An Analogy Ontology for Integrating Analogical Processing and First-principles Reasoning

Kenneth D. Forbus

Qualitative Reasoning Group
Northwestern University
1890 Maple Avenue
Evanston, IL, 60201, USA
forbus@northwestern.edu

Thomas Mostek

Qualitative Reasoning Group
Northwestern University
1890 Maple Avenue
Evanston, IL, 60201, USA
tmostek@naviant.com

Ron Ferguson

College of Computing
Georgia Institute of Technology
801 Atlantic Avenue
Atlanta GA 30332, USA
rwf@cc.gatech.edu

Abstract

This paper describes an *analogy ontology*, a formal representation of some key ideas in analogical processing, that supports the integration of analogical processing with first-principles reasoners. The ontology is based on Gentner's *structure-mapping* theory, a psychological account of analogy and similarity. The semantics of the ontology are enforced via procedural attachment, using cognitive simulations of structure-mapping to provide analogical processing services. Queries that include analogical operations can be formulated in the same way as standard logical inference, and analogical processing systems in turn can call on the services of first-principles reasoners for creating cases and validating their conjectures. We illustrate the utility of the analogy ontology by demonstrating how it has been used in three systems: A crisis management analogical reasoner that answers questions about international incidents, a course of action analogical critiquer that provides feedback about military plans, and a comparison question-answering system for knowledge capture. These systems rely on large, general-purpose knowledge bases created by other research groups, thus demonstrating the generality and utility of these ideas.

Introduction

There is mounting psychological evidence that human cognition centrally involves similarity computations over structured representations, in tasks ranging from high-level visual perception to problem solving, learning, and conceptual change [21]. Understanding how to integrate analogical processing into AI systems seems crucial to creating more human-like reasoning systems [12]. Yet similarity plays at best a minor role in many AI systems. Most AI systems operate on a *first-principles* basis, using rules or axioms plus logical inference to do their work. Those few reasoning systems that include analogy (cf.[1,37]) tend to treat it as a method of last resort, something to use only when other forms of inference have failed. The exceptions are case-based reasoning systems [27,28], which started out to provide computational mechanisms similar to those that people seem to use to solve everyday problems. Unfortunately, CBR systems generally have the opposite problem, tending to use only minimal first-principles reasoning. Moreover, most of today's CBR systems also tend to rely on feature-based descriptions that cannot match the expressive power of predicate calculus. Those relatively few CBR systems that rely on more expressive representations tend to use domain-specific and task-specific similarity metrics. This can be fine for a specific application, but being able to exploit similarity computations that are more like what people do could make such systems even more useful, since they will be more understandable to their human partners.

While many useful application systems can be built with purely first-principles reasoning and with today's CBR technologies, integrating analogical processing with first-principles reasoning will bring us closer to the flexibility and power of human reasoning. This paper describes a method for doing this. The key idea is to use an *analogy ontology* that provides a formal, declarative representation of the contents and results of analogical reasoning. The semantics of this ontology is defined by the underlying theory of analogy and similarity, structure-mapping [18]. This semantics is enforced via procedural attachment, using the analogy ontology to set up computations that are solved by special-purpose analogical processing systems, and reifying the results of these programs in a way that can be used by first-principles reasoning systems.

We start by reviewing relevant aspects of structure-mapping theory. Then we present the analogy ontology, and outline its implementation. We then describe how these ideas have been used in three systems: analogical reasoning about historical precedents, critiquing military course of action sketches via cases, and answering comparison questions in knowledge capture systems. We close by discussing related and future work.

Prelude: Structure-mapping, SME, and MAC/FAC

In structure-mapping theory [18], an analogy match takes as input two structured representations (*base* and *target*) and produces as output a set of mappings. Each mapping consists of a set of *correspondences* that align base items

with target items and a set of *candidate inferences*, which are surmises about the target made on the basis of the base representation plus the correspondences. The constraints that govern mappings, while originally motivated by psychological concerns [21], turn out to be equally important for the use of analogy in case-based reasoning, since they ensure that candidate inferences are well defined and that stronger arguments are preferred [11].

Two simulations based on structure-mapping are relevant to this paper. The first, the Structure-Mapping Engine (SME) [1,7,11], is a cognitive simulation of analogical matching. How SME works is described in [6,7,11]. Two characteristics are key to the work described here:

- SME operates in polynomial time, using a greedy merge algorithm to provide a small number of mappings that best satisfy the constraints of structure-mapping.
- SME's results are consistent with a large and growing body of psychological results on analogy and similarity [9], making its answers and explanations more likely to be accepted by human collaborators.

The second simulation, MAC/FAC, is a two-stage model of similarity-based retrieval that is consistent with psychological constraints [14] and has been used in a fielded application [16]. The key insight of MAC/FAC is that memory contents should be filtered by an extremely cheap match that filters a potentially huge set of candidates, followed by a structural match (i.e., SME) to select the best from the handful of candidates found by the first stage[1]. The extremely cheap match is based on *content vectors*, a representation computed from structured descriptions. Content vectors are useful because the dot product of two content vectors provides an estimate of the quality of match between the corresponding structural descriptions (see [14] for details).

The Analogy Ontology

The analogy ontology defines the types of entities and relationships used in analogical processing. The ontology can be grouped into *cases*, *matches*, *pragmatic constraints*, *mappings*, *correspondences*, *candidate inferences* and *similarities and differences*. We discuss each in turn.

Cases: Cases are collections of statements, treated as a unit. We use the relation `(ist-Information ?case ?fact)` to indicate that statement `?fact` is in the case `?case`.[2] Cases can be explicitly named in the KB, via the function `explicit-case-fn`, whose single argument is a term naming the case.

One important way of encoding task constraints in analogical reasoning is by specifying different collections of information about a term to be gathered from a knowledge base. In the ontology these are represented simply as additional functions. (How such functions are implemented is described in [30], and their integration into reasoning systems described below) Here are some functions that we have found useful in several tasks:

- `(minimal-case-fn ?thing)` denotes all of the statements in the KB that directly mention `?thing`, where `?thing` can be any term. This is rarely used in isolation, instead providing a convenient starting point for defining other case functions.
- `(case-fn ?thing)` is the union of `(minimal-case-fn ?thing)` and attribute information about all of the ground terms occurring in `(minimal-case-fn ?thing)`, e.g., in the case of Cyc, the attribute information consists of the `isa` statements involving the ground terms. When `?thing` is an event, all causal links between the terms are included in addition to attribute information. When `?thing` is an agent, all facts whose ground terms are those in `(minimal-case-fn ?thing)` are included. This is useful when a quick comparison is required, to see if the match is worth pursuing further.
- `(recursive-case-fn ?thing)` denotes the union of `case-fn` applied to `?thing` and to all of its subparts, recursively. What constitutes a subpart depends on the KB's ontology. For instance, if `?thing` is an event, its subparts are subevents.
- `(in-context-case-fn ?thing ?case)` denotes the subset of `(case-fn ?thing)` that intersects the case `?case`. This is used for more focused queries, e.g., Alabama in the context of the US Civil War.
- `(no-postlude-case-fn ?thing)` denotes `(case-fn ?thing)` with all causal consequences of `?thing` removed. This is useful in reasoning about alternative outcomes of a situation, since it suppresses knowledge of the actual outcomes.

The particular definitions for agent, event, causal consequences, and subparts depend on the KB used. In the KB's we have worked with to date (Cyc, SAIC, and KM) it has sufficed to enumerate short lists of predicates used to identify these concepts. For instance, in using Cyc, members of the collections `Event` and `Agent` are treated as events and agents respectively, and short lists of relationships were identified as expressing causal and mereological consequences. We expect that this technique will work for any knowledge base that includes some version of these concepts.

Matches: Matches represent a comparison between a base and target. They consist of a set of *correspondences* and *mappings*, which are described shortly. The relation `(match-between ?base ?target ?match)` indicates that `?match` is the result of SME comparing the cases `?base` and `?target`. There are two variations that are like `match-between`, except that the process they specify is slightly different. `recursive-match-between` dynamically expands the cases to resolve competing entity matches. This is useful in dealing with large, complex cases. `seeking-match-between` only considers matches that reflect required or excluded correspondences derived from task constraints. The procedural semantics of `recursive-match-between` and `seeking-match-between` are detailed

[1] Hence the name: Many Are Called/Few Are Chosen

[2] `ist-Information` is drawn from the Cyc upper ontology.

in [30].

Mappings: In structure-mapping, a mapping consists of three things: A structurally consistent set of correspondences, a set of candidate inferences, and a structural evaluation score [6,7,11]. How correspondences and candidate inferences are represented in this ontology is described below. The structural evaluation score is an estimate of match quality. The function `structural-evaluation-score` denotes the score computed by SME for the mapping. The relation `(mapping-of ?mapping ?match)` indicates that mapping `?mapping` is part of match `?match`, i.e., that it is one of the solutions SME found to the comparison. The relation `(best-mapping ?match ?mapping)` indicates that `?mapping` has the highest structural evaluation score of `?match's` mappings. Typically this is the mapping that is preferred, so it is worth being able to specify it concisely.

Correspondences: A correspondence relates an item in the base to an item in the target. Items can be entities, expressions, or functors. Correspondences have a structural evaluation score[3]. The relation `(correspondence-between ?c ?b ?t)` indicates that item `?b` corresponds to item `?t` according to correspondence `?c`. `(has-correspondence ?m ?c)` indicates that mapping `?m` contains the correspondence `?c`. (Not all correspondences participate in mappings, and only do so under specific constraints: See [6,7] for details.)

The *parallel connectivity* constraint of structure-mapping states that if a correspondence involving two expressions is in a mapping then so must correspondences involving its aligned arguments [19]. This constraint induces a structural dependency relationship between correspondences which is reified in our ontology. The relation `(structurally-supported-by ?c1 ?c2)` indicates that correspondence `?c2` links a pair of arguments in the statements linked by correspondence `?c1`. This information is used in constructing dependency networks during first-principles reasoning, so that explanations (and backtracking) will take account of analogical processing results. For example, correspondences implicated in a contradiction can be identified by a TMS, and used to formulate constraints (described next) which push SME into seeking an alternate solution.

Pragmatic constraints: Some task constraints can be expressed in terms of restrictions on the kinds of correspondences that are created during the matching process. Such statements can be inserted into the knowledge base or used in queries. The analogy ontology incorporates three kinds of constraints:

1. `(required-correspondence ?Bitem ?Titem)` indicates that its arguments must be placed in correspondence within any valid mapping. This is useful when the task itself implies correspondences whose consequences are to be explored (e.g., "If Saddam Hussein is Hitler, then who is Churchill?")

2. `(Excluded-correspondence ?Bitem ?Titem)` indicates that its arguments may not be placed in correspondence within any valid mapping. This is useful when a mapping has been ruled out due to task constraints, and alternate solutions are sought. Excluding an otherwise attractive correspondence will cause SME to look for other solutions.

3. `(required-target-correspondence ?Titem)` indicates that there must be a correspondence for target item `?Titem`, but does not specify what that must be. (`required-base-correspondence` is the equivalent in the other direction.) This is useful when seeking an analogy that sheds light on a particular individual or relationship, because being placed in correspondence with something is a necessary (but not sufficient) requirement for generating candidate inferences about it.

Candidate inferences: A candidate inference is a statement in the base that, based on the correspondences in a mapping, might be brought over into the target as a consequence of the analogy [7]. Candidate inferences have the following properties:

- *Propositional Content*. The statement about the target that has been imported from the base by substituting the correspondences from the mapping into the base statement.
- *Support*. The base statement it was derived from.
- *Support score*. The degree of structural support derived from the correspondences in the mapping.
- *Extrapolation score*. The degree of novelty of the candidate inference, derived from the fraction of it that is not in the mapping.

The support and extrapolation scores are defined in [13]. The binary relationships `candidate-inference-content`, `candidate-inference-support`, `candidate-inference-support-score`, and `candidate-inference-extrapolation-score` express the relationships between a candidate inference and these properties. Candidate inferences can postulate new entities in the target, due to the existence of unmapped entities in the support statement. Such terms are denoted using the function `analogy-skolem`, whose argument is the base entity that led to the suggestion.

Similarities and differences: Summarizing similarities and differences is an important part of many analogy tasks [21]. The main source of information about similarity is of course the set of correspondences that comprise a mapping. However, it is often useful to extract a subset of similarities and differences relevant to a particular correspondence of interest. `(similarities ?mh ?m ?exact ?dim ?other)` states that the structural support for correspondence `?mh` of mapping `?m` consists of three sets: `?exact`, representing exact matches differing only in `?mh`, `?dim`, correspondences where the entities involved vary along some dimension (e.g., different in color), and ?other, correspondences that are neither of the other two. (The set bound to ?other are candidates for rerepresentation, to improve the alignment.) Similarly, `(differences ?mh ?m ?1only ?2only)` indicates that with respect to correspondence `?mh` of mapping `?m`, `?1only` are statements

[3] The structural evaluation score of the mapping is the sum of the structural evaluation score for its correspondences. See [7] for details.

only holding in the base and `?2only` are statements only holding in the target.

Integrating 1st principles and analogical reasoning

The analogy ontology provides the glue between 1st principles reasoning and analogical processing. It provides a language for the entities and relationships of structure-mapping, so that they can be used along with other theories in logic-based reasoning systems. The intended semantics of the analogy ontology is that of structure-mapping. Just as predicates involving arithmetic typically have their semantics enforced via code, we use procedural attachment [22,38] to enforce the semantics of the analogy ontology. This is crucial because special-purpose software is needed for efficient large-scale analogical processing, where descriptions involving hundreds to thousands of relational propositions are matched. These procedural attachments provide the means for analogical processing software to be seamlessly used during first-principles reasoning. Matches and retrievals are carried out via queries, whose result bindings include entities such as matches, mappings, and correspondences, as defined above. Subsequent queries involving these entities, using other relationships from the analogy ontology, provide access to the results of analogical processing. Next we outline how this works.

Procedural attachment requires two-way communication between the reasoning system and the attached software. The reasoning system must have some means for recognizing predicates with procedural attachments and carrying out the appropriate procedures when queries involving them are made. For instance, in our FIRE reasoning system[4], *reasoning sources* provide a registration mechanism that ties procedures to particular signatures of predicate/argument combinations. For instance, `(match-between :known :known :unknown)` invokes SME, binding the result variable (third argument) to a term which can be used in subsequent queries to access the consequences of the match. The reasoning source maintains a table that translates between terms added to the reasoning system and the internal datastructures of the analogical processing system. Thus to the reasoning system, a standard unification-based query mechanism provides transparent access to analogical processing facilities.

Communication in the other direction, from analogy software to reasoning system, is handled by using the reasoner's query software from within the analogy subsystem. There are three kinds of information needed from 1st principles reasoning during an analogical query. First, SME needs basic information about the functors used in statements (e.g., are they relations, attributes, functions, or logical connectives?). This information is gathered by queries to the KB, e.g., in Cyc such questions reduce to membership in specific collections. Second, the terms denoting cases must be evaluated to gather the statements that comprise the case. This is implemented via a method dispatch, based on the case function (e.g., `case-fn`) using the techniques described in [30]. Third, the cases that comprise a case library must be collected. This is handled by querying the KB for the members of the case library.

Let us step through the process with a simple (albeit abstract) example, to see how these mechanisms work together. Consider a goal of the form `(match-between ?base ?target ?match)`. When `?base` and `?target` are bound, this goal invokes SME to match the value of `?base` against the value of `?target`. When `?base` is unbound, this goal invokes MAC/FAC, with the value of `?target` as the probe and the current case library as the memory. If `?target` is unbound, the query is considered to be an error, and the goal fails. The values of `?base` and `?target` determine how the case information fed to the software is derived. If the value is a non-atomic term whose functor is one of the case-defining functions, further reasoning is invoked to derive the contents of the case [30]. Otherwise, the value is construed as the name of a case and the appropriate facts are retrieved using queries involving `case-description`.

A successful `match-between` query results in the creation of a new term in the reasoning system to represent the match. `?match` is bound to this new term as part of the bindings produced by the goal, and the new term is linked the analogical processing data structure to support future queries. In addition, the mappings associated with the match are reified as new terms in the reasoning system, with the appropriate `mapping-of`, `best-mapping`, and structural evaluation score information asserted.

Analogical matches and retrievals involving large descriptions can result in hundreds to tens of thousands internal data structures, any of which is potentially relevant, depending on the task. Blind reification can easily bog down a reasoning session. Consequently, we use a *lazy reification* strategy. Because of the designs of SME and MAC/FAC, there can at most be only a handful of mappings created by any query[5], reifying mappings always makes sense. Correspondences and candidate inferences are reified on demand, in response to goals involving them (e.g., `correspondence-of`, `candidate-inference-of`).

We have implemented these ideas in two reasoning systems, DTE[6] and FIRE. DTE used ODBC-compliant databases to store its knowledge base, a logic-based TMS reasoning engine [10] as its working memory, a general-purpose mechanism for procedural attachments (including hooks for GIS systems and a diagrammatic reasoner) and a prolog-style query system which provides access to all of these facilities for application systems. FIRE is the

[4] FIRE = Integrated Reasoning Engine. The F is either Flexible or Fast, and we hope both.

[5] SME's greedy merge algorithm with its usual parameter settings only produces the top interpretation plus at most two more close interpretations. MAC/FAC's default settings allow it to return at most three candidates from the MAC stage, which means there can be at most nine mappings created in the FAC stage.

[6] DTE = Domain Theory Environment.

successor to DTE, using a higher-performance knowledge base created in collaboration with Xerox PARC, and with a streamlined mechanism for integrating other reasoning sources.

Examples

We have used the analogy ontology in a variety of systems. For concreteness, we briefly summarize how it is being used in three of them: A crisis management analogical reasoner, an analogical critiquer for military courses of action, and answering comparison questions in a knowledge capture system. We focus on showing how the analogy ontology enabled the smooth integration of analogical and first-principles reasoning in these systems.

Crisis management analogical reasoner. When reasoning about international crises, analysts commonly rely on analogy (cf. [24,32,33]). In the DARPA HPKB Crisis Management challenge problems, *parameterized questions* were introduced to express some typical types of queries [4]. Seeing how the analogy ontology enables such queries to be expressed and answered is a good illustration of its utility. We use two examples for illustration.

```
PQ226   Who/what is <SituationItem> in
{<EventSpec1> <ContextSpec1>, <SituationSpec1>}
similar to in {<EventSpec2> <ContextSpec2>,
<SituationSpec2>}?  How so, and how are they
different?
```

The terms in a parameterized question are specified according to the knowledge base used. Treating the first event as the base and the second as the target, this parameterized question is answered by using the following template:

```
(and (required-target-correspondence
       <SituationItem>)
     (recursive-match-between ?base ?target
                              ?match)
     (best-mapping ?match ?mapping)
     (has-correspondence ?mh ?mapping)
     (correspondence-between ?mh ?x
                      <SituationItem>)
   (similarities ?mh ?mapping ?exact
                 ?dimensional ?other)
   (differences ?mh ?mapping ?other
                ?obj1-only ?obj2-only))
```

For example, using the Cyc KB, the question

SQ226: Who/what is IRAN in Y2-SCENARIO-CONFLICT similar to in PERSIAN-GULF-WAR?

yields the answer

IRAN in The Facts concerning Y2-SCENARIO-CONFLICT case corresponds to IRAQ in the PERSIAN-GULF-WAR case.

where the supporting structural justifications include that both were involved in hostile actions, both were in conflict with other nation states in hostile-social-actions, and so on.

Our second example:

```
PQ228 How does the analogy between
{<SituationSpec1>, <PropositionConj1>
<ContextSpec1>} and {<SituationSpec2>,
<PropositionConj2> <ContextSpec2>} [where
<AnalogyMappingSpecConj>] suggest that the latter
will turn out, and how do differences between
them affect the answer?
```

Assuming that `?bstart` and `?tstart` are the starting points for the base and target, the initial query uses the template

```
(and (required-correspondence ?bstart ?tstart)
     (seeking-match-between
        (case-fn ?bstart)
        (no-postlude-case-fn ?tstart)
         ?match)
     (best-mapping ?match ?mapping)
     (has-correspondence ?mh ?mapping)
     (correspondence-between ?mh ?bstart ?tstart)
     (similarities ?mh ?mapping ?exact
                   ?dimensional ?other)
     (differences ?mh ?mapping ?other
                  ?obj1-only ?obj2-only))
```

The follow-up query finds what candidate inferences of the mapping make predictions about what might happen after `?tstart` based on its analogy with `?bstart`:

```
(and (candidate-inference-of ?ci ?mapping)
     (candidate-inference-content ?ci
     (<CausalConnective> ?tstart
             (analogy-skolem ?prediction)))
```

Notice the use of the `analogy-skolem` function, whose argument is bound to the base event being projected. The set *<CausalConnective>* varies with the particular structure of the knowledge base; for instance, there were 9 such relations in the Cycorp KB and 5 in the SAIC KB.

For example, using the SAIC KB, the question

TQF228b: How does the analogy between Iran oppose Saudi Arabia's influence in OPEC in the Y1 Scenario and Iran oppose Saudi Aramco influence in the Azerbaijan International Operating Company in the Y2 Scenario suggest that the latter will turn out, and how do differences between them affect the answer?

yields four predictions, based on Iran's responses to a relevant Saudi action in the base case: An attack on an embassy, injury of Saudi citizens in a terrorist attack, student protest marches, and demanding an explanation of the Saudi ambassador.

Our Crisis Management Analogical Reasoner was used by both teams in the HPKB program. For SAIC, we used an automatic translator to load their KB files into a DTE knowledge base. We formulated queries, and our system's explanatory hypertext was used as the team's answer on the questions we tackled. For Cycorp, we supplied them with a server version of our system, which could either interact with the Cyc IKB directly, or could be used with DTE assimilating the Cyc IKB knowledge base. The details of the evaluation process and results can be found elsewhere [34]; suffice it to say that we got a "B" which, given the difficulty of the problems, we view as a success.

Battlespace Analogical Critiquer: Military courses of action (COAs) are plans outlining how an operation is to

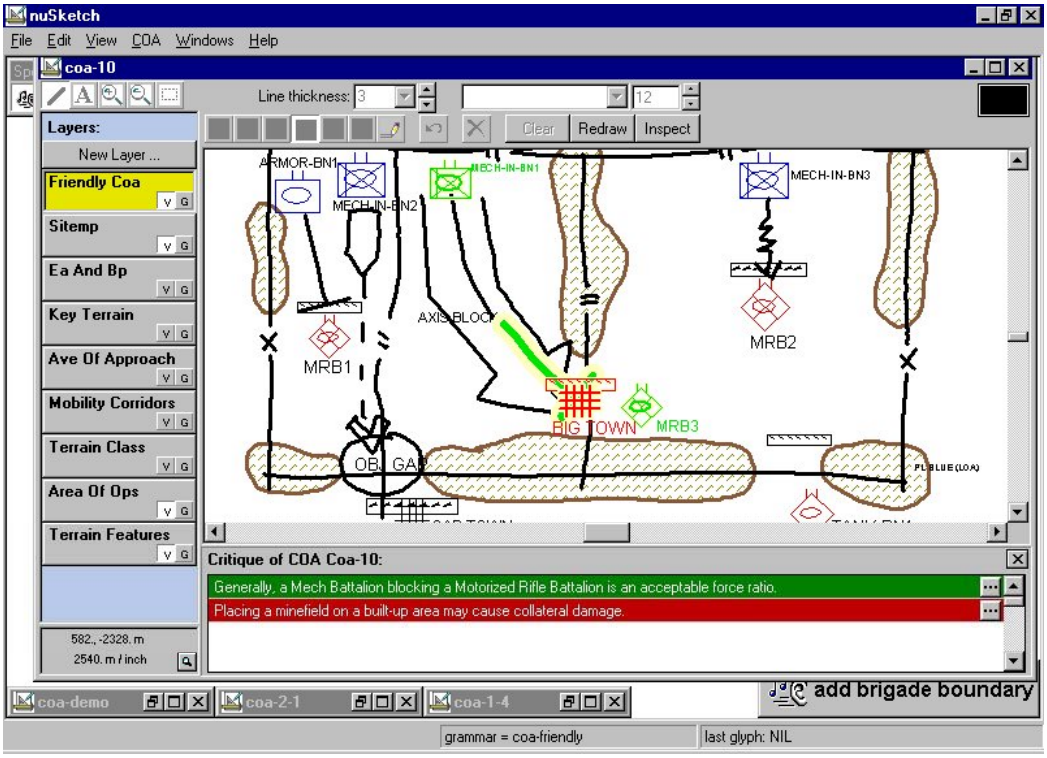

Figure 1: Analogical critiques of a course of action

be carried out. The early part of generating a COA concentrates on the *COA sketch*, a graphical depiction of the plan expressed through a graphical symbology representing military units, tasks and other relevant entities. The other challenge problem in HPKB was aimed at supporting military planners by critiquing COAs. Critiquing a sketch involves looking for violations of doctrine, ideas that might be unwise given prior experiences with the opponent, or just plain violate common sense, given that such plans are sometimes drafted in the field by people who haven't slept for days.

We created an *analogical critiquer* that provides feedback, both positive and negative, about a COA sketch based on *critique cases* authored by military personnel. The KB used is from Cyc, augmented with domain theories for spatial and geographic reasoning. Critique cases are generated by first using the same sketching tool (nuSketch COA Creator [8]) to generate a COA sketch, then using a special interface to select subsets of that COA to be encoded in cases. (A COA sketch can be used to create multiple cases, if different points can be illustrated by various aspects of it.) Each case must have a *critique point*, a proposition that is the key idea that that aspect of the case is making. Critique points are marked as positive or negative, with explanatory text added to phrase the idea in military terms. The author must also select (graphically) a subset of the sketch's propositions to be included in case. This *reminds-me* subset is what drives retrieval. A subset of that, in turn, is identified as the necessary conditions for the critique to be valid. The authoring tool installs a justification of the critique point based on the necessary antecedents. This justification will provide the candidate inference that raises the suggestion of the critique point on retrieval.

The critiquer can be invoked by a planner as desired when using the nuSketch COA Creator to create a COA sketch. Internally the critiquer uses the current sketch as a probe to retrieve critiques via the following query:

```
(and (match-between ?critique <current sketch>
                    ?match)
     (mapping-of ?mapping ?match)
     (candidate-inference-of ?ci ?mapping)
     (candidate-inference-content ?ci
       (causes–prop-prop ?antecedents
                         ?critique)))
```

This query retrieves all critiques that are potentially relevant to the current sketch, since the only `causes-prop-prop` statement in the case is inserted by the authoring environment. The final relevance check (not shown) evaluates `?antecedents`, ensuring that they do not contain any skolems (indicating a serious mismatch) and that they are true in the current sketch. The results are presented visually to the user, as shown in Figure 1.

While this critiquer was not completed in time to be used in the official HPKB evaluations, reviews of a preliminary version by military officers not involved in its creation have led to its continued development as a component in training systems and performance support systems to be used by the US military [35].

Comparison questions in knowledge capture

One of the bottlenecks in the creation of AI systems is the difficulty of creating large knowledge bases. Research in the DARPA Rapid Knowledge Formation program is tackling this problem by creating systems that can be "taught" interactively by domain experts, without AI experts or knowledge engineers in the loop. Two end-to-end systems have been constructed. Cycorp's KRAKEN system uses a combination of natural language and forms, controlled via a dialogue manager, to interact with experts. SRI's SHAKEN system [3] uses concept maps. Our group provides analogical processing services for both teams. So far this has focused on providing answers to comparison questions (e.g., "How is *X* similar to *Y*?" and "How are *X* and *Y* different?), which are used both by users to figure out what needs to be added and to assess the KB.

SHAKEN uses the KM representation system [3], which is frame-based, and it is written in Common Lisp. These factors, plus the fact that we were generating output for users directly, led to the decision to implement a special-purpose KM interface rather than using the analogy ontology directly in the implementation. On the other hand, for interfacing with KRAKEN we created an Analogy Server, using our FIRE reasoning engine and a simple KQML vocabulary of messages for controlling it. The Analogy Server's knowledge base was kept up to date by routines in KRAKEN, and called by it when users wanted to ask comparison questions.

While these systems are still very much in progress, in Summer 2001 there was an independent evaluation carried out where domain experts (biology graduate students) used these systems to create knowledge bases for a chapter of a biology textbook. The evaluation details can be found in [35]. For this paper, the key thing to note is that the analogy ontology successfully enabled KRAKEN to use analogical processing facilities, as intended.

Related Work

There have been a number of systems that capture some aspects of reasoning by analogy. Winston [39] describes a system that extracts rules from precedents, but was only tested with very small (10 or so propositions) examples. Special-purpose matchers and retrievers have been used for exploiting analogy in problem solving and planning (cf. [1,37]), but such systems lack the flexibility to deal with large knowledge bases and cover the range of phenomena that SME and MAC/FAC handle. Other cognitive simulations of analogical processing, including ACME and ARCS [31], LISA[25], IAM [26], have only been tested with small examples, and some are known to fail when tested with descriptions even one-tenth the size of what was needed to handle the problems described here. No previous analogy systems have been successfully used with multiple, large general-purpose knowledge bases created by other research groups.

While the majority of today's CBR systems have moved to feature-vector representations, there are a number of systems that still use relational information. Some examples include [2,17,28,29]. We believe that the ideas described here could be used in these systems, and that using psychologically realistic matchers, retrievers, and similarity metrics could add value to them.

Discussion

The analogy ontology provides a key advance in creating general-purpose reasoning systems that are closer to providing human-like flexibility. By providing formal representations for concepts used in analogical reasoning, first-principles reasoning systems can use analogical matching and retrieval as resources for tackling complex problems, and analogical processing systems can in turn use first-principles reasoning to extract knowledge from general knowledge bases to create cases and test the validity of candidate inferences. We have shown that this analogy ontology enables the creation of systems that simply were not possible before, systems that can tackle complex problems, involving cases that are literally an order of magnitude larger than found in the rest of the analogy literature (involving thousands of propositions per case), and interoperating with general-purpose, large-scale knowledge bases created by other research groups.

While the analogy ontology here relies on structure-mapping theory, it does not depend at all on the details of our current simulations of structure-mapping. Most cognitive simulations of analogy today are consistent with structure-mapping to the extent needed by this ontology [20], so if more capable matchers or retrievers were created, they could be used instead. There are a number of limitations of the ontology as it stands, the most significant being that it currently does not capture the ideas of *alignable* and *non-alignable* differences [21] appropriately. Formalizing these concepts has proven to be quite subtle, but we continue to work on it.

Acknowledgements

This research was supported by the DARPA High Performance Knowledge Bases and Rapid Knowledge Formation programs, and by the Artificial Intelligence program of the Office of Naval Research.

References

1. Blythe, J. and Veloso, M. (1997) Analogical replay for efficient conditional planning, *Proceedings of AAAI-97*, pages 668-673.

2. Branting, K. L. 1999. *Reasoning with Rules and Precedents – A Computational Model of Legal Analysis.* Kluwer.

3. P. Clark, J. Thompson, K. Barker, B. Porter, V. Chaudhri, A. Rodriguez, J. Thomere, S. Mishra, Y. Gil, P. Hayes, T. Reichherzer. Knowledge Entry as the Graphical

Assembly of Components. *First International Conference on Knowledge Capture*, October 21-23, 2001.
4. Cohen, P., Schrag, R., Jones, E., Pease, A., Lin, A., Starr, B., Gunning, D., and Burke, M. 1998. The DARPA High Performance Knowledge Bases Project. *AI Magazine*, Winter, 1998.
5. Cohn, A. (1996) Calculi for Qualitative Spatial Reasoning. In Artificial Intelligence and Symbolic Mathematical Computation, LNCS 1138, eds: J Calmet, J A Campbell, J Pfalzgraf, Springer Verlag, 124-143, 1996.
6. Falkenhainer, B., Forbus, K., and Gentner, D. (1986, August) The Structure-Mapping Engine. Proceedings of AAAI-86, Philadelphia, PA
7. Falkenhainer, B., Forbus, K., Gentner, D. (1989) The Structure-Mapping Engine: Algorithm and examples. Artificial Intelligence, 41, pp 1-63.
8. Ferguson, R., Rasch, R., Turmel, B., and Forbus, K. 2000. Qualitative spatial interpretation of course-of-action diagrams. *Proceedings of QR-2000.* Morelia, Mexico.
9. Forbus, K. 2001. Exploring analogy in the large. In Gentner, D., Holyoak, K. and Kokinov, B. (Eds) *Analogy: Perspectives from Cognitive Science*. Cambridge, MA: MIT Press.
10. Forbus, K. and de Kleer, J., *Building Problem Solvers*, MIT Press, 1993.
11. Forbus, K., Ferguson, R. and Gentner, D. (1994) Incremental structure-mapping. *Proceedings of the Cognitive Science Society,* August.
12. Forbus, K., & Gentner, D. (1997). Qualitative mental models: Simulations or memories? *Proceedings of the Eleventh International Workshop on Qualitative Reasoning*, Cortona, Italy.
13. Forbus, K., Gentner, D., Everett, J. and Wu, M. 1997. Towards a computational model of evaluating and using analogical inferences. *Proceedings of CogSci97.*
14. Forbus, K., Gentner, D. and Law, K. (1995) MAC/FAC: A model of Similarity-based Retrieval. *Cognitive Science*, 19(2), April-June, pp 141-205.
15. Forbus, K. and Usher, J. 2002. Sketching for knowledge capture: A progress report. *Proceedings of IUI-2002*, ACM Publications, January, San Francisco.
16. Forbus, K.D., Whalley, P., Everett, J., Ureel, L., Brokowski, M., Baher, J. and Kuehne, S. (1999) CyclePad: An articulate virtual laboratory for engineering thermodynamics. *Artificial Intelligence*. **114**, 297-347.
17. Friedrich Gebhardt, Angi Voß, Wolfgang Gräther, Barbara Schmidt-Belz. 1997. *Reasoning with complex cases.* Kluwer, Boston, 250 pages. March.
18. Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. Cognitive Science, 7, 155-170.
19. Gentner, D. (1989). The mechanisms of analogical learning. In S. Vosniadou & A. Ortony (Eds.), *Similarity and analogical reasoning* (pp. 199-241). London: Cambridge University Press. (Reprinted in *Knowledge acquisition and learning*, 1993, 673-694.)
20. Gentner, D., & Holyoak, K. J. (1997). Reasoning and learning by analogy: Introduction. *American Psychologist, 52,* 32-34.
21. Gentner, D., & Markman, A. B. (1997). Structure mapping in analogy and similarity. *American Psychologist, 52,* 45-56. (To be reprinted in *Mind readings: Introductory selections on cognitive science,* by P. Thagard, Ed., MIT Press)
22. Greiner, R. and Lenat, D. 1980. A representation language language. *Proceedings of AAAI-80.*
23. Gross, M. and Do, E. (1995) Drawing Analogies - Supporting Creative Architectural Design with Visual References. in *3d International Conference on Computational Models of Creative Design*, M-L Maher and J. Gero (eds), Sydney: University of Sydney, 37-58.
24. Heuer, R. J. 1999. *Psychology of Intelligence Analysis*. Center for the Study of Intelligence. Government Printing Office, US Government.
25. Hummel, J. E., & Holyoak, K. J. (1997). LISA: A computational model of analogical inference and schema induction. *Psychological Review.*
26. Keane, M. T. (1990). Incremental analogising: Theory & model. In K. J. Gilhooly, M. T. G. Keane, R. H. Logie, & G. Erdos (Eds.), *Lines of thinking* (Vol. 1, pp. XX). Chichester, England: Wiley.
27. Kolodner, J. L. (1994). Case-based reasoning. San Mateo, CA: Morgan Kaufmann Publishers.
28. Leake, D. (Ed.) 1996. *Case-Based Reasoning: Experiences, Lessons, and Future Directions*, MIT Press.
29. Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D., Wess, S. (Eds.), *Case Based Reasoning Technology – from Foundations to Applications.* Lecture Notes in Artificial Intelligence 1400, Springer, 1998
30. Mostek, T., Forbus, K. and Meverden, C. 2000. Dynamic case creation and expansion for analogical reasoning. *Proceedings of AAAI-2000*. Austin, Texas.
31. Thagard, P., Holyoak, K. J., Nelson, G., & Gochfeld, D. (1990). Analog retrieval by constraint satisfaction. Artificial Intelligence, 46, 259-310.
32. Neustad, R. and May, E. 1988. *Thinking in time: The uses of History for Decision Makers*. Free Press.
33. IET, Inc. and PSR Corp. 1999. HPKB Year 2 Crisis Management End-to-end Challenge Problem Specification. http://www.iet.com/Projects/HPKB/Y2/Y2-CM-CP.doc
34. http://www.iet.com/Projects/HPKB/
35. Rasch, R., Kott, A. and Forbus, K. 2002. AI on the battlefield: An experimental exploration. *Proceedings of IAAI 2002*.
36. Schrag, Robert. http://www.iet.com/Projects/RKF/
37. VanLehn, K., & Jones, R. M. (1993). Integration of analogical search control and explanation-based learning of correctness. In S. Minton (Ed.), Machine learning methods for planning (pp. 273-315). San Mateo, CA: Morgan Kaufman.
38. Weyhrauch, R. 1978. Prolegomena to a theory of formal reasoning. Stanford CS Department CS-TR-78-687, December, 1978
39. Winston, P. 1982. Learning New Principles from Precedents and Exercises," *Artificial* Intelligence **19**, 321-350