

# Qualitative Spatial Reasoning for Question-Answering: Axiom Reuse and Algebraic Methods

Tomás E. Uribe and Vinay Chaudhri and Patrick J. Hayes\* and Mark E. Stickel

Artificial Intelligence Center  
SRI International  
333 Ravenswood Ave.  
Menlo Park, CA. 94025  
uribe@ai.sri.com

\* Institute for Human & Machine Cognition  
The University of West Florida  
40 South Alcaniz Street  
Pensacola, FL 32501  
phayes@ai.uwf.edu

## Abstract

Knowledge-based question answering relies on declarative knowledge and an inference procedure such as theorem proving. In this paper, we explore question answering based on spatial knowledge. We first consider a broad general-purpose axiomatic theory covering different aspects of qualitative spatial representation such as topology, orientation, distance, size, and shape. Since it can be expensive to build such a theory from scratch, we *heuristically slice* out a spatial subset of the Cyc knowledge base as a starting point for our work. We also explore a number of techniques to support efficient reasoning. The first is the RCC8 calculus, supported by the use of composition tables. We present a general-purpose mechanism for integrating composition tables into a first-order theorem prover. We also present a novel calculus to support reasoning with orientation. Finally, since a theoretical expressiveness analysis of such a broad spatial theory is not feasible, we develop a test suite of questions as a qualitative measure of the knowledge it captures. We also show how such a theory can serve as a special-purpose reasoning module in a larger system that is not limited to spatial queries.

## Introduction

Question answering based on declarative knowledge and a deductive engine can greatly enhance the performance of a knowledge-based system. Reasoning capabilities are directly related to the system's ability to answer questions. Furthermore, reasoning allows testing knowledge as it is entered, finding gaps and errors early in the knowledge base (KB) creation process.

Spatial inferences are fundamental to human question answering. Any large KB designed to handle a broad range of questions must include spatial reasoning. In particular, we focus on the task of performing *qualitative spatial reasoning*. Significant work on qualitative spatial reasoning has already been done, and numerous spatial representations and calculi to support efficient reasoning are available (Cohn *et al.* 1997; Cohn & Hazarika 2001). Our primary goal in this paper

is not to invent new spatial representation formalisms, but instead capitalize on existing results, and construct a module that can provide spatial reasoning services in the context of a larger system that is not necessarily restricted to spatial knowledge.

Our work is being done in the context of SHAKEN, a knowledge entry system developed as part of DARPA's Rapid Knowledge Formation (RKF) program. The primary goal of SHAKEN is to allow efficient knowledge entry by subject matter experts (SMEs) who are not necessarily knowledge engineers, or experts in logic. Therefore, the deductive component of the system must be as automatic and transparent as possible, so as not to burden users with tasks that are not central to the knowledge entry and editing process. At the same time, it must support the spatial inferences that the domain expert would find natural.

We first describe the architecture of the spatial reasoning subsystem and provide some background on the relevant components, and how they fit in the overall SHAKEN system. We then describe how an initial general-purpose spatial theory was extracted from Cyc. Next, we discuss two particular spatial theories, namely, RCC8 and our new orientation algebra, and how we integrate them into our theorem prover. We give a comprehensive list of questions that we used to evaluate the spatial theory and show how the spatial reasoning module integrates into SHAKEN. We conclude with a summary, conclusions, and directions for future work.

## Architecture

SHAKEN is a system for knowledge entry through the graphical assembly of concepts (Clark *et al.* 2001). SMEs enter knowledge into SHAKEN by modifying and combining concepts, starting from a predefined set of basic concepts and slots. Each concept is represented as a graph, but can also be translated to a textual representation in first-order logic. Concepts are edited and combined through the Web-based SHAKEN graphical interface (Thomere *et al.* 2002). This GUI also lets users ask questions, using templates. Each template has several variables that are specified by choosing values from pull-down menus.

Figure 1 describes the high-level architecture of

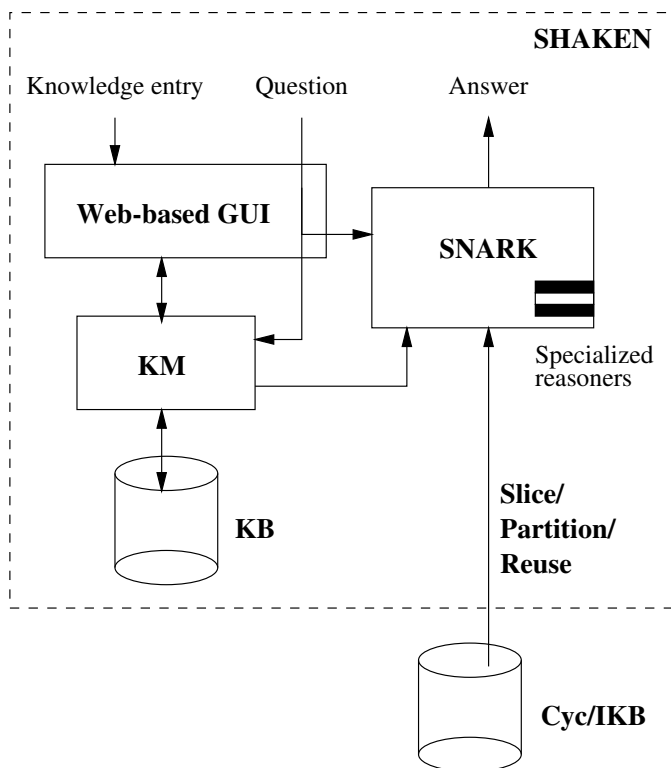


Figure 1: Architecture Outline

SHAKEN that is relevant to this paper (other shaken subsystems include analogy, metrics, and knowledge-based testing and diagnostics). Knowledge entered into SHAKEN is stored and managed using knowledge machine (KM) (Clark & Porter 1999). KM is an object-oriented knowledge representation system with the representation power of full first-order logic, extended with a STRIPS representation of actions. KM handles many of the object-oriented queries, as well as queries involving process representation. Questions that require spatial inference are dispatched to the SNARK prover. We discuss the KM-SNARK interface in the latter part of this paper.

## SNARK

SNARK (SRI's New Automated Reasoning Kit) (Stickel, Waldinger, & Chaudhri 2001) is an automated theorem-proving program being developed in Common Lisp. The principal inference rules in SNARK are resolution and paramodulation, and its style of theorem proving is similar to that of OTTER (McCune 1994). Some distinctive features of SNARK are its support for special unification algorithms, sorts, nonclausal formulas, answer construction for program synthesis, procedural attachment, and extensibility by Lisp code.

## IKB

IKB is a small subset of the Cyc<sup>©</sup> knowledge base (Cy-Corp 1997) that has been distributed to the participants in DARPA's Rapid Knowledge Formation program. IKB is written in the MELD knowledge representation language, and provides a broad and comprehensive coverage for concepts and axioms for an upper ontology. In addition, it contains domain-specific knowledge for several domains, for example, crisis management.

IKB provides a first-order axiomatization of basic spatial representational primitives. It includes axiomatization of more than 65 spatial predicates, which include 15 different kinds of containment and 7 different kinds of covering. Representations are also available for shapes, boundaries, regions, and convex hulls. Here are some example predicates:

```
objectFoundInLocation(object, location)
in-ContGeneric(object1, object2)
between(x, y, z)
near(x, y)
touches(x, y)
```

Here,  $x$ ,  $y$  and  $z$  can be either objects or locations: an object represents its own location.

## Shaken Knowledge Base

The KB, also called the *component library*, contains a collection of components representing (1) general knowledge about common physical objects and events, states of existence, and core theories, including time, space, and causality, and (2) more specialized knowledge about microbiology and biological warfare agents. By a component, we mean a coherent set of axioms that describe some abstract phenomenon (e.g., the concept of "invade") and are packaged into a single representational unit.

In the SHAKEN KB, the spatial predicates appear as predefined slots, which SMEs can use when defining new concepts. Since SHAKEN adopts the strategy of providing only a limited vocabulary, it supports only a handful of basic spatial relations such as *above*, *below*, *along*, and *inside*.

## Examples

Here are a few examples of questions that require spatial reasoning:

- If *DNA-strand* is part of *nucleus* and *nucleus* is inside *cell*, then *DNA-strand* is inside *cell*.
- If *Cell11* and *Cell12* are disjoint, and *DNA-strand* is part of *Cell11*, then *DNA-strand* is not part of *Cell12*.
- If *nucleus* is inside *Cell* and *Cell* is at location  $x$ , then *nucleus* is at location  $x$ .

## Axiom Reuse

We now describe the process of reusing the Cyc/IKB axioms. We apply the terminology of (Chaudhri *et al.*

2000), which identifies the following generic knowledge reuse steps:

1. The *translation* step is a syntactic transformation from the language of the reused knowledge base to that of the target application.
2. The *comprehension* step refers to understanding the semantics of the reused ontology.
3. The *slicing* step selects the relevant subset of the pre-existing axioms.
4. The *reformulation* step, also known as *morphing*, transforms the representation of the input theory into that of the target language.
5. The *merging* step is the process of combining the different KBs being used, eliminating redundant information.

Our previous work on knowledge reuse (Chaudhri *et al.* 2000) was limited to only taxonomic knowledge. In the present paper we extend the reuse work to include full first-order axioms. We now describe in more detail how each step was performed in the particular case of reusing Cyc’s IKB axioms.

## Translation

Since IKB axioms are encoded in MELD, we need a way to import the axioms into the SNARK theorem prover. To support this functionality, we constructed a translation module. The translation required us to address three kinds of problem: constant name mapping, syntactic transformation, and axiom expansion. Constant name mapping involved identifying equivalent constant names between MELD and SNARK’s input language. Syntactic transformation involved ensuring that SNARK is able to accept the MELD syntax. Axiom expansion was needed because for many of the axioms, Cyc uses special-purpose reasoning modules, and the axioms defining them are not explicitly in the theory. For example, one such axiom involves `#$AntiTransitiveBinaryPredicate`:

```
(#$isa #$in-ContGeneric
  #$AntiTransitiveBinaryPredicate).
```

This axiom is translated to

```
(#$implies
  (#$in-ContGeneric ?x ?y)
  (#$not (#$in-ContGeneric ?y ?x)))
```

Similar axiom expansions were needed for axioms involving reflexive, transitive, and symmetric relations.

## Comprehension

To comprehend the spatial ontology, we extensively used IKB’s knowledge base browser. To further enhance the effectiveness of comprehension, we used the knowledge base clustering tools that allowed us to group similar axioms next to each other (Mehrotra *et al.* 2002). We also constructed predicate dependence graphs showing how predicates were defined in terms of each other.

Even though the tools were a considerable help, we still had to invest significant manual effort to gain a good understanding of the ontology.

## Slicing

In our previous work on slicing, we developed an algorithm to slice out a portion of a knowledge base taxonomy (Chaudhri *et al.* 2000). The present work extends that algorithm to include full first-order axioms.

The slicing problem may be formally stated as follows: Given a knowledge base  $B$  and a *seed*  $S$  containing the constant symbols we are interested in, we want to extract a set of axioms  $A \subseteq B$  that captures the *relevant* knowledge about  $S$ . We say that  $A$  is *complete with respect to*  $S$  if any theorem built from the symbols in  $S$  that follows from  $B$  also follows from  $A$ . We say that  $A$  is *minimal with respect to*  $S$  if no subset of  $A$  is complete with respect to  $S$ .

A complete slice for seed  $S_0$  can be obtained as follows:

1. Let  $S = S_0$ . The set  $S$  will contain the symbols and formulas in  $B$  considered to be of interest so far.
2. For all the classes and individuals in  $S$ , compute their upward taxonomic closure of the taxonomy of  $B$ , and add them to  $S$ .
3. For all the relations in  $S$ , identify the classes appearing in type constraints on those relations, compute their upward taxonomic closure, and add them to  $S$ .
4. For all rules in  $B$  that mention any constant symbol in  $S$ , identify all the constant symbols appearing in them. Compute their upward taxonomic closure and add them to  $S$ . Repeat until no more rules can be added to  $S$ .

At the end of this process, the axioms in  $S$  are precisely the following:

- All the axioms with `instance-of` and `subclass-of` relations involving any members of  $S$ .
- All the axioms involving type constraints on the relation symbols in  $S$ .
- All the axioms that mention any symbol in  $S$  (and, in particular, any symbol in  $S_0$ ).

The slice produced by the above algorithm is therefore complete, but not always minimal. In our previous work, we show how one could compute a minimal taxonomic closure that preserves completeness. In a similar spirit, we propose here the following heuristics that reduce the set of rules that are considered in Step 4 of the algorithm.

- Any rule in  $B$  that mentions a class  $X$  that is not in  $S_0$ , where  $X$  is a subclass of some class in  $S_0$ , need not be included in the slice. The intuitive justification for this step is that since  $X$  is not in  $S_0$ , it will not appear in any query. Furthermore,  $X$  is a subclass of some class  $Y$  in  $S_0$ , and since classes are

defined top-down by specializing their axioms, an axiom mentioning  $X$  is inherently about some subclass of  $Y$ . For example, if the concept of `tree` is in  $S_0$ , then any axiom mentioning `redwood-tree` is inherently about concepts that are more specialized than `tree`. Such axioms do not need to be included in the slice, unless `redwood-tree` happens to be a part of the seed.

- Any rule in  $B$  that mentions a relation  $X$ , and classes or non-relation individuals  $Y_1, \dots, Y_n$ , none of which is in  $S_0$ , need not be included in the slice. The intuitive justification for this step is that such axioms are of interest only if they apply to a class that is of interest. For example, an axiom such as (`#$oppositeDirections #$North #$South`) need not be included in the slice if both `#$North` and `#$South` are not in  $S_0$ .
- Polarity considerations similar to those of resolution and indexing strategies can be used, depending on the queries of interest. For example, if the predicates of interest only appear with negative polarity in both the goal and a rule  $f$  (e.g., as a conjunct in the antecedent  $a$  of an implication  $a \supset c$ ), then  $f$  need not be included in the slice. On the other hand,  $f$  is accepted if any predicate in  $S$  appears with positive polarity in  $f$  (e.g., in the consequent of an implication or either side of an `iff` formula).

## Reformulation

The predicate expansion step considered in the translation step can be viewed as reformulation.

## Merging

The Cyc spatial axioms, encoded in MELD, serve as a background theory that is implicit, but not present, in the SHAKEN KB, encoded in the KM syntax. Since both are given to SNARK to support the reasoning, it is not necessary to merge the axioms into one single knowledge base.

This axiom extraction process yielded a total of 704 clauses. With these IKB axioms as background facts, SNARK can successfully answer simple spatial questions such as the ones described above.

We also consider a final *extension* step, by which the reused axioms are augmented with new ones. In our case, this takes the form of integrating special-purpose reasoning, as described below.

## Built-in Reasoning

Using a full first-order theory such as the one in IKB, it is possible to obtain a good coverage of different aspects of spatial representation. However, such a theory is not always efficient to reason with. Therefore, we consider here two spatial calculi that have efficient reasoning support. This combines the generality of first-order theorem proving with the efficiency of specialized reasoning.

## Orientation Algebra

The representation of orientation is a fundamental aspect of spatial representation. Geographic directions, and notions of parallel, antiparallel, and layers can be reduced to the fundamental primitives of orientation. Here we consider a specialized theory for three-dimensional orientation that has only one primitive (Hayes 2001). We now give a brief description of this theory.

The relative orientation of objects is captured using a fixed set of directions

$$\{\text{up, down, left, right, forward, back}\},$$

abbreviated as U,D,L,R,F,B, respectively, a unary function  $\text{op}(X)$ , which gives the opposite direction of  $X$  (so, for instance,  $\text{op}(\text{left}) = \text{right}$ ), and a binary function

$$C(X, Y),$$

which describes how direction  $Y$  is transformed after a  $\pi/2$  clockwise rotation over axis  $X$  (so, for instance,  $C(\text{forward}, \text{left}) = \text{up}$ ). This gives the following table for  $C$ :

$X \backslash Y$	U	F	R	D	B	L
U	U	L	F	D	R	B
F	R	F	D	L	B	U
R	B	U	R	F	D	L
D	U	R	B	D	L	F
B	L	F	U	R	B	D
L	F	D	R	B	U	L

If we let  $\text{dir}(x)$  be the direction of  $x$ , that  $a$  and  $b$  are parallel can be expressed as

$$\text{dir}(a) = \text{dir}(b);$$

that they are antiparallel, as

$$\text{dir}(a) = \text{op}(\text{dir}(b));$$

that they are orthogonal, as

$$\text{dir}(a) \neq \text{dir}(b) \wedge \exists x. \text{dir}(a) = C(x, b) .$$

The resulting algebra can be built into SNARK using rewrite rules or procedural attachment. Formulas built from the interpreted function symbols and predicates of the theory can then be efficiently evaluated.

## Region Connection Calculus

Like orientation, topological spatial relations are also fundamental for qualitative spatial reasoning. The RCC8 *region connection calculus* (Cohn *et al.* 1997) is a well-known technique for reasoning about them. RCC8 defines a set of jointly exhaustive and pairwise disjoint (JEPD) binary relations between regions and a composition table that encodes compatibility of relations among triples of regions. We now describe how specialized reasoning with JEPD binary relations and composition tables is possible in the SNARK theorem-prover.

For any pair of regions  $x$  and  $y$ , exactly one of the following eight primitive relations holds:

TPP( $x, y$ )	( $x$ is a tangential proper part of $y$ )
NTPP( $x, y$ )	( $x$ is a nontangential proper part of $y$ )
DC( $x, y$ )	( $x$ is disconnected from $y$ )
EC( $x, y$ )	( $x$ is externally connected to $y$ )
PO( $x, y$ )	( $x$ partially overlaps $y$ )
EQ( $x, y$ )	( $x$ is equal to $y$ )
NTPPi( $x, y$ )	( $y$ is a nontangential proper part of $x$ )
TPPi( $x, y$ )	( $y$ is a tangential proper part of $x$ )

A *composition table* specifies which of the JEPD binary relations can possibly hold between  $x$  and  $z$  given those that hold between  $x$  and  $y$  and between  $y$  and  $z$ . The composition table for RCC8 thus encodes the information in statements such as

$$\begin{aligned} \text{TPP}(x, y) \wedge \text{TPP}(y, z) \supset \\ \text{TPP}(x, z) \vee \text{NTPP}(x, z) . \end{aligned}$$

As in this example, where it is uncertain whether  $x$  is a tangential or nontangential proper part of  $z$ , knowledge of what relation holds between a pair of regions may be inexact and represented by a set of possible primitive relations. Many useful inexact relations can be defined as disjunctions:  $x$  is a proper part of  $y$  iff  $x$  is a tangential or nontangential proper part of  $y$ ;  $x$  is a part of  $y$  iff  $x$  is a proper part of or equal to  $y$ ;  $x$  and  $y$  are discrete iff  $x$  is disconnected from or externally connected to  $y$ .

RCC8 reasoning can be done by constraint propagation among triples of nodes in a graph whose nodes represent regions and whose edges are labeled by subsets of

$$\{\text{TPP}, \text{NTPP}, \text{DC}, \text{EC}, \text{PO}, \text{EQ}, \text{NTPPi}, \text{TPPi}\} .$$

Inconsistency is detected when an edge is labeled with the empty set.

Although this constraint propagation approach is incomplete in general, it is powerful and efficient enough for us to want to make it available in SNARK. We previously incorporated this kind of reasoning into SNARK by using constrained resolution (Bürckert 1991) in conjunction with a constraint propagation procedure. However, this approach had a number of drawbacks: it required that much code (for the constraint propagation procedure) be added to SNARK, it used multiple representations of information both as literals in the theorem-prover and as nodes and edges in the constraint propagation procedure, and the interface between the reasoners was complicated.

We now use a simpler approach that enables us to easily adapt SNARK to reason with composition tables for sets of JEPD binary relations including RCC8 and Allen's temporal intervals (Allen 1983). This requires encoding a pair of nodes and the label of the edge between them in a single literal and providing procedural attachments for relation composition, intersection, and inverse.

For RCC8, all the relations are replaced by the single ternary relation `RCC8-relation` whose first and second

arguments are regions and whose third argument is an encoding of the set of possible primitive relations between the regions. When using a bit-vector encoding,  $A$  being a proper part of  $B$  can be written as

$$\text{RCC8-relation}(A, B, \#b11000000),$$

with the eight bits corresponding in order to the eight relations above. The interpretation is that  $A$  might be a tangential proper part of  $B$  (since the first bit is 1),  $A$  might be a nontangential proper part of  $B$  (since the second bit is 1),  $A$  cannot be disconnected from  $B$  (since the third bit is 0), and so on. Negative RCC8 information is also encoded by positive literals. For example, that  $A$  is not equal to  $B$  would be written as

$$\text{RCC8-relation}(A, B, \#b11111011) .$$

Literals of the form `RCC8-relation`( $A, B, \#b11111111$ ) and `RCC8-relation`( $A, B, \#b00000000$ ) can be simplified to `true` and `false`, respectively.

We use a superior enhanced encoding that uses an eight-element list instead of a bit-vector. The elements of the list are 1 if the relation might hold, and a variable (instead of 0) if the relation cannot hold. Each such variable is required to be unique—it does not occur anywhere else—and will be written as “?”.

This encoding allows a theorem-prover's ordinary factoring and subsumption operations to reason appropriately about sets of possible relations. For example, the clause

$$\text{RCC8-relation}(A, B, [1, 1, ?, ?, ?, 1, 1, 1])$$

can be derived by factoring the clause

$$\text{RCC8-relation}(A, B, [1, 1, ?, ?, ?, 1, ?, ?])$$

$$\vee \\ \text{RCC8-relation}(A, B, [?, ?, ?, ?, ?, 1, 1, 1]) .$$

Because the clause

$$\text{RCC8-relation}(A, B, [1, 1, ?, ?, ?, ?, ?])$$

is a substitution instance of each of the clauses

$$\text{RCC8-relation}(A, B, [1, ?, ?, ?, ?, ?, ?])$$

and

$$\text{RCC8-relation}(A, B, [?, 1, ?, ?, ?, ?, ?]),$$

either of the latter could be used to eliminate the former by subsumption.

The remaining task of automating RCC8 reasoning in SNARK is adding the axioms

- (1) `RCC8-relation`( $x, x, [?, ?, ?, ?, ?, 1, ?, ?]$ )
- (2) `RCC8-relation`( $x, y, l_1$ )  $\wedge$  `RCC8-relation`( $x, y, l_2$ )  
 $\supset$  `RCC8-relation-intersection`( $l_1, l_2, x, y$ )
- (3) `RCC8-relation`( $x, y, l_1$ )  $\wedge$  `RCC8-relation`( $y, z, l_2$ )  
 $\supset$  `RCC8-relation-composition`( $l_1, l_2, x, y, z$ )

The first axiom expresses reflexivity of the RCC8 EQ relation. In the second axiom, the

$$\text{RCC8-relation-intersection}(l_1, l_2, x, y)$$

predicate symbol is procedurally attached so that when  $l_1$  and  $l_2$  are instantiated to lists representing sets of possible primitive relations it will return a literal of the form

`RCC8-relation( $y, x, [v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7]$ )`

whose third argument encodes the (inverse of the) intersection of  $l_1$  and  $l_2$ . Note the argument order  $y, x$  in the consequent versus  $x, y$  in the antecedent, so that this axiom can compute inverse as well as intersection.

In the third axiom, the

`RCC8-relation-composition( $l_1, l_2, x, y, z$ )`

predicate symbol is procedurally attached so that when  $l_1$  and  $l_2$  are instantiated to lists representing sets of possible primitive relations it will return a literal of the form

`RCC8-relation( $x, z, [v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7]$ )`

whose third argument encodes the set of possible primitive relations (obtained by use of the composition table) between  $x$  and  $z$  that are consistent with the relationship between  $x$  and  $y$  being in  $l_1$  and the relationship between  $y$  and  $z$  being in  $l_2$ .

Besides being simple and easy to use, the current approach also provides easy paths to greater completeness and expressiveness.

Constraint propagation is efficient at the cost of being incomplete, since local consistency of a constraint graph may be insufficient to demonstrate total consistency (see (Allen 1983) for an example that is locally but not totally consistent). Total consistency can be tested after constraint propagation by backtracking search among alternative labelings when edges are labeled by nonsingleton sets of relations. In the current approach for using constraint propagation in a theorem-prover, the search among alternatives can be invoked implicitly by “unfactoring” literals that represent nonsingleton sets of relations into multiple literals and relying upon the theorem-prover to do complete reasoning about disjunctions. For example, the clause

`RCC8-relation( $A, B, [1, ?, ?, ?, ?, ?, ?, ?]$ )`  
 $\vee$

`RCC8-relation( $A, B, [?, 1, ?, ?, ?, ?, ?, ?]$ )`

can be derived by unfactoring the clause

`RCC8-relation( $A, B, [1, 1, ?, ?, ?, ?, ?, ?]$ ),`

forcing the theorem-prover to test the cases separately. Thus, the example in (Allen 1983) is found to be inconsistent by SNARK when given in unfactored form even though constraint propagation alone (or SNARK using only the factored form) fails to detect the inconsistency.

The current approach allows nonground terms as region arguments of `RCC8-relation`. Regions might be proven outside the RCC8 reasoner to be equal. These are possibilities that can pose substantial difficulty if the region arguments must be used as nodes in a separate constraint propagation procedure but they are handled naturally in the current approach.

## Improving Performance

Proof search using unrestricted resolution can be time and space consuming, particularly if the set of background axioms is large. Much effort has thus been devoted to identifying resolution restrictions and strategies to reduce the search space while preserving the completeness of the resolution method.

As an example of the effectiveness of these restrictions, answering a sample question using the reused IKB axioms required the generation of 29,000 formulas when unrestricted resolution was used. This took 5 minutes on a 700MHz Sun workstation with 1 gigabyte of RAM. Answering the same question using *ordered resolution* generated 1,200 formulas, and took 3 seconds on the same machine.

We want to achieve this consistently, and transparently to the user. The efficiency of reasoning over large sets of axioms can be improved by *partitioning* the axioms and restricting inferences to occur mostly within each partition, with fixed rules determining what inferred formulas can be communicated between different partitions (Amir & McIlraith 2000). Each partition can have a separate associated inference method.

We are investigating the application of partitioning techniques to improve the efficiency of the axiomatized spatial reasoning as part of the reformulation step for knowledge reuse. As a simple initial step, partitions can be used to generate term and predicate orderings, for use with the ordered resolution restriction (see above), preserving completeness.

## Test Question Suite

Most work on qualitative spatial representation has focused on characterizing the expressiveness of a narrowly defined calculus or algebra. It is difficult to do an expressive analysis of a theory that does not fit into a narrow scope. Motivated by this, we have started to develop a test suite of spatial queries. We plan to significantly expand this test suite so that a broad class of spatial inferences can be tested. We anticipate that the test suite will contain queries that cannot be answered by any single theory. We hope that such a test suite will drive the development of spatial reasoning systems. We list here only a few representative queries:

1. Suppose that A is touching B and B is inside C and C is at D. Is A at D?
2. If A is between B and C, and both B and C are at D, where is A?
3. A cell is a container. A nucleus is inside a cell. The cell is under a microscope. Where is the nucleus?
4. DNA-strand is a group of nucleotides. One of the members happens to be an adenine. DNA-strand is in the nucleus of the same cell as in the previous query. Where is that nucleotide?
5. If A is between B and C, and both B and C are inside D, and D is convex, where is A?

6. If A is inside B, is A inside the convex hull of B?
7. Suppose that A is touching B and B is inside C and C is at D. Suppose in addition that D is touching (or adjacent to) E. Is A touching (or adjacent to) E?
8. If C is part of A, D is part of B, and A and B are externally connected, then C and D are discrete (that is, C is disconnected from or externally connected to D).
9. If C is part of A, D is a nontangential proper part of B, and A and B are externally connected, then C and D are disconnected.

### KM-SNARK integration

We now take a closer look at how a spatial reasoning module can provide services in the context of a larger system. Since SHAKEN uses KM as the primary storage of knowledge, we must find a way so that when a question is posed to KM, the spatial reasoning module can be invoked when the need for spatial inference arises. This requires inventing an appropriate hybrid reasoning architecture.

One possible fully general alternative is to translate all the KM axioms into SNARK, and use only SNARK for answering questions. Such a solution is not cost-effective.

A more pragmatic approach is to invoke SNARK whenever a spatial inference is needed. A central problem in such an architecture is which facts to ship from the KM knowledge base to SNARK while answering a particular question.

While answering a question in SHAKEN, we always start from a concept, and compute all the slot values of that concept. Many of the slot values are Skolem individuals that are computed using KM's inference rules, which themselves may introduce new slot values. In general, this computation is recursive, and the number of ground facts produced can be infinite. In SHAKEN, while answering questions for a concept, all its slot values are always computed. This is necessary because the concepts are usually shown as graphs, and the slot values are essential to the graph representation. As an initial interface between KM and SNARK, one can simply export the top-level concept slot values.

As a simple example, we may ask the question

```
(|_Prokaryotic-Gene926| |has-part| |_Promoter916|)
```

from a set of ground facts that includes

```
(|_Prokaryotic-Gene926| |has-part| |_Template898|)
(|_Template898| |has-region| |_Promoter916|)
```

The question is transformed to

```
(|physicalParts| |_Prokaryotic-Gene926|
      |_Promoter916|))
```

while the facts are transformed to

```
(|physicalParts| |_Prokaryotic-Gene926|
      |_Template898|)
(|in-ContGeneric| |_Promoter916| |_Template898|)
```

The IKB axioms that relate `PhysicalParts` and `in-ContGeneric` can then be used by SNARK to answer the question.

Of course, these slot values are not always a complete set of facts necessary to answer spatial reasoning questions about a concept. We are considering the following extensions to generalize this solution. We could compute the slot values for spatial slots for several levels. For example, given a concept  $C$ , we will compute values  $V_1, \dots, V_n$  for its spatial slots. We will recurse by computing values for spatial slots for each of  $V_1, \dots, V_n$ . This process either will terminate naturally or can be cut off by setting a limit on the number of facts generated. Alternatively, instead of performing this computation in KM, we could export to SNARK just those rules in KM that apply to spatial slots. We plan to explore these alternative hybrid reasoning approaches in our future work.

### Summary and Conclusions

Our emphasis for future work is to prove the formal properties of our slicing procedures, to complete the hybrid reasoning interface between KM and SNARK, and to significantly expand our test suite of questions. We are also investigating the relationship between our orientation algebra and the two-dimensional orientation algebra of (Isli & Cohn 1998), when generalized to the 3D case.

To summarize, we extracted a broad general-purpose spatial theory from Cyc's IKB, and extended it to support efficient reasoning. Specifically, we considered an implementation of RCC8 calculus and algebra for reasoning with orientations. We considered an initial version of a test suite of questions to test spatial theories, and sketched a design of how a spatial theory may be integrated into a larger system. Finally, although we use spatial representation as our main example, we expect that the techniques for knowledge reuse and hybrid reasoning developed here will be more generally applicable.

### Acknowledgments

This research was supported by DARPA under contract N66001-00-C-8019.

### References

- Allen, J. F. 1983. Maintaining knowledge about temporal intervals. *Communications of the Association for Computing Machinery* 26(11):832–843.
- Amir, E., and McIlraith, S. 2000. Partition-based logical reasoning. In *7th Intl. Conference on Principles of Knowledge Representation and Reasoning (KR 2000)*.
- Bürckert, H.-J. 1991. *A Resolution Principle for a Logic with Restricted Quantifiers*, volume 568 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag.

- Chaudhri, V. K.; Stickel, M. E.; Thomere, J. F.; and Waldinger, R. J. 2000. Using prior knowledge: Problems and solutions. In *AAAI 2000*. IEEE Press.
- Clark, P., and Porter, B. 1999. *KM: The Knowledge Machine*. <http://www.cs.utexas.edu/users/mfkb/km.html>.
- Clark, P.; Thompson, J.; Barker, K.; Porter, B.; Chaudhri, V.; Rodríguez, A.; Thomere, J.; Gil, Y.; Hayes, P.; and Reicherzer, T. 2001. Knowledge entry as the graphical assembly of components. In *First Intl. Conference on Knowledge Capture*.
- Cohn, A. G., and Hazarika, S. M. 2001. Qualitative spatial representation and reasoning: An overview. *Fundamenta Informaticae* 1-2:1-29.
- Cohn, A. G.; Bennett, B.; Gooday, J.; and Gotts, N. M. 1997. Qualitative spatial representation and reasoning with the region connection calculus. *Geoinformatica* 1:1-44.
- CyCorp. 1997. Cyc public ontology. <http://www.cyc.com>.
- Hayes, P. 2001. Orientation algebra. University of West Florida. Draft manuscript.
- Isli, A., and Cohn, A. G. 1998. An algebra for cyclic ordering of 2D orientation. In *Proc. of 15th AAAI Conference*, 643-649. AAAI/MIT Press.
- McCune, W. 1994. *OTTER 3.0 Reference Manual and Guide*.
- Mehrotra, M.; Bobrovnikoff, D.; Chaudhri, V.; and Hayes, P. 2002. A clustering approach for analysis of knowledge bases. Draft manuscript.
- Stickel, M. E.; Waldinger, R. J.; and Chaudhri, V. K. 2001. *A Guide to SNARK*. <http://www.ai.sri.com/snark/tutorial/tutorial.html>.
- Thomere, J.; Rodríguez, A.; Chaudhri, V.; Mishra, S.; Ericksen, M.; Clark, P.; Barker, K.; and Porter, B. 2002. A Web-based ontology browsing and editing system. SRI International. Draft manuscript.