Identification of Relations in Region Connection Calculus: 9-Intersection Reduced to 3^+-Intersection Predicates

Chaman L. Sabharwal and Jennifer L. Leopold

Computer Science Department, Missouri University of S&T,
Rolla, MO – 65409, USA
{chaman,leopoldj}@mst.edu

Abstract . The intersection between objects relates to a class of problems where either precise intersection is required or imprecise intersection is acceptable. The calculation of intersection between two 2D/3D objects is a computation-intensive process. For qualitative spatio-temporal reasoning, it is sufficient to know the existence of intersection instead of the precise intersection. In order to identify RCC8 relations, the 9-Intersection model considers the pairwise intersection of interiors, boundaries, and exteriors of objects. It was determined that the 9-Intersection is sufficient for identifying spatial relations. Later, it was shown that a 4-Intersection model is sufficient to achieve the same results making the definition (and implementation) of the RCC8 relations worth studying in greater detail. Herein we prove that the 9-Intersection model can be further reduced to almost three intersection predicates, producing a 3^+-Intersection model. This results in improved algorithmic and computational efficiency as a consequence of fewer predicates and faster intersection operations.

Keywords: Spatial Reasoning, Qualitative Reasoning, Region Connection Calculus, Spatial Object Intersection.

1 Introduction

Imprecision and uncertainty are widespread in the physical world [1]. A ubiquitous task in QSR is the intersection between objects [2, 3, 4, 5]. Typically, the intersection between objects refers to a class of problems where either precise intersection is required [4] or imprecise intersection is acceptable [2, 3]. The computational complexity varies depending on the quality of intersection. The precise calculation of intersection between two 2D/3D objects is a computation-intensive process. For the qualitative spatial reasoning (QSR) domain, it is sufficient to know the existence of intersection instead of the quantity of intersection (i.e. precisely where the intersection occurs and what the intersection is). In particular, the calculation of the intersection predicate IntInt(A, B), intersection between the interiors objects A and B, is more complex than other types of intersections [6]. In this paper, we prove that 9-Intersection model can be further reduced to almost three intersection predicates,

F. Castro, A. Gelbukh, M.G. Mendoza (Eds.): MICAI 2013, Part II, LNAI 8266, pp. 362–375, 2013.

3^+-Intersection framework. This reduces the computation time considerably while retaining the same accuracy as the 9-Intersection model. Specifically, it not only reduces the number of intersections, it also replaces the slow intersection algorithm with a faster odd parity test to accomplish the same intersection detection task.

There is a wide class of applications for intersection detection in areas such as geometric modeling [4], virtual reality [7], and Geographical Information Systems for qualitative spatial reasoning (QSR) [2]. The determination of intersection between concave objects is much more complex than that of convex objects. Also the intersections between concave objects do not form closed algebra, as intersection between two concave objects may not result in a concave object, but rather a collection of disjoint objects violating the closedness property. Concave objects can be segmented into convex objects; this renders the task to working with convex objects only. So the finite intersection algebra is closed for convex objects. A bounded region, which is non-empty connected set, partitions the space into three parts: interior, boundary and exterior, see Fig. 1.

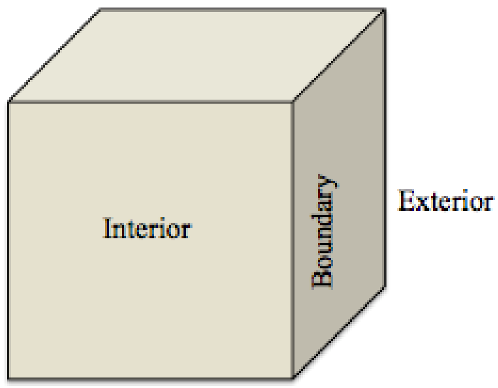


Fig. 1. The interior, boundary, and exterior of a bounded convex region

Perhaps the most well known formal model for qualitative spatial representation and reasoning is RCC8, which distinguishes spatial relations by employing first order logic [5] or a 9-Intersection model [2] that compares the intersections of one region's interior, boundary, and exterior with those of another region. In order to identify RCC8 relations, the 9-Intersection model uses nine intersection predicates, see Table 1, where shaded entries correspond to the 4-Intersection model [3]. The RCC8 distinct topological relations corresponding to the 9-Intersection model are displayed in Fig. 2. RCC8 forms a jointly exhaustive and pairwise distinct (JEPD) set of relations and a composition table provides a basis for qualitative spatial reasoning [2]. Originally the 9-Intersection model was designed independently of the logical foundations. First order logic is useful for knowledge acquisition and deriving inferences, whereas 9-Intersection is useful for knowledge representation and implementation. RCC8 may be considered as the spatial counterpart of Allen's thirteen interval relations among intervals of time [8].

Table 1. 9-Intersection 3x3 matrix and Reduced 4-Intersection 2x2 Matrix (Shaded) for Calculating RCC8 Relations

	Interior	Boundary	Exterior
Interior	Int(A)∩Int(B)	Int(A)∩Bnd(B)	Int(A)∩Ext(B)
Boundary	Bnd(A)∩Int(B)	Bnd)∩Bnd(B)	Bnd(A)∩Ext(B)
Exterior	Ext(A)∩Int(B)	Ext(A)∩Bnd(B)	Ext(A)∩Ext(B)

The paper is organized as follows: Section 2 gives a very terse description of the related work, the reader may refer to the referenced papers for details; Section 3 describes the innovation, related mathematical foundations and efficiency considerations of the authors' work, Section 4 is devoted to optimizing an application, Section 5 draws conclusions, followed by references in Section 6.

2 Background

For any two regions A and B in 2D/3D space, the spatial relation between A and B is denoted by R(A, B). As hierarchically depicted in Fig. 2 based on connectivity, there are eight RCC8 relations: DC (discrete), EC (external connection), PO (partial overlap), EQ (identical), TPP (tangential proper part), NTPP (non-tangential proper part), TPPc, and NTPPc. Converse relations are defined to make these relations jointly exhaustive and pairwise distinct (JEPD), thereby avoiding ambiguous interpretations of the data. For example, the converse of "A is a proper part of B" is "A contains B"; instead of specifying "B is a proper part of A" as PP(B, A), in RCC8 this converse relation is denoted by PPc(A, B). Similarly the TPPc(A, B) and NTPPc(A, B) relations are defined. For symmetric relations such as DC, EC, PO, EQ, no (distinct) converse relation is defined.

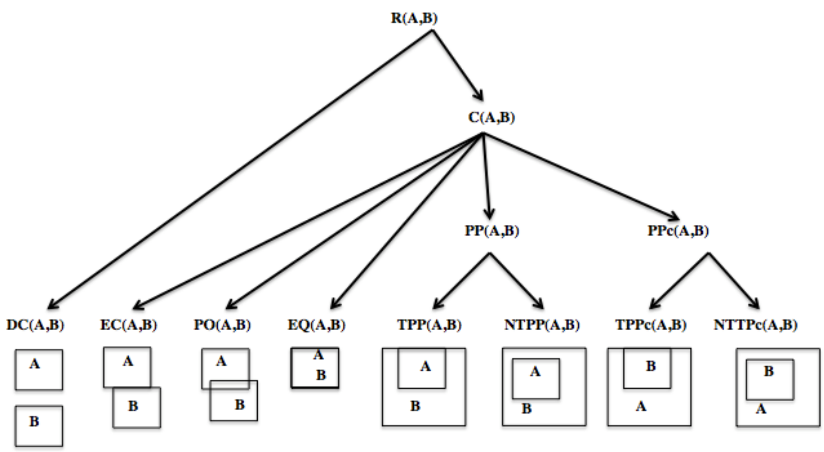


Fig. 2. Hierarchy of the RCC8 JEPD relations

Each of the RCC8 relations can be described uniquely by using a 9-Intersection matrix between two regions A and B, where symbol Int represents the region's interior, Bnd denotes the boundary, and Ext represents the exterior, see Fig. 3. For example, the predicate IntInt(*A, B*) is a binary relation that represents the truth value of intersection, Int(*A*)$\cap$Int(*B*), between the interiors of region A and region B; the truth value of this function is either true (for non-empty intersection) or false (for empty intersection) for that intersection. Similarly, there are other predicates for the intersection of A's interior, boundary, or exterior with those of B.

It was established that the 9-Intersection model presents a sufficient set of intersection predicates for identifying RCC8 spatial relations [2, 5]. In qualitative spatial reasoning in 3D [6], it was observed that one of the intersection predicates does not contribute any knowledge in distinguishing the relations. This predicate was discarded from the algorithm implementation without sacrificing the accuracy in results. The implementation extensively used the remaining eight intersections, thus the name 8-Intersection. For a mathematical proof, see Theorem 1 in Section 3. Later it was observed from Table 2 and Fig. 3 and analytically proved that a 4-Intersection version yields the same results as the 9-Intersection model [3]. In Table 2, the shaded entries are sufficient to distinguish RCC8 relations. In the 8-Intersection model, there are 64 entries, whereas in the 4-Intersection model there are 32 entries; however only 26 entries (shaded) actually are used to classify all the eight relations.

Table 2. The 8-Intersection and 4-Intersection Vectors. Only the Shaded Entries are used to distinguish RCC8 Relations.

	IntInt	BndBnd	IntBnd	BndInt	IntExt	BndExt	ExtInt	ExtBnd
DC	F	F	F	F	T	T	T	T
EC	F	T	F	F	T	T	T	T
EQ	T	T	F	F	F	F	F	F
PO	T	T	T	T	T	T	T	T
TPP	T	T	F	T	F	F	T	T
NTPP	T	F	F	T	F	F	T	T
TPPc	T	T	T	F	T	T	F	F
NTPPc	T	F	T	F	T	T	F	F

In Fig. 3, this table is translated into a decision tree for visual understanding of the 4-Intersection model. In Section 3, we lay the ground work to improve the efficiency of the intersection predicates in Table 2 and Fig. 3, so as to make the most expensive Interior-Interior intersection predicate the least used, and yet achieve the same results.

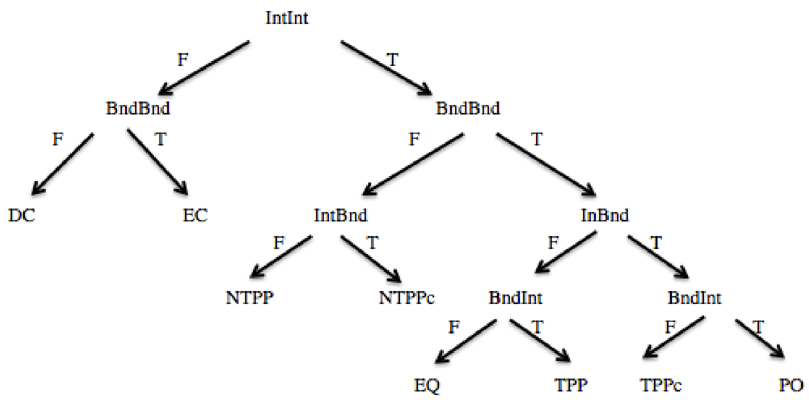


Fig. 3. Decision tree using 4-Intersection model. Each relation is determined with at most 4 predicates

3 Mathematical Analysis

3.1 Mathematical Foundations

Herein we present mathematical analysis to support the reduction in the number of required intersection predicates in RCC8 and VRCC-3D+, a region connection calculus for the 3D objects [9]. There is no need to check the predicate ExtExt(*A, B*), the exterior-exterior intersection predicate, to determine any relation [6]; for bounded regions, the intersection between the exteriors of two regions is always trivially non-empty and thus the intersection predicate ExtExt(*A, B*) contributes no knowledge in distinguishing the relations. We provide mathematical proof of this observation in Theorem 1. Note that we use the following notation in the proofs: A^i stands for the interior of set A; A^b represents the boundary of set A; A^e corresponds to the exterior of set A; A^c denotes the complement of set A, and $\bar{A}$ connotes the closure of set A.

Theorem 1. Let there be two bounded sets A and B, it is shown that $A^e \cap B^e \neq \emptyset$, (i.e., intersection predicate ExtExt(A, B) is always true).

Proof. Since A and B are bounded, their exteriors A^e and B^e are unbounded.
The closure, $\bar{B}$, of B is bounded.
Suppose $A^e \cap B^e = \emptyset$. Then $A^e \subseteq (B^e)^c = \bar{B}$.
This is a contradiction because A^e, which is unbounded, is a subset of $\bar{B}$ which is bounded.
Therefore the supposition is false.
Hence the theorem holds good.

This proves that the original 9-Intersection predicates can be replaced with eight intersection predicates, thus the name 8-Intersection. The next theorem shows that the predicate IntInt(*A, B*) is not independent of predicate IntBnd(*A, B*); in fact, IntBnd(*A, B*) implies IntInt(*A, B*). Similarly BndInt(*A, B*) implies IntInt(*A, B*). Thus if the value of IntBnd(*A, B*) or BndInt(*A, B*) is true, it ensures that we do not need to compute the value of predicate IntInt(*A, B*), it is true by *de facto*. In Table 3, the five relations entries (PO, TPP, TPPc, NTPP, NTPPc) have IntBnd(*A, B*) or BndInt(*A, B*) value true, thus IntInt(*A, B*) is true without computation. In the entries for relations, DC, EC, EQ, both IntBnd and BndInt are false. However BndBnd is false for DC and true for EC, and EQ. Thus BndBnd resolves identification of DC without considering IntInt. For the intersection predicate entries for two relations EQ and EC, both BndBnd and IntInt(*A, B*) are false. They can be simplified and made more efficient as shown in Theorem 4.

Theorem 2. For any two regions *A* and *B*, if the predicate IntBnd(*A, B*) is true, then the predicate IntInt(*A, B*) is true. That is, if $A^i \cap B^b \neq \emptyset$, then it is shown that $A^i \cap B^i \neq \emptyset$.

Proof. Let IntBnd(*A, B*) be true. Then $A^i \cap B^b \neq \emptyset$.

Let $x \in A^i \cap B^b$. Then $x \in A^i$ and $x \in B^b$.

Since $x \in A^i$ and A^i is an open set, there a neighborhood $N_r(x) \subset A^i$ for some positive real r, r>0.

Since $x \in B^b$ and B^b is the boundary set, then every neighborhood $N_y(x)$ of x intersects both the interior and exterior of *B*. That is $N_y(x) \cap B^i \neq \emptyset$ and $N_y(x) \cap B^e \neq \emptyset$. In particular, $N_r(x) \cap B^i \neq \emptyset$. Now we have $N_r(x) \subset A^i$ and $N_r(x) \cap B^i \neq \emptyset$. Therefore $N_r(x)$ has points common to A^i and B^i .

Hence $A^i \cap B^i \neq \emptyset$ and the predicate IntInt(*A, B*) is true

Similarly if $A^b \cap B^i \neq \emptyset$, then $A^i \cap B^i \neq \emptyset$.

Note that the converse of Theorem 2 is not true. We may have IntInt true without IntBnd(*A, B*) or BndInt(*A, B*) or both true. For example, in the case of equal objects or externally connected objects, IntInt(*A, B*) is true despite the fact that both IntBnd(*A, B*) and BndInt(*A, B*) can be false. Theorems 3 and 4 prove that IntInt(*A, B*) need not be directly calculated. So IntInt(*A, B*) can be replaced with a much simpler point-in-object odd-parity rule. Briefly, when IntBnd(*A, B*) and BndInt(*A, B*) are false, and BndBnd(*A, B*) is true, if a semi-infinite ray from the centroid C of one object, A, intersects the boundary of the other object, B, an odd number of times, then C is inside B, hence IntInt(A, B) is true. Note that the converse may not be true. Thus it simplifies the decision tree and IntInt(A, B) computation, where EC(*A, B*) and EQ(*A, B*) are special siblings, see Fig. 4(b). Those relations can be distinguished with a much simpler test than IntInt(*A, B*). For representing this intersection we denote it by ROI meaning “Ray-Object Intersection-ray through the interior point of one object with the boundary of the other object”.

Theorem 3. If the predicate BndBnd(*A, B*) is true, and BndInt(*A, B*) and IntBnd(*A, B*) are false, then the predicate EQ(*A, B*) is true if and only if there is a point common to the interior of *A* and the interior of *B*.

Proof. Let the predicate BndBnd(*A, B*) be true, BndInt(*A, B*) and IntBnd (*A, B*) be false.

Therefore $A^b \cap B^b \neq \emptyset$, $A^b \cap B^i = \emptyset$ and $A^i \cap B^b = \emptyset$.

The proof is presented in two parts, using necessary and sufficient conditions.

Necessary: If $A=B$, then $A^i=B^i$. It follows that every point of A^i is in B^i, thus A^i and B^i share at least one point,

Sufficient: Let A^i and B^i have a point in common. That means $A^i \cap B^i \neq \emptyset$. We show that $A=B$.

Since $A^b \cap B^i = \emptyset$, then $B^i \subseteq (A^b)^c$. This means $B^i \subseteq A^i \cup A^e$ where $A^i \cap A^e = \emptyset$, by definition.

Since $B^i \cap A^i \neq \emptyset$ and $B^i \subseteq A^i \cup A^e$ with B^i connected and $A^i \cap A^e = \emptyset$,

we have $B^i \subseteq A^i$. Similarly we can derive that $A^i \subseteq B^i$.

Therefore $B^i = A^i$, or $A^i = B^i$. Since $A^b \cap B^i = \emptyset$ and $A^i \cap B^b = \emptyset$, $A^b = B^b$.

Now $\bar{A} = \bar{B}$. Hence we have proved that $A = B$.

Theorem 4. If predicate BndBnd(*A, B*) is true, and BndInt(*A, B*) and IntBnd (*A, B*) are false, then the predicate EQ(*A, B*) is true if and only if the predicate EC(*A, B*) is false.

Proof. Let predicate BndBnd(*A, B*) be true, BndInt(*A, B*) and IntBnd (*A, B*) be false.

That is, $A^b \cap B^b \neq \emptyset$, $A^b \cap B^i = \emptyset$ and $A^i \cap B^b = \emptyset$.

Under these conditions, using necessary and sufficient conditions we have,

(a) Necessary: Let EQ(*A, B*) be true. Then $A \subseteq B$ and $B \subseteq A$.

Then $A \subseteq B$ implies $A^i \subseteq B$. Since A^i is open and B^i is the largest open set contained in *B*, then $A^i \subseteq B^i$. Similarly $B^i \subseteq A^i$. Thus $A^i = B^i$ implying $A^i \cap B^i \neq \emptyset$, hence EC(*A, B*) is false.

(b) Sufficient: Let EC(*A, B*) be false,

Then by definition, one of the following is true: $A^i \cap B^i \neq \emptyset$, $A^i \cap B^b \neq \emptyset$, $A^b \cap B^i \neq \emptyset$, $A^b \cap B^b = \emptyset$.

Since it is given that $A^i \cap B^b = \emptyset$, $A^b \cap B^i = \emptyset$, $A^b \cap B^b \neq \emptyset$,

the only possibility is $A^i \cap B^i \neq \emptyset$.

We have seen in Theorem 3 that if $A^i \cap B^i \neq \emptyset$, then $A=B$.

Hence the predicate EC(*A, B*) is false implies that predicate EQ(*A, B*) is true.

This completes the proof.

Note that it is also true that "EQ(*A,B*) is false if and only EC(*A,B*) true".

3.2 Efficiency Considerations

In the 4-Intersection model, four of the RCC8 relations can be identified with four intersection predicates, two of the relations can be identified with three intersection predicates, and the remaining two relations can be identified with two intersection predicates. In Table 2, it is clear from looking at the IntInt and BndBnd columns that those are the most useful predicates in the sense that they are sufficient to partition the RCC8 relations into three classes: {DC, EC}, {NTTP, NTTPc}, and {PO, EQ, TPP, TPPc}. That heuristic is the basis for the conceptual decision tree shown in Fig. 3. It was proved mathematically that all other intersections do hold good *de facto* [3].

The average number of predicate computations to determine an RCC8 relation using the 4-Intersection model is 3.25; see Table 3. Yet there is inherent inefficiency even in the 4-Intersection model. IntInt is computed and used first for the identification of every relation, and it is the most computation-intensive of all the predicates.

Theoretically the intersection predicates can be executed in any order. If we rearrange predicate computations in accordance with the result presented in Section 3.1, it improves the efficiency. Additionally, the ray-object intersection (ROI) based on the odd-parity rule can be seamlessly integrated in place of the IntInt predicate, thereby providing more efficiency. The computational cost of implementing ROI is less than half of the computation in implementing IntInt.

Table 3. Shaded Vectors Used For 4-Intersection Model, Average Computation is 3.25 Predicates Per Relation

RCC8	IntInt	BndBnd	IntBnd	BndInt
DC	F	F	F	F
EC	F	T	F	F
PO	T	T	T	T
EQ	T	T	F	F
TPP	T	T	F	T
NTPP	T	F	F	T
TPPc	T	T	T	F
NTPPc	T	F	T	F

Herein we show that the efficiency can be further enhanced in the following ways: (1) six of the RCC8 relations can be identified with at most 3-Intersection predicates (see Table 4, Fig. 4(a)) because the IntInt predicate is not necessary to determine those relations; (2) for the remaining two RCC8 relations which need to be resolved with predicate IntInt (Fig. 4(b)), we use an odd-parity rule to speed up the computation of IntInt; and (3) the average number of predicate computations for determination of an arbitrary RCC8 relation can be reduced to less than 3 (see Table 4), which results in a saving of at least 12% in computation time, see Table 5.

In Table 4, the last column is denoted by Hint in the header, hybrid intersection representing IntInt or ROI; the value of IntInt is T/F as usual, the value of ray-object intersection, ROI, is O/E for odd even parity.

In terms of tree terminology, in this case, the average distance from the root is smaller if the root predicate is “farther away” from the IntInt predicate. This is a considerable improvement in implementation performance when a test is executed thousands of times. Our observation and intuition comes from the fact, that if IntBnd is true, then IntInt is true by *de facto.*

Table 4. Shaded Vectors used for 3^{+}-Intersection Model Average Computation is less than 3 Predicates Per Relation

RCC8	BndBnd	BndInt	IntBnd	Hint
DC	F	F	F	F
EC	T	F	F	E
PO	T	T	T	T
EQ	T	F	F	O
TPP	T	T	F	T
NTPP	F	T	F	T
TPPc	T	F	T	T
NTPPc	F	F	T	T

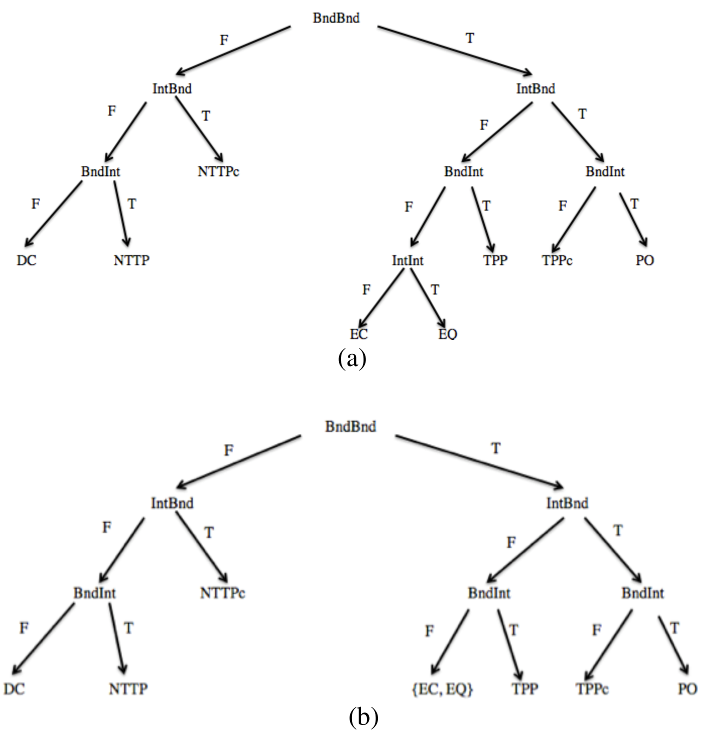


Fig. 4. (a) Modified version of 4-Intersection model with IntInt as the last test. (b) Since IntInt is the last predicate, it can be implemented with a simpler, less rigorous test.

The average number of predicate tests for a relation has been reduced to less than 3. This amounts to saving at least 12% in computation time, see Table 5.

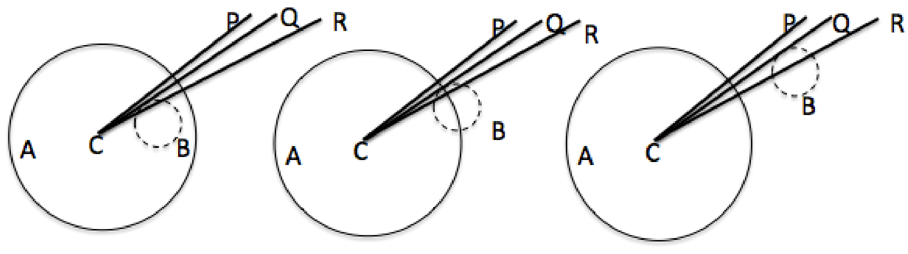
Table 5. Percent Speedup Gain By 3^{+}-Intersection Model over other Models

Model	Number Of Predicates	Average Required Predicates	Percent Relative Gain
9-Intersection	9	9	210
8-Intersection	8	8	176
4-Intersection	4	3.25	12
3^{+}-Intersection	3^{+}	2.9	

3.3 Example Ray-Object Intersection, Point-in-Object Odd Parity Rule

In general, the odd parity rule to test a point inside an object is standard. However, the test for predicate IntInt(A,B) is not that trivial. Even parity may lead to IntInt(A,B) being true (Fig. 5(a,b)) or false (Fig. 5(c)), whereas odd parity also could result in IntInt(A,B) being true (Fig. 6(a,b,c)). For IntInt(A,B) we note that: (1) this parity test is inconclusive, (2) even parity is not the opposite of odd parity, and (3) there is no clear cut way to determine the test point C in A to establish the truth of IntInt(A,B). Here we describe the hurdles in the computation of the IntInt(A,B) predicate and how to circumvent them.

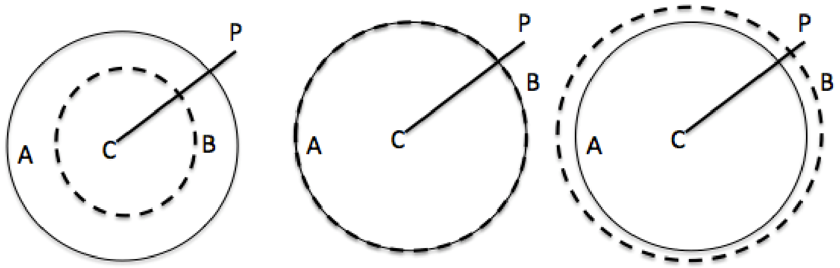
Example 1. *Consider the case depicted in Fig. 5: (a) the object B is inside A; (b) B overlaps A, (c) B is outside object A.* Let C be an arbitrary point in A. The semi-infinite ray CP does not intersect B in each case. The ray CQ intersects tangentially at a point (counted as double point) on B in each case. The ray CR cross intersects B at two points in each case. In each case ray-object intersection parity is even. But this does not tell anything about IntInt(A,B), which is true in Fig. 5(a,b) and false in Fig. 5(c). In Fig. 5(a,b), if C were selected to be inside B then there is odd parity. So the problem becomes one of selecting the test point as well. This will be revisited when we analyze Example 3 and Fig 7.



(a) (b) (c)

Fig. 5. Object B is inside, overlaps, and is outside of object A, in (a), (b), and (c) respectively. The ray CP does not intersect B, CQ tangentially intersects B, and CR cross intersects B at two points. In (a) and (b) IntInt(A,B) is true, whereas in (c) IntInt(A,B) is false. Thus even parity is inconclusive.

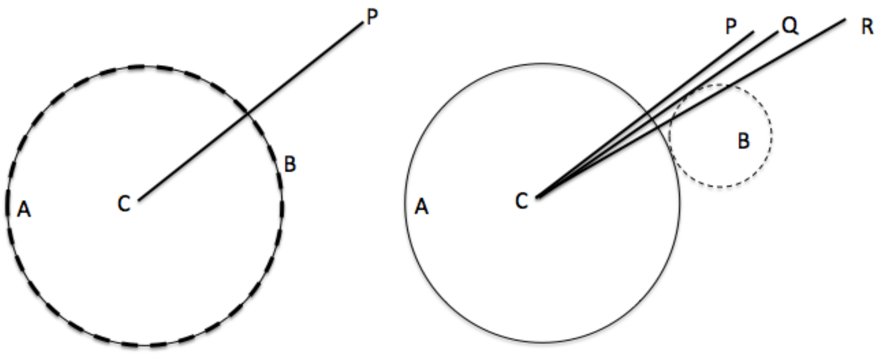
Example 2. *Consider the case depicted in Fig. 6: (a) the object B is inside A; (b) B equals A, (c) B is contains object A.* Let C be an arbitrary point in A. In each case, the semi-infinite ray CP intersects B, ray-object intersection parity is odd and IntInt(A,B) is true. One would be tempted to think the odd parity rule works all the time. However it depends on how you select the test point C. In Fig. 5(a) and Fig. 6(a), B is inside A, the ray CP intersects B, parity is not odd (even) in Fig. 5(a), but parity is odd in Fig. 6(a). In both the cases IntInt(A,B) is true. Thus it is inconclusive to apply the parity rule unless one has selected the test point, C, appropriately.



(a) (b) (c)

Fig. 6. Object B is inside, identical, outside of object A. The ray CP intersects all of them in one point. Odd intersection parity and IntInt(A,B) is true.

Example 3. *Consider the case depicted in Fig. 7: (a) the object B is equals A; (b) B is touches object A.* Referring back to the decision tree construction shown in Fig. 4(a, b), suppose that predicate BndBnd(*A,B*) is true, and IntBnd(*A,B*) and BndInt(*A,B*) are false. There are only two such configurations, examples of which are depicted in Fig. 7(a,b). The test point *C* can be located at any position in *A*. In the case Fig. 7(a) the ray-object intersection parity is odd and IntInt(*A,B*) is true. In Fig. 7(b) the ray-object intersection parity is even and IntInt(*A,B*) is false. So we arrive at the conclusion, that under these conditions, we can unambiguously apply the odd parity rule to test IntInt(*A,B*).



(a) (b)

Fig. 7. BndBnd(A,B) is true, IntBnd(*A,B*), and BndInt(*A,B*) is false. The odd parity rule holds in (a) and IntInt(*A,B*) is true, whereas in (b) odd parity does not hold and IntInt(*A,B*) is false.

4 Optimizing an Application

In general, the qualitative spatial reasoning problem concerns determining the spatial relations among pairs of objects (or regions) in a given collection. Qualitative reasoning is utilized in many applications and requires computation over data sets of hundreds of thousands of 2D/3D points. Consequently, many attempts have been made to speed up the determination of RCC8 relations. VRCC-3D+, a region connection calculus for 3D objects, utilizes three of the 9-Intersection predicates (specifically, IntInt, IntBnd, and BndInt) to detect occlusion in 2D. Yet it also qualifies the connectivity between objects using the RCC8 relations, albeit considered using 3D rather than the more traditional 2D data. Here we briefly describe the historical approaches used to reduce computation time and explain how our approach extends the previous approaches to optimize algorithmic and computation time

Again the primary question is if we are presented with two spatial objects, how do we find the RCC8 relation between them? Without additional information, it is equally likely to be any one the eight JEPD RCC8 relations. Let us first enumerate some of the strategies employed in the algorithms for relation identification:

(a) The brute force and worst case approach is to compute each of the nine intersection predicates and test each of the eight RCC8 relations to determine which one is the actual relation between the two objects (or regions) of interest.

(b) To avoid testing all the eight relations, a composition table [2] is used to consider what is known about the spatial relations between other pairs of objects in the scene, and eliminate the impossible relations between the pair of objects of interest This reduces the number of possible relations to test. However, for each possible relation, 9-Intersection predicates still are computed.

(c) To improve the computation time of the algorithm, Quinlan's ID3 algorithm is used to order the 9-Intersection predicates based on which predicate will provide the most information about the RCC8 relations that could hold for the pair of objects of interest (again, using knowledge of what is already known about the spatial relations between other objects in the scene) [10]. The algorithm orders the predicates, but does not necessarily reduce the number of intersection predicates that must be computed.

(d) To reduce the computational overhead of the 9-Intersection predicates, it was determined that all the nine predicates are not necessary to distinguish the RCC8 relations [6]. The 9-Intersection predicates were replaced with 8-Intersection, then subsequently 4-Intersection predicates were determined to be sufficient for identifying any RCC8 relation [3].

Prior to and independent of steps (b)-(d), an attempt was made to sort the predicates based on complexity [11]. It was determined that such a calculation would depend on hardware implementation of the predicates (e.g., availability of

GPUs). No conclusive decision could be made as to which predicate is faster to implement and apply.

This made the problem more interesting from an algorithmic point of view and the subject of further exploration. The number of predicates used to classify a relation is of paramount importance. Even a slight speed up can make an enormous difference in the usability of an application. Each operation requires significant computation "under the hood", so our goal is to avoid it as much as possible. As shown in Section 3, we have reduced the computation of predicates without sacrificing the accuracy. One predicate IntInt was justifiably removed from the 4-Intersection decision tree, see Fig. 4(b). There are two RCC8 relations that require the IntInt predicate. But functionality of that predicate can be supported with the more efficient ray-object intersection. While there is a little conceptual difference, there is significant mathematical and computational difference.

5 Conclusion

For qualitative spatial reasoning, a major task is to compute the intersections between regions. This paper provides a shorter and more efficient path to determine RCC8 relations. Specifically we have seen that 9-Intersection was reduced to 8-Intersection, which was further reduced to 4-Intersection models in the past. In this paper we have moved a step further to reduce predicate computations to almost three intersections, 3^+-Intersection. In this way, six of the RCC8 relations are identified with at most 3-Intersections. The remaining two RCC8 relations are resolved with a 4^{th} intersection which replaces the complex IntInt predicate with a simple point-in-object test using an odd-parity rule. The average number of predicate tests for a relation has been reduced to less than 3. This amounts to saving at least 12% in computation time, see Table 5. This is considerably more efficient than the previously used intersection frameworks, particularly when there are tens of thousands of regions to be analyzed in a dataset, and hundreds of thousands of data points. We intend to use this improved characterization in our future work in spatial reasoning, and hope others will find it useful as well.

References

[1] Dutt, S.: Approximate Spatial Reasoning Integrating qualitative and quantitative constraints. International Journal of Approximate Reasoning 5, 307–331 (1991)

[2] Egenhofer, M.J.: Deriving the Composition of Binary Topological Relations. Journal of Visual Languages and Computing 5(2), 133–149 (1991)

[3] Sabharwal, C., Leopold, J.: Reducing 9-Intersection to 4-Intersection for Identifying Relations in Region Connection Calculus. In: Proceedings of the 24th International Conference on Computer Applications in Industry and Engineering (CAINE 2011), Honolulu, Hawaii, November 16-18, pp. 118–123 (2011)

[4] Sabharwal, C.L., Factor, J.D.: Cross Intersection Between Any Two C^0 Parametric Surfaces. In: Proceedings of the International Australian Computer Graphics Conference, Ausgraph 1988, Melbourne, Australia, pp. 37–42 (1988)

[5] Randell, D.A., Cui, Z., Cohn, A.: A Spatial Logic Based on Regions and Connection. In: Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning, pp. 165–176 (1992)
[6] Albath, J., Leopold, J., Sabharwal, C.: Visualization of Spatio-Temporal Reasoning Over 3D Images. In: Proceedings of the 2010 International Workshop on Visual Languages and Computing (in Conjunction with the 16th International Conference on Distributed Multimedia Systems), Oak Brook, IL, October 14-16, pp. 277–282 (2010)
[7] Held, M., Klosowski, J.T., Mitchell, J.S.B.: Evaluation of Collision Detection Methods for Virtual Reality Fly-Throughs, Proc. 7th Canad. In: Gold, C., Robert, J.M. (eds.) Proc. 7th Canad. Conf. Computat. Geometry, Québec City, Québec, Canada, August 10-13, pp. 205–210 (1995)
[8] Allen, J.F.: Maintaining knowledge about temporal intervals. Communications of the ACM 26, 832–843 (1983)
[9] Sabharwal, C.L., Leopold, J.L.: Evolution of Region Connection Calculus to VRCC-3D+. In: Wang, P.P. (ed.) New Mathematics and Natural Computation (World Scientific) Cognition, Intelligence & Language. World Scientific Publishing Co. Pte. Ltd. (to appear)
[10] Eloe, N., Leopold, J., Sabharwal, C., McGeehan, D.: Efficient Determination of Spatial Relations Using Composition Tables And Decision Trees. In: IEEE Symposium on Computational Intelligence for Multimedia, Signal and Vision Processing (CIMSIVP), Singapore, pp. 1–7 (2013)
[11] Albath, J., Leopold, J., Sabharwal, C., Perry, K.: Efficient 3D Qualitative Spatial Reasoning with RCC-3D. In: Proceedings of KSEM 2010, pp. 470–481 (2010)