \mathbf{A} is denoted as $\text{limit}(\mathbf{A})$.*

[64, p5]

Definition 3.4 *The closure of a set \mathbf{A} is the set \mathbf{A} together with its limit points, denoted by $\overline{\mathbf{A}}$ (or \mathbf{A}^-).*

[64, p6]

Using these background definitions, a single dimensional dense space can be created and convex intervals in this space can be defined without the use of order.

Definition 3.5 *An A-Space is a topological space (\mathbf{A}, τ) such that there is a one to one mapping: $\mathbf{A} \rightarrow \mathbb{R}$.*

Thus A-Space has the properties of linearity and density. An object in A-Space is represented as an interval. An interval may be convex or non-convex.

Definition 3.6 An interval X is any set in A-Space, and is denoted as **interval**(X).

An interval X in A-Space is convex if

$$\forall a, b \subset X, \neg (\bar{a} \cap b = \emptyset \wedge a \cap \bar{b} = \emptyset)$$

Otherwise an interval is non-convex.

In the remainder of this thesis we will be dealing exclusively with convex intervals.

An A-Space may be represented as an interval that has limit points and is the interval in which all the other intervals exist.

Definition 3.7 In an ordered A-Space, the A-Space and the intervals within that space are assigned an ordering.

In any given ordered interval, the limit points are assigned names with respect to that order. One limit point is called the *begin point* (denoted $b(\text{interval})$) and the other is called the *end point* (denoted $e(\text{interval})$). If an A-Space, \mathbf{A} , is given an ordering such that one limit point precedes (the precedes relation is denoted as ξ ; see Appendix A for details) the other limit point, then the first point is the begin point and the second is the end point (represented as $b(\mathbf{A}) \xi e(\mathbf{A})$).

The A-Space can be assigned an ordering, which can be transferred to all of the intervals that inhabit the A-Space. The process of transferring the ordering from the A-Space to the intervals in that A-Space is described in Appendix B.

3.2 NS-Relation

When defining an ordered A-Space there are two possible ways of selecting the begin point from the alternative limit points of an A-Space. Each choice indicates an alternative ordering of the points in the environment. The ordering of the points, as designated by the choice of the begin point, is called the **flow**.

The flow provides a consistent and structured framework for representing and analysing the relationships between objects. Alternative flows define different

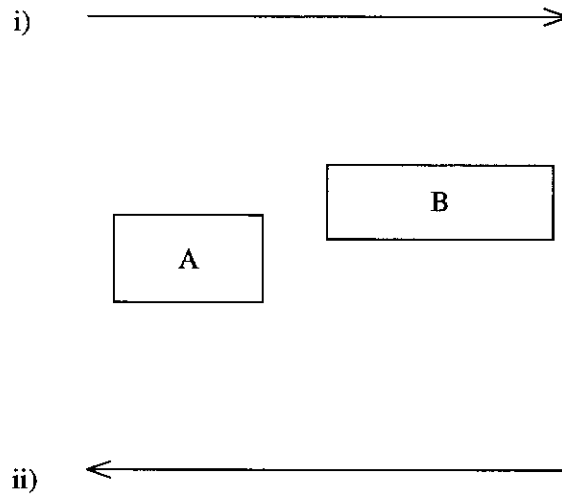


Figure 3.1: Flows i) and ii) represent different orderings along the same axis.

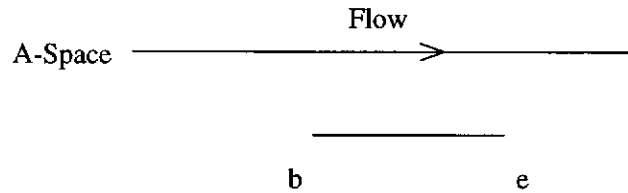


Figure 3.2: The ordering of the limit points of the interval as designated by the flow in A-Space.

orders. These orderings do not affect the actual physical relationship between the objects. An example of alternative flow descriptions is given in Figure 3.1 where in Figure 3.1i, A is *before* B and in Figure 3.1ii, B is *before* A.

The flow of an A-Space allows an ordering of all the intervals to be defined with respect to that A-Space. Given an arbitrary interval Y, the ordering of the points of the interval is designated with respect to the flow of the A-Space. The first limit point of the interval is the *begin point* of the interval with respect to the flow of A-Space (see Figure 3.2).

Some characteristics of the relationship between objects remain consistent irrespective of the flow used, and other characteristics change with respect to the flow used. In Figure 3.1 objects A and B are *disjoint*, and depending on the flow, A precedes B or B precedes A. The *invariant* aspect of the relationship is the relation *disjoint* (non-intersecting). The *variant* aspect of the relationship, dependent on the flow used, indicates the ordering of the objects. A spatial

relationship between any two objects can be described in terms of its variant and invariant components and used as a basis of abstraction for relationships between objects in multi-dimensional space. The invariant element of a spatial relation is called the *A-Relation* (see Section 3.2.1). The variant element of the relationship is called the *Orientation* (See Section 3.2.2).

Definition 3.8 *A spatial relation that is described in terms of its variant and invariant elements, with respect to the flow, is called an NS-Relation. In 1-D, an NS-Relation has the following format:*

$$(A\text{-Relation}, \text{Orientation})$$

where A-Relation describes the invariant part of the relation and Orientation describes the variant part of the relation.

3.2.1 A-Relations

There are six different types of invariant relations into which all of the thirteen different interval relations [3] can be classified.

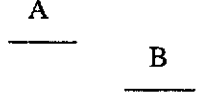
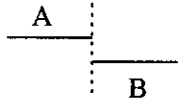
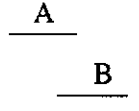
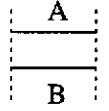
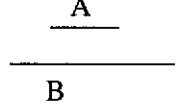
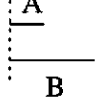
Definition 3.9 *An A-Relation is a relation that is used to describe the flow invariant component of the relationship between two intervals. There are six A-Relations, {disjoint, overlaps, meets, during, equal, ends}.*

Table 3.1 illustrates the six different types of A-Relation that may be used to describe the relationship between objects. The set of conditions necessary for each A-Relation (Column 2) and a diagrammatic representation of each relation (Column 3) are shown. For example, the relation *ends* is described as one interval being a strict subset of the other with the further condition that exactly one limit point of each interval is equal to a limit point of the other interval. An example of *ends* is illustrated in column 3.

3.2.2 Orientation

The flow imposed on an ordered A-Space allows for an ordering of the intervals within the space.

Table 3.1: A-Relations.

<i>A-Relation name</i>	<i>Definition</i>	<i>Pictorial example</i>
disjoint	$A \text{ disjoint } B \text{ iff } (A \cap B = \emptyset)$	
meets	$A \text{ meets } B \text{ iff } (A \cap B \neq \emptyset \wedge (\exists x \in \text{limit}(A), \exists y \in \text{limit}(B) \Rightarrow (x = y) \wedge ((A - x) \cap (B - y) = \emptyset)))$	
overlaps	$A \text{ overlaps } B \text{ iff } (A \cap B \neq \emptyset \wedge (A - (A \cap B)) \neq \emptyset \wedge (B - (A \cap B)) \neq \emptyset)$	
equal	$A \text{ equal } B \text{ iff } A = B$	
during	$A \text{ during } B \text{ iff } ((A \subset B \vee B \subset A) \wedge (\forall x \in \text{limit}(A) \wedge \forall y \in \text{limit}(B) \Rightarrow x \neq y))$	
ends	$A \text{ ends } B \text{ iff } ((A \subset B \vee B \subset A) \wedge (\exists x \in \text{limit}(A), \exists y \in \text{limit}(B) \Rightarrow x = y))$	

Definition 3.10 An *Orientation* is a relation used to describe the flow variant component of a relation between two intervals. The variant component of the interval relation is described in terms of the relationship between the end points of the two intervals.

The format of the orientation is defined in terms of the ordering of points:

interval operator interval

where **interval** indicates the intervals involved, and **operator** determines the two limit points used and the ordering of those limit points. Three operators $<$, $>$, $><$ are used to distinguish between different orientations.

1. $<$ (given as $A<B$) indicates that the begin point of the first interval precedes the begin point of the second interval, which is equivalent to $b(A) \xi b(B)$ with respect to the ordered A-Space.
2. $>$ (given as $A>B$) indicates that the end point of the first interval follows the end point of the second, which is equivalent to $e(B) \xi e(A)$, again with respect to the ordered A-Space.
3. The composite operator $><$ (given as $A><B$) indicates that both $>$ and $<$ are true, i.e. $A<B$ and $A>B$.

The operator $><$ is flow invariant and is only used to differentiate between the two types of during relations.

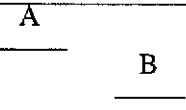
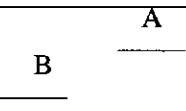
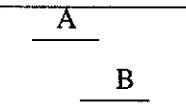
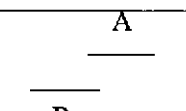
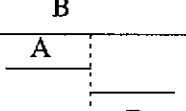
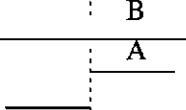
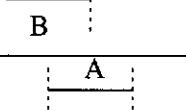
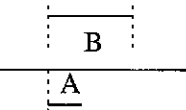
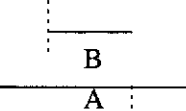
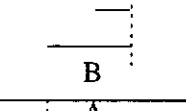
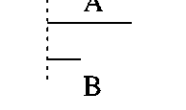
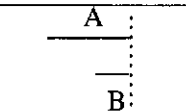
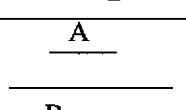
3.2.3 New Spatial Relations

The complete set of thirteen interval relations can be constructed through combinations of pairs of A-Relations and orientations. The orientation allows for the discrimination between different members of a group of relations that are described by one A-Relation.

Table 3.2 enumerates all instances of the six A-Relations. With the A-Relations *disjoint*, *overlaps*, *meets* and *during* there are two ways of ordering an interval **A** with respect to a second interval **B**. The *ends* relation has four instances, one for each of the different ways of ordering that A-Relation. There is only one entry for the *equals* relationship; *equals* is the only relation that is flow-independent.

Each instance of the four interval relations *disjoint*, *overlaps*, *during* and *meets* can be described using two alternative orientations. For consistency, a single orientation (Column 4) will be used. Column 5 shows the equivalent Allen operators.

Table 3.2: A-Relation instances and their Orientation discriminators.

<i>A-Relation</i>	<i>Instance</i>	<i>Valid Orient.</i>	<i>Standard Orient.</i>	<i>Allen's Operators</i>
disjoint		$A < B, B > A$	$A < B$	A before B B inv(before) A
		$B < A, A > B$	$B < A$	B before A A inv(before) B
overlaps		$A < B, B > A$	$A < B$	A overlaps B B inv(overlaps) A
		$B < A, A > B$	$B < A$	B overlaps A A inv(overlaps) B
meets		$A < B, B > A$	$A < B$	A meets B B inv(meets) A
		$B < A, A > B$	$B < A$	B meets A A inv(meets) B
equal				A equal B B equal A
ends		$B > A$	$B > A$	A starts B B inv(starts) A
		$B < A$	$B < A$	A finishes B B inv(finishes) A
		$A > B$	$A > B$	B starts A A inv(starts) B
		$A < B$	$A < B$	B finishes A A inv(finishes) B
during		$B > < A$	$B > < A$	B during A A inv(during) B
		$A > < B$	$A > < B$	A during B B inv(during) A

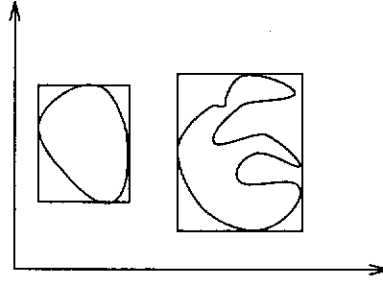


Figure 3.3: Each 2-D object is represented in terms of its bounding box.

Table 3.1 shows that when dealing with space, the six primitive A-Relations can be used to describe relationships between intervals without any notion of order. The subsequent imposition of order (via orientation) adds to the A-Relation such that a unique representation for each of the thirteen relations between two intervals is possible.

3.3 n-D Spatial Reasoning

Relations in multi-dimensional space (n-D) can be represented using n 1-D NS-Relations. An approximate representation of the object is made in terms of the extreme points of the object along each axis (See Figure 3.3 for an example). A bounding box is one method that can be used to simplify the shape of a given object. Other types of approximations such as minimum bounding circle may be used [59]. We have chosen bounding boxes as they can be represented as independent intervals along each dimension.

This approximation of an object's shape allows reasoning about the object's relationships with respect to other objects. Transforming relations between systems that use different axes and/or origins will not be developed in this thesis. A global origin with fixed perpendicular axes is chosen and the bounding boxes are determined relative to this coordinate frame.

Definition 3.11 *An n-D NS-Relation is defined to be a set of n 1-D NS-Relations, one for each dimension as:*

$$\{ NS-Relation_1 - NS-Relation_2 \dots - NS-Relation_n \}$$

or

$$\{ (A-Rel_1, Orient_1)_{axis_1}, (A-Rel_2, Orient_2)_{axis_2} \dots (A-Rel_n, Orient_n)_{axis_n} \}$$

The set of 1-D NS-Relations is listed in an order determined by the axes. Thus an n-D NS-Relation can usually be described as the following ordered set:

$$\{ (A-Rel_1, Orient_1), (A-Rel_2, Orient_2) \dots (A-Rel_n, Orient_n) \}$$

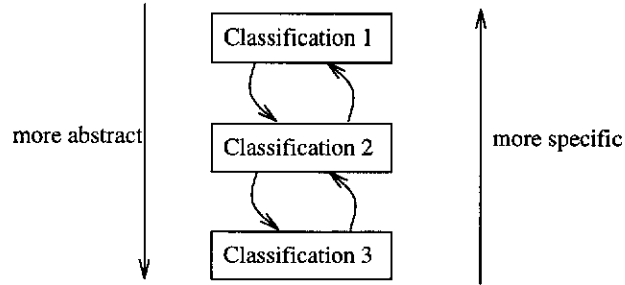
The ordered set of relations may also be represented as follows:

$$\langle A-Rel_1-A-Rel_2-\dots-A-Rel_n \rangle \langle Orient_1-Orient_2-\dots-Orient_n \rangle$$

Both notations will be used in this thesis.

3.3.1 n-D Classifications

Based on the variant and invariant components, we can build a hierarchy to allow for the querying of object relations at different levels of abstraction. There are three layers in this hierarchy and they are described as Classifications 1, 2 and 3.



Definition 3.12 *A relation that is described in terms of both the variant and invariant aspects of the relationship is designated as belonging to Classification 1 (C1). This classification allows for a unique description to be given to each of the 13 base interval relations. In n-D, the format of a Classification 1 relation is:*

$$\langle A-Rel_1-A-Rel_2-\dots-A-Rel_n \rangle \langle Orient_1-Orient_2 \dots -Orient_n \rangle$$

Definition 3.13 *A Classification 2 (C2) relation is a relation that has been described solely in terms of the invariant component of the interval relation.*

In n -D, the format of a Classification 2 relation is:

$$\langle A\text{-Rel}_1\text{-}A\text{-Rel}_2\text{-}\dots\text{-}A\text{-Rel}_n \rangle$$

This classification groups the 13 base relations into 6 different relations according to the invariant component of the base relation. For both Classification 1 and 2 the order of the relations along each dimension is maintained.

Definition 3.14 A Classification 3 (C3) relation is a relation that only exists in n -D when $n > 1$ and describes the interval relation in terms of an unordered set of A-Relations.

The format for a Classification 3 relation is:

$$\{A\text{-Rel}_i, A\text{-Rel}_j, \dots A\text{-Rel}_l\} \text{ where } i, j, l \in \{1 \dots n\}$$

Listing relations without indicating the axis to which they apply means that we do not know or care how the objects are oriented with respect to the axes, but we do know the qualitative, coarse relations between the objects.

The relationship between Classifications 1 & 2 and 2 & 3 as well as the transition functions can be formally defined, as shown in the following subsections.

3.3.1.1 Transforming an n -D relation from Classification 1 to Classification 2

The process of transferring from Classification 1 to Classification 2 involves identifying the A-Relations used to describe the C1 Relation. A relation described solely in terms of an A-Relation (the invariant component) is a C2 relation.

$$\begin{aligned} f_1: NS\text{-}Relation^n &\rightarrow A\text{-}Relation^n \\ \langle NS\text{-}Rel_1, NS\text{-}Rel_2 \dots NS\text{-}Rel_n \rangle &\rightarrow \langle A\text{-}Rel_1, A\text{-}Rel_2 \dots A\text{-}Rel_n \rangle \\ &\text{or} \\ \langle (A\text{-}Rel_1, Orient_1), \dots (A\text{-}Rel_n, Orient_n) \rangle &\rightarrow \langle A\text{-}Rel_1, \dots A\text{-}Rel_n \rangle \end{aligned}$$

The transformation from Classification 1 to Classification 2 involves projecting the A-Relations out of the NS-Relations. This results in a loss of information and thus the transformation is many-to-1.

3.3.1.2 Transforming an n-D relation from Classification 2 to Classification 1

The inverse process of transforming a relation from Classification 2 to Classification 1 requires that a description of the invariant component be supplied. A single C2 relation may match many C1 relations. However, the given C2 relation and the C1 relations share the same A-Relation.

$$\begin{aligned}
 f_1^{-1}: A-Relation^n &\rightarrow NS-Relation^{mn} \\
 \langle A-Rel_1, A-Rel_2 \dots A-Rel_n \rangle &\rightarrow \{ \langle (A-Rel_1 \text{ Orient}_{11}) \dots (A-Rel_n \text{ Orient}_{1n}) \rangle \\
 &\quad \langle (A-Rel_1 \text{ Orient}_{21}) \dots (A-Rel_n \text{ Orient}_{2n}) \rangle \\
 &\quad \cdot \\
 &\quad \cdot \\
 &\quad \langle (A-Rel_1 \text{ Orient}_{k1}) \dots (A-Rel_n \text{ Orient}_{kn}) \rangle \}
 \end{aligned}$$

The f_1^{-1} transformation involves enumerating all NS-Relation instances of the given A-Relations.

3.3.1.3 Transforming an n-D relation from Classification 2 to Classification 3

The transformation from Classification 2 to Classification 3 removes the ordering of the A-Relations. Thus there is no restriction as to which axis should be associated with a particular A-Relation.

To illustrate this we define the set $A-Rel = \{\text{disjoint, overlaps, during, meets, equals, ends}\}$ and the set $\Gamma_n^1(A) \equiv A$ contains exactly n elements [40, p23]. Then the transformation of a C2 relation to a C3 relation is:

$$\begin{aligned}
 f_2: A-Relation^n &\rightarrow \Gamma_n^1(A) = \{ a_i \mid a_i \in A-Rel, 1 \leq i \leq n \}. \\
 A-Rel_1 - A-Rel_2 \dots A-Rel_n &\rightarrow \{ A-Rel_1, A-Rel_2 \dots A-Rel_n \}
 \end{aligned}$$

3.3.1.4 Transformation of an n-D relation from Classification 3 to Classification 2

The inverse transformation f_2^{-1} requires that all the possible combinations of the n A-Relations for n dimensions be enumerated.

$$f_2^{-1}: \Gamma_n^1(A) = \{ a_i \mid a_i \in ARel, 1 \leq i \leq n \} \rightarrow A\text{-Relation}^{dn}$$

$$\begin{aligned} \{ A\text{-Rel}_1, A\text{-Rel}_2 \dots A\text{-Rel}_n \} &\rightarrow \{ A\text{-Rel}_1 - A\text{-Rel}_2 \dots A\text{-Rel}_n \\ &A\text{-Rel}_2 - A\text{-Rel}_1 \dots A\text{-Rel}_n \\ &\vdots \\ &A\text{-Rel}_a - A\text{-Rel}_b \dots A\text{-Rel}_d \} \end{aligned}$$

The f_2^{-1} transformation involves enumerating all unique permutations of the A-Relations.

The transformation between Classification 1 and Classification 3 can be constructed by composing the other transformations as follows:

$$f_3: NS\text{-Relation}^n \rightarrow A\text{-Relation}^n \rightarrow \Gamma_n^1(A) = \{ a_i \mid a_i \in ARel, 1 \leq i \leq n \}.$$

$$f_3^{-1}: \Gamma_n^1(A) = \{ a_i \mid a_i \in ARel, 1 \leq i \leq n \} \rightarrow A\text{-Relation}^{dn} \rightarrow NS\text{-Relation}^{mdn}$$

3.3.2 Examples

We examine two examples of an n-D system: Section 3.3.2.1 examines 2-D NS-Relations whilst Section 3.3.2.2 examines 3-D NS-Relations.



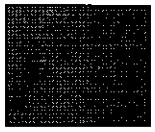
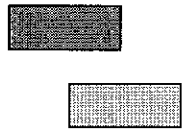
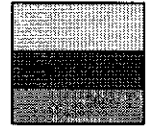
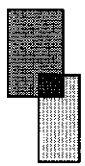



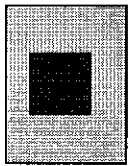
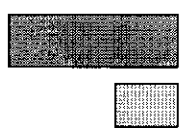
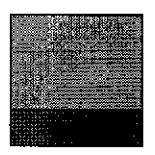

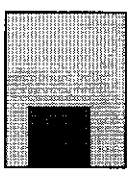
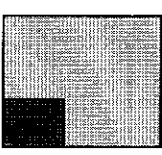
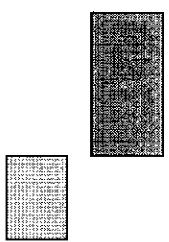
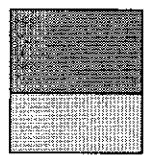
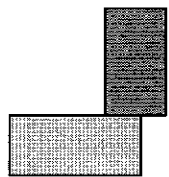
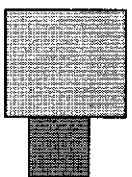


3.3.2.1 2-D NS-Relations

This section describes a two dimensional system using the n-D representation outlined in the previous section. A 2-D NS-Relation can be expressed in the following format:

$$A\text{-Relation}_1 - A\text{-Relation}_2 \text{ orient}_1\text{-orient}_2$$

There are 13^2 (169) possible base relations between two 2-D objects. These 169 relations can be described using the C1 2-D NS-Relations. A more abstract representation, where there are 36 (6^2) different C2 NS-Relations, describes the relationship in terms of the flow-independent characteristics of the relationship. The 21 different 2-D NS-Relations at Classification 3 describe a relationship independent of flow and axis.

Table 3.3: Pictorial Representation of the C3 A-Relations.

	disjoint	equal	overlaps	during	ends	meets
disjoint						
equal						
overlaps						
during						
ends						
meets						

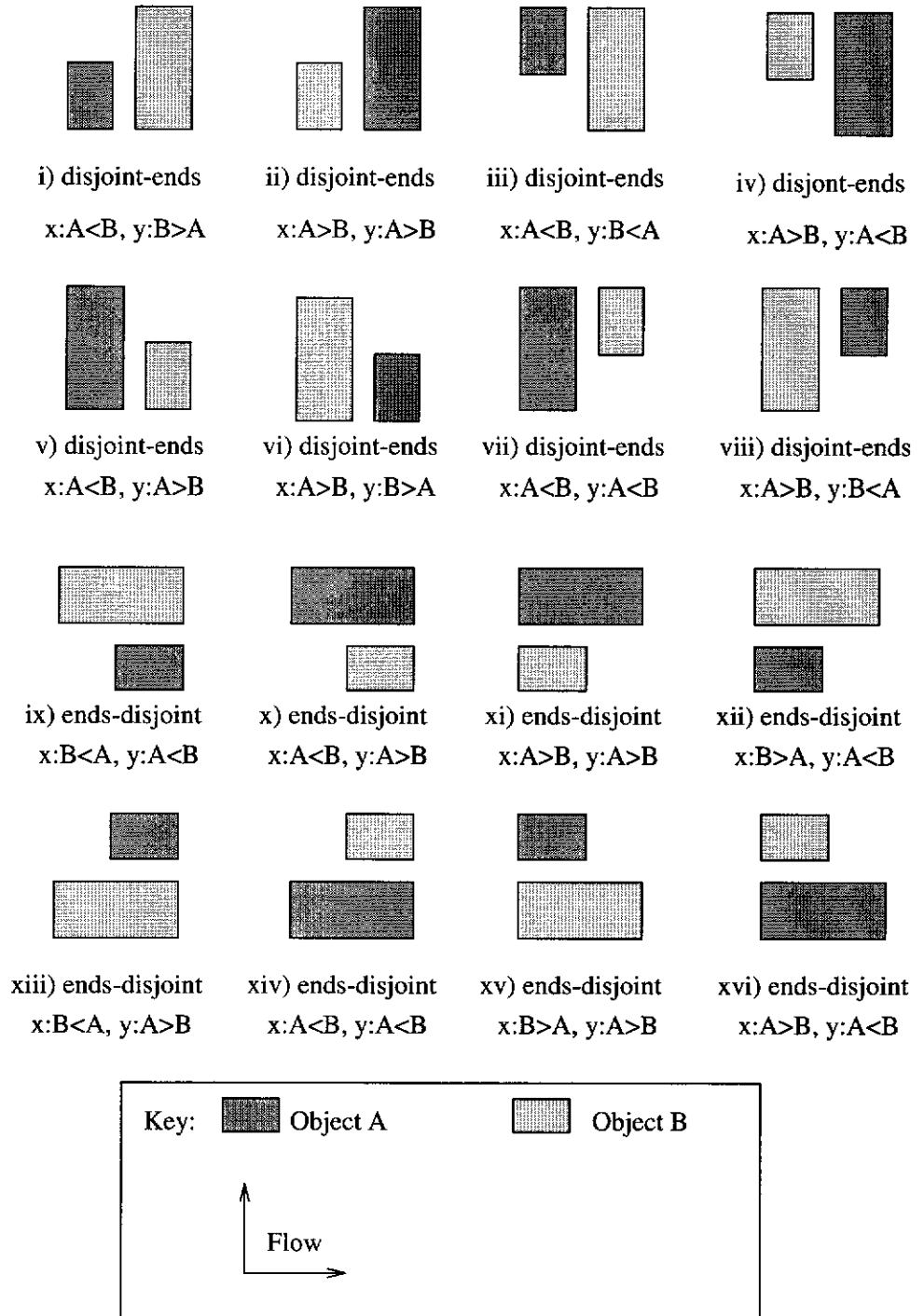


Figure 3.4: Examples of the {disjoint, ends} relation.

A pictorial representation of the C3 2-D NS-Relations is given in Table 3.3.

All the C1 2-D NS-Relations for the C3 2-D NS-Relation (*ends, disjoint*) are given in Figure 3.4. Note that Figures 3.4(i-viii) list the C1 2-D NS-Relations for the C2 2-D NS-Relation *disjoint-ends*, and Figures 3.4(ix-xvi) list the C1 2-D NS-Relations for the C2 NS-Relation *ends-disjoint*.

3.3.2.2 3-D NS-Relations

A 3-D NS-Relation can be expressed in the following format:

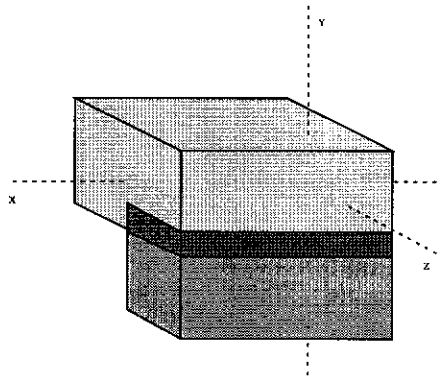
$$A-Relation_1-A-Relation_2-A-Relation_3 \text{ Orient}_1-Orient_2-Orient_3$$

There are 2196 (13^3) base relations between any two 3-D objects. The 2196 base relations are described by the C1 3-D NS-Relations. 216 C2 3-D NS-Relations are used to describe the relationship independent of the direction of the flow. There are 56 C3 3-D NS-Relations.

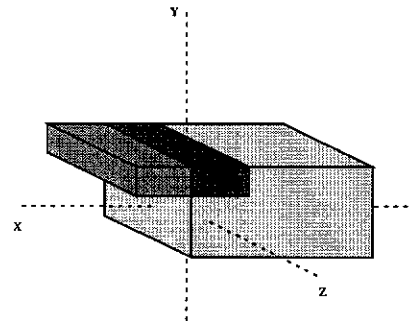
There are six C2 2-D NS-Relations that can be made up of the relation that consists of *equal*, *overlaps* and *ends*. Table 3.4 shows the six C2 3-D NS-Relations and the possible orientations (36) that could be used to create the 48 C1 3-D NS-Relations. Five of the six C2 3-D NS-Relations are illustrated in Figure 3.5 (a C1 example for each is chosen to illustrate the type).

Table 3.4: Permutations of the equal, overlaps and ends relations in 3-D (equals is flow independent, hence introduces no constraint along the respective axis).

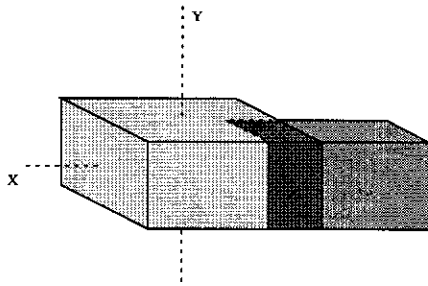
Relation	Possible Orientations
equal-overlaps-ends	Y: $A < B, B < A$ Z: $A < B, A > B, B > A, B > A$
equal-ends-overlaps	Y: $A < B, A > B, B < A, B > A$ Z: $A < B, B < A$
overlaps-equals-ends	X: $A < B, B < A$ Z: $A < B, A > B, B < A, B > A$
overlaps-ends-equal	X: $A < B, B < A$ Y: $A < B, A > B, B < A, B > A$
ends-overlaps-equal	X: $A < B, A > B, B < A, B > A$ Y: $A < B, B < A$
ends-equal-overlaps	X: $A < B, A > B, B < A, B > A$ Z: $A < B, B < A$



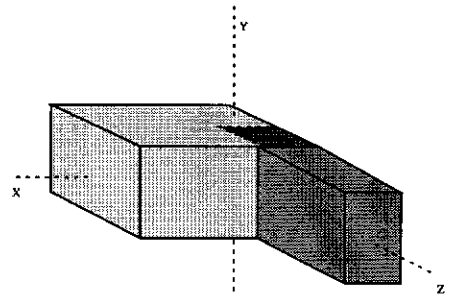
equal-overlaps-ends



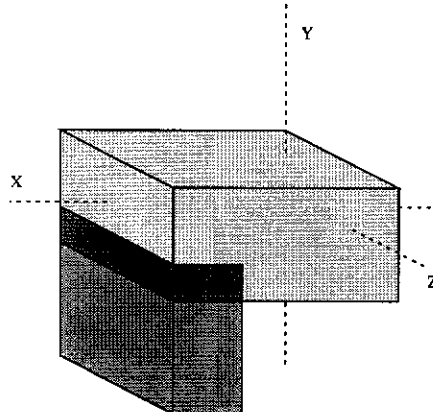
overlaps-ends-equal



overlaps-equal-ends



ends-equal-overlaps



ends-overlaps-equal

Figure 3.5: Classification 1 examples of five different C2 3-D Relations.

The C1 3-D NS-Relations for the C2 3-D NS-Relation equal-ends-overlaps are given in Figure 3.6.

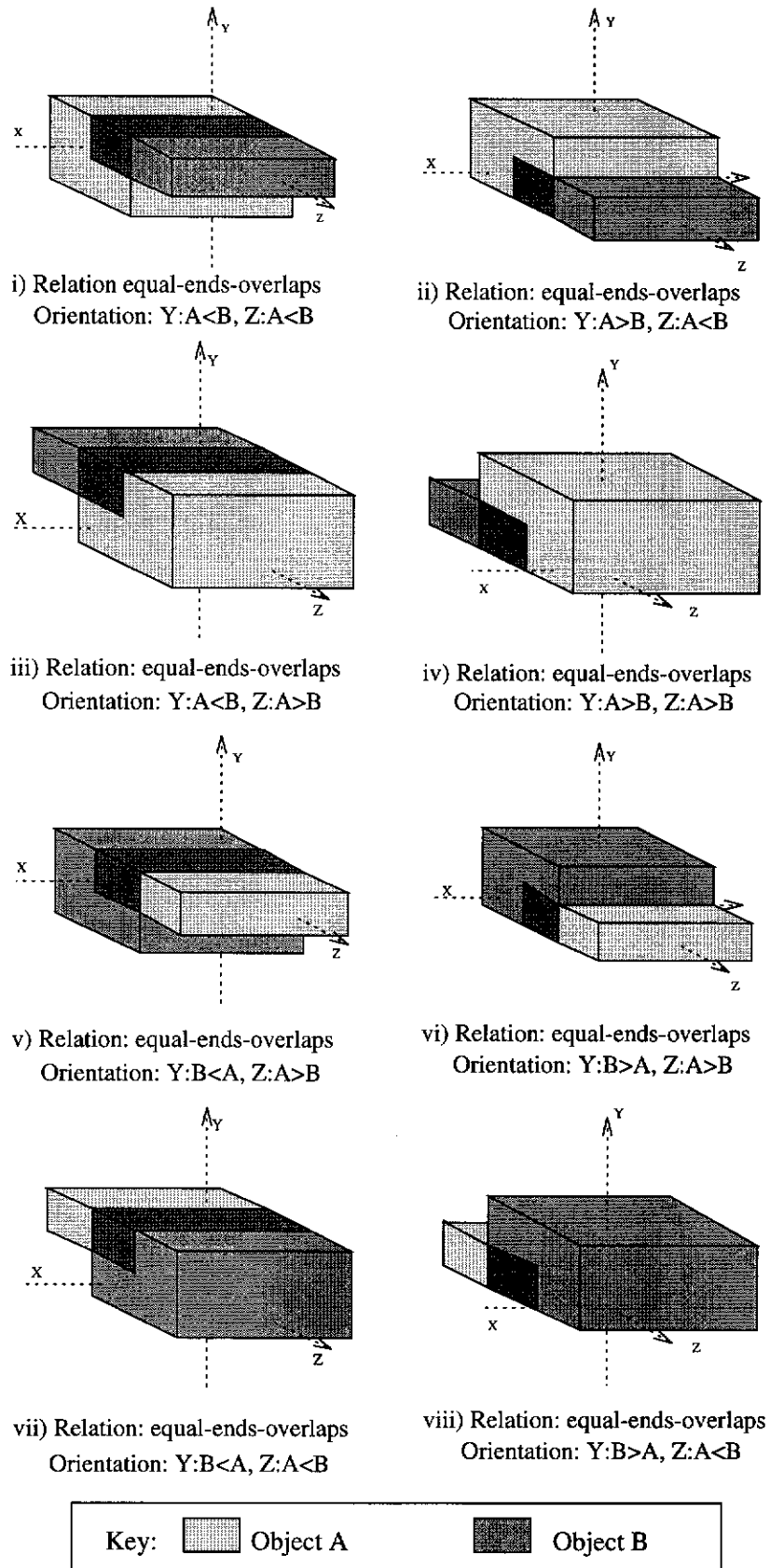


Figure 3.6: The eight C1 3-D NS-Relations for the C2 3-D NS-Relation equal-ends-overlaps.

3.4 Queries

In this chapter, the focus has been upon the process of representing objects in space in terms of intervals along multiple axes. The relationships between objects have been expressed in terms of the relationship between the intervals representing these objects along each dimensional axis. We have also provided a method of classifying these relations according to similarities between relations when both flow independence and axis independence are considered.

From this representation of objects in space, a query language can be specified to provide the means of accessing information about relations between objects in space. The multiple levels of classification allow for a powerful hierarchy of queries to be formed about object relations.

The query language provides the only access to the spatial information. It thus represents both the power and the limitations of the type of representation of spatial information proposed in this thesis.

In this section we demonstrate a querying system based on 1-D, 2-D and 3-D representations outlined in the previous sections. We use logical operators and variables to enable sophisticated queries about object relations to be constructed.

In this section, querying uses the following representation of NS-Relations:

$$A \text{ disjoint } 1 < 2 \text{ B}$$

to describe the NS-Relation instead of

$$A \text{ disjoint B } A < B$$

that we have been using so far in this chapter.

A query may contain variables, which are used to either return a value of interest (i.e. satisfying the query) or are used as part of the constraint(s) represented by the query and their specific values are of no interest to the querist. Variables may be used to represent objects and relations.

Three types of variables are used to represent unknown values within a query:

1. The value of an anonymous variable is of no concern; it is used to determine if there is a value (any value) that will match this aspect of the query. An anonymous variable is represented as $??a$. Anonymous variables may

be used as the orientation for queries associated with the relation equals (equals has no orientation).

Example 3.1 *Anonymous variable*

(??a *disjoint* 1<2 B)

Is there an interval that satisfies the relation *disjoint* 1<2 with respect to interval B? The query only determines the existence of such an object and does not return the actual value.

◇

2. A numbered variable places the restriction that any match to this variable must match other instances of the same numbered variable used in this query. This allows for restrictions to be placed on the query matching the variable. Note that the actual value of the result is of no concern, although the existence of such a value is relevant. A numbered variable is represented as ??n where *n* is a unique number representing this numbered variable.

Example 3.2 *Numbered variable*

(and ??1 *disjoint* 1<2 B ??1 *overlaps* 1<2 C)

Is there an interval that satisfies both given relations to the given intervals? The example uses an operator “and” which is defined later and is given in prefix notation.

◇

3. Named variables are of interest to the querist. The matches to these variables are returned. A named variable is represented as ?name where *name* is the unique name used to identify the variable. To avoid confusion there is a restriction on the variable names: the name itself cannot begin with “?”. .

Example 3.3 *Named variable*

(?A *disjoint* 1<2 B)

What are the names of the intervals that satisfy the given relation to interval B?

◇

The discussion of queries is divided into two main sections: atomic queries and composite queries. Both the atomic and composite queries can be further subdivided into 1-D, 2-D and 3-D queries as discussed in the following sub-sections.

3.4.1 Atomic queries

3.4.1.1 1-D Queries

The general form of a 1-D query using NS-Relations to express the relationship between two intervals is as follows:

interval1 A-Relation Orientation interval2

There are three different types of elements that make up a query: the intervals, the A-Relation and the orientation.

1. Intervals are used to represent the objects or elements of the environment for which the relationship will be described using the NS-Relation. The interval name may be known or unknown as part of the search. In an environment where the interval is described by name, the following are possible values in a query:

A an object named A

??a an anonymous interval

??1 an interval that is restricted to a specific instance

?G a named variable whose value is returned to the user

2. An A-Relation is used to determine the pattern of the relation between the intervals. An A-Relation may be one of the following six types: *disjoint*, *meets*, *overlaps*, *during*, *equal* and *ends*.

Example 3.4 *Queries with different A-Relations*

(interval1 *disjoint* orient interval2)

- The A-Relation between the intervals is *disjoint* and the other elements are known;

(interval1 *overlaps* orient interval2)

- The A-Relation between the intervals is *overlaps* and the other elements are known;

(interval1 ??a orient interval2)

- The A-Relation between the intervals is unrestricted and the other elements are known;

(interval1 ??1 orient interval2)

- The A-Relation is restricted to a particular type (possibly determined by some other part of the query) and the other elements are known;

(interval1 ?A orient interval2)

- The A-Relation is not known yet and is to be established, and the other elements are known.

◇

3. The orientation between the intervals is used to differentiate between various A-Relations. It defines an ordering of the end points of the two intervals. The following examples show the role of orientation in queries.

Example 3.5 *Queries with different Orientations*

(interval1 *disjoint* 1<2 interval2)

- the relation is *disjoint* and the first interval precedes the second;

(interval1 *during* 1><2 interval2)

- the relation is *during* and the first interval encompasses the second;
(interval1 *during* ??a interval2)
- the relation is *during* with no specific orientation indicated;
(interval1 *disjoint* ??1 interval2)
- the relation is *disjoint* with a possibly predefined orientation;
(interval1 *overlaps* ?A interval2)
- the relation is *overlaps* with an unknown orientation which is to be established;
(interval1 ??a 1>2 interval2)
- interval1 terminates after interval2.

◇

The three types of elements that make up 1-D queries can be combined together in a variety of combinations where each element is either known or unknown.

Example 3.6 *Queries with different names, A-Relations and orientations*

A *disjoint* 1<2 B

- returns true if A is *disjoint* to B with an orientation where the begin point of A is less than the begin point of B;

?A *disjoint* 1<2 B

- returns all the intervals that are *disjoint* to B and the begin point of ?A is less than the begin point of B;

?A *disjoint* 1<2 ?B

- returns all the intervals that are *disjoint* to each other and the begin of ?A is less than the begin of ?B;

A ?A 1<2 ?B

- returns all the A-Relations and the intervals where the begin point of A precedes the begin point of that interval.

◇

3.4.1.2 2-D Queries

A 2-D query is expressed in terms of two 2-D objects and a 2-D NS-Relation. A 2-D NS-Relation consists of two 1-D NS-Relations. Thus, a 2-D query has the following structure:

object1 Relation1 Orient1 Relation2 Orient2 object2

The first A-Relation/Orientation pair (Relation1 Orient1) is used to express the objects' relationship along the x-axis (the first 1-D NS-Relation) and the second pair (Relation2 Orient2) is used to express the objects' relationship along the y-axis (the second 1-D NS-Relation). Objects 1 and 2 are represented by their interval approximations (their interval along each axis). For dimensions higher than one, the interval approximation of the objects will be referred to as the object.

The three different types of components that make up a 2-D query have been discussed in the previous section.

Example 3.7 *Sample 2-D queries*

(A *disjoint* 1<2 *disjoint* 1<2 B)

- verify whether A and B are non-intersecting and B is to the upper right of A;

(A *disjoint* 1<2 ??a ??a B)

- verify whether A is before B along the x-axis;

(A ??a ??a *during* 1><2 B)

- verify whether A is *during* 1><2 B along the y-axis;

(??a ?A ??a ?B ??a B)

- determine the A-Relations (along both axes) of all objects with respect to B.

◇

3.4.1.3 3-D Queries

A 3-D query consists of two objects and a 3-D NS-Relation (three pairs of A-Relation/Orientation used to describe relations along the x, y and z axes). The general form for a 3-D Query is:

object1 Relat1 Orient1 Relat2 Orient2 Relat3 Orient3 ***object2***

The 3-D query is a straightforward extension of the 2-D query. The additional NS-Relation (Relation3 Orient3) is used to express the constraint in the third dimension. Axis independent relational queries are discussed in Section 3.4.2.

Example 3.8 *Sample 3D queries*

(A *disjoint* 1<2 *disjoint* 1<2 *disjoint* 1<2 B)

- verify whether A precedes and is non-intersecting to B along all axes;

(A ??a 1<2 ??a 1<2 ??a 1<2 B)

- verify whether A starts before B in all three dimensions;

(?A ??a ??a *during* ??a ??a 1<2 B)

- determine which intervals occur *during* B along the y-axis and start before B along the z-axis.

◇

3.4.2 Composite n-D queries

Atomic n-D queries, as described in the previous section, provide a system for making a limited number of queries about the given environment. To increase the utility of atomic queries, a set of logical operators are defined for combining atomic queries into composite queries. This section first discusses the methods for making composite queries and then demonstrates the power of the system.

Three logical operators – conjunction, disjunction and negation – are used to create compound queries from the atomic queries. We use pre-fix notation to express composite queries.

1. All the atomic queries (q_i) in the **conjunction** ($\wedge q_1 \dots q_n$) must be true for the conjunction to be true. Binding of variables is ordered from left to right through the conjunction. Bindings that are made in the first query are applied to the second query. This process of binding variables in consecutive atomic queries is repeated until the final atomic query is reached.

- *(and (A disjoint 1<2 B) (A overlaps ??a ?A))*

Is A before B and if so does A overlap any intervals, and if so which ones?

2. **Disjunction** ($\vee q_1 \dots q_n$) of atomic queries requires at least one of the queries (q_j) be true for the entire query to be true. Bindings are made individually and all the bindings are returned as different instances of the solution. All solutions are determined.

- *(or (B overlaps ??a ??a) (C ?A ??a D))*

Does B overlap anything or what is the A-Relation between C and D?

3. **Negation** is used as a special instance of conjunction, where the negated query is used to eliminate specific bindings made as part of the non-negated query.

- *(and (?A disjoint ??a B) (not (?A overlaps ??a C)))*

What objects are disjoint to B and not overlapping C?

Negation is only used to help restrict conjunction queries. De Morgan's theorem can be used to reduce some other forms of negation into this format.

Example 3.9 *Resolving not*

Find all the pairs of objects whose relation is not disjoint-overlaps 1<2-1>2.

$$\begin{aligned} &(\text{not } ?A \text{ disjoint-overlaps } 1<2-1>2 ?B) \rightarrow \\ &(\text{and } (?A ??a-??a ??a-??a ?B) \\ &(\text{not } ?A \text{ disjoint-overlaps } 1<2-1>2 ?B)) \end{aligned}$$

This is translated into the search for all relations where they do not have the relationship disjoint-overlaps 1<2-1>2.

◇

Compound queries can be used to provide a variety of different tasks in querying. A selection is listed for the 1-D, 2-D and 3-D queries. The 1-D queries are selected to highlight some of the possibilities of composite queries, while the 2-D and 3-D queries look at issues specific to 2-D and 3-D respectively.

3.4.2.1 1-D Composite Queries

Compound queries demonstrate the true expressive power available with numbered and named variables. They allow for bindings to be made and thus restrictions to be propagated between the atomic components of the composite query.

Example 3.10 1D composite queries

(and (??1 *disjoint* ??a A) (?A *overlaps* ??a ??1))

What are the names of the intervals that overlap an interval that is *disjoint* to A?

(and (?A *disjoint* ??a B) (?A *overlaps* 1<2 C))

What intervals are *disjoint* to B and *overlap* 1<2 C?

◇

A variety of new queries can be defined in terms of atomic and composite 1-D queries. For instance, the heuristic query “between” can have two possible interpretations:

Case 1: non-intersecting “between”

One interval is between two intervals and it does not intersect either interval (*nibetween* see Figure 3.7i)

$$\begin{aligned} A \text{ nibetween } B \ C \rightarrow & (\text{or } (\text{and } (A \text{ disjoint } 1<2 \ B) \\ & (A \text{ disjoint } 1>2 \ C)) \\ & (\text{and } (A \text{ disjoint } 1>2 \ B) \\ & (A \text{ disjoint } 1<2 \ C)))) \end{aligned}$$

Case 2: intersecting “between”

An interval relation between two intervals may be intersecting (*ibetween* see Figure 3.7ii) and it may intersect one or both.

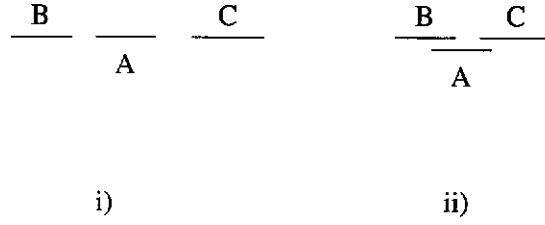


Figure 3.7: Two interpretations of between: i) non-intersecting between : A nibetween B C and ii) intersecting between: A ibetween B C.

$$\begin{aligned}
 A \text{ ibetween } B \ C \rightarrow & \text{(or (and (B ??a } 1 < 2 \ A) \\
 & (C ??a } 1 > 2 \ A)) \\
 & \text{(and (C ?? } 1 > 2 \ A) \\
 & (B ??a } 1 < 2 \ A)))
 \end{aligned}$$

Similarly, the relation “left-of” has two possible interpretations:

Case 1: non-intersecting left-of

A is to the left of *B* and the two intervals do not intersect each other (*nileft-of*).

$$A \text{ Nileft-of } B \rightarrow (A \text{ disjoint } 1 < 2 \ B)$$

Case 2: intersecting left-of

A is to the left of *B* and the two intervals may intersect each other (*ileft-of*):

$$A \text{ ileft-of } B \rightarrow (A \text{ ??a } 1 < 2 \ B)$$

A variety of such queries can be constructed in terms of combinations of the primitive A-Relation/Orientation pairs and in terms of other composite queries.

3.4.2.2 2-D Composite Queries

Composite queries may also be formed in 2-D. For example, an object is *between* if it is *between* in either dimension.

$$\begin{aligned}
 2\text{-D: } A \text{ nibetween } B \ C \rightarrow & \text{(or (and (A disjoint } 1 < 2 \text{ ??a ??a } B) \\
 & (A \text{ disjoint } 1 > 2 \text{ ??a ??a } C)) \\
 & \text{(and (A disjoint } 1 > 2 \text{ ??a ??a } B)
 \end{aligned}$$

(A (*disjoint* 1<2 ??a ??a C))
 (and (A ??a ??a *disjoint* 1>2 B)
 (A ??a ??a *disjoint* 1<2 C))
 (and (A ??a ??a *disjoint* 1<2 B)
 (A ??a ??a *disjoint* 1>2 C)))

2-D: A ibetween B C \rightarrow (or (and (A ??a 1<2 ??a ??a B)
 (A ??a 1>2 ??a ??a C))
 (and (A ??a 1>2 ??a ??a B)
 (A ??a 1<2 ??a ??a C))
 (and (A ??a ??a ??a 1>2 B)
 (A ??a ??a ??a 1<2 C)))
 (and (A ??a ??a ??a 1<2 B)
 (A ??a ??a ??a 1>2 C)))

The above examples define two objects to be “between” in 2-D if they are “between” in either of the dimensions. Alternative interpretations could be defined where two objects are “between” in 2-D only if they are “between” in both individual dimensions.

3.4.2.3 3-D Queries

In 3-D there is greater flexibility for expressing composite queries when compared to 1-D or 2-D. Each atomic 3-D query has 3 components (one for each dimension) and a composite query may impose restrictions on any one dimension (3 possibilities), any pair of dimensions (3 possibilities) or all 3 dimensions.

Following our example of the “between” operator we can see seven possible interpretations for “between” in 3D:

1. between along the x-axis only
2. between along the y-axis only
3. between along the z-axis only
4. between along the x and y axes

5. between along the x and z axes
6. between along the y and z axes
7. between along all three axes.

We can also allow for axis-independent definitions, such as A is “between” B and C along an axis. Two examples of axis-independent queries are given below.

If a single relation along one of the axes is known, there are three situations that must be checked: where the relation is true along the x, y or z axes.

```

3-D (object1 relation orient object2)
    (or (object1 relation orient ??a ??a ??a ??a object2)
        (object1 ??a ??a relation orient ??a ??a object2)
        (object1 ??a ??a ??a ??a relation orient object2))

```

When two relations are known, there are six permutations.

```

3-D (object1 RelationA OrientA RelationB OrientB object2)
    (or (object1 RelationA OrientA RelationB OrientB ??a ??a object2)
        (object1 RelationA OrientA ??a ??a RelationB OrientB object2)
        (object1 RelationB OrientB RelationA OrientA ??a ??a object2)
        (object1 RelationB OrientB ??a ??a RelationA OrientA object2)
        (object1 ??a ??a RelationA OrientA RelationB OrientB object2)
        (object1 ??a ??a RelationB OrientB RelationA OrientA object2))

```

We have demonstrated how our system for reasoning about spatial relations may be used to construct queries about object relations in 1-D, 2-D and 3-D. We have also demonstrated the range of possible n-D queries that can be defined by the user.

3.4.3 Specification

Similar to [13], we present a syntactic definition for general query construction in our system, using extended BNF notation.

$\langle A\text{-Rel} \rangle ::= disjoint \mid overlaps \mid during \mid ends \mid equals \mid meets$

$\langle \text{Ori} \rangle ::= 1 \langle 2 \mid 1 \rangle 2 \mid 2 \rangle 1 \mid 2 \langle 1 \mid 1 \rangle \langle 2 \mid 2 \rangle \langle 1$
 $\langle \text{variable} \rangle ::= \text{'??'} \mid \text{number} \mid \text{'a'} \mid \text{'?'} \text{name}$
 $\langle \text{A-Relation} \rangle ::= \langle \text{A-Rel} \rangle \mid \langle \text{variable} \rangle$
 $\langle \text{Orientation} \rangle ::= \langle \text{Ori} \rangle \mid \langle \text{variable} \rangle$
 $\langle \text{object} \rangle ::= \langle \text{variable} \rangle \mid \text{name}$
 $\langle 1\text{D-NS} \rangle ::= \langle \text{object} \rangle \langle \text{A-Relation} \rangle \langle \text{orientation} \rangle \langle \text{object} \rangle$
 $\langle \text{nD-A-Rel} \rangle ::= \langle \text{A-Relation} \rangle \{ \text{'-'} \langle \text{A-Relation} \rangle \}^{n-1}$
 $\langle \text{nD-Ori} \rangle ::= \langle \text{orientation} \rangle \{ \text{'-'} \langle \text{orientation} \rangle \}^{n-1}$
 $\langle \text{nD-NS} \rangle ::= \langle \text{object} \rangle \langle \text{nD-A-Rel} \rangle \langle \text{nD-Ori} \rangle \langle \text{object} \rangle \mid$
 $\quad \langle \text{object} \rangle \{ \langle \text{A-Relation} \rangle \langle \text{orientation} \rangle \}^n \langle \text{object} \rangle$
 $\langle \text{bi-boolean} \rangle ::= \text{or} \mid \text{and}$
 $\langle \text{uni-boolean} \rangle ::= \text{not}$
 $\langle \text{Composite} \rangle ::= \langle \text{uni-boolean} \rangle \langle \text{nD-NS} \rangle \mid \langle \text{bi-boolean} \rangle \langle \text{nD-NS} \rangle \langle \text{nD-NS} \rangle$
 $\quad \mid \langle \text{bi-boolean} \rangle \langle \text{Composite} \rangle \langle \text{Composite} \rangle$

Number is a string of numeric characters. **Name** is a string of alpha-numeric characters.

Note: When constructing Composite relations each of the $\langle \text{nD-NS} \rangle$ must be of the same dimension.

3.4.4 Classifications and Queries

Classifications provide the means of describing groups of relations such that high-level reasoning can be performed on them. The 1, 2 and 3 dimensional simple and composite queries developed in the previous sections have provided the means for representing relations at different levels of abstraction. The querying system offers the means of forming queries at any of the three levels of Classification.

Classification 1 and 2 queries are easily constructed from the simple queries using variables as shown below.

$$\begin{aligned}
\text{object1 } (\textit{disjoint } 1<2)^{C1} \text{ object2} &\rightarrow \text{object1 } \textit{disjoint } 1<2 \text{ object2} \\
\text{object1 } (\textit{disjoint})^{C2} \text{ object2} &\rightarrow \text{object1 } \textit{disjoint } ??a \text{ object2}
\end{aligned}$$

Classification 3 queries can be constructed using composite queries.

$$\begin{aligned}
\text{object1 } (\textit{disjoint, overlaps})^{C3} \text{ object2} &\rightarrow \\
&(\text{or } (\text{object1 } \textit{disjoint } ??a \textit{ overlaps } ??a \text{ object2}) \\
&(\text{object1 } \textit{overlaps } ??a \textit{ disjoint } ??a \text{ object2}))
\end{aligned}$$

3.5 Conclusion

This chapter has demonstrated a relational system for reasoning about multi-dimensional spatial relations. The system uses the concept of flow and axis independence to provide a means of classifying relations into groups that share common characteristics. A description of the complete set of 13 primitive interval relations is given for the spatial domain in terms of NS-Relations. The NS-Relation is defined in terms of the invariant (A-Relation) and variant (orientation) component of the relation.

The basic 1-D representation is generalised to n-D. The difficulties of reasoning with a large (multi-dimensional) space are reduced by using a system of hierarchic classifications that allow for the grouping of relations that share common characteristics. Finally, a set of query primitives are defined (based on the NS-Relation), which allow for the construction of a comprehensive set of queries. This query language provides the only means of access to information about spatial relations.

This concludes the section dealing with multi-dimensional spatial reasoning. In the next chapter we begin our work on representing the intervals which have been used to represent the spatial information. The next chapter details a representation of intervals for a simple subset and then shows how the querying system of this chapter is applied to the representation.

Chapter 4

Interval Representation

The best material model of a cat is another, or preferably the same, cat.

A. Rosenblueth and N. Wiener

The previous chapter examined a method of reasoning with multi-dimensional space. This method involved representing objects in space in terms of their intervals along perpendicular axes. Reasoning about spatial relations was performed in terms of these interval-based representations.

For the rest of the thesis, the problem of representing this interval information shall be the focus of our effort. This problem has been identified as being non-trivial and directly comparable to the problem of representing temporal intervals.

Figure 4.1 illustrates the division of the representation problem. At the highest level is the spatial information. The middle level is the interval-based representation of these spatial objects. The lowest level is the actual representation of the interval information.

We shall first formally define interval representations, and then the four different kinds of interval representations.

Definition 4.1 *An interval representation $\underline{IR} = \langle Int, IA, \phi \rangle$ is a triple where Int is the set of intervals, IA is the set of all possible interval relations (2^{13}) and ϕ is a set of triples $\langle interval_a \text{ label } interval_b \rangle$ defining the mapping between the intervals and their relations, where $interval_a, interval_b \in Int$ and $label \in IA$.*

The terms “label”, “interval relation” and “edge” will be used interchangeably.

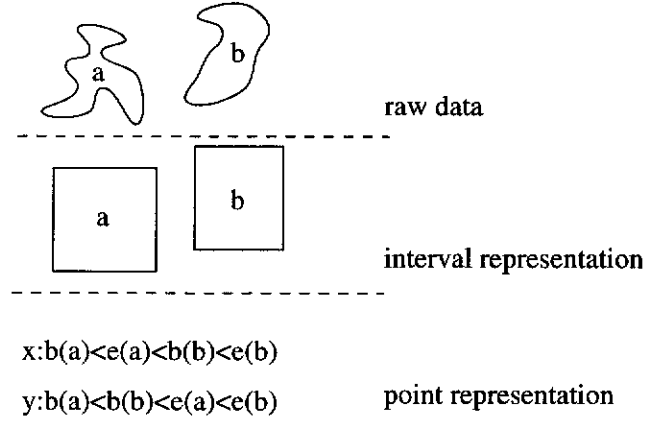


Figure 4.1: The three levels of representation: raw data represents the quantitative information; the interval representation is the interval approximation of the object; and the point representation is the point representation of intervals.

There are two criteria by which we discriminate between different interval representations. By restricting the types of interval relations that may be used to create an interval representation, we are able to define the following:

Definition 4.2 A *Primitive IR* is an interval representation where the set of labels is restricted to primitive interval relations, i.e. $\{disjoint\ 1<2, disjoint\ 1>2, overlaps\ 1<2, overlaps\ 1>2, equals, during\ 1><2, during\ 2><1, meets\ 1<2, meets\ 1>2, ends\ 1<2, ends\ 1>2, ends\ 2<1, end\ 2>1\}$.

Definition 4.3 A *Disjunctive IR* is an interval representation where at least one label in the IR is a disjunctive interval relation.

Our second criteria is the number of relations that are explicitly known. An unknown relation is a disjunction of all thirteen primitive interval relations. There are two cases: where all relations are explicitly known, and where only some of the relations are explicitly known.

The two criteria allows us to define four interval representations as follows:

1. a primitive interval representation where all relations are explicitly known
2. a primitive interval representation where some relations (but not all) are explicitly known

3. a disjunctive interval representation where all relations are explicitly known
4. a disjunctive interval representation where some relations (but not all) are explicitly known.

In this chapter we shall describe the method of representing spatial information using an interval representation and show, for one type of interval representation (all are explicitly known and every relation is a primitive interval relation), the method of representing these intervals.

The representation of the other types of interval sets will be the subject of the following two chapters. The systems described in those two chapters are interchangeable with the interval representation described in this chapter, with respect to the overall spatial reasoning system. They differ only in the types of information to be represented and the method of representation used.

4.1 Intervals and end points

The problem of representing intervals is not new. In the domain of temporal reasoning this problem has generated a variety of different solutions. Allen's [2] interval-based reasoning system promotes a powerful method of representing intervals as the most primitive unit and develops a method of closure for propagating information about intervals.

Despite its virtues, Allen's system does not provide a complete closure (all possible propagations) of the interval information. Alternative methods have used different combinations of primitives (points and intervals) and propagation methods (closure and non-closure).

These different combinations are motivated by desire to minimise different costs associated with the representation. The costs are:

- the space to represent the information,
- the construction time and
- the search time (to determine the relationship between a pair of intervals).

The construction time and search time are non-complementary. Space used to represent the information plays a secondary consideration.

The method of interval representation used in this thesis is based upon a point based representation of intervals that uses non-closure methods of propagating relational information. For the simplistic interval representation of this chapter, many of the details of the representation will not be apparent. Chapters 5 and 6 provide greater detail of the proposed system.

For the rest of this section, the issues of representing intervals in terms of points are discussed.

Definition 4.4 *An interval \mathbf{A} consists of two end points. The begin point of interval \mathbf{A} is denoted as ‘ a ’ and the end point is denoted as ‘ A ’. The relationship between two points x and y is represented in either of the following formats:*

xy to illustrate that $x < y$

(xy) to illustrate that $x = y$

4.1.1 Interval relations to point relations

Using the end point representation of intervals, it is possible to derive an end point-based description of primitive interval relations.

Definition 4.5 *A primitive interval relation can be expressed in terms of four point relations. The four point relations that will be considered for any two intervals \mathbf{X} and \mathbf{Y} are: xy , xY , yX and XY . For every unambiguous relationship (primitive interval relation) between intervals \mathbf{X} and \mathbf{Y} , there is an equivalent unique conjunction of point relations (see Table 4.1). A conjunction is used as all of the point relations must be true if the corresponding interval relation is true.*

It is now possible to represent interval relations in terms of conjunctions of point relations.

4.1.2 Point relations to interval relations

The information about interval relations, as represented by the point-based representation of the intervals, must be derivable from the point representation. By

Table 4.1: The interval relations described in terms of the relationship between the end points of the intervals.

<i>XY Interval Relation</i>	<i>xy</i>	<i>xY</i>	<i>yX</i>	<i>XY</i>
disjoint 1<2	<	<	>	<
disjoint 1>2	>	>	<	>
meets 1<2	<	<	=	<
meets 1>2	>	=	<	>
overlaps 1<2	<	<	<	<
overlaps 1>2	>	<	<	>
during 2><1	<	<	<	>
during 1><2	>	<	<	<
ends 2>1	=	<	<	<
ends 1>2	=	<	<	>
ends 2<1	>	<	<	=
ends 1<2	<	<	<	=
equal	=	<	<	=

reversing the process from the previous section, an interval relation between two intervals X and Y can be determined by establishing the relationship of four key point relations: *xy*, *xY*, *yX* and *XY*. Thus the method of interval representation using point relations is complete.

Table 4.2 demonstrates that, for each of the interval relations, there are key point relations that will determine the actual interval relation. For example, the relation *disjoint 1<2* is true if and only if the relation *y>X* is true.

For the problem of determining if a given interval relation is true between two intervals, a maximum of three point queries is required. When determining the interval relation between two intervals, a careful ordering of the point relations to be queried makes it possible to reduce the average number of point queries needed to determine the interval relation.

Seven of the thirteen relations can be identified (or ruled out) by determining the point relationship between *xy* and *XY*. Thus if the checks are performed in

Table 4.2: Interval relations defined in terms of their significant point relations.

<i>Interval Relation</i>	xy	xY	yX	XY
disjoint $1 < 2$			$>$	
disjoint $1 > 2$		$>$		
meets $1 < 2$			$=$	
meets $1 > 2$		$=$		
overlaps $1 < 2$	$<$		$<$	$<$
overlaps $1 > 2$	$>$	$<$		$>$
during $2 > < 1$	$>$			$<$
during $1 > < 2$	$<$			$>$
ends $2 > 1$	$=$			$<$
ends $1 > 2$	$=$			$>$
ends $2 < 1$	$>$			$=$
ends $1 < 2$	$<$			$=$
equal	$=$			$=$

the following order: xy , XY , yX and xY ; then only three of all interval relations (namely *meets $1 > 2$* , *disjoint $1 > 2$* and *overlaps $1 > 2$*) require all four checks to be performed. Hence, the efficiency of the querying system depends on the checks being performed in the correct sequence.

We shall be considering a variety of situations for representing intervals using point relations and determining interval relations from point relations. In some situations it is not possible to unambiguously determine all of the four point relations used to describe a given interval (see Chapters 5 and 6).

In such a situation the point relations that can be determined unambiguously may be used to restrict the possible interval relations to a subset of the 13 primitives. For example, if it is known that the begin point of **A** and the begin point of **B** are equal, then (from Table 4.1) the relationship between intervals **A** and **B** is one of the following: *ends $2 > 1$* , *ends $1 > 2$* or *equal*.

Given this ability to represent primitive interval relations as point relations

and the ability to query this point representation to extract information about interval relations, we shall now consider the simplest case: a primitive interval representation where every relation is explicitly known.

4.2 Interval Representation: The Simple Case

Even with the simplest type of representation – a set of intervals where every relation between every pair of intervals is explicitly known and each of these relations is a primitive interval relation – the considerations of representation space, construction time and search time are an issue.

Two methods of representation are considered. The first represents all interval relations (though this is the closure method there is no need to propagate information at construction time as all information is available), and the second represents a minimal set of information about intervals by representing them in terms of end points.

4.2.1 Closure

For a set of intervals where every interval relation is known and each interval relation is a primitive interval relation, an explicit representation would contain n intervals and $n(n-1)$ primitive interval relations.

For this situation the construction time is 0 (as all relations are known), search space is $n(n-1)$ and search time is constant (table lookup).

4.2.2 Linear Ordering

Alternatively, the intervals and the interval relations may be represented in terms of their end points and the relationships between these end points. For the problem where every interval relation is known and every known relation is a primitive interval relation, the graph used to represent the point relations can be reduced to a linear ordering of end points.

Definition 4.6 *A primitive interval representation based upon points may be represented as a Point Graph $(PG) = (N, \varphi)$ where N is the set of nodes (all*

end points of intervals) and φ is a set of pairs $\langle n_i, n_j \rangle$ defining a relation on the nodes; n_i and $n_j \in N$.

$\langle n_i, n_j \rangle$ is an edge that represents the relationship “<” between points represented by nodes n_i and n_j . A pair of points that are equal are represented as a single node that has a compound label (representing the two end points). The relation $n_i > n_j$ may be represented as $\langle n_j, n_i \rangle$.

The construction of the graph PG from the interval representation is a two part process (see Algorithm 4.1) of converting the intervals into end points and then converting interval relations into conjunctions of point relations. These point relations may then be added to the graph. Every interval relation in the set of interval relations IR can be converted into a conjunction of point relations $\langle \text{pr}_a \wedge \text{pr}_b \wedge \text{pr}_c \wedge \text{pr}_d \rangle$.

```

Given interval representation  $\langle \text{Int}, \text{IR} \rangle$ 
 $N = \emptyset, \varphi = \emptyset$ 
{the points from the intervals}
foreach interval  $\in \text{Int}$ 
   $N' = N + B(\text{interval}) + E(\text{interval})$ 
  {the begin < end relation}
foreach interval  $\in \text{Int}$ 
   $\varphi' = \varphi + B(\text{interval}) < E(\text{interval})$ 
  {the point relation from the interval relations}
foreach  $\langle \text{pr}_a \wedge \text{pr}_b \wedge \text{pr}_c \wedge \text{pr}_d \rangle \in \text{IR}$ 
  foreach  $\langle p_x \text{ relation } p_y \rangle \in \langle \text{pr}_a \wedge \text{pr}_b \wedge \text{pr}_c \wedge \text{pr}_d \rangle$ 
     $\varphi' = \varphi + \langle p_x \text{ relation } p_y \rangle$ 

```

Algorithm 4.1 *The procedure for constructing a point graph from a primitive interval representation.*

This type of representation allows for a linear ordering to be constructed. The point graph representation can be reduced to a linear ordering by removing all redundant edges from the graph (full minimisation).

Let G be a graph and \mathbf{ab} the relation between two nodes in the graph G . Let the function $search(G, rel)$ be a boolean function, where the first argument is a graph and the second a point relation. The function $search$ returns true if relation rel can be found by performing a search of the graph G , otherwise the function returns false (i.e. if false or not known).

Definition 4.7 Given $G = (N, \varphi)$, redundant edges is a set φ_r such that $\varphi_r \subset \varphi$, and φ_r are defined as the edges that can be removed from the graph such that every relation represented by these edges can be deduced by searching the graph $(N, \varphi - \varphi_r)$.

Thus we can state that the edge used to represent the relation \mathbf{ab} is redundant with respect to the graph G if the following is true:

$$search(G - \mathbf{ab}, \mathbf{ab}).$$

Definition 4.8 A fully minimised graph is a graph $G = (N, \varphi)$ where none of the edges in φ are redundant ($\varphi_r = \emptyset$).

See [1] for a formal analysis of graph minimisation (also called Transitive Reduction).

If all redundant edges are removed from the graph PG , and PG is derived from a primitive interval representation, where all relations are explicitly known, then the resulting graph is a linear ordering of points. To remove the redundant edges every relation $(2n-1)$ must be checked in $O(n)$ time. Thus the linear ordering would be achieved in $O(n^2)$ time.

Our non-closure method presented here is not as efficient, in terms of search time and construction time, as the closure method described above. It is presented here only to show that it does provide a viable method of representing information about intervals.

The non-closure method, however, requires less space than the closure method. For a problem where space is paramount, this solution would be preferred over the closure method.

In this simple situation the closure method does not need to propagate information to determine every interval relation (as they are already known). For

other more complex situations the necessity to propagate information in order to perform closure significantly increases the cost of constructing the representation (NP-Complete).

The simplistic case of this section provides the opportunity to represent a minimal amount of information about the interval relations. As each relation between the intervals is known and is primitive, it is possible to construct a linear ordering of the end points. Thus for n intervals there are $2n$ end points and $2n-1$ point relations. The relationship between end points can be determined in (on average) $2 \times (n/2)$ time [62]. It takes up to four point searches to determine an interval relation. The worst case would be $4n$ time.

4.3 Results

In this chapter an algorithm has been described using a system based on linear ordering for representing information about intervals, for sets of intervals where every interval relation between a pair of intervals in the set is explicitly known and that relation is a primitive interval relation.

Though this is not an optimal solution, it does provide the basis for constructing a complete system that encompasses the entire process of representing spatial information (see Figure 4.1). In Chapters 5 and 6 more complex problems will be solved by expanding the minimisation technique outlined in this chapter. These new methods will provide the same information that the linear order system provides (though they represent different types of interval information) and are thus equivalent with respect to the high-level reasoning and subsequent queries on the representation.

This section on results does not emphasise the performance of the system (with respect to search time, construction time and representation space) but demonstrates the ability to connect the components together. The raw data about spatial relations is represented as intervals. Intervals are represented as points. Queries about object relations are described in terms of interval relations. These interval queries are then converted into point queries. Finally, the point query (a search) is performed upon the point representation (see Figure 4.2).

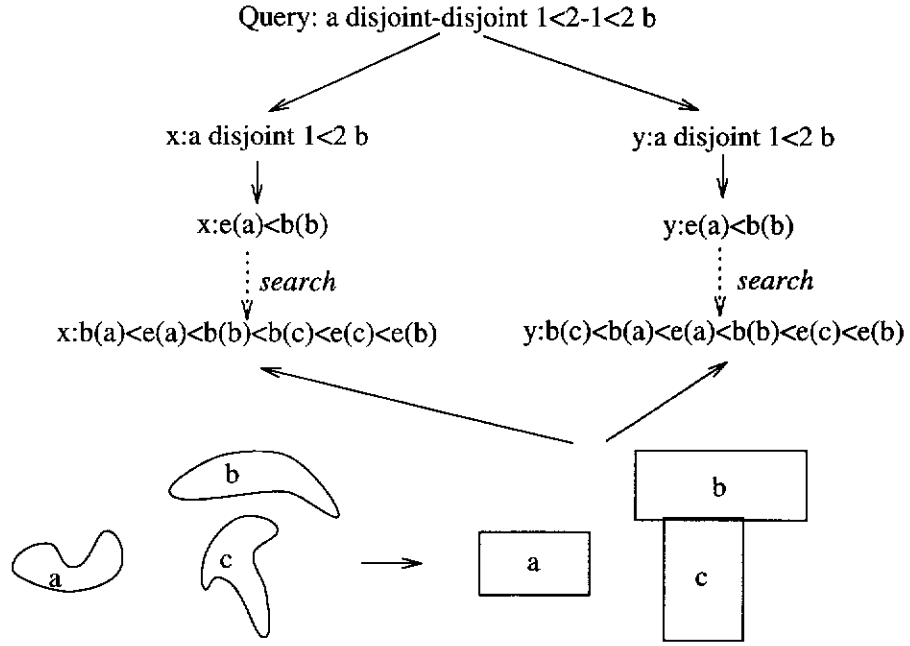


Figure 4.2: The complete system entails the representation of raw spatial data as intervals, the representation of these intervals as points, the specification of n-D queries, the processing of n-D queries into single dimensional interval queries, the conversion of interval queries to point queries and the search of point queries in a point representation.

The system described has been implemented in C++ on an SG R4400 processor.

In the next section, we introduce an artificially generated dataset (randomly choosing the positions of the objects). The 2-D environment depicted in Figure 4.3 will be used for the example queries. Examples of queries are given in English and are then translated into the syntax accepted by the query system.

For each example the query is converted into the equivalent point queries that are performed by the low level point search system. Bindings of variables are performed from left to right.

4.3.1 Simple queries

The following are examples of simple queries performed on a sample environment (Figure 4.3) using the system described in this chapter. The results were generated from a program. Queries presented to the querying program and results

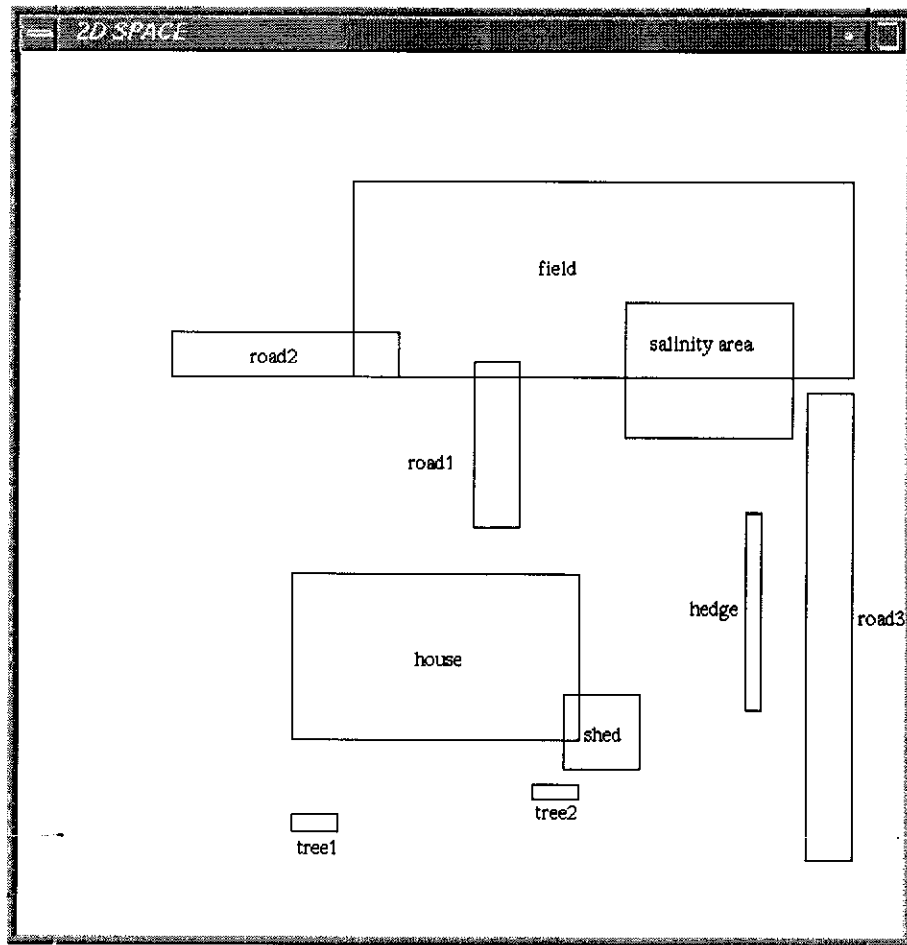


Figure 4.3: A 2-D example environment. The flow runs from top to bottom and from left to right.

from the program are prefixed with '*P*'.

Query1: Is the object *road2* disjoint to and followed by object *road3* along both axes?

Result 1 - true

P: (and *road2* disjoint-disjoint 1<2-1<2 *road3*)

x-axis: *road2* disjoint 1<2 *road3* \Rightarrow b(*road3*) > e(*road2*)

y-axis: *road2* disjoint 1<2 *road3* \Rightarrow b(*road3*) > e(*road2*)

***P*:Status:true**

◇

Query2: What is the name of all the objects that are *disjoint* to and preceded by the object *road2* along both axes?

Result 2: road3, hedge, shed and tree2

P:(and road2 disjoint-disjoint 1<2-1<2 ?object)

x-axis: road2 disjoint 1<2 ?object \Rightarrow b(?object) > e(road2)

y-axis: road2 disjoint 1<2 ?object \Rightarrow b(?object) > e(road2)

The x-axis query binds ?object to all the objects whose begin point is greater than e(road2) and then uses those bindings to test if those objects also have their begin point greater than e(road2) in the y-axis.

P: (?object= road3) (?object= hedge) (?object= shed) (?object= tree2)

◇

Query3: What are all the objects *disjoint* to and preceded by another object along both axes?

Result 3: (road2 to road3), (road2 to hedge), (road2 to shed), (road2 to tree2), (road1 to shed), (road1 to tree2)

P:(and ?object1 disjoint-disjoint 1<2-1<2 ?object2)

x-axis: ?object1 disjoint 1<2 ?object2 \Rightarrow b(?object2) > e(?object1)

y-axis: ?object1 disjoint 1<2 ?object2 \Rightarrow b(?object2) > e(?object1)

The x-axis query binds the pairs of objects to ?object1 and ?object2 if the begin of ?object2 is greater than the end point of ?object1. It then takes those bindings and checks to see if that relation is true for the y-axis.

*P: ((?object1= road2)(?object2= road3))
((?object1= road2)(?object2= hedge))
((?object1= road2)(?object2= shed))
((?object1= road2)(?object2= tree2))
((?object1= road1)(?object2= shed))
((?object1= road1)(?object2= tree2))*

◇

Query4: What is the orientation of the two objects *road2* and *hedge* that are disjoint along both dimensions?

Result 4: $1 < 2, 1 < 2$

P: (and road2 disjoint-disjoint ?or1-?or2 hedge)

x-axis: $\text{road2 disjoint ?or1 hedge} \Rightarrow (\text{or } b(\text{road2}) > e(\text{hedge}) \text{ } b(\text{hedge}) > e(\text{road2}))$

y-axis: $\text{road2 disjoint ?or2 hedge} \Rightarrow (\text{or } b(\text{road2}) > e(\text{hedge}) \text{ } b(\text{hedge}) > e(\text{road2}))$

The query checks to see if either of the disjoints are true for the x-axis by checking that $b(\text{road2})$ greater than the $e(\text{hedge})$ or that the $b(\text{hedge})$ is greater than the $e(\text{road2})$. If one is true then checks to see if it is disjoint for the y-axis. Only returns a value if a relation was found for both axes.

P: (?or1 = 1 < 2) (?or2 = 1 < 2)

◇

Query5: For all the objects that are *disjoint* along both axes to the object *road2* return the object's name and the orientation.

Result 5: $(1 < 2, 1 < 2, \text{road3}), (1 < 2, 1 < 2, \text{hedge}), (1 < 2, 1 < 2, \text{shed}), (1 < 2, 1 < 2, \text{tree2})$

P: (and road2 disjoint-disjoint ?or1-?or2 ?object)

x-axis: $\text{road2 disjoint ?or1 ?object} \Rightarrow (\text{or } b(\text{road2}) > e(\text{?object}) \text{ } b(\text{?object}) > e(\text{road2}))$

y-axis: $\text{road2 disjoint ?or2 ?object} \Rightarrow (\text{or } b(\text{road2}) > e(\text{?object}) \text{ } b(\text{?object}) > e(\text{road2}))$

The query finds all the objects whose end point is less than $b(\text{road2})$ or whose begin point is greater than the $e(\text{road2})$. Given the x-axis bindings, it checks to see if the end point of each binding is greater than the $b(\text{road2})$ or the begin point of each binding is less than the $e(\text{road2})$ along the y-axis.

P: ((?object= road3)(?or1= 1<2)(?or2= 1<2))
 ((?object= hedge)(?or1= 1<2)(?or2= 1<2))
 ((?object= shed)(?or1= 1<2)(?or2= 1<2))
 ((?object= tree2)(?or1= 1<2)(?or2= 1<2))

◇

Query6: Return the A-Relation along the y-axis, the orientation along both axes and the object's name for all the objects that are *disjoint* to the object *road2* along the x-axis?

Result 6: (*disjoint*, 1<2, 1<2, road3), (*disjoint*, 1<2, 1<2, hedge), (*during*, 1<2, 2><1, salinity area), (*disjoint*, 1<2, 1<2, shed), (*disjoint*, 1<2, 1<2, tree2), (*overlaps*, 1<2, 1<2 road1).

P: (and road2 disjoint-?rel ?or1-?or2 ?object)

x-axis: road2 disjoint ?or1 ?object \Rightarrow (or b(road2) > e(?object) b(?object) > e(road2))

y-axis: road2 ?rel ?or2 ?object \Rightarrow b(road2) ? b(?object), b(road2) ? e(?object), b(?object) ? e(road2), e(road2) ? e(?object)

The query determines if there are objects whose end point is less than b(road2) or that the begin point is greater than e(road2). For each of those bound objects it determines their relation to the object road2, using the standard four point test.

P: ((?object= road3)(?or1= 1<2)(?or2= 1<2)(?rel= *disjoint*))
 ((?object= hedge)(?or1= 1<2)(?or2= 1<2)(?rel= *disjoint*))
 ((?object= salinity area)(?or1= 1<2)(?or2= 2><1)(?rel= *during*))
 ((?object= shed)(?or1= 1<2)(?or2= 1<2)(?rel= *disjoint*))
 ((?object= tree2)(?or1= 1<2)(?or2= 1<2)(?rel= *disjoint*))
 ((?object= road1)(?or1= 1<2)(?or2= 1<2)(?rel= *overlaps*))

◇

Query7: What is the A-Relation along each axis for the objects *road2* and *road3*, given the begin of *road2* precedes the begin of *road3* along each axis.

Result 7: (*disjoint*, *disjoint*)

P: (*and road2 ?rel-?rel2 1<2-1<2 road3*)

x-axis: *road2 ?rel 1<2 road3* \Rightarrow *b(road2) < b(road3)*, *b(road2) ? e(road3)*,
b(road3) ? e(road2), *e(road2) ? e(road3)*

y-axis: *road2 ?rel 1<2 road3* \Rightarrow *b(road2) < b(road3)*, *b(road2) ? e(road3)*,
b(road3) ? e(road2), *e(road2) ? e(road3)*

This is a slight variation on the four point test as one of the points must be a particular value *b(road2) < b(road3)*. If there is a valid binding this will determine the x-axis relation. The same approach applies for the y-axis.

P: (*?rel= disjoint*)(*?rel2= disjoint*)

◇

Query8: For an object whose begin point is preceded by the begin point of the object *road2* along both axes, return the object's name and the A-Relations along each axis.

Result 8: (*overlaps*, *disjoint*, *house*), (*disjoint*, *disjoint*, *road3*), (*disjoint*, *disjoint*, *hedge*), (*disjoint*, *disjoint*, *shed*), (*disjoint*, *disjoint*, *tree2*), (*disjoint*, *overlaps*, *road1*).

P: (*and road2 ?rel-?rel2 1<2-1<2 ?object*)

x-axis: *road2 ?rel 1<2 ?object* \Rightarrow *b(road2) < b(?object)*, *b(road2) ? e(?object)*,
b(?object) ? e(road2), *e(road2) ? e(?object)*

y-axis: *road2 ?rel 1<2 ?object* \Rightarrow *b(road2) < b(?object)*, *b(road2) ? e(?object)*,
b(?object) ? e(road2), *e(road2) ? e(?object)*

The query determines all the objects whose begin point is greater than *b(road2)* and then determines the relation using the four point test (checks the other three relations) along the x-axis. For the y-axis, each of the objects bound to *?object* are then tested to see if they satisfy *b(road2) < b(?object)*. If they do, then check to see if there is a valid relation.

P: ((?object= house)(?rel= *overlaps*)(?rel2= *disjoint*))
 ((?object= road3)(?rel= *disjoint*)(?rel2= *disjoint*))
 ((?object= hedge)(?rel= *disjoint*)(?rel2= *disjoint*))
 ((?object= shed)(?rel= *disjoint*)(?rel2= *disjoint*))
 ((?object= tree2)(?rel= *disjoint*)(?rel2= *disjoint*))
 ((?object= road1)(?rel= *disjoint*)(?rel2= *overlaps*))

◇

Query9: What is the relationship between the objects *road2* and *road3*?

Result 9: *disjoint-disjoint* 1<2-1<2

P: (and road2 ?rel-?rel2 ?or-?or2 road3)

x-axis: road2 ?rel ?or road3 \Rightarrow b(road2) ? b(road3), b(road2) ? e(road3),
 b(road3) ? e(road2), e(road2) ? e(road3)

y-axis: road2 ?rel2 ?or2 road3 \Rightarrow b(road2) ? b(road3), b(road2) ? e(road3),
 b(road3) ? e(road2), e(road2) ? e(road3)

This is a standard four point test to determine the relationship between objects road2 and road3. The same check is performed to determine the y-axis relation.

P: (?rel= *disjoint*)(?or= 1<2)(?or2= 1<2)(?rel2= *disjoint*)

◇

Query10: What is the relationship of every object to *road2*?

Result 10: (*equal-equal* road2), (*during-disjoint* 1><2-1<2 tree1), (*overlaps-ends* 1<2-2>1 field), (*overlaps-disjoint* 1<2-1<2 house), (*disjoint-disjoint* 1<2-1<2 road3), (*disjoint-disjoint* 1<2-1<2 hedge), (*disjoint-during* 1<2-2><1 salinity area), (*disjoint-disjoint* 1<2-1<2 shed), (*disjoint-disjoint* 1<2-1<2 tree2), (*disjoint-overlaps* 1<2-1<2, road1)

P: (and road2 ?rel-?rel2 ?or-?or2 ?object)

x-axis: road2 ?rel ?or ?object \Rightarrow b(road2) ? b(?object), b(road2) ? e(?object),
 b(?object) ? e(road2), e(road2) ? e(?object)

y-axis: $\text{road2} \text{ ?rel2 ?or2 ?object} \Rightarrow \text{b(road2) ? b(?object), b(road2) ? e(?object),}$
 $\text{b(?object) ? e(road2), e(road2) ? e(?object)}$

For every object it uses the four point test to determine the relationship to the object road2. It then does the same for the y-axis.

P: $((\text{?object} = \text{road2})(\text{?rel} = \text{equal})(\text{?rel2} = \text{equal})$
 $((\text{?object} = \text{tree1})(\text{?rel} = \text{during})(\text{?or} = 1 > 2)(\text{?or2} = 1 < 2)(\text{?rel2} = \text{disjoint})$
 $((\text{?object} = \text{field})(\text{?rel} = \text{overlaps})(\text{?or} = 1 < 2)(\text{?or2} = 2 > 1)(\text{?rel2} = \text{ends}))$
 $((\text{?object} = \text{house})(\text{?rel} = \text{overlaps})(\text{?or} = 1 < 2)(\text{?or2} = 1 < 2)(\text{?rel2} = \text{disjoint}))$
 $((\text{?object} = \text{road3})(\text{?rel} = \text{disjoint})(\text{?or} = 1 < 2)(\text{?or2} = 1 < 2)(\text{?rel2} = \text{disjoint}))$
 $((\text{?object} = \text{hedge})(\text{?rel} = \text{disjoint})(\text{?or} = 1 < 2)(\text{?or2} = 1 < 2)(\text{?rel2} = \text{disjoint}))$
 $((\text{?object} = \text{salinity area})(\text{?rel} = \text{disjoint})(\text{?or} = 1 < 2)(\text{?or2} = 2 > 1)(\text{?rel2} = \text{during}))$
 $((\text{?object} = \text{shed})(\text{?rel} = \text{disjoint})(\text{?or} = 1 < 2)(\text{?or2} = 1 < 2)(\text{?rel2} = \text{disjoint}))$
 $((\text{?object} = \text{tree2})(\text{?rel} = \text{disjoint})(\text{?or} = 1 < 2)(\text{?or2} = 1 < 2)(\text{?rel2} = \text{disjoint}))$
 $((\text{?object} = \text{road1})(\text{?rel} = \text{disjoint})(\text{?or} = 1 < 2)(\text{?or2} = 1 < 2)(\text{?rel2} = \text{overlaps}))$

◇

4.3.2 Composite Queries

In this section we give examples of how composite queries are constructed from simple queries. Logical operators {and, or, not} are used to logically connect individual simple queries. Low level point queries are not given in this section.

Query11: Is it true that the relationship between objects *field* and *hedge* is

during 1 > 2 along the x-axis and

disjoint 1 < 2 along the y-axis

and that the relationship between object *road3* and *hedge* is

disjoint 1 > 2 along the x-axis and

during 1 > 2 along the y-axis?

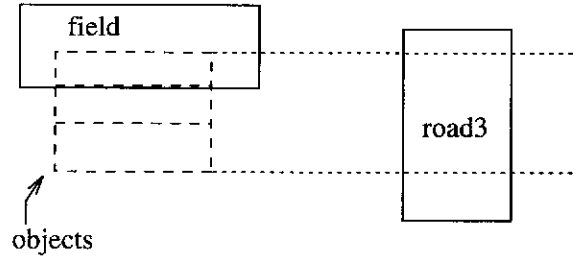


Figure 4.4: An illustration of possible positions of the *objects* with relation to the *field* and the *road3* (Query 12). The actual relation between the *field* and the *road3* may not be known.

Result 11: True

P: (and field during-disjoint 1><2-1<2 hedge road3 disjoint-during 1>2-1><2 hedge)

P: Status: true

◇

Query12: Find all the objects that have the relationship

during 1><2 along the x-axis to the object *field*, and

the begin point follows the begin point of the object *field* along the y-axis

and the object also has the relationship

during 1><2 along the y-axis to the object *road3*, and

the end point of the object is before the end point of the object *road3* along the x-axis?

The query is depicted in Figure 4.4

Results 12: (*disjoint, disjoint, hedge*), (*disjoint, disjoint, shed*), (*disjoint, disjoint tree2*)

P: (and field during-?rel 1><2-1<2 ?objects road3 ?rel2-during 1>2-1><2 ?objects)

P: ((?rel2= disjoint)(?objects= hedge)(?rel= disjoint))
((?rel2= disjoint)(?objects= shed)(?rel= disjoint))
((?rel2= disjoint)(?objects= tree2)(?rel= disjoint))

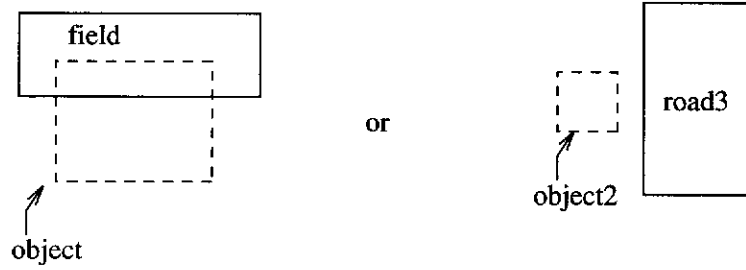


Figure 4.5: The possible position of the *object* and the *object2* as described in the composite query (Query 13).

◇

Query13: Find the names of all the objects that have the relationships

during $1 > 2$ along the x-axis

overlaps $1 < 2$ along the y-axis

to the object *field*, or have the relationship

disjoint $1 > 2$ along the x-axis

during $1 > 2$ along the y-axis

to object *road3* (see Figure 4.5).

Result 13: (house, tree1, tree2, shed, hedge), (salinity area, road1)

P: (or *field* *during-overlaps* $1 > 2$ - $1 < 2$?*object* *road3* *disjoint-during* $1 > 2$ - $1 > 2$?*object2*)

P: (?*object2*= house) (?*object2*= tree1) (?*object2*= tree2) (?*object2*= shed) (?*object2*= hedge) (?*object*= salinity area) (?*object*= road1)

◇

Query14: What are the objects that have the relationship

during $1 > 2$ along the x-axis to the object *field*,

the begin point follows the begin point of the object *field* along the y-axis

or have the relation

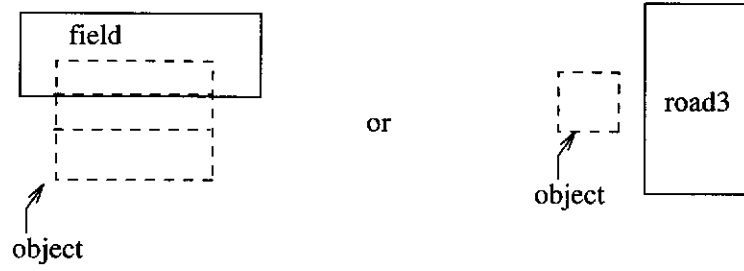


Figure 4.6: The possible position of *object* given with relation to *field* and *road3* (Query 14).

disjoint $1 > 2$ along the x-axis

during $1 > 2$ along the y-axis

to object *road3* (see Figure 4.6)?

Result 14: house, tree1, tree2, shed, hedge, (*disjoint*, hedge), (*overlaps*, salinity area), (*disjoint*, shed), (*disjoint*, tree2), (*overlaps*, road1).

P: (or field during-?rel 1 > 2-1 < 2 ?object road3 disjoint-during 1 > 2-1 < 2 ?object)

P: ((?object= house))
 ((?object= tree1))
 ((?object= tree2))
 ((?object= shed))
 ((?object= hedge))
 ((?object= hedge)(?rel= *disjoint*))
 ((?object= salinity area)(?rel= *overlaps*))
 ((?object= shed)(?rel= *disjoint*))
 ((?object= tree2)(?rel= *disjoint*))
 (?object= road1)(?rel= *overlaps*)

◇

Compound queries can be nested. The following examples demonstrate how compound queries may be nested inside each other.

Query15: What objects are

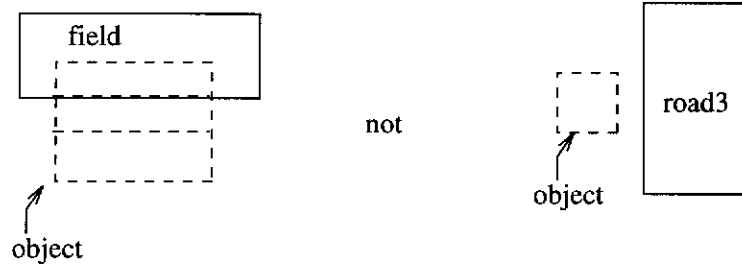


Figure 4.7: The possible position of *object* given the two constraints described by Query 15.

during 1><2 along the x-axis to the object *field*,

the begin of the object follows the begin of object *field* along the y-axis

but not

disjoint 1>2 along the x-axis

during 1><2 along the y-axis

to the object *road3* (see Figure 4.7)?

Result 15: (*overlaps*, road1), (*overlaps*, salinity area)

P: (*and field during-?rel* 1><2-1<2 ?*object* (*not road3 disjoint-during* 1>2-1><2 ?*object*))

P: ((?*rel*= *overlaps*)(?*object*= road1))

((?*rel*= *overlaps*)(?*object*= salinity area))

◇

Query16: What are the objects that are either

during 1><2 along the x-axis

disjoint 1<2 along the y-axis

to the object *salinity area* and

disjoint 1>2 along the x-axis

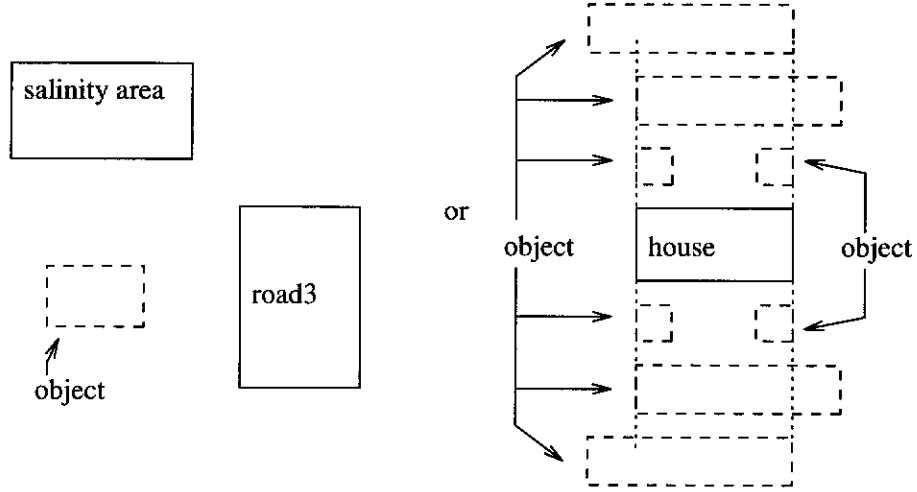


Figure 4.8: The possible positions of *object* as described in the query (Query 16).

during $1 > 2$ along the y-axis

to the object *road3*

or has the relation

the A-Relation *end* along the x-axis

the A-Relation *disjoint* along the y-axis

to the object *house* (see Figure 4.8)?

Result 16: $(1 > 2, 1 < 2 \text{ tree1}), (1 < 2, 1 < 2, \text{tree2}), \text{hedge}.$

P: (or (and salinity area during-disjoint $1 > 2 - 1 < 2$?object road3 disjoint-during $1 > 2 - 1 > 2$?object) house ends-disjoint ?or-?or2 ?object)

P: ((?object= tree1)(?or= $1 < 2$)(?or2= $1 < 2$))
 ((?object= tree2)(?or= $1 < 2$)(?or2= $1 < 2$))
 ((?object= hedge))

◇

4.4 An Application

To demonstrate how the proposed multi-dimensional spatial reasoning system can be applied, we present in this section an application from the area of Geographic

Information Systems (GIS). GIS data is characteristically large (hundreds to millions of objects) and the position and shape of every object in the dataset is precisely known. The spatial information is usually in the form of complex quantitative shapes but many systems also provide information about the bounding boxes of objects (the extreme x and y coordinates of the objects) to assist in accessing relational information about objects.

When designing a system for representing this GIS data, the types of information to be represented and the types of queries to be performed must be considered. The particular types of problems that are the focus of this work are when the information is quantitative but the types of queries are qualitative.

An example query: Show all the salinated regions that are on private land?

The information about salination and land ownership are common aspatial information that may be associated with a dataset. The spatial component of the query is much more ambiguous. What is the exact spatial relationship specified by the relation “on”?

Using the system described in Chapter 3, a qualitative query may be constructed that will accurately describe a particular relationship. It also provides a method of grouping queries according to the classifications. A large vocabulary of queries may be constructed using the querying system described in Section 3.4.

An example real dataset and the types of queries that may be performed on that dataset are presented as follows.

4.4.1 The dataset

The US Department of the Interior uses a file format called GIRAS to represent some GIS datasets. We shall use a dataset¹ in this format (Figure 4.9). The GIRAS data format provides two pieces of information of interest:

- objects (regions) described by polygons, and
- the bounding boxes of these objects.

The information is quantitative in nature and thus all possible pairwise re-

¹Available from <http://edcwww.cr.usgs.gov/doc/edchome/ndcddb/ndcddb.html>.

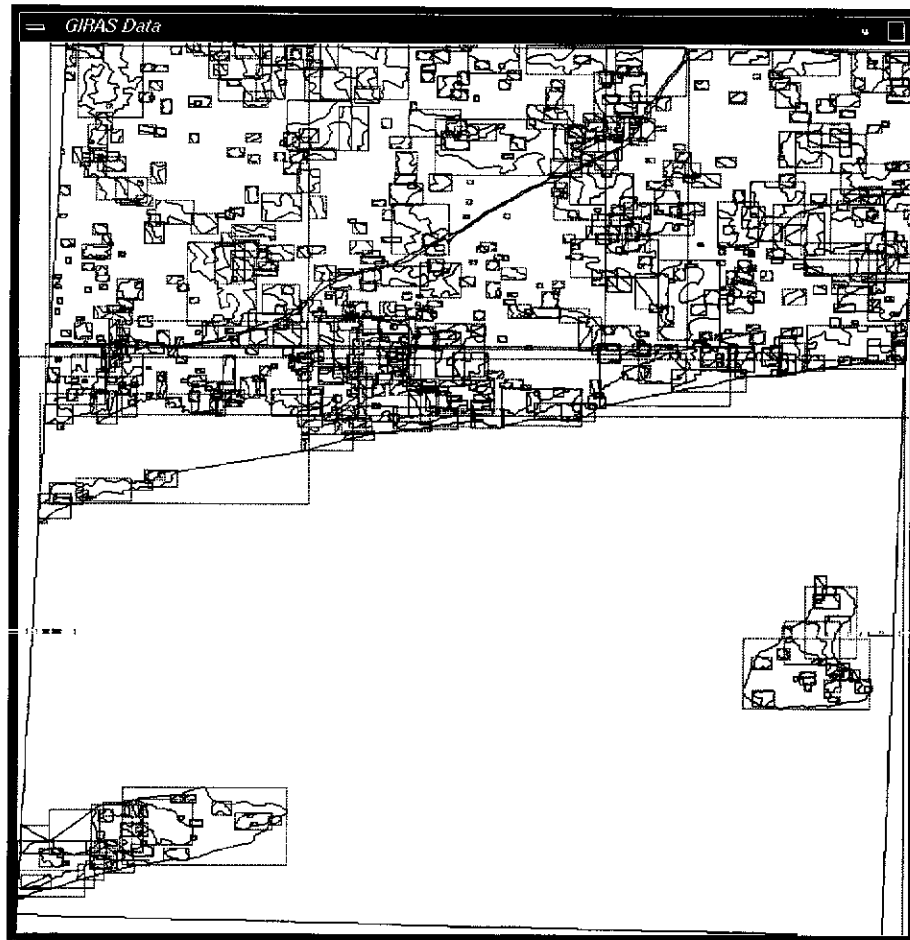


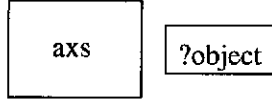
Figure 4.9: GIRAS dataset from the US Department of the Interior: Land usage in area of WILMINGTON, DE NJ PA.

lations between objects or bounding boxes are known. We shall remove the quantitative information and represent only the bounding boxes of the objects as a set of intervals, represented in terms of linearly ordered end points. Each object in the dataset has been assigned a unique name to discriminate between different objects.

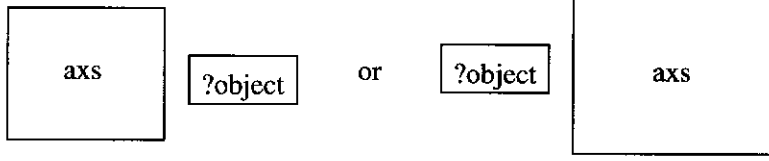
4.4.2 GIS Queries

The amount data represented by the dataset is considerable. Looking at Figure 4.9 it is difficult to determine where one object ends and another starts. If all the labels assigned to each object are included in the figure, it would quickly become unintelligible. The options are to increase the size of the picture or provide some sort of referencing system.

i) $axs \text{ disjoint-during } 1 < 2-1 > 2 \text{ ?object}$



ii) $axs \text{ disjoint-during } ??a-1 > 2 \text{ ?object}$



iii) (and $axs \text{ disjoint-during } ??a-1 > 2 \text{ ?1 ?1 during-disjoint } 1 > 2-1 < 2 \text{ ?object}$)

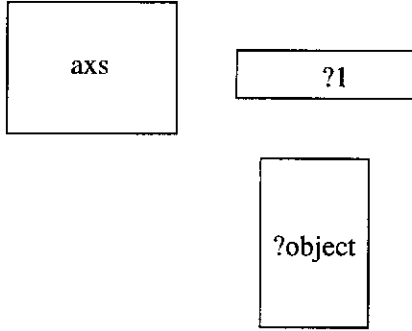


Figure 4.10: Pictorial representations of sample queries

This type of problem reflects the difficulties inherent in large complex sets of data. This section shows how the querying system allows information to be extracted from the complex dataset (such as depicted in Figure 4.9). Based upon one object in the dataset (*axs*), information is extracted about objects with respect to the object *axs*.

The information about objects is represented in terms of bounding boxes which can be represented as intervals (along 2 dimensions) which, in turn, may be represented as a linear ordering of end points of intervals.

The following examples demonstrate the usage of the spatial querying system on the sample dataset.

Example 4.1 *Queries on the dataset*

i) $axs \text{ disjoint-during } 1 < 2-1 > 2 \text{ ?object}$ (depicted in Figure 4.10i)

x-axis: $axs \text{ disjoint } 1 < 2 \text{ ?object} \Rightarrow e(axs) < b(?object)$

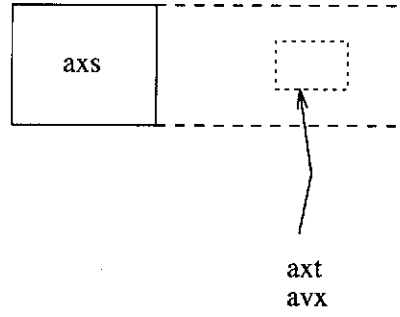


Figure 4.11: A relational depiction of the objects *avx* and *axt* to the object *axs*. The 2D relation is disjoint-during $1 < 2-1 > 2$.

y-axis: $\text{axs during } 1 > 2 \text{ ?object} \Rightarrow b(\text{axs}) < b(\text{?object}) \text{ and } e(\text{axs}) > e(\text{?object})$

By binding the results of a search for intervals whose begin points are greater than $e(\text{axs})$ on the x-axis to ?object, a simple check can be made to see if any of the intervals bound to ?object satisfy the y-axis checks.

Answer: (avx, axt)

In this section we shall construct a figure that illustrates the relations queried from the dataset. Each query will indicate the increase in knowledge about the relations between objects. Figure 4.11 depicts the relationship of the objects *avx* and *axt* to the object *axs*.

◇

ii) *axs disjoint-during ??a-1 > 2 ?object* (depicted in Figure 4.10ii)

x-axis: $\text{axs disjoint } ??a \text{ ?object} \Rightarrow e(\text{axs}) < b(\text{?object}) \text{ or } b(\text{axs}) > e(\text{?object})$

y-axis: $\text{axs during } 1 > 2 \text{ ?object} \Rightarrow b(\text{axs}) < b(\text{?object}) \text{ and } e(\text{axs}) > e(\text{?object})$

Binding all the results of the search for intervals whose begin points are greater than the point $e(\text{axs})$ or whose end points are less than the point $b(\text{axs})$ to ?object, a simple check can be made to see if any of these intervals satisfies the y-axis conditions.

Answer: (awc, axr, awd, axq, axp, axo, axl, bdc, axm, awg, axj, axh, awi, axg, axf, awj, axd, awk, axb, axa, awl, awm, awz, awn, awy, awv, awo, awr, awq, awu, awp, avb, aws, avx, axt)

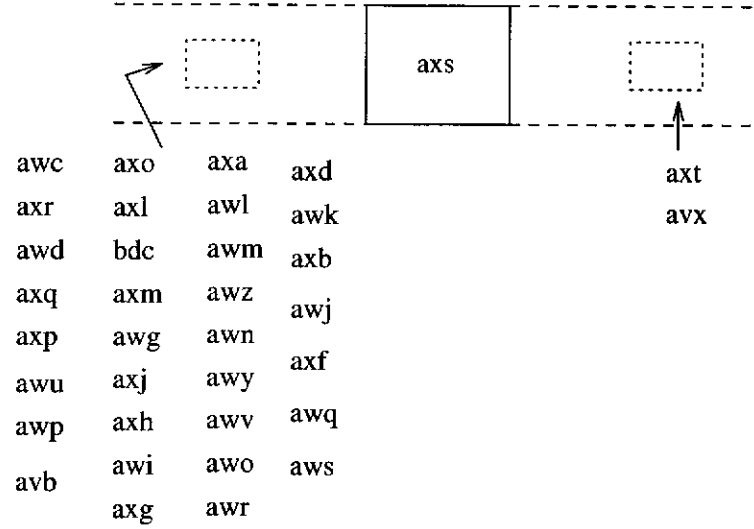


Figure 4.12: A relational depiction of the objects that satisfy the 2D query disjoint-during ?or-1><2.

Figure 4.12 shows the distribution of the objects given that we have already established the relationship between *axs* and the objects *avx* and *axt*.

◇

iii) (and *axs* disjoint-during ??a-1><2 ?1 ?1 during-disjoint 1><2-1<2 ?object)
(depicted in Figure 4.10iii)

x-axis: *axs* disjoint ??a ?1 $\Rightarrow e(axs) < b(?1)$ or $b(axs) > e(?1)$

y-axis: *axs* during 1>< 2 ?1 $\Rightarrow b(axs) < b(?1)$ and $e(axs) > e(?1)$

Binding all the results of the search for intervals whose begin points are greater than the point $e(axs)$ or whose end points are less than the point $b(axs)$ to ?1, a simple check can be made to see if any of these intervals satisfies the y-axis conditions.

and

x-axis: ?1 during 1>< 2 ?object $\Rightarrow b(?1) < b(?object)$ and $e(?1) > b(?object)$

y-axis: ?1 disjoint 1<2 ?object $\Rightarrow e(?1) < e(?object)$

The bindings from the previous query are continued and these are checked against the x-axis and y-axis conditions.

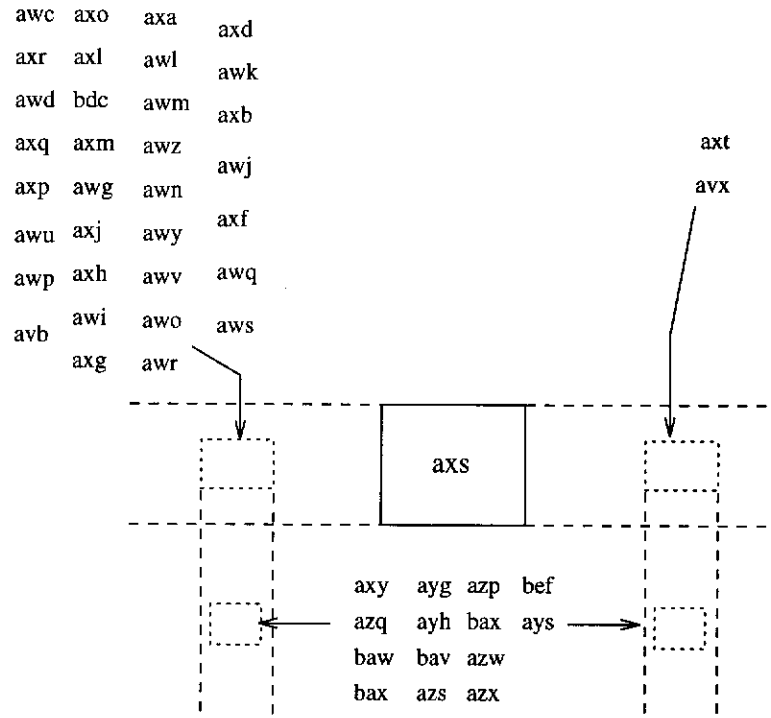


Figure 4.13: All the objects that are *during-disjoint* $1 > < 2 - 1 < 2$ to the objects to which *axs* is *disjoint-during* $??a-1 > < 2$.

Answer: (axy, azq, baw, bax, ayg, ayh, bav, azs, azp, bax, azw, azx, bef, ays, awu)

Using this new information we can now illustrate the position of the objects in Figure 4.13. This figure depicts the relative position of objects, not the actual positions. Indeed we do not know the exact relation of the objects but we do know what possible relationships they may have.

◇

iv) *axi ?relx-?rely ?orx-?ory axs*

x-axis: $axi ?relx ?orx axs \Rightarrow b(axi) ? b(axs), e(axi) ? e(axs), b(axi) ? e(axs), b(axs) ? e(axi).$

y-axis: $axi ?rely ?ory axs \Rightarrow b(axi) ? b(axs), e(axi) ? e(axs), b(axi) ? e(axs), b(axs) ? e(axi).$

Following the procedure given in Section 4.1.2 specific queries about point relations can be used to determine the interval relation.

What is the 2-D NS-Relationship between objects *axi* and *axs*?

Answer: (?relx= *disjoint*)(?orx= 1<2)(?ory= 1><2)(?rely= *during*)

◇

v) (and *axi* ?1-?2 ?3-?4 *axs* ?object ?1-?2 ?3-?4 *axs*)

x-axis: *axi* ?1 ?3 *axs* \Rightarrow b(*axi*) ? b(*axs*), e(*axi*) ? e(*axs*), b(*axi*) ? e(*axs*), b(*axs*)
? e(*axi*).

y-axis: *axi* ?2 ?4 *axs* \Rightarrow b(*axi*) ? b(*axs*), e(*axi*) ? e(*axs*), b(*axi*) ? e(*axs*), b(*axs*)
? e(*axi*).

See the above example and bind the solutions to the appropriate variables.

and

x-axis: ?object ?1 ?3 *axs*

y-axis: ?object ?2 ?4 *axs*

Given the information about the A-Relations, Orientations and key point relations for specific interval relations described in Section 4.1.1, bindings can be made to the intervals that have the same relationship.

What objects have the same relationships to object *axs* as does object *axi*?

Answer: (*axi*, *aik*, *aqm*)

Figure 4.14 shows the objects which have the same relationship to the object *axs* as the object *axi*. No relationship between the objects *axi* (or other objects of the same group) and *awc* (or other objects of the same group) can be derived from this pictorial representation except with respect to their relationship to the object *axs*. That is, the object *axi* is *during* 1><2 along the y-axis to the object *awc*. There is no horizontal information except that both are *disjoint* 1<2 to *axs*.

◇

From these queries, the information about several objects' relationships to the object *axs* have been established. More importantly, we can determine that they are the only objects that have those relationships to *axs* (as represented in the dataset). This method of querying provides a systematic and accurate method of determining the relationships between objects in complex datasets.

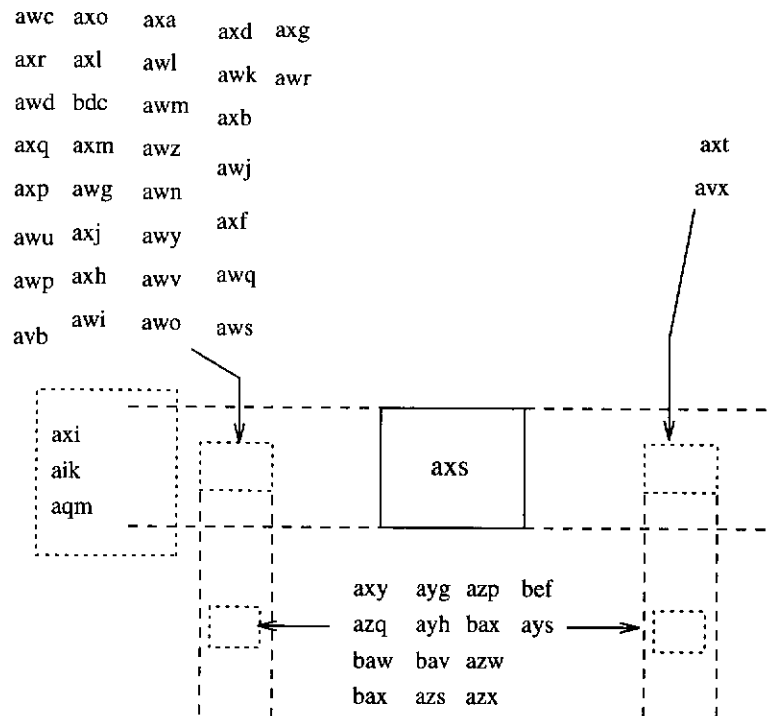


Figure 4.14: All the information about the objects that has been acquired by five different queries.

4.5 Conclusion

For a simple problem, this chapter has demonstrated the entire process of representing information about objects and their spatial relations in terms of intervals. These intervals are then represented in terms of a linear ordering of the end points of these intervals. This can only be done for one type of dataset.

Examples of the methods for searching this representation using the spatial queries of the previous chapter were given. This process involved breaking down the interval queries into point queries and then generating interval solutions from the results of these point queries. This process was performed on an artificially generated dataset and a real GIS dataset.

The representation of intervals presented in this chapter (linear ordering) is only appropriate for one simple type of representation problem. The next two chapters (Chapters 5 and 6) describe more complex types of representations and the methods introduced to represent them.

Chapter 5

Primitive Interval

Representations where some relations are unknown

The concept of progress acts as a protective mechanism to shield us from the terrors of the future.

'Collected Sayings of Muad'Dib' by the Princess Irulan

In this chapter we expand the point-based system of representing intervals to encompass the types of interval datasets where information about every pair of intervals is not explicitly known but whenever interval relations are known they are only primitive interval relations.

For a system performing closure on such a dataset, the construction process would involve propagating information about interval relations that usually leads to the creation of disjunctive interval relations. These disjunctive interval relations are then used as a part of the construction process. Thus this type of problem is usually grouped together with the problem of representing disjunctive datasets.

For our non-closure based representation, this type of dataset provides an interesting problem and we are able to provide an efficient method of representing this type of information.

The system of linearly ordering points presented in the previous chapter is

not applicable as the relationship between every pair of end points is not explicitly provided. Thus a graph-based representation will be used to represent the point information. To minimise the cost of construction (space) and search time, a method of minimisation will be used that requires low construction time (Section 5.1).

Extracting point relations from this graph is the process of searching from a given point for another point. An efficient method of search is presented in Section 5.2.

The performance of the representation system with respect to a control is presented in Section 5.3. A variety of different types of graphs are randomly generated and a suite of searches performed on each of these graphs.

A discussion of the relevance of the interval representation presented in this chapter is given in Section 5.4 and the conclusions in Section 5.5.

5.1 Graph-based Interval Representation

A primitive IR can be represented in terms of the relationships between end points of the intervals in the dataset. When the relations between every pair of intervals is not known (is not primitive and explicit), the linear ordering method of the previous chapter cannot be used. We shall examine the alternative methods for representing intervals and interval relations as points and point relations in a graph.

5.1.1 Graph construction

The method of graph construction presented in this chapter takes a set of interval relations and incrementally constructs a point-based representation. Every interval relation is converted from its interval representation to an equivalent point-based representation. The information about these point relations is then added to the Point Graph (PG). The fact that the begin point of every interval precedes the end point of the same interval is also added to the graph, i.e. the begin point of interval **A** is less than the end point of the same interval ($a < A$).

There are many alternative methods of modifying the above representation to optimise a particular aspect of the access to or representation of the interval information.

There are three main strategies that are commonly used. Closure minimises the access time to the information at the cost of space and time to construct the representation. Full minimisation reduces the amount of data used to represent the information at the cost of access time to and the construction time of the representation. Simply leaving the data as is, optimises the construction time and results in variable access time and representation space. No single representation is, or can be, optimal in all aspects.

Rather than constructing a graph using consistent data, random values are used for each interval relation added to the graph. Consistency checks are then performed to determine if the new interval relation to be added to the graph is consistent with the information already represented in the graph. Thus the time given for constructing the graph representation also includes the time taken to determine the consistency of every interval relation with respect to the graph before it is added. In order to determine the effectiveness of different methods of representing interval information, a control dataset is created that represents consistent information about interval relations in a graph upon which no modifications have been made.

5.1.1.1 Local minimisation

We propose a system of *local minimisation* that has low, although not minimal, cost for construction, storage space and search. Local minimisation is a variant of full minimisation that provides significant savings with respect to the cost of construction.

The method described in this section reduces the number of edges used to represent the information about interval relations. When a new edge is added to the representation, a check is made to see if any of the edges adjacent to the new edge are redundant. These redundant edges are removed. The result is a minimisation strategy that only checks areas local to the addition of the

new information. Although this strategy does not guarantee the removal of all redundant edges it does remove many of the edges at a much reduced cost when compared to the full minimisation strategy.

Definition 5.1 A path of length x exists between two nodes, n_i and n_j , if there exists exactly x edges that may be used to form a sequence of edges that connect n_i and n_j . i.e. a path of length 3 is $\{n_i n_a, n_a n_b, n_b n_j\}$.

Definition 5.2 Two nodes, n_i and n_j , are adjacent with distance x if the shortest path between n_i and n_j is of length x .

Definition 5.3 A graph $G = (N, \varphi)$ is locally minimised up to “ n ” if, whenever a new edge $\langle n_i, n_j \rangle$ is added, all redundant edges of adjacency with distance up to n are removed.

The system of minimisation described hereafter uses an adjacency of one edge. After adding ab to a partially constructed graph G , the following checks are required:

Given a graph with points **a** and **b** that both precede a third point **n**, adding an edge between **a** and **b** will make one of the current edges redundant (see Figure 5.1i).

$$\text{if } \exists n \in N \mid an, bn \in \varphi \Rightarrow \varphi' = \varphi - (an) \quad (5.1)$$

Given a graph where **a** precedes some point **n** that precedes **b** then the addition of an edge between **a** and **b** will be redundant (see Figure 5.1ii).

$$\text{if } \exists n \in N \mid an, nb \in \varphi \Rightarrow \varphi' = \varphi - (ab) \quad (5.2)$$

Given a graph where points **a** and **b** are both preceded by a third point **n**, adding an edge between **a** and **b** will cause one of the edges to be redundant.

$$\text{if } \exists n \in N \mid na, nb \in \varphi \Rightarrow \varphi' = \varphi - (nb) \quad (\text{see Figure 5.1iii}) \quad (5.3)$$

The algorithm for constructing a locally minimised graph of size 1 is as follows:

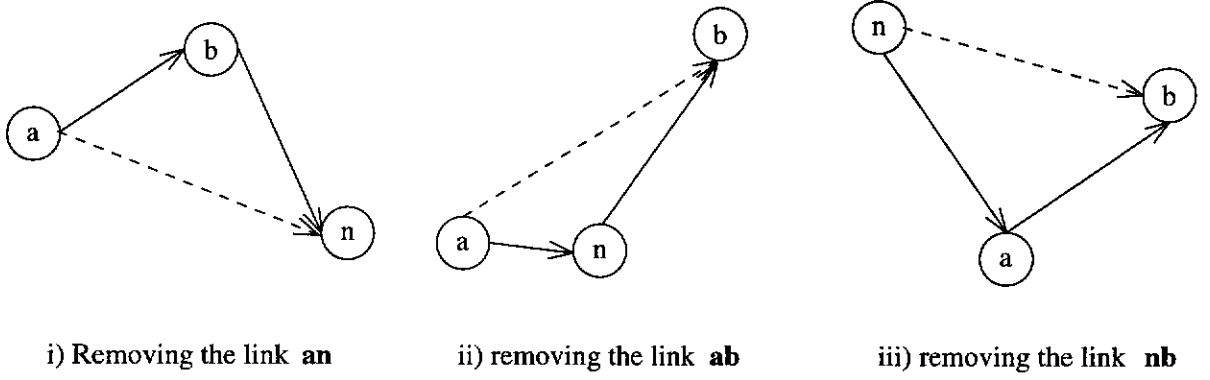


Figure 5.1: Three different situations where the three different local minimisation rules are used when adding the edge $\langle a, b \rangle$.

Given nodes $a, b \in N$ when adding edge ab :

```

if not search( $G, ba$ )
   $\varphi' = \varphi + (ab)$ 
  if  $\exists n \in N \mid an, bn \in \varphi \Rightarrow \varphi' = \varphi - (an)$ 
  if  $\exists n \in N \mid an, nb \in \varphi \Rightarrow \varphi' = \varphi - (ab)$ 
  if  $\exists n \in N \mid na, nb \in \varphi \Rightarrow \varphi' = \varphi - (nb)$ 

```

Algorithm 5.1 *Adding an edge between nodes using the local minimisation rules.*

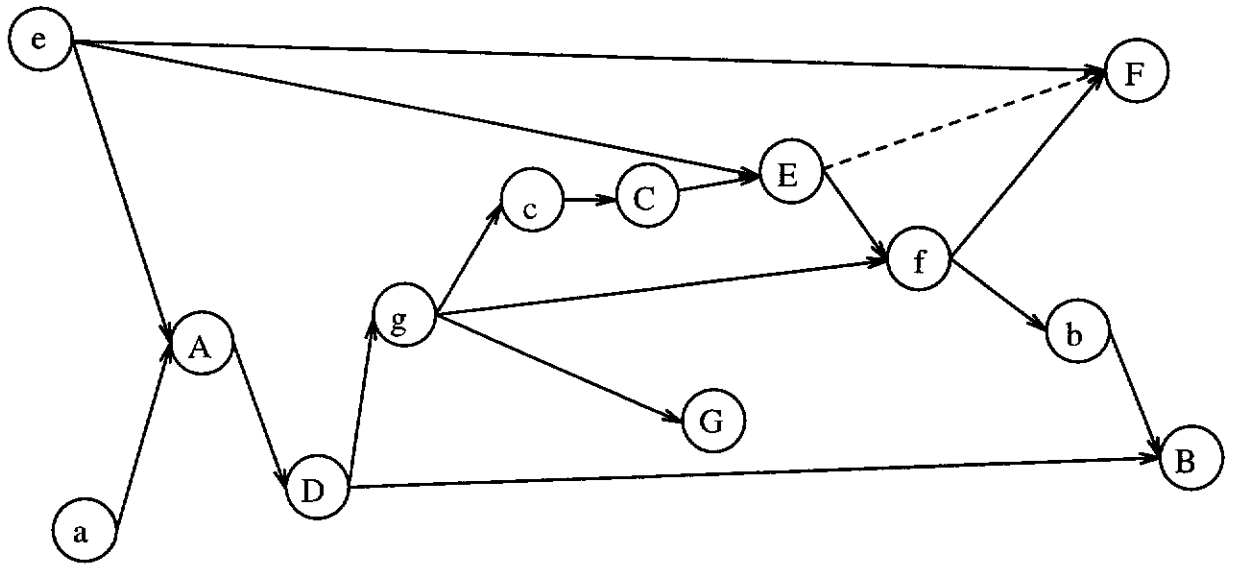
5.1.1.2 An Example

Figure 5.2 helps illustrate the effect of local minimisation on a graph when new edges are added to the graph. A new edge is added after a check of the current graph is performed to see if the new information to be added would contradict the current information in the graph. If no contradiction occurs, then the edge is added and the local minimisation rules are applied.

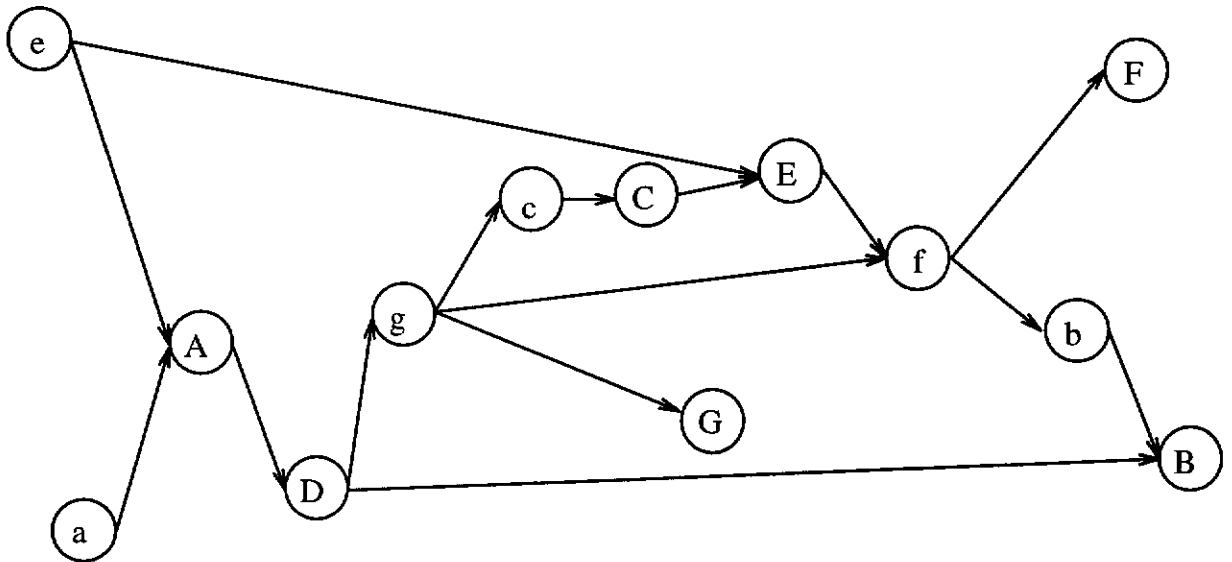
In Figure 5.2i the addition of the edge EF causes two changes to be made to the graph. Firstly the connection from e to F is removed as it is redundant (Equation 5.3) and secondly, the actual edge EF is removed (Equation 5.2) as it is also redundant. The final graph is shown in Figure 5.2ii.

5.1.1.3 Analysis

Local minimisation does not eliminate a large number of the connections between the nodes. The number of edges removed due to local minimisation is dependent



i) before



ii) after

Figure 5.2: Addition of extra information using local minimisation.

upon the density of the graph (the number of edges). A locally minimised graph normally contains redundant information i.e. it is not fully minimised.

The search for relations between nodes in a locally minimised graph differs from the search algorithms used to search a fully minimised graph. Search algorithms for fully minimised graphs can take advantage of the lack of redundant edges in optimising their search. Locally minimised graphs cannot rely upon this assumption as they may contain redundant edges. The presence of redundant

edges will result in an increase in the number of possible search paths a search algorithm must consider and thus the cost of the search would increase. Heuristics may be used in the search process to reduce the cost of searching a graph that contains redundant edges.

The problem of searching a graph with redundant edges is handled by a similar mechanism for pruning the search space in a fully minimised graph. A fully minimised graph will have search convergences (multiple paths between two points) and a search that keeps track of the nodes that have been previously searched will reduce the number of possible search paths. A locally minimised graph will have an increased number of search convergences compared to the equivalent fully minimised graph and thus efficient handling of the convergence problem is more significant.

A positive side effect of the increased number of redundant edges between nodes is that, on average, the search will progress faster through the graph (by skipping local areas). Figure 5.3 demonstrates an example where the search of the locally minimised representation would be faster than the search of the equivalent fully minimised representation. A fully minimised graph would construct a path of length 5 between a and f whereas a locally minimised graph (as in this example) requires a path of length 2. The worst case search situation of local minimisation is equivalent to full minimisation. The issues of density and connectivity will be discussed further in Section 5.3.

5.1.1.4 Extending minimisation

Local minimisation is one method of organising the information about the relationships between points in the given graph. The choice of an adjacency value of 1 is only one possible method of local minimisation. For certain applications it may be practical to spend more time on construction (adjacency > 1) to minimise space. The cost of increasing the adjacency from 1 to 2 is significant. Given that there are m edges for every node then the cost of maintaining an adjacency of 1 is $(m \times m - 1)$ checks per addition, and for adjacency 2 it is $(m \times m - 1) \times (m - 1 \times m - 1)$ checks per addition. The adjacency 2 check is the determination

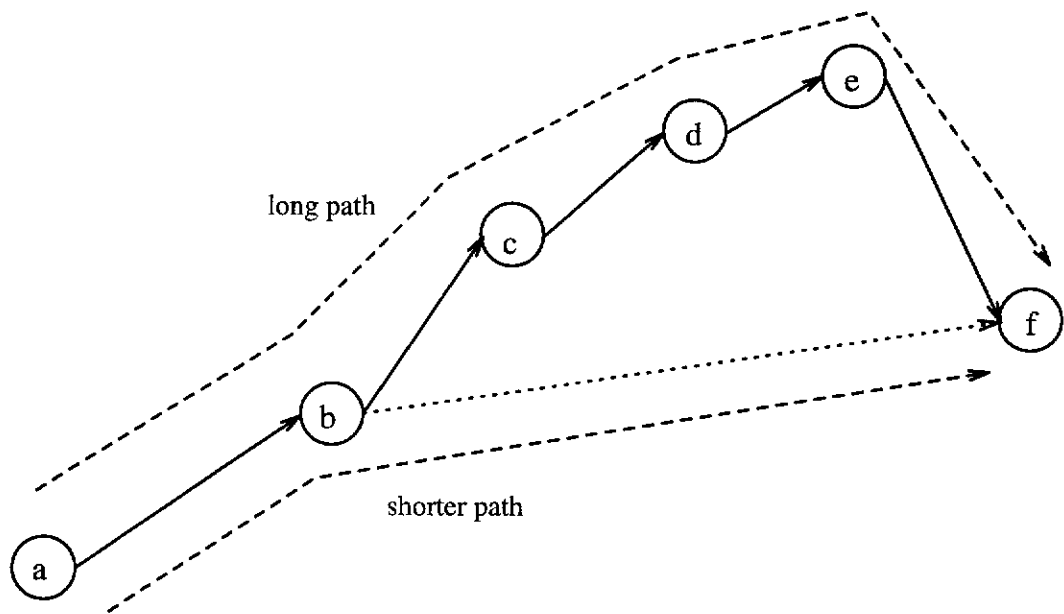
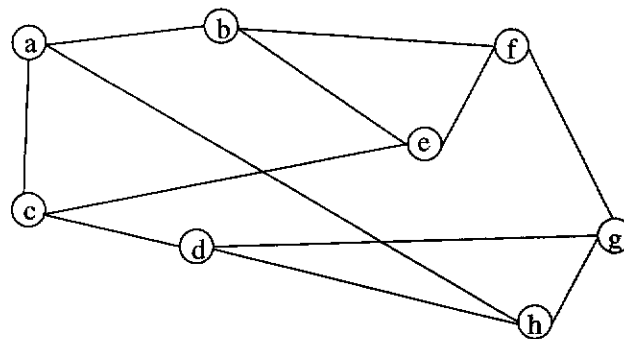


Figure 5.3: A search for the relationship *af* on a locally minimised graph results in a search that follows the path *abf* whereas a fully minimised graph (no *bf*) would require the search to use the path *abcdef*.

i)



ii)

abf, bfe, hgf, bfg
 abc, acd, cdg, bec
 ahg, hdg, cdh, adh
 ace, cef, ecd, cdh

Figure 5.4: For the graph given in i) all the triples in which two of the components are a distance of two or less links from “a” are listed in ii).

of all the triples that are two links or less from “a” (see Figure 5.4).

Full minimisation (or transitive reduction, see [1]) is equivalent to a local minimisation with an adjacency of a number greater than the maximum path

length between any two nodes. Thus full minimisation can be considered to be an extreme type of local minimisation. The benefit of a system of local minimisation with a low adjacency value over a system with a higher adjacency value is in time of construction.

The construction time increases as the adjacency increases. The effect of increasing the adjacency on search will be examined in the next section.

5.1.1.5 Construction costs

The construction time of a graph is dependent upon two factors: the time taken to verify if the relation is already represented in the graph (constant time), and the time taken to search if the inverse relation is true (one search). If neither is true then a new edge can be added in constant time.

Once the new link has been added then local minimisation can be performed. The three minimisation rules (Section 5.1.1.1) are applied with respect to the adjacent nodes. The time taken to perform this is dependent upon the number of nodes adjacent to the given two nodes. The number of adjacent nodes is dependent upon the *density* of edges between the nodes and the size of the adjacency. See Section 5.3 for an empirical evaluation.

The amount of space taken to represent the information using local minimisation falls between the actual number of edges to be represented and the minimum number needed to represent the information. The exact performance of the system will be evaluated in Section 5.3. The next section will examine the search process.

5.2 Search

The representation of intervals and their relations described in this chapter provides a point-based representation of this information. Thus the search for intervals and their relations can be translated into an equivalent search for points and point relations.

5.2.1 Point search

The search for (or query of) the relation between two points is a process of determining if there is a path between the two points in the graph representation. We provide algorithms for determining if there is a path between two points.

The most primitive operation is the search for a target point (destination node) from a given start point (starting node) in the graph in a given *direction*. The *direction* indicates one of two possible search strategies:

- < - searching for a target point that precedes the given start point; and
- > - searching for a target point that is preceded by the given start point.

Thus from a *start point* a search is performed for a *target point* in a given *direction*. A version of a breadth first search is used, and the algorithm is outlined as follows:

```
pointsearch(Start_point, Direction, Target_point)
begin
  current_list = Start point
  while not (Target_point ∈ current_list)
    new list = next(current_list, Direction)
    current list = new list
  end
```

Algorithm 5.2 *Searching for a Target point in the graph.*

The function *next* returns the points adjacent (size 1) to the points that are members of the current list in the given *direction*. A search for a particular point in a given *direction* from a start point proceeds by creating a list of current points (initially the start point) and searching from the points in the list of current points in the *direction* indicated. A new point is added if it is linked in the given *direction* from a point in the current list. Points are removed from the current list when every link in the given direction, adjacent to the point, has been added to the current list. The search continues until the target point is located (member of the current list) or the current list is empty (not found).

Using this primitive search algorithm any point relation can be found if there exists a path between the start point and the target point in the given direction. If the search for a target point (a) in a given direction ($<$) from a start point (b) is successful then a specific relation is true ($a < b$). Determining the point relation between two points involves searching the graph to see if any of the three point relations are true $\{a < b \text{ or } a > b \text{ or } a = b\}$ (see Algorithm 5.3).

```

Given  $a, b \in N$ 
if  $a = b$ 
    return  $a = b$ 
else
    if (pointsearch( $a, <, b$ ))
        return  $a > b$ 
    else
        if (pointsearch( $a, >, b$ ))
            return  $a < b$ 
        else
            return unknown

```

Algorithm 5.3 *Establishing the relation between two points “a” and “b”.*

5.2.2 Costs

The search results depend on the dataset, that is, the number of nodes, the number of edges and the order in which the information is added. There are two extreme situations that occur in *full minimisation*:

1. A situation where $n-2$ nodes are ordered with respect to 2 nodes but not with respect to each other (see Figure 5.5) is the highest number of edges situation, where the number of edges is $(n - 2) \times 2$ and the search time is constant (maximum 2 edges away).
2. The lowest number of edges situation is a linear ordering of points. There are $n-1$ edges in the system and the search time is: $(n - 1)$ checks performed for an unsuccessful search and $n/2$ (on average) for a successful search.

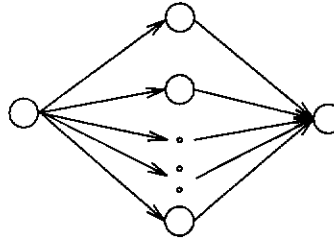


Figure 5.5: A fully minimised graph where the number of edges is a maximum.

There are two extrema for a system of *local minimisation* with an adjacency of 1:

1. Highest number of edges. Imagine a system where random edges are added to the system and the local minimisation rules are never needed. This system continues to have edges added until no more edges may be added without the local minimisation rules being used. This situation is the highest number of edges case for local minimisation of adjacency of 1. The number of edges is $n^2/4$. The search time is, at most, the time taken to search 2 edges (each node is at most two edges away from any other node).
2. Linear ordering (see above).

The relationship between the number of edges and the search time is not linear. There is a worst case situation for the search where the number of edges is high, but the level of connectivity is low [33, 63]. Thus when considering the costs of search, the data being represented can have significant and complex influence. The most expensive component of local minimisation is the search time. The number of edges and nodes determine the search time. Section 5.3 will demonstrate the effect of the number of edges and nodes on the search.

5.2.3 Search improvements

The presence of redundant edges increases the possible number of search paths in the representation and thus the cost of performing search on the representation. The very nature of the data being represented (intervals) and the way in which the data is represented (points) provides opportunities for search efficiencies.

5.2.3.1 History

The most obvious cost saving for breadth first search is removing the extra searches due to different search paths converging to the same point. The simplest method to keep track of the search progression is to keep track of the nodes encountered as the search progresses.

We represent the nodes in a graph as the set N . Initially all members of the set N are also members of the set of unmarked nodes UM . As the search progresses, the encountered nodes are transferred from the set UM to the set M . At all times the union of the set of unmarked nodes UM and the set of marked nodes M is equal to the set of nodes N and $M \cap UM = \emptyset$. If a search branch reaches a node that is a member of the set of marked nodes M then the search branch is discontinued.

Set membership can be efficiently determined by adding an extra boolean field to the definition of a node: $\langle \text{name}, \text{searched?} \rangle$. At the beginning of a search all nodes have their “searched?” field set to false. As the search progresses nodes encountered during the search have their “searched?” field set to true.

5.2.3.2 Begins and ends

The points being represented are end points of intervals. This fact can be used to reduce the search. The search for a path between two interval points may be satisfied or terminated upon encountering the companion point for that interval. For instance, since it is always true that the begin of an interval precedes the end of the same interval, if the search in direction $<$ for a *begin point* of an interval encounters the *end point* of that interval then the search is satisfied because *begin point* $<$ *end point*.

5.2.4 Interval search

A search for an interval relation can be divided into point searches. Searching for an interval relation is equivalent to searching for each of the point relations in the conjunction that represents the interval relation. The search improvements of Section 4.1 can be applied to reduce the cost of this search.

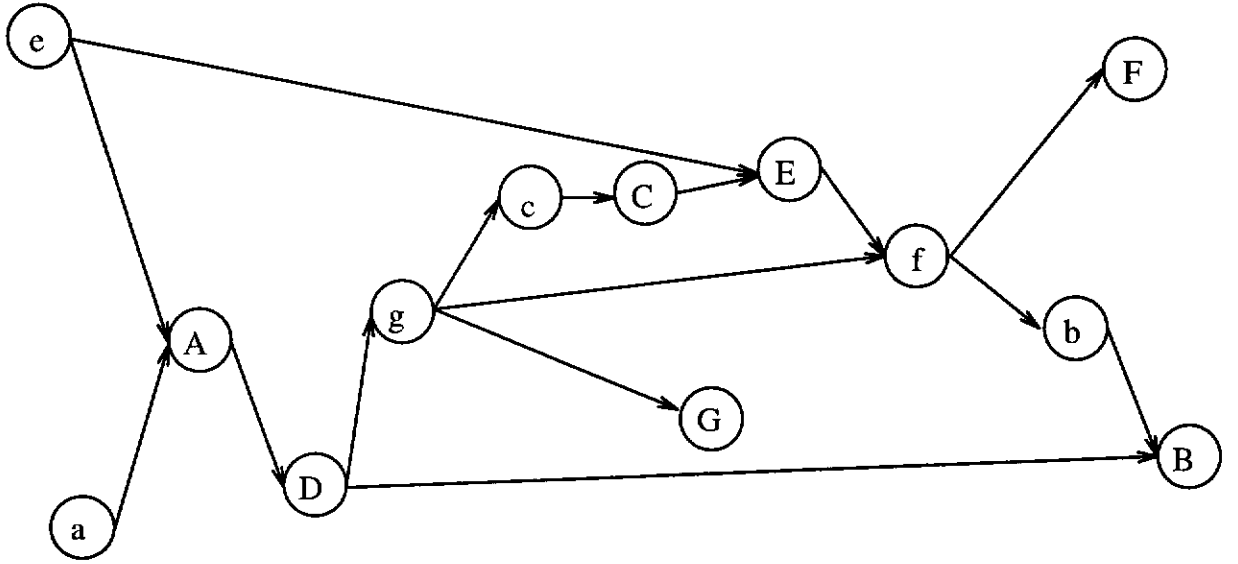


Figure 5.6: The final graph from Figure 5.2.

Let us examine the process of determining the relationship between intervals **B** and **A** in Figure 5.6.

The first step is to consider the relationships between the four significant point relations between **B** and **A**: ba , bA , aB and BA . Searching the graph produces the relations shown in Table 5.1 and the corresponding diagrammatic representation is shown in Figure 5.7. Note, that due to the factors discussed in Section 4.1.2, only three of the point relations need to be checked to determine this interval relation.

Let us now examine the relationship between intervals **C** and **G** in Figure 5.6. Table 5.2 contains the relation between the end points of **C** and **G**.

The relations between cG and CG are not defined in the graph. Thus, there is ambiguity and there are five possible relationships between the intervals **C** and **G** (see Table 5.2).

Table 5.1: The results of the four point queries and the deductive result between intervals **B and **A**.**

ba	bA	aB	BA	<i>Result</i>
$>$	$>$	$<$	$>$	disjoint 1 $>$ 2



Figure 5.7: The relationship between intervals A and B.

Table 5.2: Point relation results for intervals C and G. Shows the intervals that match.

<i>cg</i>	<i>cG</i>	<i>gC</i>	<i>CG</i>	<i>Results</i>
>	*	<	*	disjoint 1>2 meets 1>2 overlaps 1>2 during 2><1 ends 2<1

5.3 Test Results

In this section we shall demonstrate the performance of the local minimisation system. The three components of the evaluation are the random generation of the datasets, the control algorithm and the statistical methods for evaluating algorithm performance.

The local minimisation method is not restricted to particular types of datasets. Thus all datasets for evaluating the algorithms are generated using random techniques. The method for constructing a graph is as follows:

Given a number of intervals (*size*), the end points of these intervals become nodes (*N*) of a graph (*G*). Given a number of links to be added to each node (*initial density*), each link is added if the link is consistent with the current graph. This consistency is determined by checking to see if the converse of the relation represented by the link to be added is true (i.e. for <, the converse is >, =). This check is performed by searching the graph to see if these relations are true.

Using this method a control representation can be constructed. This control represents the basic method of constructing a consistent graph and shall be used to compare other methods with respect to the construction, the representation space and the searching of the graph.

The local minimisation algorithm differs from the above method of constructing a graph by only one factor. After a new edge is added to the graph, a check is made to see if the local minimisation rules come into affect. Edges are removed from the graph according to the local minimisation rules.

This results section will focus upon providing a comparison between the control and local minimisation methods of constructing a graph.

Statistical analysis of the performance of these two methods requires that randomised datasets be generated for a variety of different initial densities for a number of different sized datasets. For each initial density of each different sized graph, thirty datasets are generated.

The performance of each algorithm is judged by the time taken for construction, the time to search for point relations in the graph and the number of relations needed to represent the graph. The search time is the average time it takes to determine the truth value of a point relation in the graph for 1000 different searches. An individual search consists of randomly choosing two points and then searching to see if a given relation is true for those two points. The averages for construction time, search time and the number of edges from the thirty datasets represent a statistical evaluation of the algorithm's performance.

A local minimisation algorithm will use an adjacency of 1.

5.3.1 Data

The following results were obtained using a SG R4400 processor using a program written in C++.

The results have been generated for the control and local minimisation for four different sized graphs, with a variety of different initial densities.

The four data sizes are 100 intervals (200 points) given in Table 5.3, 200 intervals (400 points) given in Table 5.4, 300 intervals (600 points) given in Table 5.5 and 400 intervals (800 points) given in Table 5.6. In each case averages (of thirty datasets) are given for construction time, number of edges and search time.

Table 5.3: Results for data size 200 (100 intervals).

	Control			Local Minimisation		
initial density	construction time (secs)	number of edges	search time (secs)	construction time (secs)	number of edges	search time (secs)
10	8	2135	3.5	8	1217	3.1
20	18	3424	4.2	18	1295	3.2
30	29	4457	4.1	27	1307	3.5
40	34	5097	4.4	37	1325	3.4
50	41	5789	4.0	47	1325	3.4
60	49	6354	4.0	55	1323	3.6
70	56	7054	3.8	66	1322	3.4
80	63	7742	3.6	76	1292	3.5
90	71	8351	3.5	87	1281	3.5
100	71	8609	3.4	96	1285	3.6
110	75	8986	3.3	108	1219	3.6
120	80	9245	3.3	115	1290	3.5
130	86	9934	3.1	127	1255	3.6
140	89	9961	3.2	137	1229	3.5
150	89	10368	3.1	144	1211	3.4

5.3.2 Analysis

The analysis of the performance of the local minimisation against the control algorithm is made by comparing the construction time, search time and the number of edges of each algorithm over a number of different sized graphs and for a number of different initial densities. Such an analysis follows.

5.3.2.1 Construction

The construction time is the time taken to construct a graph of a given size with a given initial density. The process of construction begins by creating a number of nodes and then adding links between these nodes. The addition of a link to the

Table 5.4: Results for data size 400 (200 intervals).

	Control			Local Minimisation		
initial density	construction time (secs)	number of edges	search time (secs)	construction time (secs)	number of edges	search time (secs)
10	53	4569	10.4	54	3166	10.5
30	200	10794	16.6	192	3762	12.0
50	364	15717	16.8	326	3887	12.3
70	484	19252	16.2	454	3962	12.3
90	663	23962	14.8	570	4106	12.3
110	688	25538	14.9	701	4026	12.4
130	873	29926	14.2	877	3740	12.2
150	909	31998	13.2	1007	3592	12.0

Table 5.5: Results for data size 600 (300 intervals).

	Control			Local Minimisation		
initial density	construction time (secs)	number of edges	search time (secs)	construction time (secs)	number of edges	search time (secs)
10	184	7013	23.0	160	5419	20.6
30	831	18106	38.9	723	7222	25.1
50	1694	28356	52.7	1325	7063	27.3
70	2504	37781	37.8	1941	6770	27.8
80	3225	46230	35.9	2551	6471	28.6
90	3910	54198	33.7	3106	6243	28.4
110	4430	60708	35.8	3666	6126	28.0
130	5239	68594	31.0	4323	6037	28.2

graph requires that a check be performed to determine if the new link contradicts the current graph representation. This check is a search of the current graph to see if a relation exists that contradicts the relation to be represented.

Table 5.6: Results for data size 800 (400 intervals).

	Control			Local Minimisation		
initial density	construction time (secs)	number of edges	search time (secs)	construction time (secs)	number of edges	search time (secs)
10	385	9430	39.3	355	7678	35.6
30	1807	24791	75.5	1702	11030	44.3
50	3976	38709	79.8	3275	11060	47.5
70	5942	51961	74.8	4886	10657	48.6
90	8117	64721	75.7	6473	10427	50.2
110	10398	76463	69.0	8053	10045	50.3
130	12133	88552	65.5	9545	9870	51.5

An algorithm may also include the modification of the graph representation. Local minimisation requires that the adjacent nodes be checked for redundant links.

The construction times for both local minimisation and the control are given in Figure 5.8. The y-axis represents the construction time and the x-axis represents the initial density.

Each of the graphs demonstrates that the construction times for both local minimisation and the control are similar. For the graph of size 100, the construction times for the control are consistently below those for local minimisation. For the graph of size 400, the reverse is true.

It would be expected that local minimisation be consistently more costly to construct than the control, considering the extra checks involved for local minimisation. The behaviour of the control and the local minimisation algorithms can be explained in part by the existence of a search process in the construction algorithm. This search performance will change along the patterns indicated in the section on search (Section 5.3.2.3).

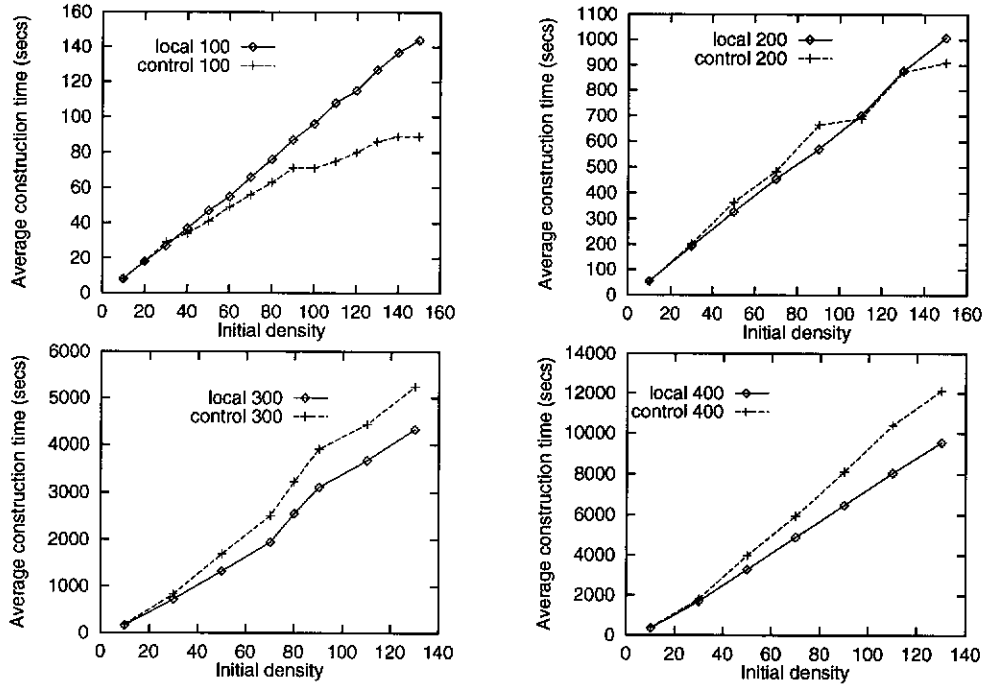


Figure 5.8: The relationship between the construction times of the control and the local minimisation are given for the four datasizes (intervals) at different densities.

5.3.2.2 Number of edges

For the control algorithm, only consistency checking affects the number of edges, whilst for the local minimisation algorithm the reductions caused by the local minimisation rules are also a factor.

Figure 5.9 and Figure 5.10 depicts the relationship between the number of edges in the local minimisation representation against the number of edges in the control. The graph gives the values for four different sized datasets where the y-axis represents the the number of edges in the locally minimised graph and the x-axis represents the number of edges in the control. Each data plot indicates the averages for the control and local minimisation number of edges that are derived from graphs that have the same initial density.

The pattern that emerges for the comparison of the four different size datasets graphed is that there is a peak point for the local minimisation. Preceding that point the gradient is relatively sharp and after the point the gradient decreases more slowly. The explanation for this behaviour is based around the concept of

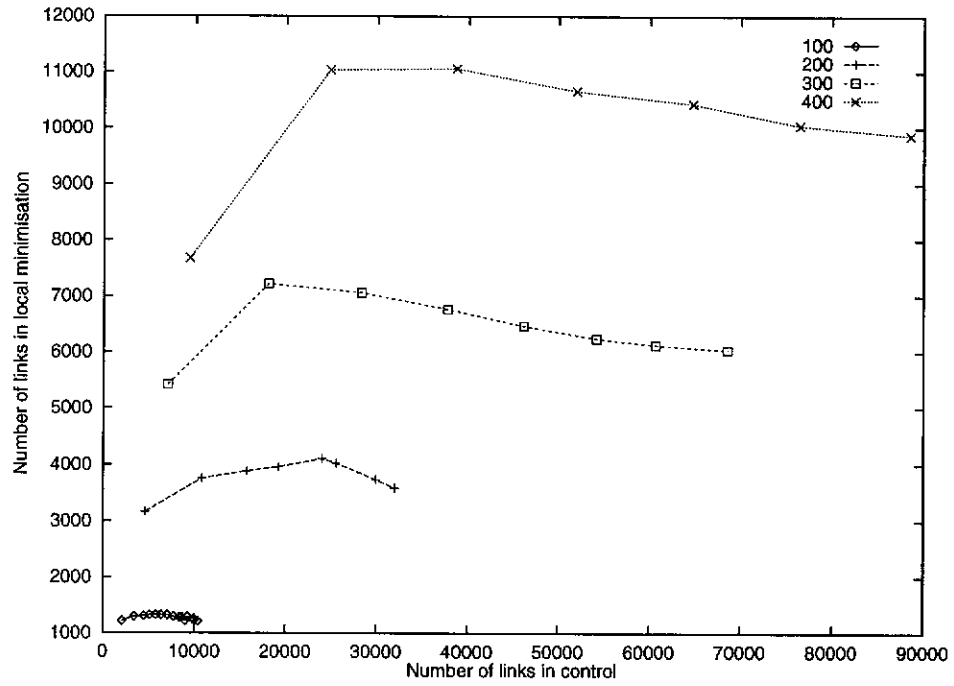


Figure 5.9: The relationship between the number of edges in the local minimisation and control algorithms for four datasizes.

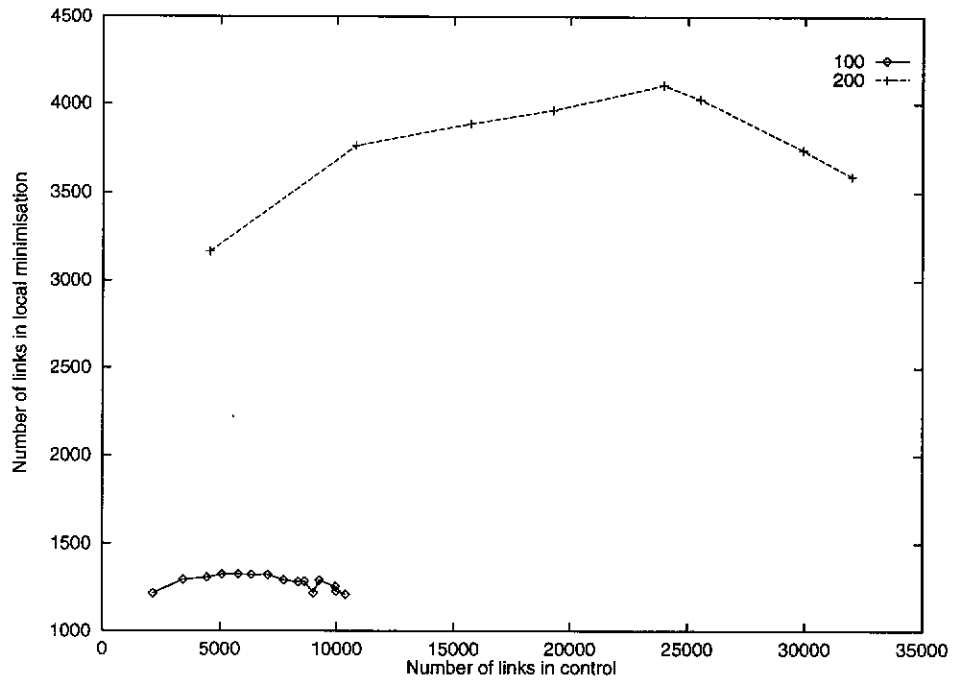


Figure 5.10: The relationship between the number of edges in the local minimisation and control algorithms focusing on size 100 and 200.

the critical density of a dataset. When the number of edges in a dataset reaches a particular value, the local minimisation rules begin to have a substantial affect on the number of edges used to represent the information. The critical value is based upon the number of edges in the graph and the size of the graph.

The local minimisation rules are not used heavily when the number of edges in the graph is low (sparse), but once the number of edges reaches the critical value the rules begin to have a significant effect on the total number of edges present. This effect increases and will continue to increase until it is a fully minimised graph. The issue is further complicated by the effect of connectivity (see the next section).

5.3.2.3 Search times

A search is the time taken to determine the truth of a given relation between two points in a graph. For this evaluation, the search points are selected randomly from the set of nodes of the graph. The relation to be tested between these two points is also randomly selected. The point relationship represented by these two random points and the random relation is checked by searching for that relation in the graph.

For each dataset generated for each initial density for each size, 1000 searches are performed. The average search time for performing 1000 searches for thirty datasets is used to indicate the search performance.

Given an initial density, we compare the search times for the two methods of representing the information (control and local minimisation) for four different data sizes in Figures 5.11, 5.12, 5.13 and 5.14. The initial density is given along the x-axis and the search time is given along the y-axis.

The above mentioned graphs depict the search time with respect to the initial density of the graph. Figure 5.15 depicts the search time against the actual number of edges in the graph. In this graph the behaviour of the search time for the locally minimised graph is consistent with the pattern depicted in the control graph. The locally minimised graph uses a low number of edges to represent information (and the subsequent low search time) whereas the control steadily

increases the number of links.

Of particular interest is the behaviour of the control graph as the initial density increases and thus the number of edges in the graph increases. By examining this behaviour, we can compare the overall performance of the locally minimised graph and the control. It will also provide insights to the patterns indicated in the empirical data.

The relationship between the search time and density (number of edges per node) can be explained in terms of connectivity [33, 63]. The connectivity of a graph is the level of interconnection of the nodes in the graph. A graph with low connectivity is one where the average shortest path between each pair of nodes is high. Conversely, a graph with high connectivity is one where the average shortest path between nodes is low.

The time taken to perform a search is dependent upon two factors. The increase in the density of the graph indicates that there are more paths to follow and thus increases the search time. The increase in connectivity of the graph indicates the length of the path between two points is decreasing. As the length of the path between any two nodes decreases then the time taken to traverse this path will decrease.

The behaviour of a search in a graph will depend upon the connectivity of the graph and the density of the graph. An indication of the behaviour of the search can be determined by examining four different extreme situations that may occur.

Qualitative terms are used to indicate the connectivity and density of the graph. For connectivity, *high* indicates the average size of a path is approaching 1 or 2 links. Whereas *low* indicates the average size of a path is approaching $(n - 1)$ links, where n is the number of nodes in the graph. Similarly, density is *low* when the total number of links in the graph is similar to the number of nodes. Density is *high* when the total number of links in the graph is approaching $(n \times n - 1)$, where n is the number of nodes in the graph.

Three of the four extreme situations are:

- The density is low and the connectivity is low; leading to good search performance as there is low number of paths to follow.

- The density is high and the connectivity is low; leading to poor search performance as there are many *long* paths to follow.
- The density is high and the connectivity is high; leading to good search performance as the many paths but those paths are short.

The final situation where the connectivity is high and the density is low will not often occur as the situation represents an unlikely event, i.e. given the few links present they lead to a highly connected graph.

For each of the four graphs 5.11, 5.12, 5.13 and 5.14 the control follows this pattern. As more edges are added to these graphs their density and connectivity would increase until constant search time would occur (as every node would be linked to every other node, closed).

In contrast, the number of edges in a locally minimised graph decreases as the number of edges added to the graph increases (due to the effect of redundant adjacent edges being removed). The locally minimised graph approaches full minimisation as the number of edges added to the graph continues to increase.

5.3.2.4 Summary

In this chapter, we have introduced the concept of local minimisation as a method for providing an efficient means of representing information about point relations. To demonstrate the effectiveness of our system, the performance of the local minimisation system was compared to a control. In both the control and the local minimisation systems, the costs of determining the consistency of the representation were included in the construction costs.

For a number of different sized and density datasets we have demonstrated (using empirical data) the performance of the locally minimised system. This performance was judged by three criteria: the construction time, the number of edges in the graph and the search time. To determine the local minimisations behaviour with respect to the construction time, the representation space and the search time, a comparison of the performance of the local minimisation system is judged purely in terms of its comparative performance to that of the control.

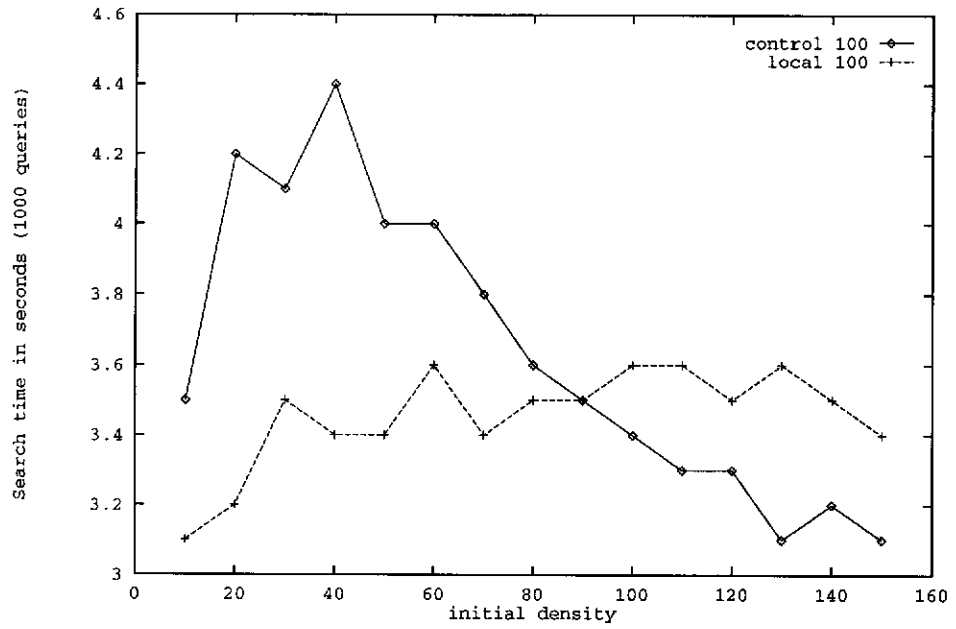


Figure 5.11: The relationship between the search times for local minimisation and the control are projected against the initial density for a graph of size 100.

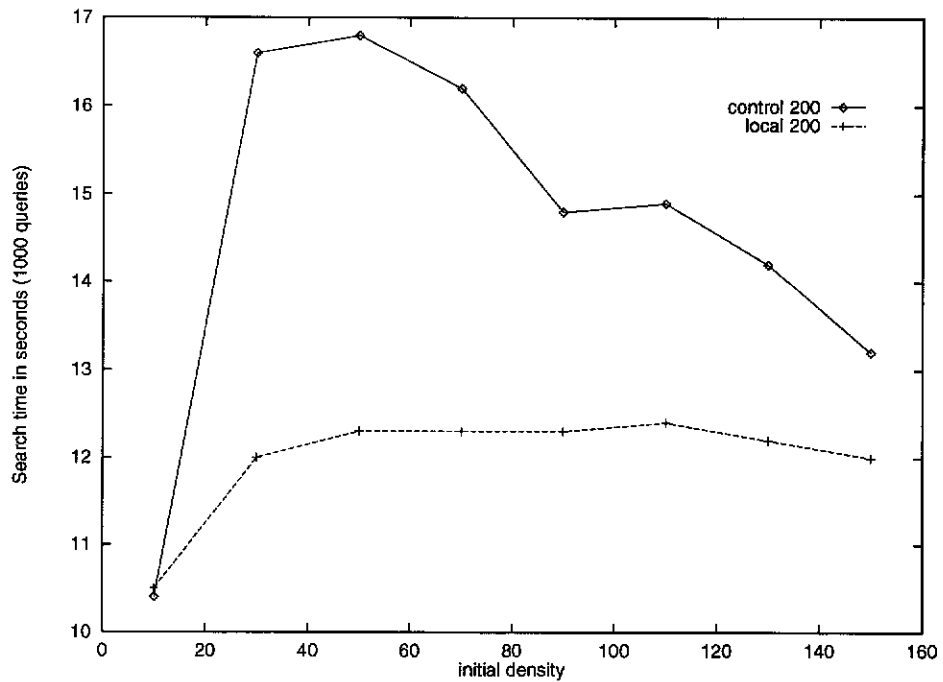


Figure 5.12: The relationship between the search times for local minimisation and the control are projected against the initial density for a graph of size 200.

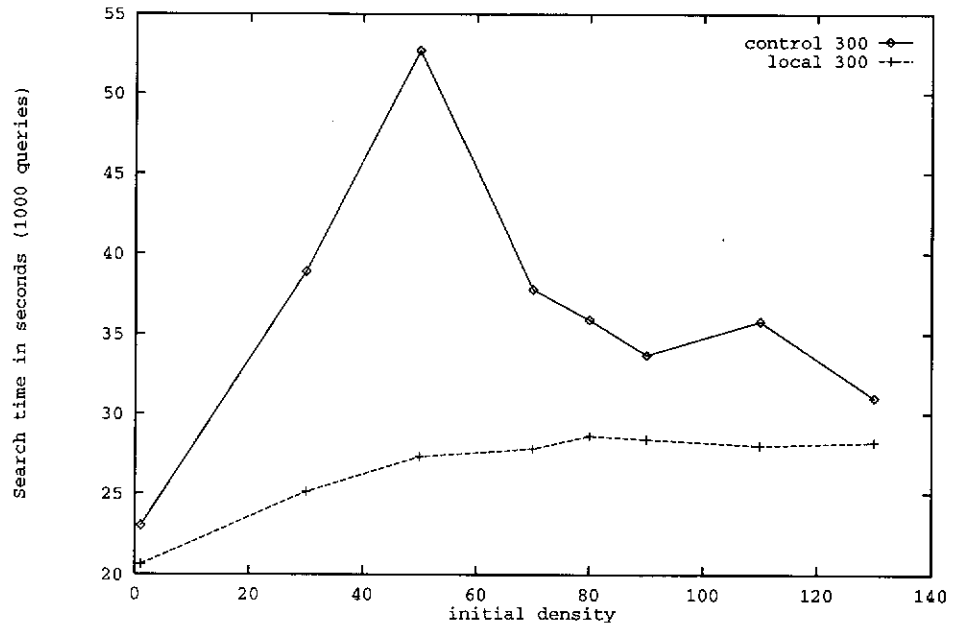


Figure 5.13: The relationship between the search times for local minimisation and the control are projected against the initial density for a graph of size 300.

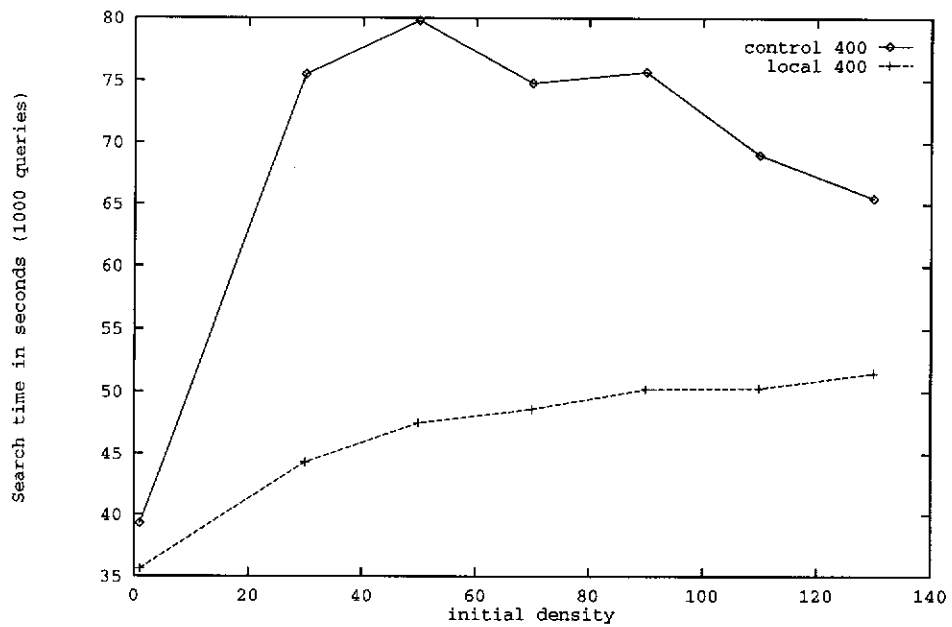


Figure 5.14: The relationship between the search times for local minimisation and the control are projected against the initial density for a graph of size 400.

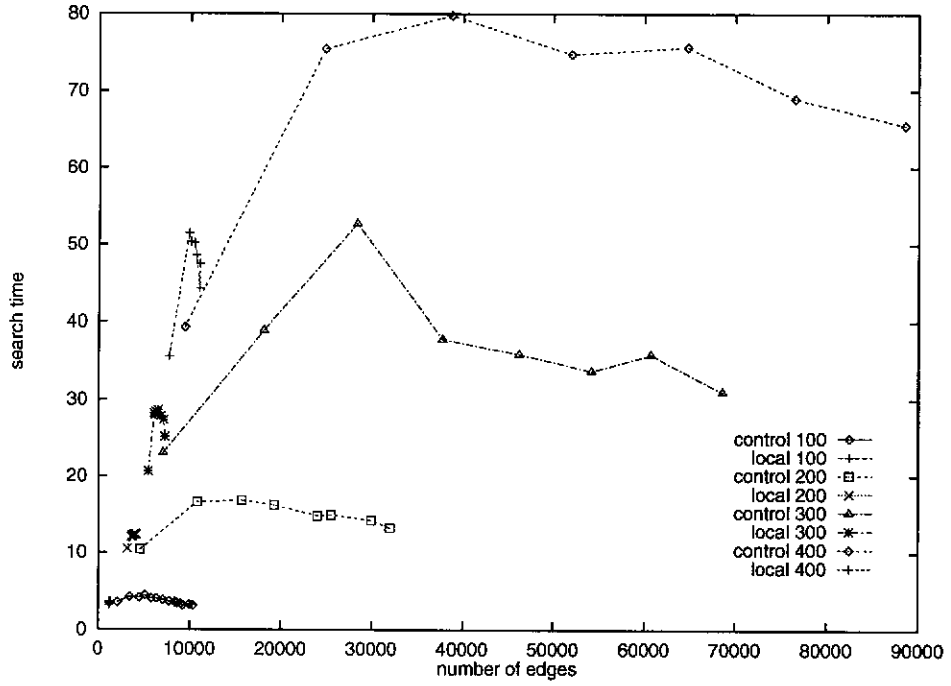


Figure 5.15: An indication of the relationship between the number of edges (x-axis) and the search time (y-axis) for the control and locally minimised graphs of sizes 200, 400, 600 and 800.

Our analysis of the local minimisation strategy indicates how the system reduces the number of edges needed to represent the information. We have shown that the construction time of the local minimisation strategy is of the same order as the control. The number of edges used to represent the information is near minimal (approaches minimal as the number of edges added to the representation is very dense). Thus the search time to determine a point relation in the local minimised representation is comparable to a fully minimised representation.

When choosing a balance between construction time, representation space and search time, local minimisation provides low construction costs and representation space with an adequate performance for search time. This adequate performance is based upon it being more effective than the control except when the control is very dense (closure).

Our system of local minimisation provides a minimisation system that has nearly the same costs for representation space and search time as full minimisation. The system does not require exhaustive checks to construct the representation and thus the construction time is very low (same order as the control).

There are several motivations for providing a comparative analysis of the performance of our local minimisation strategy over providing upper bounds of the “performance”.

Upper bounds provide an extreme upper limit for the algorithm but do not indicate either the lower bound or the average performance of the algorithm. The type of data used for the algorithm may greatly influence the performance of the algorithm making the upper bound values irrelevant. Indeed, for the problem given in this chapter, the performance of the algorithm is driven by the data (density and connectivity). The structure of the data significantly influence the performance of the system.

Evaluations of the performance of the alternative systems (Allen [76]) rely upon consistent data. The costs of generation of this consistent data is omitted from the results and provides an artificial evaluation of the closure and non-closure based systems. Determining consistency is a significant part of the problem. Thus the costs should also be indicated.

5.4 Discussion

In Chapter 4, it was shown that a linear ordering of end points may be used to accurately represent interval relations for the problem where all interval relations were primitive and every relation was explicitly known. This chapter has extended this notion of a point-based representation to encompass problems where intervals are primitive but may not all be explicitly known.

One of the significant benefits of the closure-based representation is that a minimal labelling of all interval relations could be constructed. This minimal labelling provides the least ambiguous interval relation for each interval relation with respect to every other relation in the representation. Thus every other interval relation and consequences pertaining to those relations would be used to determine the minimal label for every interval relation in the representation.

The locally minimised point-based representation does not propagate information during construction. All propagation is performed during the search process for determining the relationships between points. The paths used by the search

mechanism represent the process of utilizing other relations information to provide information about this point relationship.

The search mechanism will continue to search for a path between two points until one is found or until it is determined that no possible path exists between the two points. Once the relationships between all four key end points for an interval relation have been determined (being either a point relation or unknown), the interval relationship is minimal. There are only 29 possible relations (see Table 5.7) that may be derived from the possible combinations of four point relations having the values $\{<, >, =, *\}$, where $*$ represents the unknown relation or a relation that does not affect the overall interval relation.

Twenty-seven of these relations correspond to the relations derivable from primitive interval relations as given in Allen's Transitivity Table [2]. This provides a theoretical foundation for the derivation of the relations Allen has generated for his table. There are two relations (indicate special in Table 5.7) that are exceptions as they may be derived from primitive point relations but not via the direct propagation given in Allen's Table.

Indeed, closure of this representation can be constructed by determining every relation between every pair of interval relations. Given that the search time to determine these relations is polynomial, a restricted algebra that corresponds to the 29 relations given above has polynomial closure.

The restricted algebra represented by this problem is small, it represents one type of interval representation. Our point-based system does provide an efficient solution to representing and querying the information about interval relations (with respect to the restricted algebra). The main objective of this chapter has been to introduce the concepts that shall be discussed further in the next chapter.

5.5 Conclusion

This chapter has focused upon the problem of representing sets of intervals where every interval relation between a pair of intervals may not be explicitly known and those that are known are primitive interval relations. A graph-based method of representing the end points of intervals and their relations is used to provide

Table 5.7: Twenty-nine valid relations

Interval Relation	Conjunction of PRs				Special
	ab	aB	bA	AB	
disjoint 1<2	<	<	>	<	
disjoint 1>2	>	>	<	>	
meets 1<2	<	<	=	<	
meets 1>2	>	=	<	>	
overlaps 1<2	<	<	<	<	
overlaps 1>2	>	<	<	>	
during 1><2	<	<	<	>	
during 2><1	>	<	<	<	
ends 1<2	<	<	<	=	
ends 1>2	=	<	<	>	
ends 2<1	>	<	<	=	
ends 2>1	=	<	<	<	
equal	=	<	<	=	
{disjoint 1<2, meets 1<2, overlaps 1<2}	<	<	*	<	
{disjoint 1<2, meets 1<2, overlaps 1<2, during 2><1, ends 2>1}	*	<	*	<	
{disjoint 1<2, meets 1<2, overlaps 1<2, during 1><2, ends 1<2}	<	<	*	*	
{disjoint 1<2, meets 1<2, overlaps 1<2, overlaps 1>2, during 1><2, during 2><1, ends 2>1, ends 1>2, ends 2<1, ends 1<2, equal}	*	<	*	*	Y
{disjoint 1>2, meets 1>2, overlaps 1>2}	>	*	<	>	
{disjoint 1>2, meets 1>2, overlaps 1>2, during 2><1, ends 2<1}	>	*	<	*	

Table 5.7: Twenty-nine valid relations (cont.)

Interval Relation	Conjunction of PRs				Special
	ab	aB	bA	AB	
{disjoint 2>1, meets 1>2, overlaps 1>2, during 2><1, ends 1>2}	*	*	<	>	
{disjoint 1>2, meets 1>2, overlaps 1<2, overlaps 1>2, during 1><2, during 2><1, ends 2>1, ends 1>2, ends 2<1, ends 1<2, equal}	*	*	<	*	Y
{overlaps 1<2, overlaps 1>2, during 1><2, during 2><1, ends 2>1, ends 1>2, ends 2<1, ends 1<2, equal}	*	<	<	*	
{overlaps 1<2, during 2><1, ends 2>1}	*	<	<	<	
{overlaps 1<2, during 1><2, ends 1<2}	<	<	<	*	
{overlaps 1>2, during 2><1, ends 2<1}	>	<	<	*	
{overlaps 1>2, during 1><2, ends 1>2}	*	<	<	>	
{ends 2>1, ends 1>2, equal}	=	*	*	*	
{ends 2<1, ends 1<2, equal}	*	*	*	=	
completely unknown	*	*	*	*	

an efficient method of representing and retrieving this interval information.

Local minimisation provides a cost effective method for reducing the amount of data required to represent an equivalent amount of information to that of the control. With regards to construction costs, local minimisation is equivalent to the control, there is a minimal decrease for larger datasets (see Figure 5.8). The search performance of this representation approaches that of a fully minimised system.

This method can be applied to any interval representation whose interval relations can be reduced to point relations. For a system using four end point relations (conjunction) it is able to represent twenty-nine different interval relations, twenty-seven of which directly relate to the deductions performed in Allen's

transitivity table [2].

In the next chapter we shall increase the expressive ability of our graph-based representation by allowing disjunctions between point relations to be represented. This will allow the representation of an increased number of interval relations, and also the representation of sets of intervals where the relation between every pair of intervals may not be known and those that are known may be disjunctive.

Chapter 6

Disjunctive Interval Representations

The butterfly counts not months but moments,
And has time enough.

Rabindranath Tagore

To increase the expressive ability of the system described in the previous chapter, we introduce the concept of being able to represent disjunctions of point relations and disjunctions of conjunctions of point relations.

This provides us with the ability to represent interval relations that are disjunctions of primitive interval relations. Thus allowing the representation of sets of intervals where the relationship between every pair of intervals may not be known and those that are known may be disjunctions of primitive interval relations.

Our method of representing disjunctions of intervals as disjunctions of conjunctions of point relations (where the conjunction of point relation corresponds to a primitive interval relation), allows the possibility of representing disjunctions of point relations that do not correspond to any interval relation. Thus the solution presented here not only has the expressive ability of Interval Algebra but also relations beyond that algebra [28]. We shall restrict ourselves to the discussion of the problem of representing disjunctions of intervals using disjunctions of conjunctions of end points.

We begin by defining these disjunctive relations and demonstrating how they are to be represented and integrated into the graph-based representation introduced in the previous chapter (Section 6.1).

The introduction of disjunctions causes serious complications with respect to the search process. A search may only proceed if issues of disjunctions are accounted for. The results of the search may provide multiple alternative solutions, with which there is room for significant efficiencies (Section 6.2). We examine effectiveness of our representation and the search in Section 6.3 to demonstrate the utility of the system presented in this chapter.

Section 6.4 compares the performance of the disjunctive representation against the system of the previous chapter for a variety of different densities of disjunctions. A summary of the significant issues of this chapter is presented in Section 6.5 and conclusions in Section 6.6.

6.1 Disjunctive Relations

A disjunction of interval relations expresses the *uncertainty* about the relationship between two intervals. The disjunction contains a list of the primitive relations that *may be* true for a pair of intervals. Only one of these primitive interval relations is true, thus the disjunction is an exclusive-or. Each of the interval relations in the disjunction may be represented as a conjunction of point relations. Thus each disjunction is represented in disjunctive normal form (DNF) as:

$$(\text{point-rel}_{11} \wedge \dots \wedge \text{point-rel}_{1m}) \vee (\text{point-rel}_{n1} \wedge \dots \wedge \text{point-rel}_{nm})$$

Given a disjunctive interval representation, our aim is to determine the existence of those relations between intervals that rely upon consistent information. This is achieved by organising the information about the disjunctions of interval relations such that inconsistencies can be easily identified.

6.1.1 Weak links

To represent these disjunctive interval relations we shall extend the graph representation introduced in the previous chapter. The nodes of this graph represent

the end points of intervals, and the edges between nodes represent the relations between end points.

When a relation between two intervals is represented by a single primitive interval relation then this relation is absolute. Hence the point relations, representing the interval relation, could be directly added to the graph. Such edges can never be removed, and henceforth these types of links shall be called *solid links*.

Definition 6.1 A solid link is a monotonic binary relationship. A solid link is of the form $\langle n_i, n_j \rangle$ where n_i and n_j represent nodes, and n_i precedes n_j .

The information represented in a disjunctive interval relation represents the alternative interval relations that may hold for a given pair of intervals. The uncertainty of a disjunction means that one, and no more, of the disjuncts holds. In other words, the other disjuncts will be considered false when one disjunct is proven. Thus the information represented by the respective point relations is uncertain. If we add the respective links to the representation for every disjunction, some of them will need to be removed when the uncertainty is resolved. To distinguish between the monotonic and nonmonotonic links, we introduce *weak links* to represent nonmonotonic relations, whereas solid links are used to express monotonic relations.

Definition 6.2 A weak link $\langle n_i, n_j, conj \rangle$ is an edge that represents a nonmonotonic relationship between two points represented by nodes n_i and n_j , and *conj* represents the conjunction (of point relations) associated with the weak link (See Figure 6.1).

For example, the disjunctive interval relation:

$$(\mathbf{A} \text{ disjoint } 1 < 2 \mathbf{B} \vee \mathbf{A} \text{ disjoint } 1 > 2 \mathbf{B})$$

is represented in terms of its point relations

$$(a < b \wedge a < B \wedge b > A \wedge A < B) \vee (a > b \wedge a > B \wedge b < A \wedge A > B)$$

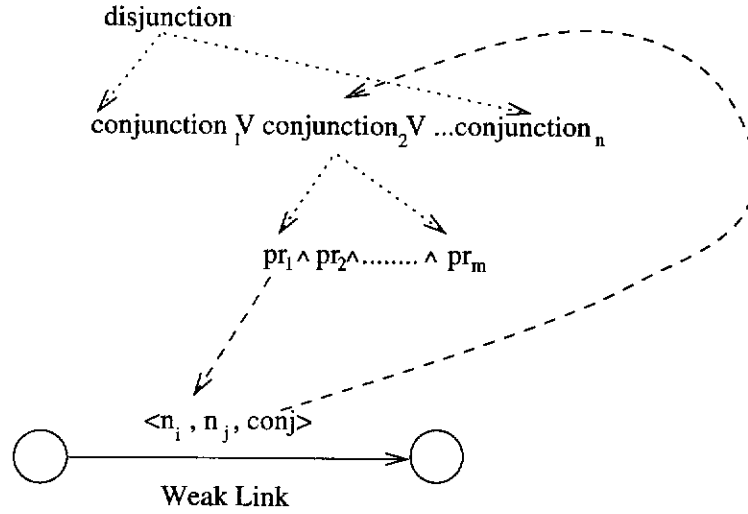


Figure 6.1: The relationship between the disjunction and the entries in the graph.

Each point relation is associated with a weak link in the graph depicted in Figure 6.2. A symbolic representation is given for the conjunction to relate the individual weak links to the conjunction.

The augmented graph containing disjunctions is defined as follows.

Definition 6.3 The *Disjunction Graph* $DG = (N, \varphi, \theta)$ is a triple where N is a set of nodes (all end points), φ is a set of pairs $\langle n_i, n_j \rangle$ defining a mapping between the nodes and solid links where $n_i, n_j \in N$, and θ is a set of triples $\langle n_i, n_j, conj \rangle$ indicating the mapping on nodes imposed by the weak links.

6.1.1.1 Adding weak and solid links

The addition of weak and solid links is dependent upon what information is currently represented in the graph. The following indicates what checks must be performed when adding weak or solid links.

A solid link indicates that the relationship between the two points is known. If any weak links exist between these two points then these weak links must be removed from the graph, as they are either proven or disproven by the relation associated with the solid link.

The algorithm to add the solid links to the graph is given in Algorithm 6.1.

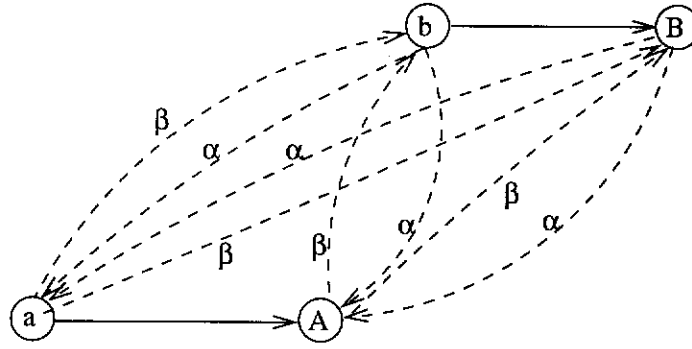


Figure 6.2: The representation of weak links in a graph. The links indicate the relation “less than” and the weak links are labeled to indicate the disjunct to which they belong i.e. α to indicate $A \text{ disjoint } 1 > 2 B$ and β to indicate $A \text{ disjoint } 1 < 2 B$.

The effect of removing a weak link (proven or disproven) to the disjunctions is discussed in Section 6.1.2.

```

if  $\neg ((a, b) \in \varphi)$  then                                {search for a solid link }
  if  $\neg (\text{pointsearch}(a, <, b) \ \& \ a = b)$  then      {search for a contradiction }
    begin
       $\varphi = \varphi + \langle a, b \rangle$                                 {add a link}
      foreach  $\langle a, b, conj \rangle \in \theta$                     {remove proven weak links}
         $\theta = \theta - \langle a, b, conj \rangle$ 
      foreach  $\langle b, a, conj \rangle \in \theta$                     {remove disproven weak links}
         $\theta = \theta - \langle b, a, conj \rangle$ 
    end

```

Algorithm 6.1 *Adding a solid link*

To add a solid link between two nodes in a graph, a check must be made to determine if the actual solid link between those two nodes already exists. A search is performed to determine if a solid path (a path made of only solid links) exists that contradicts the new link to be added. If neither are true, then the solid link is added. The addition of the solid link may make weak links redundant. A check is made to see if the weak links between the two nodes (to which the solid link is being connected) are proven or disproven. The local minimisation procedure of the previous chapter is then performed on the solid links. This check

of the weak links does not remove all possible redundancies caused by adding a solid link. This will be discussed further in Section 6.2.5.

Adding a weak link to a graph (Algorithm 6.2) can have no other direct effect other than the weak link that is being added being proven or disproven (that the relation is proven or disproven by the information already represented in the graph). Before a weak link can be added to the graph, a check is made to see if the relation is already represented in the graph (search the graph to see if the relation already holds). A similar check must be made to see if the current graph contradicts the weak link (search the graph for the contradictory relation).

```

Given weak link  $\langle a, b, conj \rangle$ 
if  $\neg$  (pointsearch( $a, b, >$ )) then           {check if no solid link}
    if  $\neg$  (pointsearch( $a, b, <$ ) or  $a = b$ ) {check if contradiction}
         $\theta = \theta + \langle a, b, conj \rangle$            {add new weak link}

```

Algorithm 6.2 *Adding a weak link between two nodes*

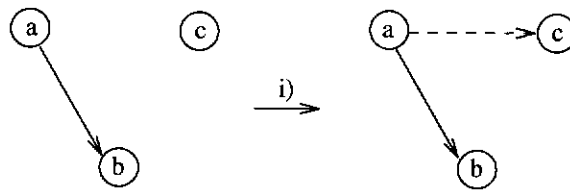
Consider the examples in Figure 6.3.

Nodes, representing two different points, may be proven to be equal. Two such nodes are combined into one node (representing two or more end points). This is only possible if no solid link (or path of solid links) exists between these two nodes, otherwise this would contradict the equality relation. The solid links and the weak links of each node are redirected to the combined node. Any contradiction that results from combining the solid links indicates that an overall contradiction has occurred. Weak links may be proven or disproven as a consequence of making the two nodes equal. This in turn may cause other weak links to become proven or disproven. This may cascade to cause other pairs of nodes to be made equal. Contradictions are avoided by removing the initial relation that lead to the contradiction.

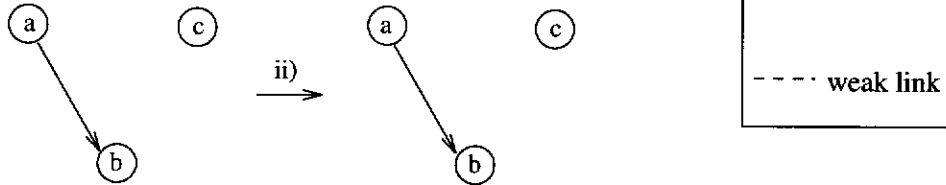
6.1.2 Disjunction resolutions

There are several situations where the information represented by the disjunctions may be modified by the addition of new information to the representation.

i) a weak link is added between nodes a and c after a check is made to see if a strong path exists between these two nodes



ii) the weak link between a and b cannot be added because a solid link already exists between these two nodes



iii) the weak link between a and c is removed because the new solid link makes it redundant

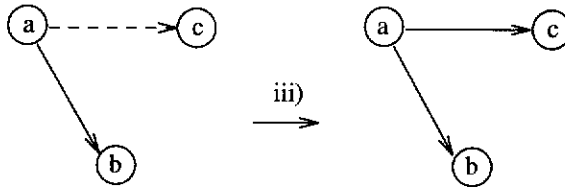


Figure 6.3: Adding links to a network. i) A weak link is added between a and c; ii) A weak link is not added between a and b; and iii) A weak link is replaced by a solid link between a and c.

There are three significant events that may occur: a weak link being proven, a weak link being disproven or a new disjunction being added. These three events may cause other situations to occur i.e. a disjunction being proven or disproven (contradiction).

6.1.2.1 Removing point relations

The rules used to determine the effect of proving or disproving a weak link are based upon the disjunctions. There are two assumptions underlying the representation of disjunctive relations:

- Every disjunction must be true (or there is a contradiction)
- The disjunctions are represented in just-one disjunctive normal form.

A conjunction of point relations represents a single interval relation. The maintenance of each conjunction during the graph construction process follows the logical properties of conjunctions. Proving one point relation causes that point relation to be removed from the respective conjunction, and the conjunct's respective weak link is removed from the graph.

Disproving a point relation causes the entire conjunction to be removed from the disjunction, and each conjunct's respective weak links in the graph are also removed.

For example, the effect of proving or disproving the relation d on a disjunction is shown below:

if $((a \wedge b) \vee (c \wedge d) \vee (e \wedge f)) \wedge \neg d \Rightarrow ((a \wedge b) \vee (e \wedge f))$,

the relation $(c \wedge d)$ is removed as d is disproven.

However, if $((a \wedge b) \vee (c \wedge d) \vee (e \wedge f)) \wedge d \Rightarrow ((a \wedge b) \vee c \vee (e \wedge f))$,

then d is proven but c is not removed.

As we are dealing with disjunctions that have “exclusive or” properties, when an interval relation in the disjunction is proven (each element of the conjunction is true), it is removed from the disjunction and every other interval relation of that disjunction is disproven (the conjunctions and their respective weak links). Removing all the elements of a disjunction causes a contradiction to occur (all disproven). It is assumed that one of the elements of the disjunction is true, and if none are true then the disjunction is contradictory with respect to the representation.

One property of the combination of the disjunction of interval relations and their conjunction of point relations is the proving of common elements of the conjunctions in a disjunction (Distributive Law). If a particular point relation is true for all the interval relations (conjunctions) of a given disjunction, then that point relation can be added to the graph as a solid link. If this point relation is false then the entire disjunction would be false, and thus a contradiction would occur.

The distributive law can be used to prove relations to be true:

$$((a \wedge b) \vee (a \wedge c)) \Rightarrow a \wedge (b \vee c)$$

a is true for both conjunctions, thus it is proven.

If 'A' is false then the disjunction is false and so a contradiction exists.

6.1.2.2 Adding disjunctions

The addition of a new disjunction requires that weak links are added to the graph to represent the point relations of the disjunction. If a disjunction already exists that shares the same pair of intervals then the intersection of the two disjunctions is determined. This intersection now represents the relationship between the two intervals.

For example:

$$\begin{aligned} & \mathbf{A} \{ \text{disjoint } 1<2, \text{ meets } 1<2, \text{ overlaps } 1<2 \} \mathbf{B} \cap \\ & \mathbf{A} \{ \text{disjoint } 1<2, \text{ meets } 1<2, \text{ during } 1><2 \} \mathbf{B} \Rightarrow \\ & \mathbf{A} \{ \text{disjoint } 1<2, \text{ meets } 1<2 \} \mathbf{B} \end{aligned}$$

6.1.3 Disjunction Table

The weak links used to represent the point relations of a disjunction provide only a partial representation of the disjunction. They represent only the point relations and the conjunctive associations between the point relations. A disjunctive table is used to represent the relationship between the conjunctions and organising point relations with respect to their conjunctions.

This information is also used to restrict the search process such that information about interval relations cannot be constructed using relations from different conjunctions of the same disjunction.

Definition 6.4 A disjunction of interval relations is represented by a disjunction of up to 13 unique primitive interval relations as follows:

$$\text{Disjunction of interval relations} = \langle \text{interval relation}_1 \vee \dots \vee \text{interval relation}_n \rangle, \\ n \leq 13$$

Definition 6.5 A Disjunction Table (dtable) is a collection of disjunctions of interval relations.

$$\begin{aligned}
\text{Disjunction Table} = [& \langle \text{Disjunction of interval relations} \rangle_1 \\
& \langle \text{Disjunction of interval relations} \rangle_2 \\
& \cdot \\
& \cdot \\
& \langle \text{Disjunction of interval relations} \rangle_m]
\end{aligned}$$

A *dtable* is used to represent all the disjunctions used in a representation.

For example, if the relationships between intervals **A** & **B** and **A** & **C** are given by:

- i) (**A** disjoint 1<2 **B** \vee **A** disjoint 1>2 **B**)
- ii) (**A** disjoint 1<2 **C** \vee **A** overlaps 1<2 **C**),

the respective *dtable* contains the following:

$$(a < b \wedge a < B \wedge b > A \wedge A < B) \vee (a > b \wedge a > B \wedge b < A \wedge A > B) \quad (6.1)$$

$$(a < c \wedge a < C \wedge c > A \wedge A < C) \vee (a < c \wedge a < C \wedge c < A \wedge A < C) \quad (6.2)$$

Upon this structure the disjunctive rules may be applied. Every point relation in every conjunction of every disjunction will have an equivalent weak link in the graph. Searches for the relationship between two points use the *dtable* to determine if the path is reliant upon inconsistent information.

6.2 Search

The process of establishing the relationship between two intervals becomes the process of determining the relationships between the significant pairs of end points of the two intervals (see Chapter 4.1).

In our graph-based representation, determining the relationship between two end points is the process of conducting a search of the graph to determine if a path exists between the two end points. There are two situations that lead to inconsistency for path construction that must be avoided. The first situation occurs when the point relations used to construct that path are weak links that represent point relations from different conjunctions of the same disjunction (see Section 6.2.1).

And the second situation, where the point relations used to construct the path are directly inconsistent with respect to each other (see Section 6.2.5).

6.2.1 Inconsistency via disjunctions

There are restrictions on which point relations may be used to construct paths as defined by the disjunctions of interval relations (*dtable*). It is the inability of Point Algebra to restrict the usage of point relations from different conjunctions of the same disjunction that prevents Point Algebra from accurately representing Interval Algebra.

For instance, the relations **A** *disjoint* $1<2$ **B** and **A** *disjoint* $1>2$ **B** would be represented as the conjunctions 6.3 and 6.4 respectively.

$$(a < b \wedge a < B \wedge b > A \wedge A < B) \quad (6.3)$$

$$(a > b \wedge a > B \wedge b < A \wedge A > B) \quad (6.4)$$

The two alternative orders, defined by the above examples, are illustrated in Figure 6.4. Without restrictions on the use of the point relations (as used with Point Algebra), a disjunction of these two relations would allow the following combination of relations to be deduced:

$$(a < b \wedge a < B \wedge b < A \wedge A < B).$$

This conjunction of point relations implies that the interval relation **A** *overlaps* $1<2$ **B** is true, which was not the intended statement of the disjunction: **A** *disjoint* $1<2$ **B** \vee **A** *disjoint* $1>2$ **B**.

The Point Algebra representation allows for relations to be deduced that were not intended to be represented. Thus the representation allows for the deduction of relations that are inconsistent with what was intended to be represented.

To prevent this situation from occurring we use the *dtable* to restrict the point relations that may be used to construct a valid solution to a point query. By recording the conjunction of every weak link used to construct a path, we are able

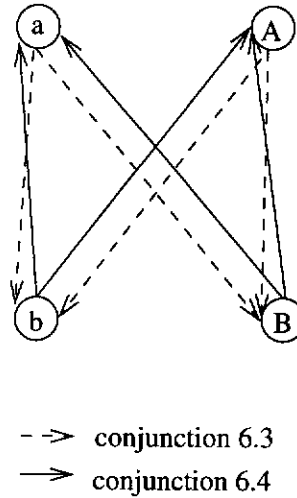


Figure 6.4: The two alternative interval relations A disjoint $1 < 2$ B and A disjoint $1 > 2$ B .

to restrict the point relations that may be used such that different conjunctions of the same disjunction will never be used to produce an individual solution.

The process of propagating the restrictions implied by the disjunctions in performing closure on PA is prohibitively expensive. Our non-closure system of maintaining minimal (near) information reduces the necessity for propagation to search time, thus performing the minimum propagation necessary.

6.2.2 Restricting Search

In this section we examine the mechanics of restricting certain combinations of point relations when constructing a search path between two points in the representation. The information detailing which combinations of relations can and cannot be used is represented in the *dtable*.

We shall use the term *search* to denote the process of determining the relationship between two points in our representation. The term *branches* will be used to indicate the state of the search. A search, expressed in the format $\langle \text{point}, \text{point}, \text{relation} \rangle$, involves determining if there is a path between the two points in the “direction” indicated by the relation, where $\text{relation} \in \{<, >, =\}$.

Definition 6.6 *Given a path $\{\langle n_1, n_2, conj_a \rangle, \dots, \langle n_{j-1}, n_j, conj_w \rangle\}$ each conjunction $conj_i$ in the path expresses a restriction on what new edges may be added*

to this path. A solid link may be considered to be a weak link with no dependencies (conjunction = NULL).

We are not interested in exactly which edges have been used, but the conjunctions from which these edges are derived. Thus it is only necessary to record the conjunctions upon which the individual search paths are dependent. A dependency is a list of these conjunctions that represent the weak links that have been used to construct a path.

$$\text{dependency} = \{ \text{conjunction}_\alpha, \dots, \text{conjunction}_\omega \}$$

where conjunction_i defines the restriction introduced by the weak links used to construct the path.

The construction of a search path must take into account the restrictions introduced as the path is constructed.

Definition 6.7 A branch $B_k = \langle n_i, \text{dependency} \rangle$ represents a path that is an element of the state of the search, where n_i is the current node and dependency is the list of the restrictive conjunctions previously used to construct a path from a given initial node to the current node n_i .

Definition 6.8 A state of the search is defined to be the branches $\{B_1, B_2, \dots, B_n\}$ where B_i is a branch.

The following considers the process of searching for a point relation using the system of branches.

When an individual branch reaches a point n_p where there are m alternative paths for the branch to continue upon, m copies of the branch are created to represent the alternative paths. A list of the possible links from a given node indicates the possible new paths from that node:

$$L = \{ L_1, L_2 \dots L_m \}$$

where L_i is a pair $\langle n_i, \text{conjunction}_i \rangle$ and for n_i there is an edge in the graph $\langle n_p, n_i, \text{conjunction}_i \rangle$

Given a Graph G and a current node n_p , we apply a function (Function 6.1) to determine the set of links for the given node. This algorithm looks for all the nodes that are directly *after* (the links are directional) the given node and creates

a list of links from each of those nodes (a similar algorithm may be used to search for those *before*).

```

lol( $n_p$ )
  listoflinks = null
  foreach  $\langle n_i, n_j \rangle \in \varphi$                                 {all solid links from node}
    if  $n_i = n_p$  then listoflinks = listoflinks +  $\langle n_j, \text{nil} \rangle$     {if linked to n, then add}
  foreach  $\langle n_i, n_j, \text{conj} \rangle \in \theta$                         {all weak links}
    if  $n_i = n_p$  then listoflinks = listoflinks +  $\langle n_j, \text{conj} \rangle$     {if linked to n then add}
  return listoflinks

```

Function 6.1 *The function for determining the list of links (lol)*

Using these preliminary definitions, the process of determining the relationship between two points can itself be defined. This process takes into account the restrictions represented by the information in the *dtable*.

A search (Algorithm 6.3) begins at a given point and initially has no dependencies (LB). The search proceeds towards the target point in a given direction. For each of the current states, a set of possible new links towards the solution is determined (L). For each of these new links that are compatible with the current state, a new state is created having a dependency that is the combination of the current state and the new path's dependencies. If this new state actually represents a solution then that solution is recorded (SOLUTION), otherwise the new state is recorded to be used for the continuing search (LBHOLD).

The issue of compatibility of a link and the current state is determined by checking to see if the link's dependency is consistent with every member of the states' dependency (see Function 6.2). Two dependencies are only inconsistent if they are members of different conjunctions of the same disjunction (this information is represented in the *dtable*).

The solution to the point query is a list of the alternative sets of dependencies, at least one of which must be true if the point relation is true.

Definition 6.9 *A dependency list $\{dependency_1, dependency_2, \dots, dependency_n\}$ is a collection of all the dependencies on which a given solution is dependent.*

```

search(startnode, finishnode, direction)
LB = {⟨startnode, NULL⟩}
LBHOLD = NULL, SOLUTION = NULL
while (LB ≠ ∅)
  begin
    foreach ⟨ni, depi⟩ ∈ LB {every branch in the list}
      begin
        L = lol(ni, direction)
        foreach ⟨nj, conjj⟩ ∈ L {for each link}
          {check if the link ⟨nj, conjj⟩ can be added}
          if (compatible(depi, conjj))
            begin
              if (nj = finishnode)
                {add to the list of solutions}
                SOLUTION = SOLUTION + (depi + conjj)
              else {add new search}
                LBHOLD = LBHOLD + ⟨nj, depi + conjj⟩
              end {if}
            end {foreach}
          LB = LBHOLD {replace old for new}
          LBHOLD = NULL
        end {while}

```

Algorithm 6.3 *Search for a point relation*

```

compatible(dependency, conjunction)
  compatible = true
  foreach k ∈ dependency
    if (diseq(k, conjunction) & ¬(k = conjunction))
      {where diseq is a function that checks dtable to see if in same disjunction}
      compatible = false
  end

```

Function 6.2 *Verification of compatibility*

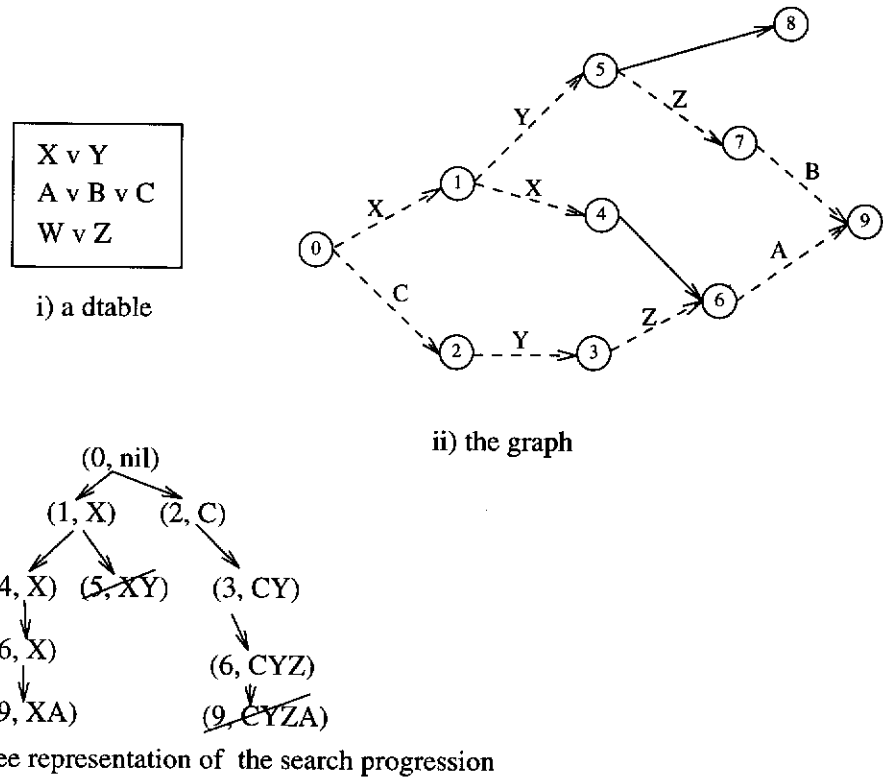


Figure 6.5: Restrictions on the search process. Where i) is the disjunction table, ii) the graph and iii) a trace of the search from 0 to 9.

For example, given the *dtable* and graph depicted in Figure 6.5, a search $(0, 9, >)$ progresses as shown in Figure 6.5iii. The branches $(5, XY)$ and $(9, CYZA)$ are discontinued as contradictions arise from their dependencies, $(5, XY)$ has contradictory dependencies as X and Y cannot both be true and $(9, CYZA)$ has the contradictory dependencies A and C .

Given the ability to determine the relationship between two points and the resultant dependencies upon which the solutions rely, the next two sections look at how this information is used.

6.2.3 Solid Solutions

The best solution to any query is one that returns a *solid solution* where a solid solution is one that does not consist of any weak links (no dependencies). Thus to determine the relationship between two points, a check is performed to see if there is a solid solution for the directions *before*, *after* and *equal*, prior to the search for a weak solution. The search mechanism from the previous chapter is

used to perform these solid searches.

As mentioned previously, it is possible for the graph to contain redundant weak solutions. These redundant weak links have a different significance to the redundant solid links. Redundant weak links may represent false information that is no longer true in the representation. Rather than performing the expensive process of removing all redundant weak links, checks are made of the solutions to queries to make sure that they are not dependent upon redundant weak links. The first check is to look for a solid solution to the query thus making any weak solution obsolete. The second check will be discussed later.

6.2.4 Weak Solutions

Weak solutions are solutions to queries that have dependencies, thus they depend upon other point relations being true.

Definition 6.10 *A Weak Solution_n (WS_n) is a pair $\langle \text{point-relation}, \text{dependency list} \rangle$ used to represent the possible relationship between two points and the alternative sets of relations upon which this may depend. “Point-relation” indicates the point relation to which the weak solution refers and the dependency list is the list of dependencies that must hold for the relation to be true (the point relation is true if one or more of the dependencies is shown to be true).*

Each dependency represents conjunctions (thus their weak links) upon which a particular path relies. If every conjunction for a dependency holds (shown to be true) then that relation is true, otherwise that dependency can no longer be used to indicate that the given relation may hold.

There are certain hierarchical properties of the dependencies that may be used to reduce the number of dependencies used in representing a relation’s dependency list. These properties may also be used to simplify the search process.

A weak solution that has a lesser number of dependencies than another weak solution is dependent upon fewer relations. Thus solutions that have a minimal number of dependencies are preferred.

This does not mean that a solution with a greater number of dependencies is discarded in favour of a solution with fewer dependencies. Each is an equally

valid possibility, the only exception occurs when one set of dependencies is a strict subset of the other. The solution that is the strict subset is retained in place of the solution with more dependencies, since the solution that is a subset represents an equivalent solution with fewer dependencies.

There are two areas of the search where this property may be used.

6.2.4.1 Efficiencies in search

A search process that evaluates all the possible links for all branches is inefficient. There is obvious scope for effective pruning of this search space.

1) Superset of solutions

Any branch whose dependencies are a superset of a known solution (equal or greater set of dependencies) is superseded. There is no need to continue searching along this path as any solution derived from this branch will be discarded when the results are returned. This is due to the fact that any solution along this path will have at least the same dependencies as the established solution. We are only interested in results with the least number of dependencies.

All branches whose dependencies are a superset of a known solution can be removed as given in Algorithm 6.4.

```

{Given a branch  $\langle n_i, \text{dep}_i \rangle$ }
solution =  $\langle \text{pr}, \text{deplist} \rangle$ 
foreach  $\text{dep}_j \in \text{deplist}$ 
  if  $\text{dep}_i \supseteq \text{dep}_j$  then
    stop branch

```

Algorithm 6.4 *Checking if the current search branch is a superset of a previous solution.*

For example, given a solution that is dependent upon the relations (abc) being true, any branch that reaches a node and has the dependency (abc) or a superset of (abc) may be a solution to the problem. However, any possible solution derived from this branch will have a minimum of (abc) as its dependency. Thus the branch need not be continued as its final solution will consist of at least the same, if not more, dependencies as a known solution.

Every time a new branch is created a check is made to see if the branch's dependencies are a superset (greater or equal) of a solution's dependencies. If it is, then the new branch can be discarded.

A solution which is a strict subset of another solution can be used to remove the second solution from the list of solutions (i.e. redundant).

Example 6.1 *Terminating the branches whose dependencies are a superset of a solution.*

Consider the graph depicted in Figure 6.6i. A search from node C to node B gives the following results:

- Solution 1 with dependency x (path = $\langle C, 1, 5, B \rangle \vee \langle C, 6, B \rangle$);
- Solution 2 with dependency yx (path = $\langle C, 1, 2, B \rangle$) and
- Solution 3 with dependency yz (path = $\langle C, 1, 2, 3, B \rangle$).

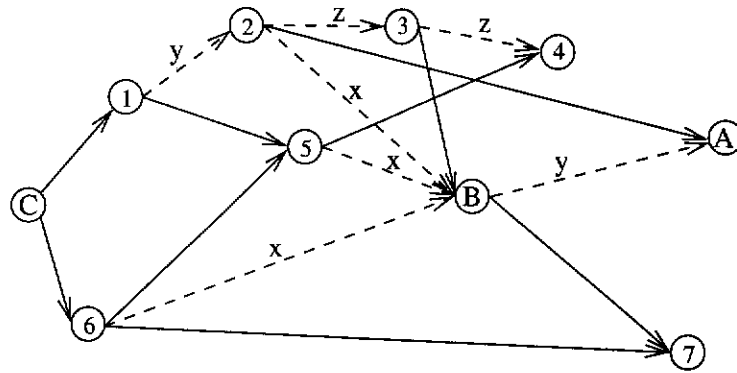
The dependency of Solution 2 is a superset of the dependency of Solution 1. Thus only the results of Solution 1 and Solution 3 are significant.

When constructing a search from node C for node A (see Figure 6.6ii), there are several branches terminated due to various reasons: search repetition (i.e. convergence at node 5), superset of a known solution (i.e. $3-zy$ is a superset of $A-y$) and superset of previous searches (i.e. $7-x$ is redundant due to previous search of 7, see the next section).

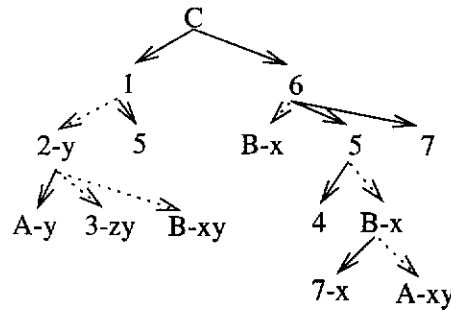
2) Supersets of Searches

In the previous chapter, the method of pruning the search space involved marking nodes to prevent the repetition of the search. Similarly, we are able to reduce the amount of redundant searching for the system described in this chapter.

As a search through the graph progresses, a given branch may encounter a node that has previously been visited by another branch of the same search. If a branch that reaches a given node has no dependencies, then any other branch that later reaches the same point is superseded by this branch. Any branch from



i)



ii)

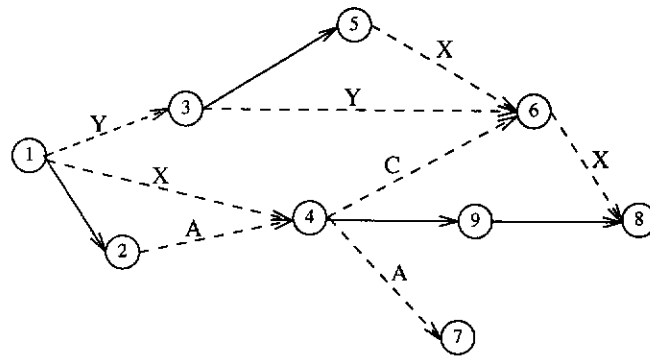
Figure 6.6: i) An example graph (weak links shown as dashed lines and solid links as solid lines) contains points A, B, C. ii) The search from point C for the point A, where elements are of the form: node - dependencies.

this point will have a greater or equal set of dependencies. This can be extended to include branches whose dependencies are supersets of a previous branch.

If a branch reaches a node which has dependencies X , and a branch that later reaches the same point has dependencies Y , then if $X \subset Y$, the later branch (with dependencies Y) is superseded by the earlier branch (with dependencies X). If the branches' dependencies are equal then, again only one branch should be continued.

There are two ways of recording branches so that redundant branches can be pruned. A global list of the branches (nodes and their dependencies) is used to record each branch as it reaches a node and its corresponding list of current dependencies (all states). Every branch checks with the global list to see if the current node is in the global list and if the dependencies of those entries in the

XvY
AvB
DvC



i) Disjunction table

ii) graph

pass

1	(1 nil)
2	(1 nil) (3 y) (4 x) (2 nil)
3	(1 nil) (3 y) (4 x) (2 nil) (5 y) (6 y) (6 xc) (9 x) (7 xa) (4 a)
4	(1 nil) (3 y) (4 x) (2 nil) (5 y) (6 y) (6 xc) (9 x) (7 xa) (4 a) (8 yx) (8 xc) (7 ya) (8 x) (6 ac) (9 a) (7 a)
5	(1 nil) (3 y) (4 x) (2 nil) (5 y) (6 y) (6 xc) (9 x) (7 xa) (4 a) (8 x) (6 ac) (9 a) (7 a) (8 a)

iii) Global search list

Figure 6.7: A search generates a global history list that helps identify repetitive branches.

global list are subsets of the current branch.

The global branch list can be kept minimal by recording only the minimal (subsets) dependencies for each node. Thus if a node that was reached by a branch with dependencies X is later traversed by a branch with dependencies Y, and Y is a subset of X, then the global entry with dependencies X can be replaced by a new entry with dependencies Y.

For example, given the disjunction table in Figure 6.7i and the graph in Figure 6.7ii, a search progresses incrementally creating a global dependency list. The growth of the dependency list is given in Figure 6.7iii. The global list is modified when new elements (and their dependencies) are added and old elements are removed. A new element is added only if it is not superseded by a current element, and an old element is removed if it is superseded by a new element, i.e. (8 yx) is removed when the new element (8 x) is added.

This results in a feasible solution. This solution, however, has problems as the size of the history can grow prohibitively and the checks for matches become expensive. One method to reduce the growth is to record a fraction of the actual

elements in the list. For example, only every second branch would be added to the global list. There is a cost associated with this heuristic, with some of the information missing, future searches may be duplicates of past ones. Hence, there is a trade-off between the storage (and search) of the history and cost in time to duplicate searches.

There is an improved method that is similar to the search heuristic of marking nodes as the search progresses (used in the previous chapter). A local history can be kept for each node. This does not change the size of the search list but does break it up into smaller lists which can be stored as a property of a node. Thus when a branch reaches a node, the checks are performed with the list maintained at that node. The search of the global list, due to its size, carries a substantial cost. With the local lists there are no extra comparisons for any history not directly associated with the current node. Figure 6.8 shows the local branch list system using the search from Example 6.7. As the searches progress, new local dependencies are created. The addition and removal of dependencies is exactly the same as in the global system except the dependencies are stored at each node.

Definition 6.11 *Given a disjunctive graph $DG = \langle N, \varphi, \theta \rangle$ we can define a set of nodes that contain a list of dependencies called N-dep, such that there is a 1-1 mapping between N and $N\text{-dep}$. $N\text{-dep}$ is a set of pairs $\langle \text{endpoint}_i, \text{dependency list}_i \rangle$ where endpoint_i is the name of point(s) represented by the node $n_i \in N$, and dependency list is the local list of dependencies that will be used to construct searches.*

The dependency list for each node is initialised to the empty set before a search is performed on the graph. The algorithm for local dependency heuristic is given in Algorithm 6.5.

When a branch reaches a particular node, the dependency of that branch is checked to see if it is a superset of any of the dependencies stored at that node (left by previous branches). If so, the branch is discontinued. Otherwise the branch's dependency is checked to see if it supersedes any of the current dependencies stored at that node. All the superseded dependencies are removed and the branch's dependency is added to the local list.

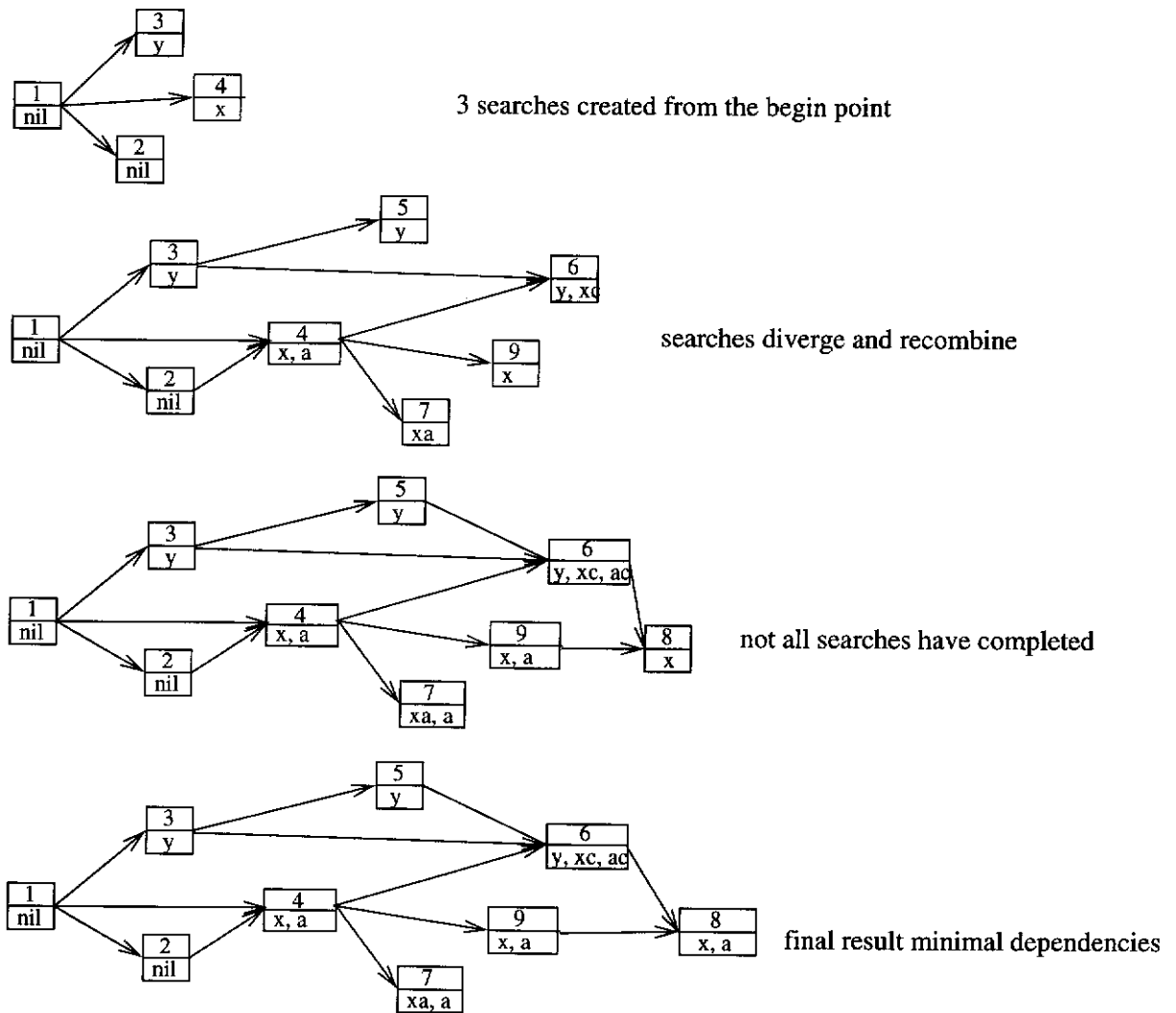


Figure 6.8: The growth of the local search lists as the search progresses. All dependencies are given with respect to the initial node 1.

6.2.4.2 Redundant information in weak solutions

Each solution path that is generated for a particular point relation is represented as a set of interval relations (conjunctions) that must be true if that particular solution path is true. No proactive elimination of redundant weak links has been performed on this representation. Thus, it is possible that a solution may rely upon information that is no longer valid in the representation.

Our alternative to the expensive process of proactively removing redundant weak links, is to check the relations used for each solution to see if these relations are contradicted in the representation. This is done by checking if there are solid solutions to the interval relations that contradict or confirm the interval relations

```

Given a branch  $\langle n_i, \text{dep}_i \rangle$ 
 $n_i = \langle \text{end point}_{n_i}, \text{deplist}_{n_i} \rangle$ 
superset = false
{check if current search is a superset of any of the local dependencies}
foreach  $\text{dependency}_j \in \text{deplist}_{n_i}$ 
  begin
    if  $\text{dep}_i \supseteq \text{dependency}_j$ 
      superset = true
    end
  if superset
    stop branch
  else
    {if not superset then check to see if local dependencies are supersets}
    {if a local dependency is a superset then remove and add current dependencies to list}
    begin
      foreach  $\text{dependency}_k \in \text{deplist}_{n_i}$ 
        if  $\text{dep}_i \subseteq \text{dependency}_k$ 
           $\text{deplist}_{n_i} = \text{deplist}_{n_i} - \text{dependency}_k$ 
         $\text{deplist}_{n_i} = \text{deplist}_{n_i} + \text{dep}_i$ 
      end {else}
  end {foreach}

```

Algorithm 6.5 *Using and maintaining local dependency lists*

upon which the solution is dependent.

If an interval relation that a path is dependent upon is proven to be false, then that entire path is no longer valid. Alternatively, if the interval relation is proven, then the path is dependent upon less uncertain relations. Thus the interval relation is removed from the set of dependencies for that path.

6.2.5 Inconsistency

So far we have examined only one type of inconsistency that may occur when using disjunctive interval relations. A second type of inconsistency that may

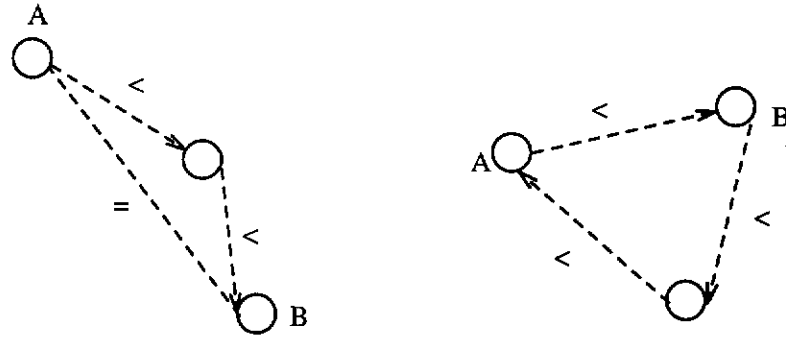


Figure 6.9: The two types of loops that are derived from consistent links (i.e. consistent with respect to each other).

occur when constructing a path is when the relations in the path contradict each other.

If (for example) $A < B$, then $A > B$ cannot hold. This type of direct contradiction is avoided as the search always progresses in the same direction, i.e. along $<$ or $>$ directions. It is possible, however, to create a logically equivalent situation if cycles are allowed. In Figure 6.9 two examples of loops that lead to contradictions are depicted.

The pruning mechanism for the search method also prohibits cycles. Whenever a node is revisited, the path is discarded, thus preventing the loop from occurring and avoiding possible contradictions.

In this chapter, no attempt has been made to check the consistency of the entire representation. Such a check is an NP-Complete problem. Thus it is possible for a representation to contain two disjunctions that will cause a contradiction.

6.2.6 Using Weak Solutions

Solid and weak point relation solutions to queries about point relations may be combined to construct solutions to interval queries. In Chapters 4 and 5 these types of interval queries have depended solely upon the use of solid solutions. In this section we shall examine the various possible combinations of weak and solid solutions.

A weak solution is expressed as: a point relation between two points and a list of dependencies. Each of these dependencies is a list of conjunctions that repre-

sent alternative paths, i.e. sets of relations that the current relation is dependent upon. Relations that have dependencies may only be combined if there exists at least one consistent union of a dependency from each of the weak solutions.

For example, given the weak solution $a < B$ with dependency $(c < D \wedge e < d)$ and the weak solution $A > B$ with dependency $(c > D \wedge e < F)$, relations $a < B$ and $A > B$ cannot be combined as their dependencies are incompatible (i.e. rely on inconsistent information: $a < B$ relies on $c < D$ and $A > B$ relies on $c > D$).

Many point relation inconsistencies are not as obvious as in the given example. The disjunction table is used to check for inconsistencies when interval relations are established from weak solutions.

Two dependencies may be combined if none of the conjunctions represented by each dependency are different conjunctions of the same disjunction. The algorithm to combine two weak solutions WS_1 and WS_2 is given in Algorithm 6.6. The *check* function determines if the two dependencies are compatible and is defined in Function 6.3.

```

newdependency =  $\emptyset$ 
 $WS_1 = \{ \langle p\text{-rel}_1, \text{deplist}_1 \rangle \}$ 
 $WS_2 = \{ \langle p\text{-rel}_2, \text{deplist}_2 \rangle \}$ 
foreach  $\text{dependency}_i \in \text{deplist}_1$ 
  foreach  $\text{dependency}_j \in \text{deplist}_2$ 
    {check if compatible}
    if  $\text{check}(\text{dependency}_i, \text{dependency}_j)$ 
      {add to new dependency list}
    newdependency = newdependency +  $\text{append}(\text{dependency}_i, \text{dependency}_j)$ 

```

Algorithm 6.6 *Combining two dependencies lists*


```

check(dependency1, dependency2)           {check if dependencies compatible}
begin
  check = true
  foreach conjunctioni ∈ dependency1
    if ¬(compatible(dependency2, conjunctioni))           {if not compatible}
      check = false
  end

```

Function 6.3 *Check is used to determine the compatibility of dependencies*

An interval relation that has been constructed using weak solutions has a dependency list that indicates relations on which the interval relation is dependent. This dependency list will be listed with the interval relation.

Example 6.2 *Weak solutions combined to determine dependencies for interval relations*

A solution that has the solid relations $a < b$, $b < A$ and $a < B$ and a weak solution $A < B$ with dependencies $(E < f \wedge A > C)$ would be listed as:

A overlaps 1 < 2 B with dependencies $(E < f \wedge A > C)$.

Figure 6.10 shows an example graph where the solutions to selected point relations are:

$b < a$ ($1 \vee 4$);

$B < A$ (4);

$a < B$ (?) and

$b < A$ (3).

The relations $b < a$ ($1 \vee 4$) and $b < A$ (3) cannot be used together as their dependencies are contradictory.

◇

6.3 Analysis

We have significantly increased the expressive ability of the system presented in Chapter 5. We are now able to represent the full set of Interval Algebra (IA)

1 v 2
3 v 4

dtable

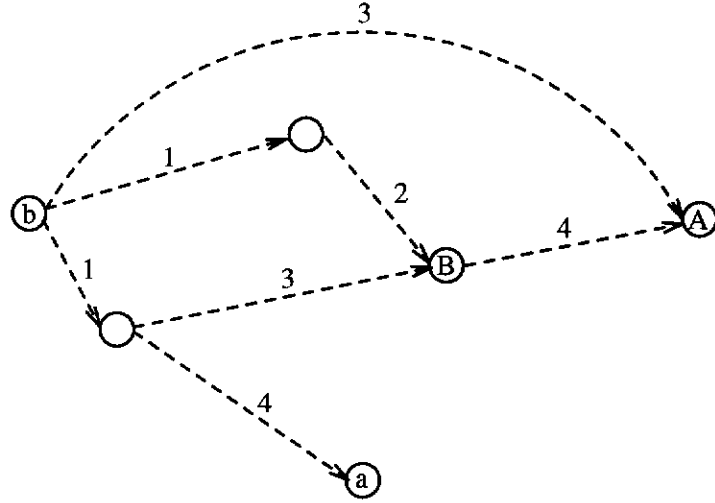


Figure 6.10: A section of a graph and the corresponding dtable that refer to the intervals A and B.

relations (2^{13}). It is also possible to extend the expressiveness of this system to relations outside of IA ($(a < b \wedge a < B) \vee (a < c \wedge A < C)$), though this will not be developed in this thesis.

For the closure-based systems, the goal is to determine all the consequences given a set amount of information. The SATisfiability problem (SAT) to determine if there is a consistent labelling of every interval relation, the Minimal Labeling Problem (MLP) to determine the minimal IA relation needed to represent each interval relation and, most significantly, the All Consistent Solutions Problem (ACSP) to determine every possible consistent combination of interval relations that will provide a consistent labelling of every interval relation (all combinations), are NP-Complete for IA.

Our non-closure system does something different, for an interval pair it returns the alternative interval relations that may be true. For each interval relation it provides all the alternative sets of interval relations upon which it is dependent.

Query: A ? B

Solution:

disjoint 1<2 if A overlaps 1<2 C and F during 1><2 G
or A overlaps 1<2 C and F overlaps 1<2 H
or A overlaps 1>2 C and J overlaps 1<2 F
overlaps 1<2 if A overlaps 1<2 B
during 1><2 if B meets 1<2 F.

Thus allowing for a form of ACSP to be developed, for each pair of intervals the system can determine the alternative possible interval relations that may be valid. For each of these possible solutions, the list of interval relations that any solution is dependent upon is given (there may be multiple alternative sets of dependencies for each interval relation).

The focus has been upon providing a method of representing interval relation (disjunctive) and demonstrating a method of propagating interval information. Its virtue will be its ability to provide information about individual interval relations (or clusters of interval relations) with respect to the entire data set without the need to extrapolate all interval information (as with closure).

There may be multiple dependencies for each weak solution. All possible combinations of dependencies are determined. Each dependency is consistent with respect to itself.

6.4 Results

In order to evaluate the performance of the system described in this chapter, its performance is compared to the simpler system described in the previous chapter. As with the previous system the results were obtained using a SG R4400 processor using a program that was written in C++.

The raw data for the two systems differs only in that the new disjunctive system is able to represent disjunctive interval information. Thus a number of different initial densities of disjunctions are evaluated for the same initial densities and sizes as used in the previous chapter.

Three data sizes 100, 200 and 300 are considered at three different disjunction initial densities (actually four, considering the primitive representation as no disjunctions) in Tables 6.1, 6.2 and 6.3. Initial densities indicate the number of links added to each node, “const” is the construction time in seconds, “number of links” is the number of solid links used in the representation, “number of weak links” is the number of weak links used in the representation and “search” is the time taken to perform 1000 queries on the representation. Many of the disjunctive links will not be added as they contradict the “current” representation. The number of disjunctions added follows a non-linear pattern (see below).

Similar to the previous chapter, the times for search, construction and the number of weak and solid links are the averages of 30 datasets.

6.4.1 Analysis

In the previous chapter (Chapter 5) our analysis of the performance of the local minimisation system was compared to a control dataset. The performance of the systems compared the construction times, search times and the number of solid links in the graphs. This provided a direct comparison of the two representations based upon the key characteristics of the representation.

In this chapter, we have introduced the property of disjunctive interval relations in an interval representation. In order to gauge the performance of this disjunctive system a variety of different initial density of disjunctions are compared. The key issue will be the density of the disjunctions, the construction time, the search time and the density of the non-disjunctive links. The control for this chapter is a locally minimised dataset with no disjunctive information (from Chapter 5).

6.4.1.1 Construction

The construction process is very similar to the process described in Chapter 5. The only difference is the addition of disjunctive information to the representation. The introduction of disjunctive information requires the construction of the disjunctive table (dtable) and the addition and maintenance of the weak links

Table 6.1: A comparison of the primitive IR system against three different initial densities of disjunctions using the disjunctive IR system for a dataset of size 200 (100 intervals). Construction and Search times are given in seconds.

		initial densities				
		10	30	50	70	90
primitive (control)	const (secs)	8	27	47	66	87
	number of links	1217	1307	1325	1322	1281
	1000 searches (secs)	3.1	3.5	3.4	3.4	3.5
disjunction density 1 (den1)	const (secs)	14	41	70	99	130
	number of links	1223	1269	1338	1259	1261
	number of weak links	98	85	57	58	52
	1000 searches (secs)	5.8	4.7	4.4	4.2	4.6
disjunction density 3 (den3)	const (secs)	13	41	70	98	128
	number of links	1222	1313	1306	1298	1297
	number of weak links	37	31	27	24	23
	1000 searches (secs)	4.7	4.2	4.2	4.2	4.1
disjunction density 5 (den5)	const (secs)	13	52	72	96	126
	number of links	1224	1324	1319	1288	1293
	number of weak links	24	20	17	16	15
	1000 searches (secs)	4.7	5.8	4.2	4.2	4.0

in the representation. There are also extra costs in adding solid links to the representation as their effect on the weak links must be determined.

Examine the graphical representation of the construction times for four different densities of disjunctions (where one is the *control*) for three different datasizes depicted in Figure 6.11. The x-axis is the initial density and the y-axis is the construction time in seconds.

There are minor changes in the gradient for the construction time for the three different densities with respect to the gradient of the control. There are slight variations from a consistent gradient for each of the three data sizes, the most

Table 6.2: A comparison of the primitive IR system against three different initial densities of disjunctions using the disjunctive IR datasets for a dataset of size 400 (200 intervals). Construction and Search times are given in seconds.

		initial densities				
		10	30	50	70	90
primitive (control)	const (secs)	54	192	326	454	570
	number of links	3166	3762	3887	3962	4106
	1000 searches (secs)	10.5	12.0	12.3	12.3	12.3
disjunction density 1 (den1)	const (secs)	85	304	531	889	975
	number of links	3066	3713	3753	3351	3617
	number of weak links	192	174	166	103	128
	1000 searches (secs)	18.2	17.9	17.7	17.0	16.0
disjunction density 3 (den3)	const (secs)	85	318	523	767	981
	number of links	3190	3767	3817	3951	3799
	number of weak links	79	65	57	54	52
	1000 searches (secs)	16.5	16.8	16.6	16.1	16.9
disjunction density 5 (den5)	const (secs)	83	308	527	755	1061
	number of links	3163	3826	3847	3824	3760
	number of weak links	48	38	44	33	29
	1000 searches (secs)	15.5	17.1	16.8	16.0	16.8

notable being *density 5* in the 300 intervals' graph.

The difference between the different densities of disjunctive intervals is less than the difference between the control and each of these densities. This leads to the conclusion that there is an inherent cost involved in resolving weak links that does not increase proportionally to the density of the disjunctive interval relations.

This increase in the construction time, demonstrated in this empirical analysis of the construction process, is consistent with the expected costs of adding and maintaining the disjunctive relations.

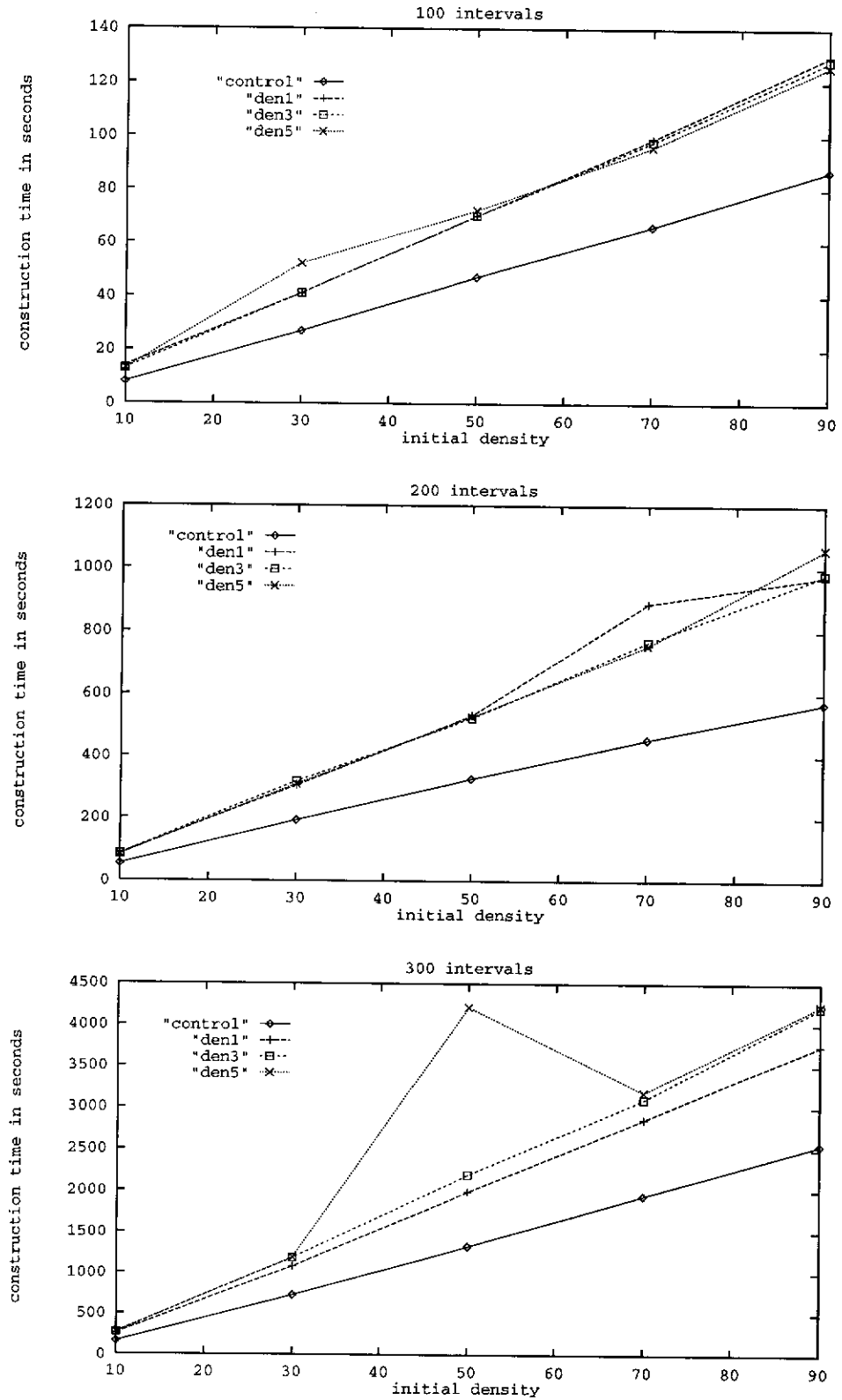


Figure 6.11: The performance of the primitive IR system vs the disjunctive IR system the with respect to construction time.

Table 6.3: A comparison of the primitive IR system against three different initial densities of disjunctions using the disjunctive IR system for a dataset of size 600 (300 intervals). Construction and Search times are given in seconds.

		initial densities				
		10	30	50	70	90
primitive (control)	const (secs)	160	723	1325	1941	2551
	number of links	5419	7222	7063	6770	2551
	1000 searches (secs)	20.6	25.0	27.3	27.8	28.6
disjunction density 1 (den1)	const (secs)	256	1080	1986	2856	3745
	number of links	5141	6703	6573	6330	6018
	number of weak links	265	203	233	163	184
	1000 searches (secs)	41.7	40.1	37.9	38.9	36.1
disjunction density 3 (den3)	const (secs)	270	1185	2192	3093	4214
	number of links	5396	7045	7040	6597	6430
	number of weak links	106	92	96	74	79
	1000 searches (secs)	36.8	37.7	41.1	37.3	45.7
disjunction density 5 (den5)	const (secs)	270	1188	4214	3178	4237
	number of links	5394	7180	6430	6719	6434
	number of weak links	69	62	79	39	48
	1000 searches (secs)	35.8	37.6	45.7	38.6	44.1

6.4.1.2 Number of links

The effect of disjunctive interval information on the number of solid links in a representation is directly attributed to the conversion of weak links to solid links (due to elements of disjunctions being proven) and the subsequent local minimisation of these new solid links. Thus the overall effect would be a minor increase in the number of solid links to be represented (the amount of “known” information) and a variable increase/decrease of the actual number of relations to be represented according to the local minimisation of these solid links.

The graphical representation of this data is given for three different initial densities of weak links and is compared to a control in Figure 6.12 for three different datasizes. The x-axis represents the initial density and the y-axis is the number of links added.

For each of the three different datasizes, the different data densities closely follow that of the control. All variation can be attributed to the effect of an increased number of solid links and the local minimisation performed when adding those solid links.

6.4.1.3 Weak links

The number of weak links used to represent disjunctions is dependent upon the number of disjunctions that can be added to the representation and the number removed due to inconsistencies and the proving of specific interval relations.

The graph in Figure 6.13 illustrates that, for higher densities of disjunctive interval relations (density 1 down to density 5), the number of weak links is greater than the lower densities. For each of the three different datasizes, three different densities of disjunctions are represented (four if you include the control that has no disjunctions) where the x-axis indicates the initial density and the y-axis indicates the number of weak links added.

There are minor fluctuations in each of the graphs, though it cannot be determined whether this is due to the relationship between the number of solid links and the number of disjunctions or simply the noise of the dataset. It is sufficient to say that there is no significant relationship between the density of the representation and the subsequent representation of the disjunctive interval relations as weak links.

6.4.1.4 Search time

The effect of the introduction of disjunctive interval relations on the search was significant. It required the recording of individual search paths at each point of the search in order to reduce redundant searches. It also required that checks be made against these records to identify these redundant searches. The disjunctive

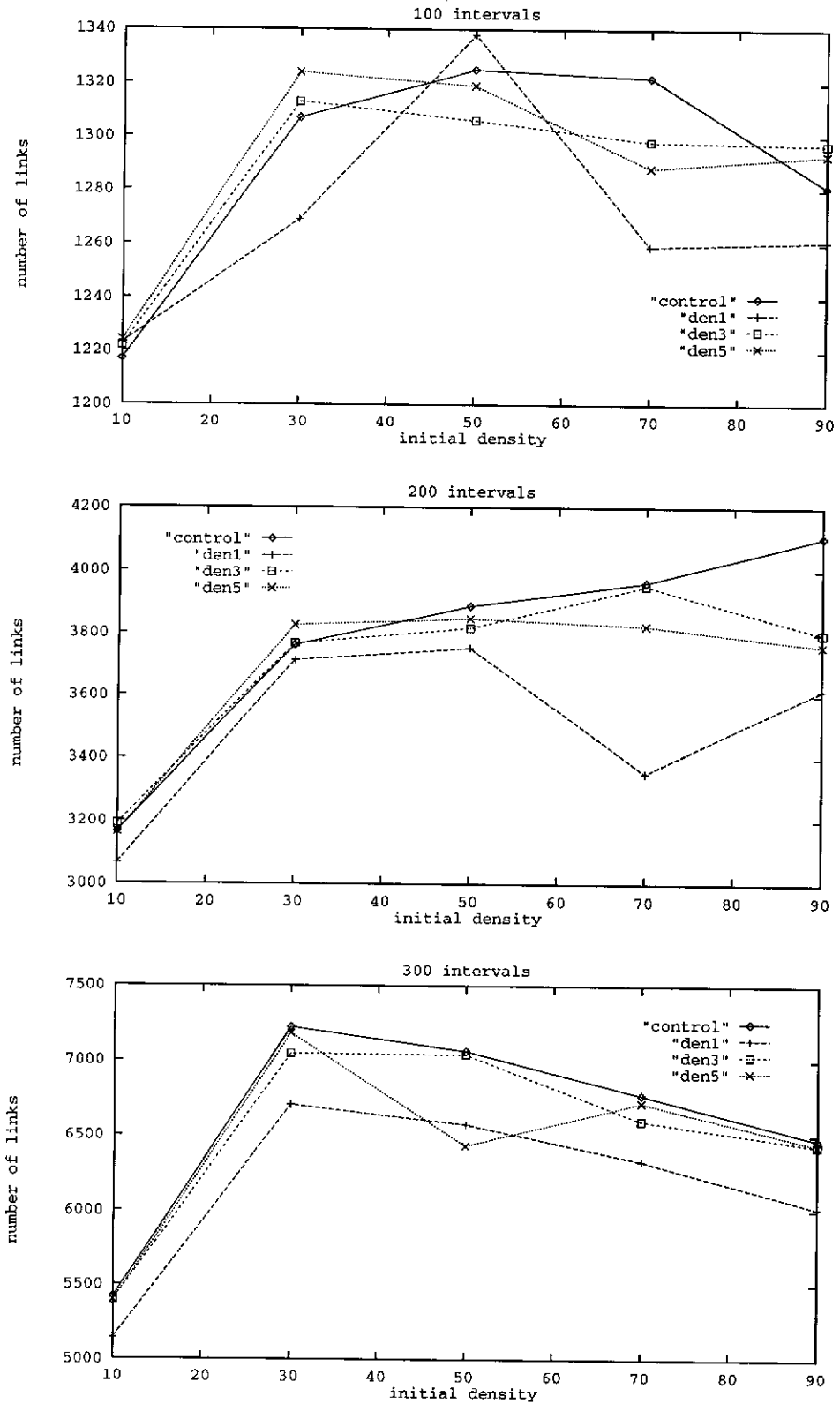


Figure 6.12: The performance of the primitive IR system vs the disjunctive IR system with respect to the number of solid links.

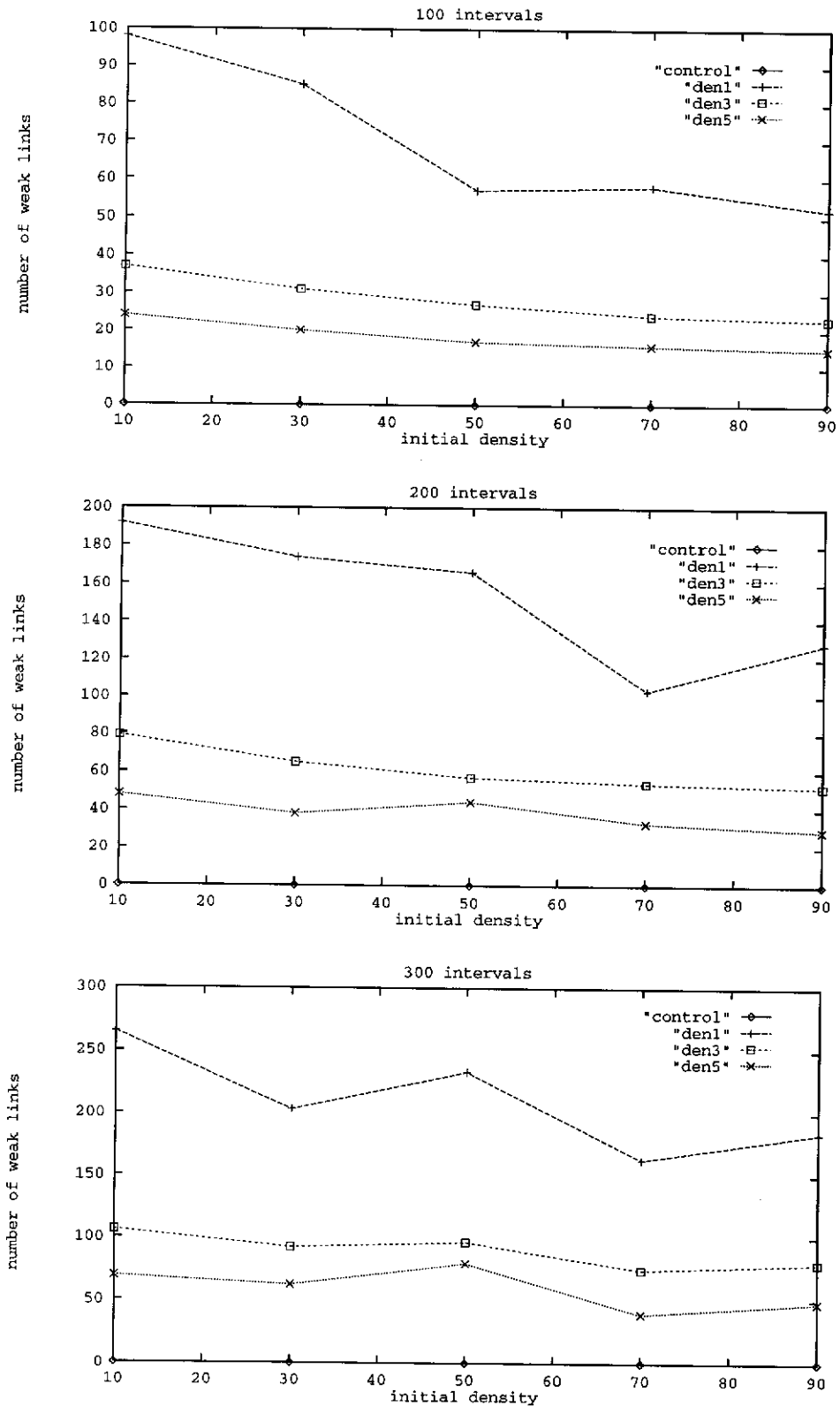


Figure 6.13: The performance of the primitive IR system vs the disjunctive IR system with respect to the number of weak links.

table would also be used to prevent the use of incompatible interval relations (members of the same disjunction) to be used as a solution. It also meant that there are multiple possible results to each individual search, and thus the search would not terminate upon the first solution but continue until all solutions are found.

A graphical comparison of the data for the search time for the three different densities of disjunctive interval relations and a control are given in Figure 6.14 for three different data sizes. The x-axis represents the number of links to be added and the y-axis indicates the search time.

Similar to the construction time, the difference between the different densities of disjunctions is much less than the difference between each density and the control. Again, this allows for the conclusion that the costs involved in maintaining the disjunctive relations are not proportional to the density of the disjunctions.

6.5 Summary

In this chapter we examined the problem of representing disjunctive interval representations. Each of the interval relations in the disjunctions can be represented as a conjunction of point relations. Each of these point relations are represented as a weak link in a graph. The disjunctions are also represented in a table (dtable) to correlate point relations to conjunctions and conjunctions to disjunctions.

Search of this representation involves constructing paths in the graph between two points. All paths that use weak links belonging to different conjunctions of the same disjunction are discarded. This is a property of the disjunctions where only one of the disjuncts may be true (exclusive-or).

A successful path may rely upon a number of weak links. Each of these paths has an associated dependency (a list of the conjunctions upon which it is dependent). The paths whose dependencies are supersets of other successful paths are redundant (dependent upon a greater set of relations than the other successful path). Our method of efficiently pruning the search also prevents the inconsistency caused by cycles.

A weak solution is a collection of all the successful paths between two given

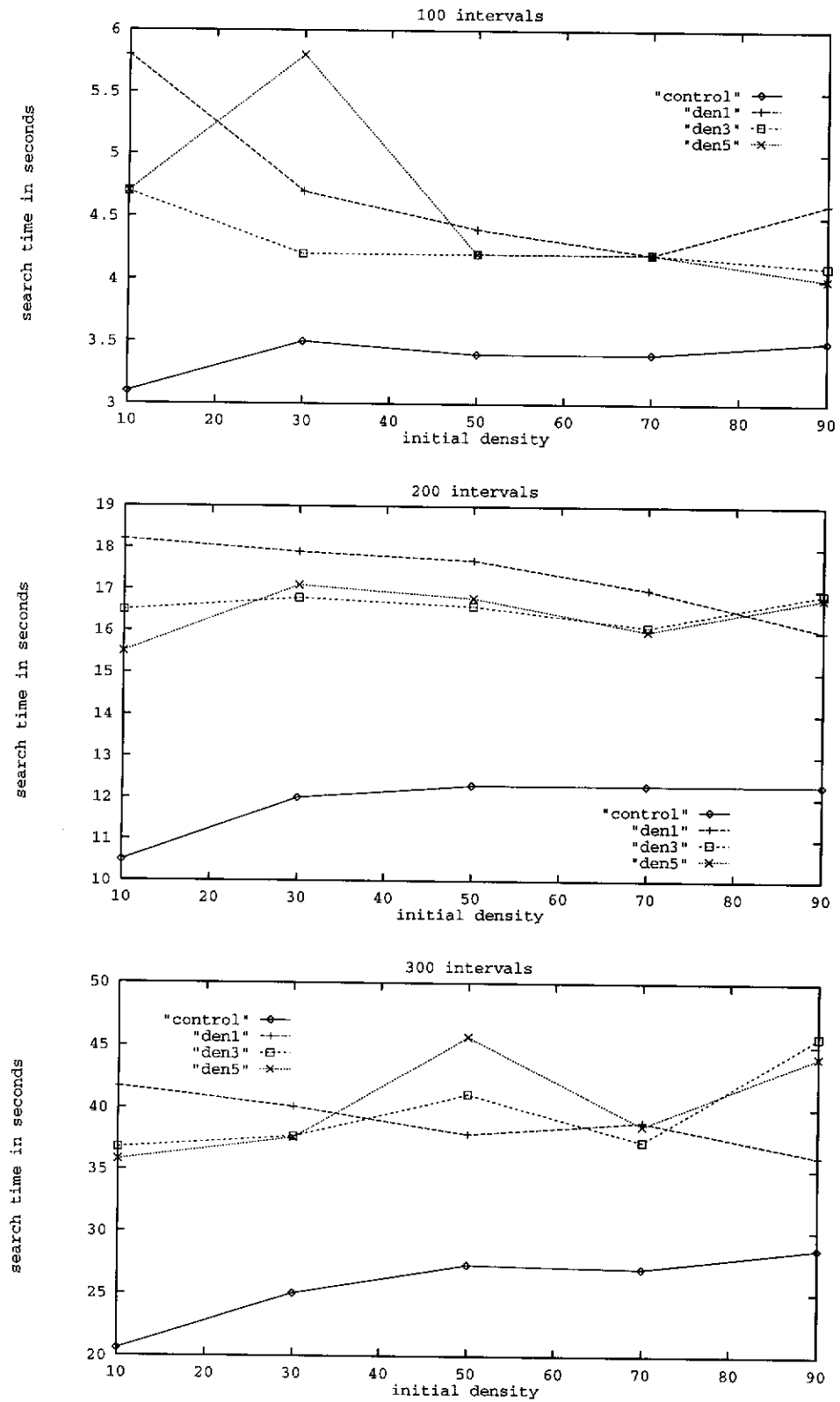


Figure 6.14: The performance of the primitive IR system vs the disjunctive IR system with respect to search time, where the x-axis is the initial density and the y-axis is the search time.

points. An interval relation is deduced from combinations of solutions to point queries. A combination of two dependent point solutions is obtained by finding consistent combinations from the dependencies of these two point solutions. Two paths are consistent if no dependency for a path is inconsistent with another dependency of another path, namely no pair of conjuncts (one element from each path) are members of the same disjunction. If no combination satisfies this constraint, then these point solutions cannot be combined. An interval relation can only be determined from the successful combination of specific point relations.

Our representation never requires construction of closure and never establishes the consistency of the entire representation. It only determines if there exists a solution to a given interval query that can be constructed from consistent information. For every possible solution it provides all constraints which must be satisfied in order for that solution to be consistent.

6.6 Conclusion

The problems of SAT, MLP and ACSP for disjunctive interval representations are NP-Complete. Our non-closure method has presented a means of determining solutions to queries about interval relations in polynomial time. Although the consistency of the entire representation cannot be determined, we can generate consistent solutions to queries. The solutions include the dependencies that indicate which relations must be true if the queried relation is true. All alternative consistent dependencies are generated as part of the solution.

This was achieved by extending the representation of the previous chapter to include disjunctive interval relations. As with the system described in Chapter 4, intervals and interval relations are represented in terms of points and point relations respectively. The disjunctive relations are represented by a disjunction table and weak links. The disjunctions are represented as disjunctions of conjunctions of point relations and these point relations as weak links in a graph. The restrictions imposed by the disjunctions are maintained by the table of disjunctive relations.

The search for point relations proceeds by determining which relations were

used to construct search paths and ensuring that the restrictions on which point relations can be used to construct a path (indicated in the disjunction table) are upheld. The efficient search of the representation using pruning techniques also prevents contradictory solutions from being generated, by not allowing cycles in the search paths.

The virtue of this type of solution is that, for representations where consistency cannot be determined in polynomial time, we can generate consistent solutions to individual queries about interval relations in polynomial time. We have presented experimental results for random datasets that show how this representation system compares to the system of the previous chapter. These results indicate that constructing the representation and determining consistent solutions for disjunctive interval representations is of a similar order to the incomplete non-disjunctive system.

Chapter 7

Conclusion

On two occasions I have been asked [by members of Parliament], ‘Pray, Mr. Babbage, if you put into the machine wrong figures, will the right answers come out?’ I am not able rightly to apprehend the kind of confusion of ideas that could provoke such a question.

Charles Babbage

7.1 Discussion

Qualitative representation of data places constraints upon the types of reasoning that may be performed upon that information. It provides the means of organising the data that allows simplistic answers to specific types of problems. The costs of representing and extracting this information are significantly less than representing and reasoning with the raw data.

Our choice of qualitative representation has been influenced by the types of information that are required from our representation. Our solution provides qualitative relational information about objects in multi-dimensional space.

A key part of the problem was the development of a system for representing this qualitative information. The presence of uncertainty in this information significantly increased the difficulty of the problem.

7.2 Summary

We began in Chapter 1 by separating the problem into two different components: the problem of representing spatial information as intervals, and the problem of representing these intervals as points.

Our review of the current literature involved examining the significant work in qualitative spatial reasoning. This review demonstrated the substantial parallels between the work in temporal reasoning and spatial reasoning in the use of points and intervals as the basic unit of representation. Issues unique to the spatial domain include multi-dimensionality and the use of global and relative axes.

The domain of temporal reasoning provided a large amount of background for the use of intervals and points as primitives for representing qualitative information. Allen [2] provided a powerful method of representing intervals and their relations using a system of closure. Due to the costs of performing this closure, alternative point-based closures have been proposed but none could capture the expressiveness of Allen's original work. Gervini [28] presented one of a few systems that proposed non-closure based representations. It provided an efficient method of representing interval relations that was nearly as expressive as Interval Algebra.

It was our intention to develop a system of multi-dimensional reasoning that represented objects in space as intervals along orthogonal axes. We would then represent these intervals using a non-closure point-based representation. These two systems could be integrated using a simple interval representation and then could be extended to problems that include uncertainty and were equivalent to Interval Algebra.

Space is dense and information about objects in space is subsequently vast. In order to organise this information, a set of assumptions were made about the types of information that would be required from the system. We then set about representing that information and determining how that information could then be extracted.

Our assumptions involved restricting the types of information that would be required about objects in space. Only qualitative relational information between

objects in space would be represented and queried. Information about precise shape, precise position and precise relation would not be required or made available by the system.

Objects in space were reduced to interval approximations along global orthogonal axes. The relationship between objects was then expressed in terms of their interval relations along each of these axes. These relations were developed around Allen's interval-based temporal model but taking into consideration the unique properties of space. Two of these properties, the lack of global flow in space and the multi-dimensionality of space, were used to produce a method of classifying relations at different levels of granularity. A querying language was developed to assist in the formation of queries about object relations in space. Simple boolean operators and variable binding extended the vocabulary of the querying language.

This led to the problem of how to represent the intervals along each dimension. Intervals were associated in terms of interval relations but it was not assumed that the data would be complete. Thus the problem was divided into three different types of interval representations: for a set of intervals where

- the relationships between every pair of intervals was explicitly known and those relations were primitive interval relations,
- the relationships between every pair of intervals was not explicitly known and those that were known were primitive interval relations,
- relations were disjunctions of primitive interval relations, regardless of whether all interval relations were known or unknown.

To demonstrate the qualitative reasoning system of Chapter 3, the simplest representation (where all pairs are explicitly known and those relations are primitive interval relations) was represented using a linear ordering of the end points of the intervals in Chapter 4. Queries about individual interval relations can be extracted from the representation by determining the relationships between particular end points.

For two datasets, one artificial and the other real, the process of performing queries upon these datasets was demonstrated. Individual multi-dimensional in-

terval queries (from Chapter 3) were converted into 1-D interval queries, which were in turn converted into point queries. The results of these point queries could be used to determine the actual interval relations required by the interval query.

Though the interval representation system given in Chapter 4 was simplistic, it did demonstrate the ability of the point representation of intervals to provide results for the interval query system developed in Chapter 3. In Chapters 5 and 6 this system was extended to allow for a larger variety of interval information to be represented.

The first stage of this expansion was to allow for incompleteness about interval relations (Chapter 5). For interval representations of sets of intervals where the relationship between every pair of intervals may not be known but those that are known are primitive interval relations, a graph-based representation of end points of intervals was introduced. Using a minimisation strategy, the information about end points could be efficiently represented (space) with minimal increase in the construction time. The search time for this representation is comparable to full minimisation. The system also provided a deductive foundation for the generation of Allen's transitivity table.

To increase the expressiveness of this system, disjunctions between intervals were represented by a new system in Chapter 6. This allowed for the representation of sets of intervals whose explicit relations were disjunctive interval relations. This was achieved by allowing disjunctions between interval relations to be represented along side the primitive interval relations. The restrictions imposed by the disjunctions were maintained by a table of disjunctions.

This led to a significant increase in the complexity of the search process. No longer were relations simply known or unknown. Relations could now be non-monotonic. Thus a search between two points would produce multiple possible solutions. Special methods were used to reduce the redundancy in the search.

Solutions to searches of individual point relations could be combined to determine interval relations only if the individual solutions were not dependent upon contradictory information. Individual solutions could consist of multiple alternative dependencies and thus a combination could be constructed between any pair

of dependencies (each from a different solution).

The performance of this system was evaluated against the system in Chapter 5. This disjunctive representation does not determine the consistency of the entire representation in polynomial time (an NP-Complete problem), but it provides consistent solutions to individual queries in polynomial time. Thus responses to queries about interval relations would indicate both the possible “relation” and any “dependencies” this relation was dependent on.

7.3 Future Directions

The greatest restriction to this system are the choices made about the types of information to be represented and queried about the system. Future developments would involve the integration of this system with quantitative systems and other qualitative systems that represent distances between and intersections of objects in space.

Freksa’s work [25] in the dynamic change of interval relations and the issues of deictic axes are both directions relevant to this thesis. The possible applications of the system defined in this thesis are many and varied. Problems ranging from Geographic Information Systems, Pictorial and Video Indexing, Robotic Navigation, DNA sequencing and Historical dating are just a few key areas of applications that may be investigated.

Appendix A

Propagating order

$\forall a$ and $\forall b$ being limit points of intervals A and B respectively, if $a \xi b$ then

$\exists X \text{ interval}(X) \mid a = B(X) \text{ and } b = E(X)$

where ξ is the symbol for the relation precedes.

The relations between the points, illustrated in Figure A.1, describe all the possible ways that **a precedes b**. We note that the situation illustrated in Figure A.1i subsumes the situation in Figure A.1ii, i.e. $B(A) < B(B) \Rightarrow B(A) < E(B)$ because $B(K) < E(K)$ holds for every interval K. Similarly Figure A.1iii subsumes the situation in Figure A.1iv. $E(A) \xi E(B)$ does not necessarily mean that $E(A) \xi B(B)$ because we could have $B(B) \xi E(A)$. Thus an expression about the ordering

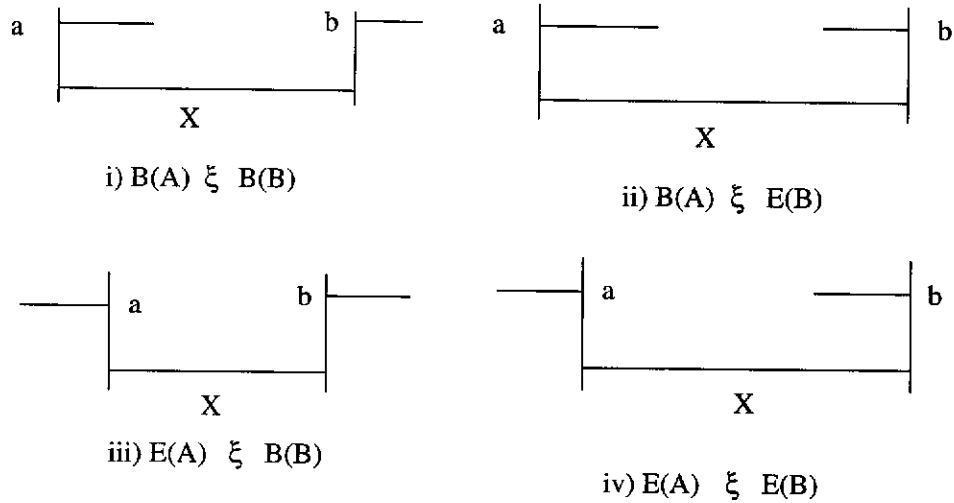


Figure A.1: Four different situations where point *a* precedes point *b*, indicated by the interval *X*

of the end points of any two intervals could be described solely in terms of the relations depicted in Figure A.1i and Figure A.1iii.

Appendix B

End points

The ordered A-Space determines how the ordering of the intervals is designated. The interval ordering is identified by assigning the first limit point to be the **begin point** and the second limit point as the **end point** (the first and second are the ordering given with respect to the A-Space).

An interval **Y** in the A-Space **A** (see Figure B.1i) has the **begin point** assigned as in Figure B.1ii). The **end point** is, for a convex interval, the other limit point of the interval. An invalid allocation of the ordering of the interval is given in Figure B.1iii). This situation can be identified by the existence of a point (**p**) along the interval **Q** that is equal to a limit point of the interval **Y**.

The formal definition is as follows:

For any interval **Y**, the limit point designations for this interval are assigned with respect to the ordered A-Space (**A**) as follows (see Figure B.2):

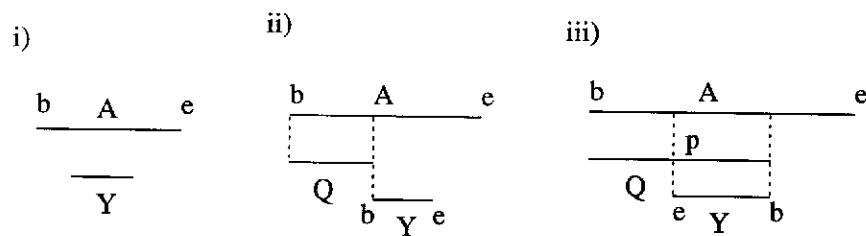


Figure B.1: i) An interval **Y** in the A-Space **A**; ii) the interval **Y** with a specific ordering; and iii) and incorrect ordering with respect to the A-Space.

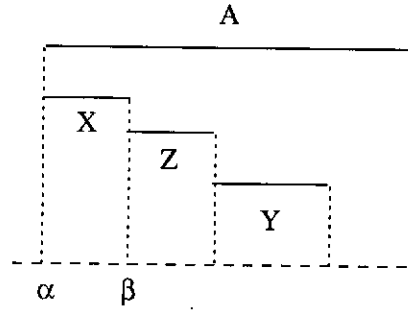


Figure B.2: Labelling an interval

Let $B = \text{limit}(Y)$

if $b(A) = B$ **then**

$b(Y) = B$ and $e(Y) = \text{limit}(Y)$ where $e(Y) \neq B$

else if $e(A) = B$

$e(Y) = B$ and $b(Y) = \text{limit}(Y)$ where $b(Y) \neq B$

else

$\exists X, \exists Y$ and $\exists Z$ are intervals

$\alpha = B(A) = \text{limit}(X)$

$\beta = \text{limit}(X) = \text{limit}(Z)$ where $\text{limit}(X) \neq \alpha$

$\text{limit}(Z) = \text{limit}(Y) = B(Y)$ where $\text{limit}(Z) \neq \beta$ and $\neg \exists \beta = \text{limit}(Y)$

$\text{limit}(Y) = E(Y)$ where $\text{limit}(Y) \neq B(Y)$

Bibliography

- [1] A. Aho, M. Garey, and J. Ullman. The transitive reduction of a directed graph. *SIAM J. Comput.*, 1(2):131–137, 1972.
- [2] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.
- [3] J. F. Allen and P. J. Hayes. Moments and points in interval-based temporal logic. *Computational Intelligence*, 5:225–238, 1989.
- [4] R. Auerbach, B. Lin, and E. Elsayed. Layout aid for the design of VLSI circuits. *Computer-Aided Design*, 13(5):271–276, 1981.
- [5] Y. Borisovich, N. Bliznyakov, A. Izrailevich, and T. Fomendo. *Introduction to Topology*. Mir Publishers, 1985.
- [6] C. C. Chang and S. Y. Lee. Retrieval of similar pictures on pictorial databases. *Pattern Recognition*, 24(7):675–680, 1991.
- [7] S. Chang and E. Jungert. A spatial knowledge structure for image systems using symbolic projections. In *Proceedings of the Fall Joint Computer Conference*, pages 79–86, Dallas, TX, November 1986.
- [8] S. Chang and E. Jungert. Pictorial data management based upon the theory of symbolic projections. *Journal of Visual Languages and Computing*, 2(3):195–215, 1991.
- [9] S. Chang, E. Jungert, and Y. Li. The design of pictorial databases based upon the theory of symbolic projections. In A. Buchmann, O. Guenther,

- T. Smith, and Y. Wang, editors, *Symposium on the Design and Implementation of Large Spatial Databases, Lecture Notes in Computer Science*, volume 409, pages 303–323. Springer-Verlag, New York, NY, 1989.
- [10] S. K. Chang, E. Jungert, and Y. Li. Representation and retrieval of symbolic pictures using generalized 2D strings. In *SPIE vol. 1199 Visual Communications and Image Process IV*, pages 1360–1372, 1989.
 - [11] S. K. Chang, Q. Y. Shi, and C. W. Yan. Iconic indexing by 2-D strings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(3):413–428, 1987.
 - [12] C. Y. Chen and C. C. Chang. An object-orientated similarity retrieval algorithm for iconic image databases. *Pattern Recognition Letters*, 14(7):465–470, June 1993.
 - [13] A. Del Bimbo, M. Campanai, and P. Nesi. A three-dimensional iconic environment for image database querying. *IEEE Transactions on Software Engineering*, 19(10), 1993.
 - [14] J. Doenhardt and T. Lengauer. Algorithmic aspects of one-dimensional layout. *IEEE Transactions on Computer-Aided Design*, CAD-6(5):863–878, 1987.
 - [15] J. Dorn. Temporal reasoning in sequence graphs. In *Proceedings of the Tenth National American Association for Artificial Intelligence AAAI-92*, pages 735–740, 1992.
 - [16] S. Dutta. Qualitative spatial reasoning: A semi-quantitative approach using fuzzy logic. In A. Buchmann, O. Guenther, T. Smith, and Y. Wang, editors, *Symposium on the Design and Implementation of Large Spatial Databases, Lecture Notes in Computer Science*, volume 409, pages 344–364. Springer-Verlag, New York, NY, 1989.
 - [17] S. Dutta. Approximate spatial reasoning: Integrating qualitative and quantitative constraints. *International Journal of Approximate Reasoning*, 5(3):307–330, May 1991.

- [18] M. Egenhofer. A formal definition of binary topological relationships. In W. Litwin and H. Schek, editors, *Foundations of Data Organization and Algorithms.*, volume 367 of *Lecture Notes in Computer Science*, pages 457–472. Springer-Verlag, New York, 1989.
- [19] M. Egenhofer. Reasoning about binary topological relations (extended abstract). In *Advances in Spatial Databases Symposium*, pages 143–160, 1991.
- [20] M. Egenhofer, E. Clementini, and P. Di Felice. Topological relations between regions with holes. *International Journal of Geographical Information Systems*, 8(2):129–142, 1994.
- [21] M. Egenhofer, A. Frank, and J. Jackson. A topological data model for spatial databases. In A. Buchmann, O. Gunther, T. Smith, and Y. Wang, editors, *Design and Implementation of Large Spatial Databases*, volume 409 of *Lecture Notes in Computer Science*, pages 271–286, New York, 1989. Springer-Verlag.
- [22] M. Egenhofer and R. Franzosa. Point-set topological spatial relations. *International Journal of Geographical Information Systems*, 5:161–174, 1991.
- [23] M. Egenhofer and J. Herring. A mathematical framework for the definition of topological relationships (extended abstract). In *Proceedings of the 4th International Symposium on Spatial Data Handling*, pages 803–813, 1990.
- [24] C. Freksa. Qualitative spatial reasoning. In D. M. Mark and A. U. Frank, editors, *Cognitive and Linguistic Aspects of Geographic Space*, pages 361–372, Dordrecht, 1991. NATO Advanced Studies Institute, Kluwer Academic.
- [25] C. Freksa. Temporal reasoning based on semi-intervals. *Artificial Intelligence*, 54:199–227, 1992.
- [26] C. Freksa. Using orientation information for qualitative spatial reasoning. In *Proceedings of the International Conference GIS - From Space to Territory: Theories and Methods of Spatio-Temporal Reasoning*, 1992.

- [27] C. Freksa and K. Zimmermann. On the utilization of spatial structures for cognitively plausible and efficient reasoning. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Chicago, October 1992. IEEE.
- [28] A. Gerevini and L. Schubert. Efficient algorithms for qualitative reasoning about time. *Artificial Intelligence*, 74(2), 1995.
- [29] A. Gerevini and L. Schubert. On computing the minimal labels in time point algebra networks. *Computational Intelligence Journal*, 74(2), 1995.
- [30] A. Gerevini, L. Schubert, and S. Schaeffer. Temporal reasoning in timegraph I-II. *SIGART Bulletin*, 4(3):21–25, 1993.
- [31] M. Ghallab and A. Mounir Alaoui. Managing efficiently temporal relations through indexed spanning trees. In *Eleventh International Joint Conference on Artificial Intelligence*, volume 2, pages 1297–1303, August 1989.
- [32] M. C. Golumbic and R. Shamir. Algorithms and complexity for reasoning about time. In *Proceedings of the Tenth National American Association for Artificial Intelligence AAAI-92*, pages 741–747, 1992.
- [33] D. G. Green. Emergent behaviour in biological systems. In D. Green and T. Bossomaier, editors, *Complex systems - From Biology to Computation*, pages 24–35. IOS Press, Amsterdam, 1993.
- [34] H. Guesgen. Four-dimensional reasoning. In *6th International Symposium on Expert Systems*, pages 99–103, 1989.
- [35] H. W. Guesgen. Spatial reasoning based on Allen’s temporal logic. Technical Report TR-89-049, International Computer Science Institute, 1947 Center Street, Suite 600 Berkeley California, 1989.
- [36] D. Hernandez. Relative representation of spatial knowledge: the 2D case. In D. M. Mark and A. U. Frank, editors, *Cognitive and Linguistic Aspects of Geographic Space*, pages 373–385, Dordrecht, 1991. NATO Advanced Studies Institute, Kluwer Academic.

- [37] D. Hernandez. Qualitative representation of spatial knowledge. Doctoral dissertation, Technische Universitat Munchen, 1992.
- [38] R. Jeansoulin. Basic tools for spatial logic. *Cognitiva*, 90:329–336, 1991.
- [39] W. Kainz. Spatial representations - topology versus order. In *Proceedings of the 4th International Symposium on Spatial Data Handling*, pages 814–819, 1990.
- [40] K. Kuratowski and A. Mostowski, editors. *Set Theory*. North-Holland, Amsterdam, 1968.
- [41] J. La Poutre and J. van Leeuwen. Maintenance of transitive closures and transitive reduction of graphs. In H. Gottler and H. Schneider, editors, *Graph-Theoretic Concepts in Computer Science*, volume 314, pages 106–120. Springer-Verlag, 1987.
- [42] P. Ladkin. Time representation: A taxonomy of interval relations. In *Proceedings of the 5th National Conference on Artificial Intelligence*, pages 360–366. AAAI, Morgan Kaufman, 1986.
- [43] P. Ladkin and R. Maddux. On binary constraint problems. *Journal of the Association for Computing Machinery*, 41(3):435–469, 1994.
- [44] P. Ladkin and A. Reinefeld. A symbolic approach to interval constraint problems. In J. Calmet and J. Campbell, editors, *Artificial Intelligence and Symbolic Mathematical Computing*, volume 737 of *Lecture Notes in Computer Science*, pages 65–84. Springer-Verlag, 1993.
- [45] P. B. Ladkin. Constraint reasoning with intervals: A tutorial, survey and bibliography. Technical Report TR-90-059, International Computer Science Institute, 1947 Center St., Suite 600, Berkeley, CA 94704-1105, USA, November 1990.
- [46] S. Lee and F. Hsu. 2D C-string: A new spatial knowledge representation for image database systems. *Pattern Recognition*, 23(10):1077–1087, 1990.

- [47] S. Lee and F. Hsu. Picture algebra for spatial reasoning of iconic images represented in 2D C-string. *Pattern Recognition Letters*, 12(7):425–435, July 1991.
- [48] S. Lee, M. Shan, and W. Yang. Similarity retrieval of iconic image database. *Pattern Recognition*, 22(6):675–682, 1989.
- [49] S. Y. Lee and F. J. Hsu. Spatial reasoning and similarity retrieval of images using 2D C-string knowledge representation. *Pattern Recognition*, 25(3):305–318, 1992.
- [50] S. Y. Lee, M. C. Yang, and J. W. Chen. Signature file as a spatial filter for iconic image database. *Journal of Visual Languages and Computing*, (3):373–397, 1992.
- [51] T. Levitt and D. Lawton. Qualitative navigation for mobile robotics. *Artificial Intelligence*, 44:305–360, 1990.
- [52] A. Mackworth. Consistency in network of relations. *Artificial Intelligence*, 8:99–118, 1977.
- [53] E. M. Markman and J. Seibert. Classes and collections; internal organisation and resulting holistic properties. *Cognitive Psychology*, 8:561–577, 1976.
- [54] R. Mohr and T. Henderson. Arc and path consistency revisited. *Artificial Intelligence*, 28:225–233, 1986.
- [55] A. Mukerjee and G. Joe. A qualitative model for space. In *AAAI 8th conf*, pages 721–727, 1990.
- [56] M. Nelson. Doctor of philosophy examiners report. September 1997.
- [57] D. Papadias and T. Sellis. The semantics of relations in 2D space using representative points: Spatial indexes. In A. Frank and I. Campari, editors, *Spatial Information Theory: A Theoretical Basis for GIS.*, volume 716 of *Lecture Notes in Computer Science*, pages 234–247. Springer-Verlag, 1993.

- [58] D. J. Peuquet. The use of spatial relations to aid spatial database retrieval. In *Proceedings of the Second International Symposium on Spatial Data Handling*, pages 459–471. 1986.
- [59] R. Rajagoplan. A model of spatial position based on extremal points. In *Workshop on Advances in Geographical Information Systems*, Arlington, Virginia, 1993.
- [60] G. Retz-Schmidt. Deictic and intrinsic use of spatial prepositions: A multidisciplinary comparison. In A. C. Kak and S. Chen, editors, *Spatial Reasoning and Multi-Sensor Fusion: Proceedings of the 1987 Workshop*, held in Pheasant Run Resort, St. Charles, Illinois, USA, pages 371–380. AAAI, Morgan Kaufmann Publishers, Inc., October 1987.
- [61] R. Rohrig. Qualitative spatial reasoning based on order relations. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 1418–1423, 1994.
- [62] R. Sedgewick. *Algorithms*. Addison-Wesley, Reading, Massachusetts, second edition, 1989.
- [63] D. Seeley and S. Ronald. The emergence of connectivity and fractal time in the evolution of random graphs. In *From Biology to Computation: Proceedings of the First Australian Conference on Complex Systems*. IOS Press, 1992.
- [64] L. A. Steen and J. A. Seebach Jr. *Counterexamples in topology*. Holt, Rinehart and Winston, New York, 1970.
- [65] J. Stillman, R. Arthur, and A. Deitsch. Tachyon: A constraint-based temporal reasoning model and its implementation. *SIGART Bulletin*, 4(3):T1–T4, 1993.
- [66] A. Tarski. On the calculus of relations. *Journal of Symbolic Logic*, 6:73–89, 1941.

- [67] P. van Beek. Approximation algorithms for temporal reasoning. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 1291–1296. AAAI, 1989.
- [68] P. van Beek. Reasoning about qualitative temporal information. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 728–734. AAAI, 1990.
- [69] P. van Beek. Reasoning about qualitative temporal information. *Artificial Intelligence*, 58:297–326, 1992.
- [70] P. van Beek and R. Cohen. Exact and approximate reasoning about temporal relations. *Computational Intelligence*, 6:132–144, 1990.
- [71] P. van Beek and D. W. Manchak. The design and an experimental analysis of algorithms for temporal reasoning. Technical report, University of Alberta.
- [72] Y. Venema. Expressiveness and completeness of an interval tense logic. *Notre Dame Journal of Formal Logic*, 31(4):529–547, Fall 1990.
- [73] Y. Venema. A modal logic for chopping intervals. *Journal of Logic Computation*, 1(4):453–476, 1991.
- [74] M. Vilain. A system for reasoning about time. In *National Conference on Artificial Intelligence*, pages 197–201. AAAI, 1982.
- [75] M. Vilain, H. Kautz, and P. van Beek. Constraint propagation algorithms for temporal reasoning: A revised report. In D. S. Weld and J. de Kleer (eds.), editors, *Readings in Qualitative Reasoning about Physical Systems*, pages 373–381. Morgan Kaufmann, San Mateo, California, 1990.
- [76] E. Yampratoom and J. F. Allen. Performance of temporal reasoning systems. *SIGART Bulletin*, 4(3):26–29, 1993.
- [77] M.-C. Yang. 2D B-String representation and access methods of image database. Master’s thesis, National Chiao Tung University, Department of Computer Science and Information Engineering, National Chiao Tung University, Hsinchu, Taiwan, July 1990.

- [78] K. Zimmerman and C. Freksa. Enhancing spatial reasoning by concept of motion. In *Proceedings of the AISB'93 Conference*. University of Birmingham, March–April 1993.