Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 99-027

Equivalence Classes of Direction Objects and Applications

Shashi Shekhar, Xuan Liu, and Sanjay Chawla

July 21, 1999

Equivalence Classes of Direction Objects and Applications*

Shashi Shekhar, Xuan Liu, Sanjay Chawla
Computer Science Department, University of Minnesota
EE/CS 4-192, 200 Union St. SE., Minneapolis, MN 55455
telephone: (612)624-8307
[*shekhar*|*xliu*|*chawla*]@*cs.umn.edu*
http://www.cs.umn.edu/research/shashi-group

Abstract

Direction is an important spatial relationship that is used in many fields such as geographic information systems(GIS) and image interpretation. It is also frequently used as a selection condition in spatial queries. In our recent work we have described a novel viewpoint to model direction as a 'spatial object' based upon the concepts of vectors, points and angles. This was a departure from the conventional approach of treating direction as a spatial relationship between objects. In this paper, based upon 'direction objects', we partition the directional space into a set of equivalence classes. By defining an algebra on equivalence classes we provide a framework to model semantics of direction predicates for qualitative spatial reasoning. We then proceed to extrapolate the definition of direction equivalence classes to define 'path' equivalence classes with an application to the landmark based route description problem.

Keywords: Directional relationships, Direction objects, Equivalence classes, Landmark based routing

[1] This work is sponsored in part by the Army High Performance Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory cooperative agreement number DAAH04-95-2-0003/contract number DAAH04-95-C-0008, the content of which does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred. This work was also supported in part by NSF grant #9631539

1 Introduction

1.1 Direction is important

Direction is a common spatial concept that is used everywhere in daily life. When people communicate about a geographic space, such as giving route descriptions[12], direction is necessary to convey the information. Direction is also frequently used as a selection condition in spatial databases[11, 3] or used for similarity accessing in image databases[17]. Example queries used in army battlefield visualization[10] are "Is there anything *over* the ridge?," "Put me in the tank *left* of that building," and "Let's move to the *north* of the tree." The first example refers to a viewer-based orientation, the second can be defined on either the intrinsic orientation of the building(object-based) or a viewer, and the third example refers to the absolute direction with respect to the tree. In order to handle the kind of queries that contain direction constraints in the selection criteria, a spatial database system should provide a way for users to model directions.

The common means of dealing with direction in GIS is to model direction as a relationship between objects[4, 19, 8, 22, 3, 16, 9, 6, 20, 15]. We view direction from a different perspective : as a spatial object. The basic approach is to model direction as a unit vector, and orientation as a set of directions. As a spatial object, direction can have its own attributes, its own operators and predicates. New spatial data types such as oriented spatial objects and unbounded spatial objects can be easily defined using direction and orientation object.

In this paper we introduce the notation of equivalence classes of direction objects to model direction predicates. The object view of direction simplifies reasoning with direction predicates as it reduces the large number of inference rules that is commonly needed. This is useful in the processing and optimization of spatial queries that contain direction constraints. Many axioms for the qualitative reasoning with direction predicates in previous work can now be understood easily in terms of direction equivalence classes. In addition, new reasoning with object-based direction predicates can be performed with our results. we will also illustrate how to apply our direction model to generate the route description based on landmarks with direction predicates. Lots of public servers provide route descriptions to users based on a road map with named roads and they give the route descriptions, such as: "Turn left onto Washington Avenue". The problem gets complicated if the given roadmap has only named landmarks, but has no named roads, such as campus maps, or skyway maps. The challenge here is that the route descriptions should be given by only using the named landmarks with proper direction objects.

1.2 Related work and our contributions

The research work on direction modeling has been carried out in several areas such as geographic information systems and image analysis. Most of the work is on how to capture the semantics of direction relationships, and further, how to do spatial reasoning on the direction[3, 4, 6]. There are two major direction reference frames used to model direction in 2D space: the cone-based model[16], and the projection-based model[4, 6]. Frank[1] compared these two models and found the projection-based reference frame to be better in many aspects. Most attention has been paid to point-based objects.

The most common way to model directions between extended objects is through the object's Minimum Bounding Rectangle(MBR), where direction relationship are obtained by applying Allen's [2] interval relations along the x and y axis, in which case, 169 different relationship[3] can be distinguished. some work based on MBR has been proposed on picture indexing in pictorial databases[17, 22] and some work aligns each boundary box to the object's major axis[13], which makes it possible to satisfy different reference frames[9]. Freska [5] proposed an alternative method: semi-intervals to formalize the one-dimensional temporal relation based on incomplete knowledge of the object. Goyal and Egenhofer [8] introduced a Direction-Relation Matrix to represent cardinal directions. Based on the projection-based frame, it partitions the space around the reference object and records into which direction tiles a target object falls. But this model still has limitations in the modeling of line objects, and it is limited to 2D space. Little work has been done on directions in 3D space [7].

The previous work modeled direction as a boolean spatial relationship between spatial objects. This seems to be a natural mapping of direction relations that is used in geographic space. But this modeling method has some limitations. Operations on direction are limited, Oriented(directed) objects and unbounded objects cannot be represented in the spatial data model. A new object direction model was proposed by us in [21] and is summarized in appendix A for convenience of reviewers. The basic approach is to model direction as a unit vector. Being modeled as a spatial object, a direction object can have its own attributes and operation set. The implementation of operators can use vector algebra, making a richer set of predicates and operators on direction feasible. Secondly, new spatial data types such as oriented objects and unbounded objects can be defined at the abstract object level.

In this paper, we extend the object model of direction by defining the equivalence classes of direction objects, so that the qualitative direction predicates are formalized as direction objects. Also, algebra is defined on the equivalence classes to formalize reasoning with direction predicates. The properties of direction predicates can be understood easily in terms of the properties of direction equivalence classes. Many Lemmas discussed in our paper were used as desirable properties(axioms) of direction predicate system in previous work, whereas these are derived properties from equivalence class of direction objects in our paper. Lemma 7 shows the lack of closure under composition, which was often assumed in previous work [4]. We show a simple reason for this using the *between* operator on direction object. Using orientation object reasoning with object-based direction predicates can be defined. Our direction model also provide a complete set of path predicates for landmark based route description.

1.3 Scope and Outline of Paper

The organization of this paper is as follows: In section 2, We introduce the basic concept of direction equivalence classes and its properties. In section 3, we do direction reasoning in both viewer-based and object-based system. Properties of direction predicates are also characterized by a set of lemmas. Finally, in future work, we apply our direction model to the landmark based route description problem, proposing a set of path predicats.

2 Basic Concepts

We have previously introduced the concept of 'direction object' [21]. Instead of modeling a direction as a relationship between two entities we model it as an object in itself. A 'direction object' represents a unit vector in the plane. We decompose the space into disjoint partitions and identify each vector with the partition in which it lies. All vector in a particular partition are considered 'equivalent'. We define an algebra on the set of partitions and use the algebraic operations to solve 'equations', where the variables are 'direction symbols'.

2.1 Direction and Orientation Object

Direction is defined as a unit vector, i.e., a vector with its magnitude equal to 1. Table 1 defines the operations on directions, using vector algebra. A brief review of vector algebra is provided in A.1.

Operations	Definition
composition	$\vec{d1} + \vec{d2} = \frac{\vec{d1}+\vec{d2}}{\lvert\vec{d1}+\vec{d2}\rvert}$ (vector addition)
deviation	$cos\theta = \vec{d_1} \odot \vec{d_2}$ (vector dot product)
reverse	$(-1) \times \vec{d1}$ (scalar product)
between[1]	$\vec{d}$ between $\vec{d_1}$ and $\vec{d_2}$ if $\exists c_1 > 0, c_2 > 0$ s.t. $\vec{d} = c_1\vec{d_1} + c_2\vec{d_2}$
among	$\vec{d}$ among $\vec{d_1}$, $\vec{d_2}$ and $\vec{d_3}$ if $\exists c_1 > 0, c_2 > 0, c_3 > 0$ s.t. $\vec{d} = c_1\vec{d_1} + c_2\vec{d_2} + c_3\vec{d_3}$

Table 1: Operations of Direction

The operations on directions can be classified into three categories. The first category is the operations that produce new directions. *Composition* and *reverse* are operations in this category. The *composition* operation is actually achieved by *vector-addition*, and the resulting vector is scaled down by its magnitude to be a unit vector, which represents the new direction. The *reverse* operator produces the reverse direction vector. The second category is to calculate the deviation between two directions. Operator *deviation* calculates the cosine of the angle between two directions, and hence gives the direction deviation of one direction from the other. A pair of vectors are orthogonal if their dot-product returns zero, i.e., they have 90^0 deviation. The last category is to test the relationship among directions. The operators *between* and *among* belong to this category. In figure 1, $\vec{d}$ is between $\vec{d_1}$ and $\vec{d_2}$; however, $\vec{d_1}$ is not between $\vec{d}$ and $\vec{d_2}$. As we will see in later sections, these three categories of direction operations make the modeling of direction concise and flexible.

Orientation is modeled as a spatial object which consists of a point of origin and N pair-wise orthogonal directions, where N is the dimension of the embedding space. The origin point and the N directions form a Cartesian coordinate system. In 3D space, the three directions may be labeled the Back-Front, Left-Right, and Below-Above directions of the orientation. Formally, we can define orientation and its operations in 3D space as follows:

[1] This definition works well as long as vector $\vec{d_1}$ is not parallel to vector $\vec{d_2}$. The parallel case can be handled in a user defined manner.

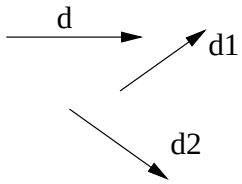


Figure 1: between operator

Orientation is a quadruple O= ⟨OP, $\vec{front}$, $\vec{right}$, $\vec{above}$⟩, where OP is a point, and $\vec{front}$, $\vec{right}$, $\vec{above}$ are three orthogonal directions. It has two operations:

- $translate(O, \vec{v}) = Orientation\ O' = \langle translate(OP, \vec{v}), \vec{front}, \vec{right}, \vec{above}\rangle$;
- $rotate(O, rotationMatrix)$ = $Orientation$ O' = $\langle OP, \vec{front}_n, \vec{right}_n, \vec{above}_n\rangle$, where $(\vec{front}_n, \vec{right}_n, \vec{above}_n)$= $(\vec{front}, \vec{right}, \vec{above})\odot\ rotationMatrix$;

An example of the rotationMatrix which rotates the orientation along the $\vec{above}$ axis for an angle θ is[18]:

$$R^{\theta}_{\vec{above}} = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Table 2 gives an illustration of these operations.

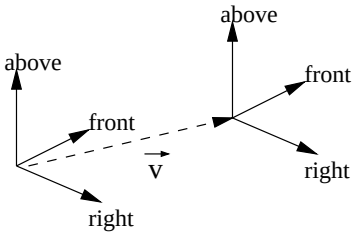
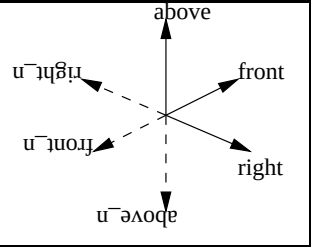
translate(O, $\vec{v}$)	rotate(O, $R^{180^0}_{\vec{above}}$)
	

Table 2: Operations on orientation

2.2 Equivalence Classes of Direction Objects

There are several ways of partitioning the space of directions. Figure 2 illustrates example partitionings for a two-dimension space of direction unit vectors. Figure 2a and 2b represent the common cone-shape divisions[16]. Figure 2a divides direction unit vectors into 4 groups, namely, $0^0 \leq \theta < 90^0$, $90^0 \leq \theta < 180^0$, $180^0 \leq \theta < 270^0$ and $270^0 \leq \theta < 360^0$, where θ denotes the angle from a coordinate axis parallel to the edge of sheet of paper to the direction vector. An alternative partitioning imposed by Figure 2a may have eight group, namely, 0^0, 90^0, 180^0, 270, $0^0 < \theta < 90^0$, $90^0 < \theta < 180^0$, $180^0 < \theta < 270^0$ and $270^0 < \theta < 360^0$. The term "partition" is used in a set-theoretical sense, i.e., partitions are pair-wise disjoint and together they cover the space of all direction unit vectors(in 2D

space). Figure 2c represents a clock-based division used in defense related applications. Some of the partitions imposed are $0^0 \leq \theta < 30^0$, $30^0 \leq \theta < 60^0$, and etc.

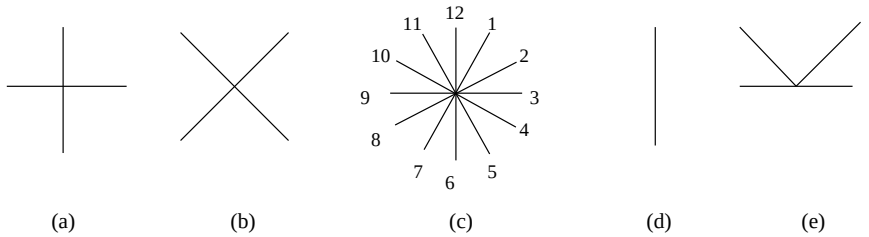


Figure 2: All except (e) are examples of symmetric partitioning. (d) is useful for path reasoning

After a partition scheme has been identified, a given $\vec{p}$ may be identified with a unique partition $[\vec{p}]$ in the partition set. Figure 3 below shows an example where the plane has been decomposed into four partitions. Each unit vector anchored at the origin is mapped into one of the four partitions. Each partition is identified with a symbol. In the example of Figure 3 these symbols are N, E, S and W, standing for North, East, South and West.

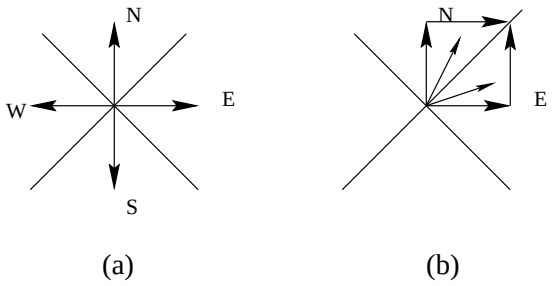


Figure 3: (a)Direction equivalence class, (b)Composition is *between*("vector sum") operation

Formally this is done by defining a relation $\sim$, on the set of all unit vectors as follows:

$$\vec{A} \sim \vec{B} \iff \vec{A} \text{ and } \vec{B} \text{ share the same partition.}$$

Lemma 1 $\sim$ *is an equivalence relation.*

Proof:

reflexive By the definition of $\sim$, $\vec{A} \sim \vec{A}$ for any $\vec{A}$ in the set of direction predicates.

symmetric If $\vec{A}$ and $\vec{B}$ are direction predicates and $\vec{A} \sim \vec{B}$ then, again by definition, $\vec{B} \sim \vec{A}$.

transitive If $\vec{A}$ $\vec{B}$ and $\vec{C}$ are direction predicates and $\vec{A} \sim \vec{B}$ and $\vec{B} \sim \vec{C}$, then $\vec{A}$ and $\vec{B}$ share the same oriented axis and $\vec{B}$ and $\vec{C}$ share the same axis. This clearly implies that $\vec{A}$ and $\vec{C}$ share the same axis or quadrant. Thus $\vec{A} \sim \vec{C}$. ■.

We now define a composition operation $\oplus$ on the equivalence classes. $[\vec{P}]$ represents a direction equivalence class which contains the vector $\vec{P}$. If two vectors $\vec{P}$ and $\vec{Q}$ share the same equivalence class then $[\vec{P}] = [\vec{Q}]$.

- Composition: $[\vec{P}] \oplus [\vec{Q}] = \{[\vec{R}] \mid \exists d_1 \geq 0 \; and \; d_2 \geq 0 \; s.t. \; \vec{R} = d_1\vec{P} + d_2\vec{Q}\}$

The composition operation $\oplus$ generalizes the ordinary vector sum operation to direction equivalence classes. A vector has a magnitude and a direction but at this time direction equivalence class only concern with direction, so the composition is related to the *between* operator, and that's why direction equivalence classes may not be closed under composition operation $\oplus$. In figure 3(b), $[\vec{N}] \oplus [\vec{E}] = \{[\vec{N}] \vee [\vec{E}]\}$.

Lemma 2 *Composition is commutative and associative.*

$$[\vec{P}] \oplus [\vec{Q}] = [\vec{Q}] \oplus [\vec{P}]$$

$$[\vec{P}] \oplus ([\vec{Q}] \oplus [\vec{R}]) = ([\vec{P}] \oplus [\vec{Q}]) \oplus [\vec{R}]$$

Proof: Follows from commutative and associative properties of vector sum operation. ■.

We explicitly define a *reverse* operation $rv([\vec{P}])$ on the direction equivalence classes defined using a symmetric partitioning. To do that we first define a symbols $[\vec{O}]$ to represent a null vector:

$$\text{Reverse} : rv([\vec{P}]) = \{[\vec{Q}] \mid \forall \vec{P} \in [\vec{P}], \exists \vec{Q} \in [\vec{Q}] \; s.t. \; \vec{P} + \vec{Q} = \vec{O}\}.$$

Each direction equivalence class has a unique reverse if the partitioning is symmetric, e.g. Fig 2(a)-(d).

Lemma 3 *Reverse distributes over Composition. i.e.,*

$$rv([\vec{P}] \oplus [\vec{Q}]) = rv([\vec{P}]) \oplus rv([\vec{Q}])$$

Proof: Follows from distributive property of scalar multiplication over vector sum in vector algebra. Note that $rv([\vec{P}])$ can also be defined in terms of equivalence class of a scalar product, i.e., $[(-1) \times \vec{P}]$. ■.

Notation	meaning
$\vec{P}$	direction object(unit-vector)
$[\vec{P}]$	equivalence class of $\vec{P}$
$[\vec{O}]$	a null vector
$\oplus$	composition of equivalence classes
rv	reverse operator of equivalence classes

Table 3: Notation table

3 Inference: Direction Predicates as Direction Equivalence Classes

In the previous section, we defined direction equivalence class. Here we model direction predicates and reasoning for viewer-based and object-based reference system with direction equivalence classes. We work with Fig 2a like partitioning. Many of the theorems and lemmas are more general and can apply to predicate systems based on other symmetric partitioning. An interesting implication of direction predicates as direction equivalence classes is lemma 9 on predicate interchangeability in composition. This property is surprising at the surface yet easy to understand given vector based interpretation using direction equivalence classes.

3.1 Viewer-based Direction Inferencing

Viewer-based direction refers to the direction relationship which is measured from a single viewer's

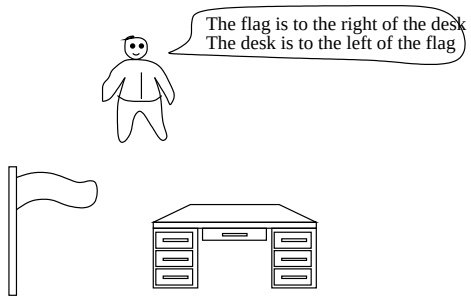


Figure 4: Viewer/Object-based Direction

perspective. There are three related components in this system: target object A, reference object B, and the viewer. The viewer has his/her own orientation, where objects A and B may or may not be oriented objects. In figure 4, the flag is to the right of the desk from the viewer's perspective.

Direction predicates for viewer-based system

We begin by defining an example set of direction predicates based on Figure 2a like partitioning in the viewer-based system as shown in Table 4. The corresponding direction equivalence classes are shown in figure 5(b). While the arrows $\vec{EA}, \vec{EL}, \vec{EB}, \vec{ER}$ have exact direction, the other four $\vec{RA}, \vec{LA}, \vec{LB}, \vec{RB}$ can point to any direction in their respective quadrants. For simplicity, we restrict our discussion in two dimensions, and the orientation of the viewer(O_v) is assumed as in figure 5(a). $O_v.\vec{above}$ and $O_v.\vec{right}$ are two orthogonal unit vectors to represent the orientation of viewer V. It can be extended to three dimensions by adding an orthogonal axis $\vec{Front}/\vec{Behind}$.

Direction predicates	$\vec{BA} \odot O_V.\vec{above}$	$\vec{BA} \odot O_V.\vec{right}$	equivalence classes	Partitions defined by angle
$SP(A,B)$	Undefined		$[\vec{O}]$	
$EA(A,B)$	1	0	$[\vec{EA}]$	$\theta = 90^0$
$EB(A,B)$	-1	0	$[\vec{EB}]$	$\theta = 270^0$
$ER(A,B)$	0	1	$[\vec{ER}]$	$\theta = 0^0$
$EL(A,B)$	0	-1	$[\vec{EL}]$	$\theta = 180^0$
$RA(A,B)$	> 0	> 0	$[\vec{RA}]$	$0^0 < \theta < 90^0$
$LA(A,B)$	> 0	< 0	$[\vec{LA}]$	$90^0 < \theta < 180^0$
$RB(A,B)$	< 0	> 0	$[\vec{RB}]$	$270^0 < \theta < 360^0$
$LB(A,B)$	< 0	< 0	$[\vec{LB}]$	$180^0 < \theta < 270^0$

Table 4: Direction Predicates for Viewer-based System

This predicate set consists of nine predicates, each of which represents one direction of object A related to object B based on the viewer V's orientation. The predicate $SP(A,B)$ refers that object A and B are in the same location. And the four predicates $EA(A,B)$, $EB(A,B)$, $EL(A,B)$, and $ER(A,B)$ represents that object A is to the exactly above, below, right, and left of B respectively from

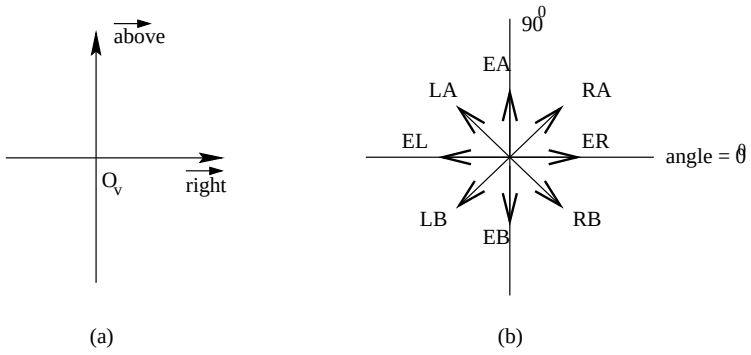


Figure 5: Direction Equivalence classes for viewer-based direction predicates

V's viewpoint. On the other hand, direction range predicates $RA(A,B)$, $LA(A,B)$, $RB(A,B)$, and $LB(A,B)$ represents that object A is within corresponding direction range of B from V's the viewpoint. For instance, $RA(A,B)$ returns TRUE iff object A is to the Right and Above of B from V's viewpoint. All the predicates are defined by the dot product between vector $\vec{BA}$ and the two axes of V's orientation.

Theorem 1 *The direction predicates are invariant with respect to viewer translation.*

Proof: Given objects A , B and viewer V with orientation $(O_V.\vec{above}, O_V.\vec{right})$, all the predicates are determined by the dot product of the vector $\vec{BA}$ with the individual orientation components $O_V.\vec{above}$ and $O_V.\vec{Right}$. Now if the perspective changes from V to $V^{'}$ with new orientation $(O_{V^{'}}.\vec{above}, O_{V^{'}}.\vec{right})$, then

$$O_{V^{'}}.\vec{above} = rot(\theta_1)(O_V.\vec{above}) \text{ and } O_{V^{'}}.\vec{right} = rot(\theta_1)(O_V.\vec{right}),$$

where $rot(\theta_1)$ is the rotation-matrix:

$$\begin{pmatrix} \cos(\theta_1) & \sin(\theta_1) \\ -\sin(\theta_1) & \cos(\theta_1) \end{pmatrix}$$

Thus the new orientations of $V^{'}$ are derived exclusively using the orientations of V. Hence all the predicates defined are translation invariant. ■

Reasoning with Direction Predicates

We can easily obtain Table 5 from the definition of rv on corresponding direction equivalence classes. For example, if A is exactly left of B, then B is exactly right of C from the same viewpoint.

Predicates	SP(A,B)	EA(A,B)	EB(A,B)	ER(A,B)	EL(A,B)	RA(A,B)	RB(A,B)	LA(A,B)	LB(A,B)
Reverse	SP(B,A)	EB(B,A)	EA(B,A)	EL(B,A)	ER(B,A)	LB(B,A)	LA(B,A)	RB(B,A)	RA(B,A)

Table 5: Reverse Relationship Between Direction Predicates

The composition operator $\oplus$ on direction equivalence classes gives us the composition Table 6 for direction predicates. The first row enumerates the eight possible direction predicates of A w.r.t to B,

$\oplus$	EA(A,B)	EB(A,B)	EL(A,B)	ER(A,B)	LA(A,B)	LB(A,B)	RA(A,B)	RB(A,B)
EA(B,C)	EA	EA $\vee$ EB $\vee$ SP	LA	RA	LA	EL $\vee$ LA $\vee$ LB	RA	ER $\vee$ RA $\vee$ RB
EB(B,C)	EA $\vee$ EB $\vee$ SP	EB	LB	RB	EL $\vee$ LA $\vee$ LB	LB	ER $\vee$ RA $\vee$ RB	RB
EL(B,C)	LA	LB	EL	EL $\vee$ ER $\vee$ SP	LA	LB	LA $\vee$ EA $\vee$ RA	EB $\vee$ LB $\vee$ RB
ER(B,C)	RA	RB	ER $\vee$ EL $\vee$ SP	ER	EA $\vee$ LA $\vee$ RA	EB $\vee$ LB $\vee$ RB	RA	RB
LA(B,C)	LA	LB $\vee$ EL $\vee$ LA	LA	EA $\vee$ LA $\vee$ RA	LA	LB $\vee$ EL $\vee$ LA	LA $\vee$ EA $\vee$ RA	U
LB(B,C)	LA $\vee$ LB $\vee$ EL	LB	LB	EB $\vee$ RB $\vee$ LB	LB $\vee$ EL $\vee$ LA	LB	U	EB $\vee$ LB $\vee$ RB
RA(B,C)	RA	RB $\vee$ RA $\vee$ ER	EA $\vee$ LA $\vee$ RA	RA	EA $\vee$ LA $\vee$ RA	U	RA	ER $\vee$ RA $\vee$ RB
RB(B,C)	RA $\vee$ RB $\vee$ ER	RB	EB $\vee$ LB $\vee$ RB	RB	U	EB $\vee$ LB $\vee$ RB	ER $\vee$ RA $\vee$ RB	RB

Table 6: Composition table for viewer-based Direction Predicates P(A,C), where P is one of EA, EB, EL, ER, LA, LB, RA, and RB

and the first column consists of the eight possible direction predicates for B relative to C from the same viewpoint. The contents of the table then shows the direction predicates for A relative to C from the viewer's viewpoint. For instance, If $EL(A,B) \oplus EA(B,C)$, the result is LA(A,C),i.e., if A is to the exactly left of B and B is to the exactly above of C, then A is within the *above* and $left$ region of C.

The properties can be characterized by a set of lemmas. Here, DPS denotes the set of direction predicates, which is {SP, EA,EB,ER,EL,RA,RB,LA,LB }.

Lemma 4 *Reverse operations on direction predicates is its own reverse, i.e.,*

$$reverse(P(A,B)) = Q(B,A) \implies reverse(Q(B,A)) = P(A,B)$$

Proof: Follows from the definition of operator rv on direction equivalence classes. rv requires a symmetric partitioning. ■.

Lemma 5 *No direction predicates except SP in DPS is symmetric, i.e.* $\forall P$, $P \in DPS$, $P(A,B)$ *and* $P(B,A)$ *can't both hold.*

Proof: Follows from having more than one direction equivalence class based on symmetric partitioning. ■.

Lemma 6 *Each direction predicate is transitive. i.e.* $\forall P$,$P \in DPS$, $P(A,B) \oplus P(B,C) \Longrightarrow P(A,C)$.

Proof: If $P(A,B)$ and $P(B,C)$ both hold, then there exists a $[\vec{R}]$ such that $P(A,B) \oplus P(B,C) \in [\vec{R}]$. By definition of $\oplus$, $[R] \oplus [R] = [R]$. Therefore the predicates are transitive. ■.

Lemma 7 *Set of direction predicate is closed under reverse but not under composition. i.e.* $\forall P$, $P \in DPS$ *and* $P(A,B)$, $\exists P' \in DPS$ *, s.t., P'(B,A) holds.*

Proof: Symmetric partitioning provides closure under reverse. The *between* semantic leads to the lack of closure for the composition. ■.

Note that previous work [4] stated that composition operation yields a set of values (not a single value) to avoid problems related to uniqueness assumptions. The equivalence classes on directions provide

a clear explanation for this. Composition of direction predicates uses *between* operator on directions leading to a set of values.

Lemma 8 *Power Set of set of direction predicate is closed under reverse and composition.*

Lemma 9 *Predicates in composition are interchangeable despite different arguments. i.e.,* $\forall P_1 \in DPS, P_2 \in DPS$, $P_1(A,B) \oplus P_2(B,C) = P_2(A,B) \oplus P_1(B,C)$.

Proof: According to the definition of equivalence classes, if a predicate holds for different argument pairs, the vector formed by the two arguments falls into same direction equivalence class. Since $\oplus$ is commutative, so the predicates,i.e., direction equivalence classes, are interchangeable. Figure 6 is an example. Here, $EA(A,B) \oplus EL(B,C) = LA(A,C) = EL(A,B) \oplus EA(B,C)$. ■.
This property is surprising at the surface yet easy to understand given vector based interpretation using

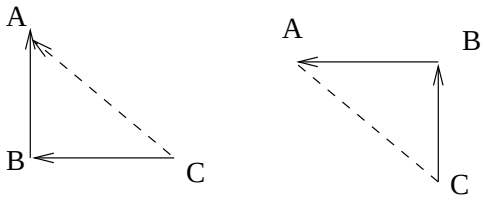


Figure 6: Interchangeability of direction predicates

direction equivalence classes.

3.2 Object-based Direction Inferencing

Object-based direction is the direction of the target object with respect to the orientation of the reference object. The reference object is an oriented object, while the target object may or may not have orientation. In figure 4, the person and the desk are oriented objects, and the person is behind the desk from the orientation of desk, and the desk is in front of the person w.r.t person's orientation.

The main difference between viewer-based and object-based direction system is that direction relationship is measured from a common viewer's viewpoint in viewer-based system, on the contrary, in object-based system, it is calculated with respective to the orientation of the reference object, and hence the referenced orientation changes according to different objects. Due to this property, direction inferencing in object-based system is more difficult, and there is little work achieved on it.

Direction predicates for Object-based system

We also begin the discussion by defining a set of direction predicates for object-based direction system as in Table 7. Similarly, we restrict our discussion in 2D. The direction predicates are similar to those

Direction predicates	SP	EA	EB	ER	EL	LA	LB	RA	RB
$\vec{BA} \odot O_B.\vec{above}$		1	-1	0	0	> 0	< 0	> 0	< 0
$\vec{BA} \odot O_B.\vec{right}$		0	0	1	-1	< 0	< 0	> 0	> 0

Table 7: Direction Predicates for Object-based System for Object B

in viewer-based direction predicates table, but now with respective to the orientation of object B, which is $(O_B.\vec{above}, O_B.\vec{right})$. If the orientation of viewer is different from the orientation of reference object B, then the direction predicate P(A,B) that describes the direction of A related to B seen from viewer will be different from the direction predicate P'(A,B) which describes the direction related to B from B's orientation. Note that translation does not affect direction predicates, as formalized in theorem 1.

Viewpoint Transformation

As we discussed above, direction changes if the orientation of the reference object changes. Figure 7 illustrates one example. Object A is to the exactly right of B in 7(a), but it is between right and below of B if B's orientation rotates 45^0, as shown in 7(b). Table 8 lists the corresponding direction predicates

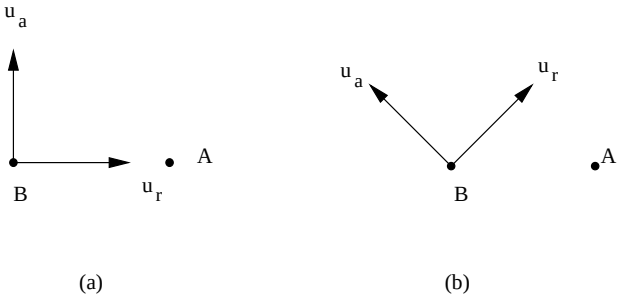


Figure 7: (a) A is ER B. (b) A is RB B

after changing the orientation of the viewer. The first row enumerates eight direction relationship of A relative to B w.r.t. object B's orientation. The first column lists the relative orientation predicates that transforms the orientation of B to orientation of object C. And the content of the table illustrates the corresponding direction predicates that correctly represent the direction relationship of A relative to B with respective to the orientation of object C.

	Direction predicate as viewed from B							
orientation of B wrt C	EA	EB	ER	RL	RA	LA	RB	LB
$AntiClock90(B,C)$	ER	EL	EB	EA	RB	RA	LB	LA
$AntiClock180(B,C)$	EB	EA	EL	ER	LB	RB	LA	RA
$AntiClock270(B,C)$	EL	ER	EA	EB	LA	LB	RA	RB
$AntiClock0to90(B,C)$	RA	LB	RB	LA	$RB \vee RA$	$LA \vee RA$	$RB \vee LB$	$LB \vee LA$
$AntiClock90to180(B,C)$	RB	LA	LB	RA	$RB \vee LB$	$RB \vee RA$	$LA \vee LB$	$RA \vee LA$
$AntiClock180to270(B,C)$	LB	RA	LA	RB	$LB \vee LA$	$LB \vee RB$	$RA \vee LA$	$RB \vee RA$
$AntiClock270to360(B,C)$	LA	RB	RA	LB	$LA \vee RA$	$LA \vee LB$	$RB \vee RA$	$LB \vee RB$

Table 8: Transformation table when the viewpoint transforms from B to C.

Inference rules for object-based system

The basic strategy of inferencing in object-based system is to transform all predicates to a common viewer by looking up table 8. then the problem is reduced to the view-based system, and we can use the same inference rules as in table 6. Figure 8 illustrates this procedure.

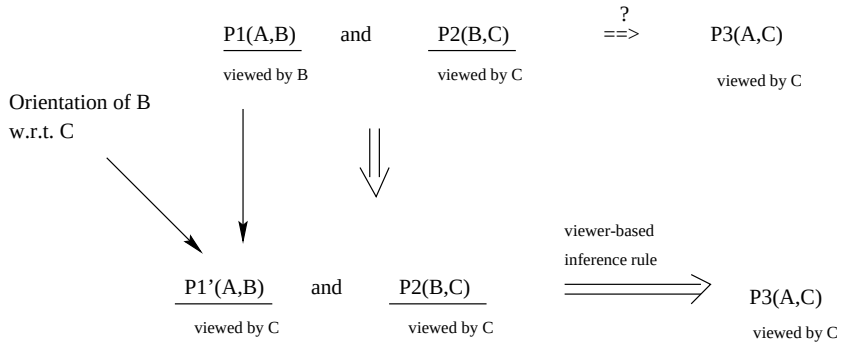


Figure 8: Strategy for Object-based direction inferencing

Here, given two relationships $P1(A,B)$(w.r.t. B's orientation) and $P2(B,C)$ (w.r.t. C's orientation), we want to infer the direction predicate P3(A,C), which describing the direction of A related to C from C's orientation. The first step is to do the viewpoint transformation by looking up in the table 4 and get the predicate P'(A,B), the problem then reduces to the inferencing within viewer-based systems.

Infer relative orientation given directions from two different viewers

One interesting application is that we can infer the relative orientation of two different viewers if we know the directions of an object from these two viewers. Table 9 illustrates this inference. The first row

$\theta(V_1, V_2)$	EA	EB	ER	RL	RA	LA	RB	LB
EA	0	180	270	90				
EB	180	0	90	270				
ER	90	270	0	180				
EL	270	90	180	0				
RA	(0,90)	(180,270)	(270,360)	(90,180)	0	90	270	180
LA	(270,360)	(90,180)	(180,270)	(0,90)	270	0	180	90
RB	(90,180)	(270,360)	(0,90)	(180,270)	90	180	0	270
LB	(180,270)	(0,90)	(90,180)	(270,360)	180	270	90	0

Table 9: relative orientation from two viewpoints

enumerates the directions of an object seen from the first viewer(V_1), and the first column enumerates the directions of the object seen from the second viewer(V_2). The contents of the table are the relative angles rotate from V_1 to V_2. For example, if the object is $exactlyabove$ from viewpoint of V_1, and $rightandbelow$ from the viewpoint of V_2, then the relative orientation changed from V_1 to V_2 is $(90^0, 180^0)$.

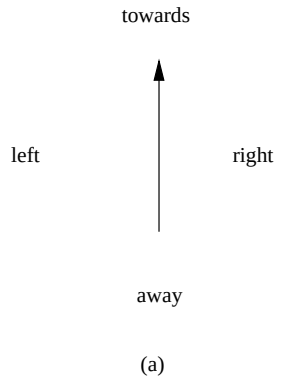
See B.2 for examples of reasoning on both viewer-based and object-based direction systems.

4 Future Work: Landmark Based Route Description

An important application in GIS is route navigation. The problem consists of two aspects: route planning and route description. In many instances, like moving around in a university campus, the route description can only be given in terms of landmarks. We will define a set of predicates which can be used to describe a route based only on landmarks. These predicates will be defined in terms of the

direction predicates discussed in Section 2 and 3.

The direction equivalence classes discussed in the previous section are for static directions in the sense that the reference object and the target object are stationary. In some cases we have to measure the direction between a moving viewer and objects, such as in route description problem. We define a new equivalence class to characterize the direction relationship between a moving viewer(a directed edge) and point objects. We begin by partitioning the plane with respect to a directed edge as shown in Figure 9.



(a)

Figure 9: Plane partition determined by the edge

There are 4 direction equivalence class here: *towards, away, left, right.* Since the orientation of the viewer is aligned with the direction of the edge, *towards* and *away* are related to equivalence classes EA and EB respectively. Similarly the equivalence class *left* and *right* are related to {LA, LB, EL } and { RA,RB, and ER }. These observations are summarized in Table 10. In table, X is an object.

path predicate	definition in terms of direction predicate
$towards(\vec{e}, X)$	$\exists$ point P $\in$ X, $\exists$ point $P_1 \in \vec{e}$,s.t., $EA(P, P_1)$
$left(\vec{e}, X)$	$\forall$ point P $\in$ X, $\exists$ point $P_1 \in \vec{e}$, s.t. $LA(P, P_1) \vee EL(P, P_1) \vee LB(P, P_1)$
$right(\vec{e}, X)$	$\forall$ point P $\in$ X, $\exists$ point $P_1 \in \vec{e}$, s.t. $RA(P, P_1) \vee ER(P, P_1) \vee RB(P, P_1)$
$away(\vec{e}, X)$	$\exists$ point P $\in$ X, $\exists$ point $P_1 \in \vec{e}$,s.t., $EB(P, P_1)$

Table 10: Defining Path Predicates

Two important issues related to the route description problem are the "choice" of landmarks for describing the path and the definition of an "acceptable" path description. We use the notion of *Voronoi regions* and *Homotopic paths* to address these issues.

Voronoi regions are geometric structures that record information about proximity. Let $P = \{p_1, p_2, \cdots, p_n\}$ be a set of points (called sites) in the two-dimensional Euclidean plane. Partition the plane by assigning every point in the plane to its nearest site. All those points assigned to p_i form the voronoi region $V(p_i)$, which contains all the points at least as close to p_i as to any other site [14], i.e.,

$$V(p_i) = \{x || p_i - x| \leq |p_j - x|, \forall j \neq i\}.$$

Homotopic paths Intuitively, two paths in a plane, with a coincident start point and end point, are homotopic if they can be "continuously deformed" into one another without any cutting or lifting. For example in Figure 10(a) f and g are homotopic, but in Figure 10(b) f and g are not homotopic because there is a "hole" in the space which obstructs the continuous deformation of f into g or vice-versa. Landmarks are viewed as holes in the plane.

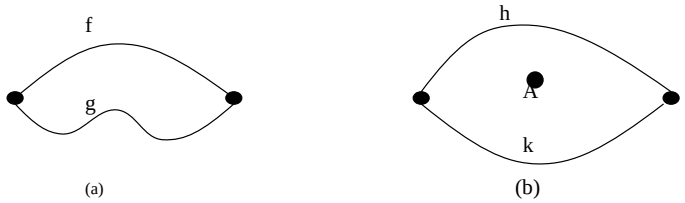


Figure 10: (a) f and g are homotopic. (b) f and g are not homotopic because of obstacle A

Problem Definition

Given: An oriented map(2D) with named point landmarks and a route in terms of node-edge-node sequences in the free space , where the coordinates for all the node-points are known.

Find: Route descriptions using landmarks and path predicates.
Correctness Constraints: The described path should be homotopic to the given exact path.

Hypothesis: Given a path P and the four predicates {towards, away, left, right}, a path can be described correctly, i.e., the path resulting from the description and the original path are homotopic.

We are working on specifying an algorithm to generate path descriptions using the directional predicates and analyzing correctness, completeness and complexity properties.

Acknowledgments

This work is sponsored in part by the Army High Performance Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory cooperative agreement number DAAH04-95-2-0003/contract number DAAH04-95-C-0008, the content of which does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred. This work was also supported in part by NSF grant #9631539 Thanks to Christiane McCarthy for helping to improve the readability of the paper.

References

[1] A.Frank. Qualitative Spatial Reasoning: Cardinal Directions as an Example. *International Journal of Geographical Information Systems*, 10(3):269–290, 1996.

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

[3] D.Papadias, M. Egenhofer, and J. Sharma. Hierarchical Reasoning about Direction Relations. In *Fourth ACM Workshop on Advances in Geographic Information Systems*, pages 105–112. ACM, 1996.

[4] A. Frank. Qualitative Spatial Reasoning about Cardinal Directions. In *Auto carto 10, D.Mark and D. White, eds., Baltimore, MD*, pages 148–167, 1991.

[5] C. Freksa. Temporal Reasoning Based on Semi-Intervals. *Artificial Intelligence*, 54:199–227, 1992.

[6] C. Freksa. Using Orientation Information for Qualitative Spatial Reasoning. *Theories and Methods of Spatio-Temporal Reasoning Geographic Space*, 639:162–178, 1992.

[7] Klaus-Peter Gapp. Basic Meanings of Spatial relations: Computation and Evaluation in 3D space. *AAAI*, 2:1393–1398, 1994.

[8] R. Goyal and M. Egenhofer. The Direction-Relation Matrix: A Representation of Direction Relations for Extended Spatial Objects . In *UCGIS Annual Assembly and Summer Retreat, Bar Harbor, ME*, 1997.

[9] G.Retz-Schmidt. Various Views on Spatial Prepositions. *AI Magazine*, 9(2):95–105, 1988.

[10] John Gurney and Elizabeth Kilpple. Composing Conceptual Structure for Spoken Natural Language in a Virtual Reality Environment . 1997.

[11] R.H. Guting. An Introduction to Spatial Database Systems. *VLDB*, 3:357–399, 1994.

[12] Open GIS Consortium Inc. Maps on us. http://www.MapsOnUs.com.

[13] E. Jungert. The observer's point of view: An Extension of Symbolic Projections. *Theories and Methods of Spatial-Temporal Reasoning in Geographic Space*, 639:179–195, 1992.

[14] Joseph O'Rourke. *Computational Geometry in C*. Cambridge University Press, 1994.

[15] D. Papadias and T. Sellis. Qualitative representation of Spatial Knowledge in Two-Dimensional Space. *VLDB Journal*, 3(4):479–516, 1994.

[16] Donna J. Peuquet and Zhan Ci-Xiang. An Algorithm to Determine the Directional Relationship Between Arbitrarily-shaped Polygons in the plane. *Pattern Recognition*, 20(1):65–74, 1987.

[17] P.W.Huang and Y.R. Jean. Using 2D C^+-String As Spatial Knowledge Representation For Image Database Systems. *Pattern Recognition*, 30(10):1249–1257, 1994.

[18] David F. Rogers. *Mathematical Elements for Computer Graphics*. McGraw-Hill Publishing Company, 1990.

[19] S. Shekhar, S. Ravada A.Fetterer, X.Liu, and C.T. Lu. Spatial Databases: Accomplishments and Research Needs. *University of Minnesota technical report, Csci TR97-016*, 1997.

[20] Shashi Shekhar, Mark Coyle, and etc. Data Models in Geographic Information Systems. *CACM*, 40(4):103–111, 1997.

[21] Shashi shekhar and Xuan Liu. Direction As a Spatial Object: A summary of Results. In *Sixth International Symposium on Advances in Geographic Information Systems*, pages 69–75. ACM, 1998.

[22] Y.I.Chang and B.Y. Yang. A Prime-Number-Based Matrix Strategy for Efficient Iconic Indexing of Symbolic Pictures. *Pattern Recognition*, 30(10):1–13, October 1997.