Mapping Self-Similar Structure: Commutative Expressions in Structure Mapping

Ronald W. Ferguson (rwf@cc.gatech.edu)
College of Computing, Georgia Institute of Technology
Atlanta, GA 30332

Abstract

Comparisons over commutative expressions constitute a special problem in analogical mapping. The unique character of these comparisons stems from the nature of commutative expressions themselves, which encapsulate particular dimensions of similarity between their arguments. In this paper, we describe commutative matching mechanisms for two analogical mapping systems (SME and MAGI), and demonstrate how use of these mechanisms can allow analogical mapping to scale over descriptions with lots of self-similar relational structure. We also discuss the cognitive implications of these mechanisms.

Introduction

There are currently several models of analogy and similarity that utilize some form of structural mapping (Falkenhainer, Forbus, & Gentner, 1989; Ferguson, 1994; Forbus, Ferguson, & Gentner, 1994; French, 1995; Gentner, 1983; Holyoak & Thagard, 1989; Hummel & Holyoak, 1997; Keane & Brayshaw, 1988; Kokinov & Petrov, 2001; Mitchell, 1993; Veale & Keane, 1997). These systems have been used to model a wide variety of phenomena, including analogy, similarity, metaphor, and symmetry detection. In all these systems, comparison is understood as a process that looks for similar systems of relational structure.

One type of relational structure, however, has only received limited attention in the study of analogy: matches of commutative relational expressions (hereafter called *commutative matches*). This is unfortunate, because commutative matches appear to be an interesting and unusual problem in analogical mapping systems. A commutative expression, such as *equal-length*(line1,line2) or *group*(circle1, circle2, circle3), by its very definition represents a kind of encapsulated self-similarity, since all its arguments are interchangeable and thus alike.

Consequently, commutative expressions represent both an opportunity and a problem for structure mapping systems. If the encapsulated similarity can be harnessed effectively in the mapping process, it adds to the power of the model. Alternatively, that encapsulated similarity can also, as we shall see, severely hamper a structure mapping model if not handled correctly. Either result also has implications for human analogical reasoning.

In this paper, we describe the commutative matching mechanisms built for two analogical mapping models: the MAGI symmetry detection model (Ferguson, in preparation) and the Structure Mapping Engine (SME; Forbus, Ferguson & Gentner, 1994, in preparation).

Why commutative expressions make structure mapping difficult

The difficulty of commutative matches is a byproduct of how structure mapping constraints are enforced by SME and other structure mapping models. SME creates a mapping as a set of alignments between expressions and entities in two descriptions (the *base* and *target*). The constraints of structure mapping are as follows:

Identicality: Only expressions with identical predicates, or functional expressions that share a parent match, may be matched to one another.

One-to-one: Each base item must match only one target item, and vice-versa.

Parallel connectivity: Each matched expression must also map all its arguments.

Systematicity: Mappings should encourage deeper, interconnected systems of matches over shallow, less connected sets of matches.

To enforce these constraints, SME constructs mappings in a local-to-global process. SME begins by building local *match hypotheses* (MHs) between potentially matching expressions. These MHs are then gathered into rooted trees of matched expressions called *kernel mappings*. Kernel mappings are in turn used to construct global mappings.

The "anatomy" of mapped expressions in SME is demonstrated in Figure 1. SME builds an MH between each pair of expressions or entities that meet the identicality constraint (such as the two *above* relations and their paired arguments). These MHs reflect the structure of the expressions that they match, so that sets of MHs tend to form rooted match trees (kernel mappings).

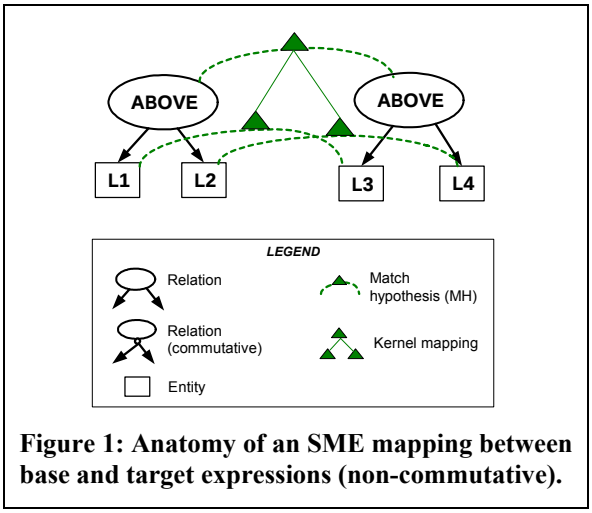


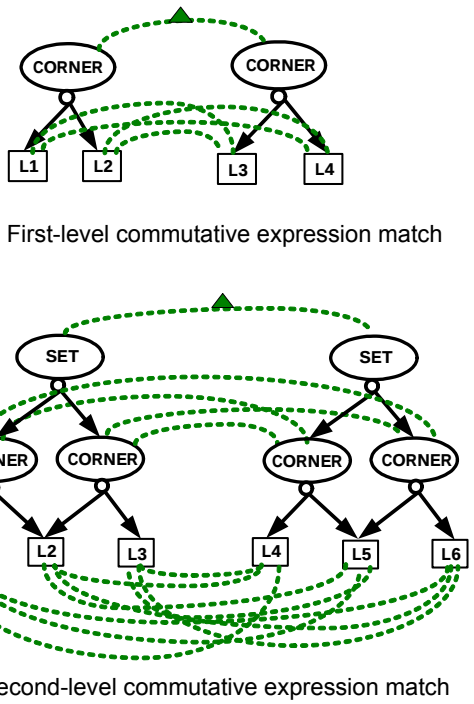
Figure 1: Anatomy of an SME mapping between base and target expressions (non-commutative).

Figure 1 shows a match between non-commutative relational expressions. SME's representation language, however, defines both non-commutative and commutative relations. [1]

It is at the kernel mapping stage that commutative matches introduce complications into the mapping process. Kernels that map non-commutative expressions are guaranteed, once created, to be structurally consistent (i.e., to meet the one-to-one, identicality, and parallel connectivity constraints). Sets of consistent kernels are then gathered via a greedy merge process (Forbus & Oblinger, 1990) to maximize the systematicity of the resulting global mapping.

This system works well for sets of non-commutative expressions. However, for commutative matches, the situation is more complex, because kernel mappings that contain commutative matches cannot guarantee structural consistency, since they must also maintain the commutativity of the matched expressions themselves, which in turn may violate the one-to-one constraint.

We can see the problem clearly in Figure 2, which depicts commutative matches in SME. In contrast to Figure 1 where the parent match is structurally consistent with all the child matches, in Figure 2-A's commutative match, not all the child matches (shown as dotted lines) are structurally consistent with one another. One MH, for example, matches L1 to L3, while another matches L1 to L2. Together these matches violate the one-to-one constraint.



(A) First-level commutative expression match

(B) Second-level commutative expression match

Figure 2: Why commutative matches in structure mapping are problematic.

This problem is exacerbated if there are multiple levels of commutative expressions—a common occurrence in visual representations such as the corner groups represented in Figure 2-B. In this case, in which the grouped *corner* expressions are matched, entity correspondences cannot be determined without choosing the particular *corner* expressions to match, which in turn are indeterminate because the *set* expressions are also commutative.

This problem is actually a special case of the much larger problem of mapping self-similar relational structure, which we will return to in the discussion.

Here we describe the commutative matching mechanism developed for the SME 3 (Forbus et al., 1994) and MAGI (Ferguson, in preparation) mapping systems. As a contrast, we also include a brief description of the mechanism used in SME 2 (Falkenhainer et al., 1989), which is different and has not previously been described in depth (this description is based instead on the SME 2 source code).

Commutative matches in SME 2

The original Structure-Mapping Engine (Falkenhainer et al., 1989), or SME 2, handles commutative matches in three stages: 1) generating a set of child matches; 2) generating the argument match permutation sets; and 3) cloning the commutative MH for each match permutation set.

In stage 1, SME creates match hypotheses for all possible pairings of the arguments sets. So, when two commutative "parent" expressions with N arguments are matched, the match hypothesis for them creates N^2 child match hypotheses.

In stage 2, SME generates the argument match permutation sets. This is designed to solve the *child selection problem*: exactly which N of the N^2 child matches should be accepted in the global mapping? To handle this, it generates all consistent sets of N child matches from the set of N^2 matches. For commutative matches with N arguments, there are N! possible permutation sets.

In the final stage, SME then "clones" the commutative match hypothesis. Instead of simply treating all the child matches as children of the commutative match, SME creates a number of "clones" of the original commutative match, each of which contains a single permutation of the commutative match's arguments.

Each clone is by definition structurally inconsistent with all the others (pairwise they violate the one-to-one constraint, since for each pair of permutations, there must be at least one base item that is mapped to different target items). For this reason, the clones will not interfere with each other in the construction of mappings—no more than one clone can be used in any particular kernel or global mapping.

While this mechanism has a very high computational cost, it is not without its advantages. While expensive, the

[1] A commutative predicate defined in SME assumes that all its arguments are interchangeable. Some frame-based representation languages (e.g., Kokinov & Petrov, 2001), allow a selected subset of the relation's arguments to be interchangeable.

time and memory costs depend solely on the number and arity of the commutative matches, which are commonly used, but are often a small subset of any given base or target. In addition, for many commutative matches, the identicality constraint limits the number of interpretations to a handful early in the process, reducing processing cost. Finally, once the cloned matches are built, they are structurally consistent, which means that SME can treat the cloned commutatives matches just like non-commutative matches, greatly simplifying later processing.

Commutative matches in SME 3

The handling of commutative expressions was significantly revised in SME 3 (Forbus et al., 1994), although the exact mechanism has not been previously published. SME 3 is also known as incremental or I-SME.

SME 3 takes a different approach than SME 2 to commutative matches. For commutative matches, SME 3 does not attempt to solve the problem by ensuring that all argument permutations are considered. Instead, it handles permutation sets only when forced to, using *lazy resolution* of commutative matches.

To avoid creating explicit match permutations, SME 3 uses a *commutatives table* to encode them implicitly (Figure 3). This method, which is similar to lazy evaluation methods in associative-commutative unification (Lincoln & Christian, 1989), creates a matrix of matching base and target items (Figure 3-A). Each table cell represents a potential MH. By the one-to-one constraint, only one item in each row and column can be true. To make this table notation work better with our mapping notation, we tilt the table clockwise into a diamond shape (B), and use "+" and "-" to indicate valid and invalid MHs, respectively.

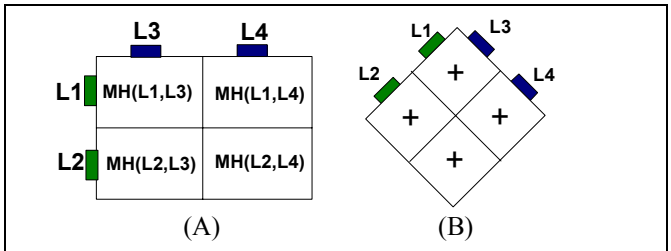


Figure 3: A commutatives table represents all potential child matches for a single commutative match. (A) shows the full table. (B) is a simplified version, rotated clockwise to aid clarity.

We can now easily demonstrate the utility of SME 3's mechanism. For example, consider the match between the commutative expressions *corner*(L1,L2) and *corner*(L3,L4) depicted in Figure 4. Here, the match table takes the place of the multiple match hypotheses shown in Figure 2-A. The table itself is connected to the parent match by a *possible-children link* (indicated by the double line).

As soon as one match is chosen or eliminated, the commutative match is fully constrained and can be integrated into the rest of the match. Suppose that MH(L1,L3) (i.e., a match between L1 and L3) is invalid. Then MH(L1,L4) and MH(L2,L3) are true, and we can create a consistent and complete match tree for the figure (Figure 5). Similarly, for a second-order commutative match, we can determine the set of correspondences as particular possible matches are eliminated (Figure 6).

The use of lazy resolution of commutative matches has some interesting implications. Suppose that the arguments do not fully resolve and are left partially or fully ambiguous. In SME 2, all permutations would be considered, and each might be included in a mapping. In SME 3, however, the commutative match is purposely left *unresolved*, with the assumption that it represents a many-to-many match where the particular individual correspondences are not only unknown, but also unimportant.

It may seem strange that SME 3 simply keeps the commutative matches unresolved, but this result appears to be a closer fit to human comparisons than SME 2's model.

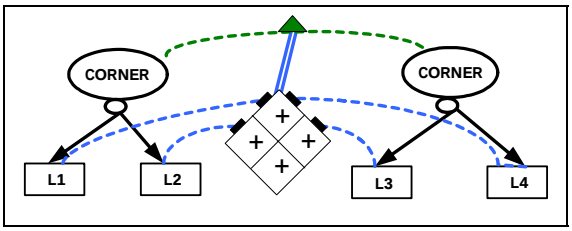


Figure 4: A commutative match utilizing a commutativity table

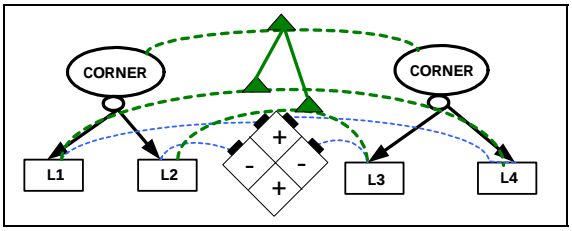


Figure 5: Resolved binary commutative match

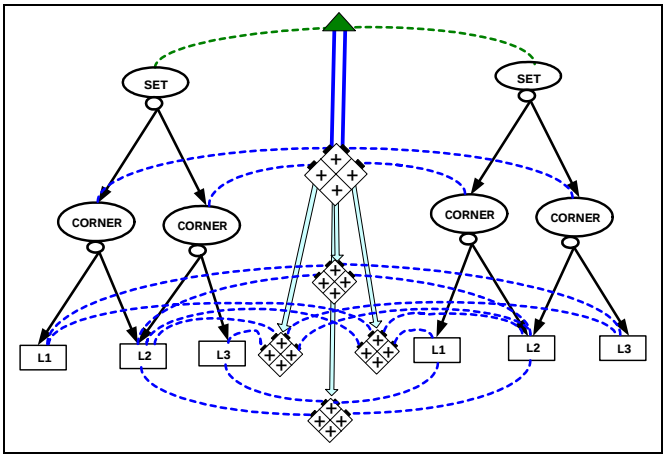


Figure 6: Mapping of a second-order data structure using commutative tables

Take, for example, an analogy that compares a basketball team with a baseball team. All five basketball players correspond to all nine baseball players, and without further constraining matches, which baseball player corresponds to which basketball player is unimportant. If additional information is added, two players may be placed into a specific match—for example, if a basketball guard and the baseball pitcher had both suffered recent injuries. But in general, a person making this comparison would simply not consider enforcing a one-to-one match.

Along with being more cognitively plausible, we can demonstrate how this commutative match mechanism makes analogical mapping more tractable. In addition, our example shows how the commutative mechanism implicitly forms a group-based many-to-many mapping.

Here, we use a modified version of the oft-used analogy between an atom and a solar system (Falkenhainer et al., 1989). The original versions of these descriptions (Figure 7) have only one electron and one planet. Using these descriptions as the beginning template, we constructed two description sets. Each set contained instances with 1 to 10 electrons (for the atom descriptions) or 1 to 10 planets (for the solar system descriptions). In the *repeated structure* set, each planet or electron is represented separately by repeating the description for each repeating element. In the *commutative expression* set, the electrons and planets were represented by a single commutative *and* relation.

We ran each set of atom/solar-system comparisons using SME 3 on a 2.6 GHz Windows 2000 Workstation running Franz Allegro Common Lisp, recording for each run the processing time, the number of MHs produced, and the memory used.

The result was a clear victory for commutative expressions. As Figure 8 (processing time) and Figure 9 (total MHs created) show, the use of repeating structure scales poorly in SME. This is not unexpected—again, this kind of self-similar structure is a problem for all structural matchers, not just SME. In contrast, the performance of the set using commutative expressions is relatively flat for less

Solar-System:

```
(cause (gravity (mass sun) (mass planet1))
       (attracts sun planet1))
(greater (temperature sun)
         (temperatureplanet1))
(cause
   (and (greater (mass sun) (mass planet1))
        (attracts sun planet1))
   (revolve-around planet1 sun))
```

Rutherford-Atom:

```
(cause
  (opposite-sign (charge nucleus)
                 (charge electron))
  (attracts nucleus electron))
  (cause
     (and (greater (mass nucleus)
                   (mass electron))
          (attracts nucleus electron))
       (revolve-around electron nucleus))
```

Figure 7: Relational descriptions used for the simulation.

than 8 items, and in general takes a fraction of the computational resources of the mappings using repeated structure. Similar results were found for memory use. These results show that, for large amounts of repeating relational structure, commutative expressions make mapping tractable, allowing it to scale in a way that repeated structure does not.

In addition, the repeating structure mappings attempted to match each electron and each planet specifically, while the code using the commutative expressions matched all the relational structure found in the repeating structure condition, but did not resolve the mapping of the set of planets to the set of electrons. In other words, while retaining a structural match of the whole system of relations, it also saw the planets and electrons as a many-to-many group match, where there was no reason to match any particular electron to any particular planet. Again, we argue that this is a cognitively plausible result.

This example, then, shows the utility and cognitive plausibility of the mechanism.

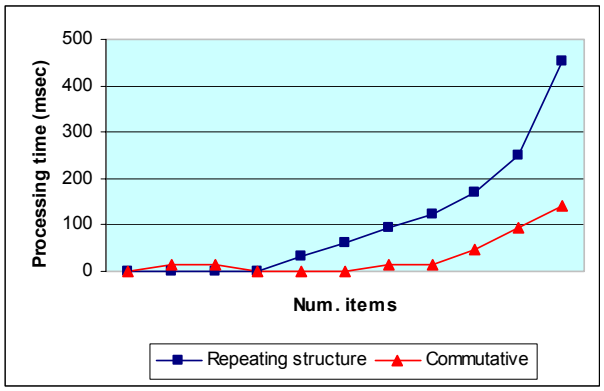


Figure 8: Processing time (in msec) for comparisons with repeating structure vs. comparison utilizing commutative expressions.

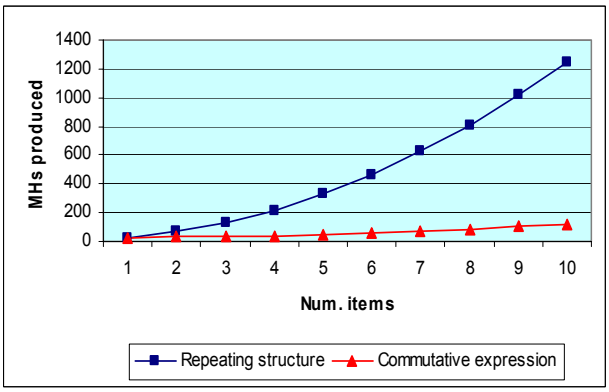


Figure 9: Number of match hypotheses produced for comparisons with repeating structure vs. comparisons utilizing commutative expressions.

Commutative matching in MAGI

Commutative expressions are even more difficult to handle in systems that, instead of mapping base and target descriptions, find self-similarity within a single description.

MAGI is a system that models symmetry and repetition detection as structure mapping (Ferguson, 1994, in preparation). MAGI has been used in a variety of domains, including diagrams illustrating physical laws (Ferguson & Forbus, 1998), logic circuits diagrams (Ferguson, 1994), and children's fables (Ferguson, 2001). It has also been used to model perceptual symmetry phenomena, including effects of qualitative visual structure (Ferguson, Aminoff, & Gentner, 1996) and the preference for symmetry about the vertical axis (Ferguson, 2000).

MAGI performs symmetry mapping by mapping a description back onto itself to find the maximal area of self-similarity. It also utilizes two additional mapping constraints:

Limited self-matching: An expression is not allowed to map to itself unless that self-match is the result of two non-identical parent matches.

Maximal differentiation: Symmetry mappings should maximize the interconnectivity and size of structure on either side of a symmetry mapping, and minimize the amount of relational structure that crosses across the two mapped portions.

If commutative mapping in SME is difficult, commutative mapping in MAGI is doubly hard. We must not only control for matches between commutative expressions, but also between a commutative expression and *itself*. MAGI allows a commutative expression to match itself, but only with the proviso that its arguments be permuted to avoid self-matching of its arguments. The limited self-match constraint is designed to block the largest, and yet also most uninformative possible mapping: a complete self-match, where every item maps to itself.

However, a simple extension of SME's mechanism suffices to handle these conditions. Figure 10 shows two corner relations (assumed now to be in the same description rather than in two descriptions). The corners can map to one another, as before, but in a twist that only occurs in a symmetry mapping, they may also map to themselves.

In this case, the commutatives for MAGI have to ensure that the limited self-match constraint is enforced—that an expression not match to itself unless its parent or child matches are not self-matches.

Therefore, to enforce the proviso that the commutative matches not completely self-match, binary commutatives (such as the *corner* self-matches in Figure 10) are constrained so that their arguments map to one another.

The result is that the mapping actually uses *three* commutative tables. The center table matches the corners to each other, and at the time depicted, the commutative match is unresolved. In addition, each corner relation has a commutative table handling the match to itself. In each of those tables, however, the limited self-match over a binary predicate has automatically resolved them so that the self-match permutes its arguments.

The careful reader will notice that because of this, commutative binary self-matches will always resolve. Given the frequency of binary commutative predicates in visual representations (e.g., c*orner, parallel, perpendicular*, *intersects*, and *beside*) this greatly increases the efficiency of MAGI's mapping algorithm.

We note that the current implementation enforces the proviso only in the case of binary commutatives. Higher arities have thus far proved difficult and inefficient, and are an area of future research. The proviso is difficult to enforce because a local non-self match can have a global effect—if one commutative somewhere in a kernel mapping does not map an argument to itself, the entire kernel meets the limited self-matching constraint. However, the frequency of binary commutatives in visual domains still makes the existing mechanism extremely useful.

Discussion

It is interesting to note that the solution of commutative expression matches has often been overlooked in cognitive models of analogical mapping. Most existing systems ignore the problem entirely (Hofstadter & Mitchell, 1992; Holyoak & Thagard, 1989; Hummel & Holyoak, 1997; Keane & Brayshaw, 1988; Veale & Keane, 1997), although the Copycat and Tabletop systems explicitly address grouping (Mitchell, 1993; French, 1995), and ACME addresses

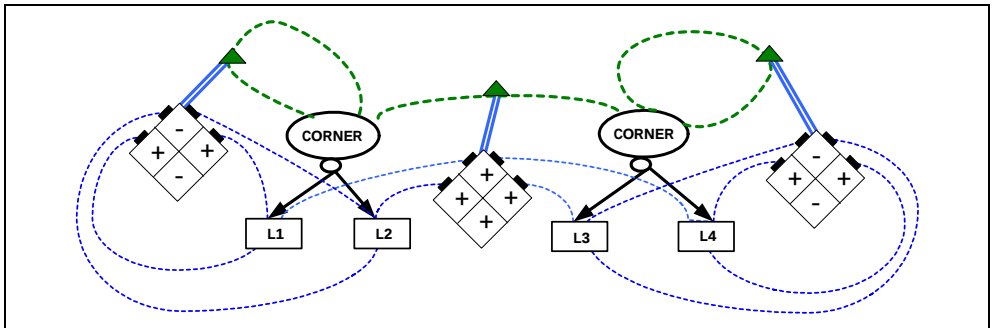


Figure 10: Example of commutative mapping in MAGI. Corners can map to each other commutatives, but may also map to themselves with the proviso that their arguments do not self-match.

many-to-many matches (Holyoak & Thagard, 1989).

This is true even though descriptions in the analogical mapping literature often involve domains that contain some form of symmetry or repetition. These domains include solar systems (Gentner, 1983), multiple radiation beams or lasers (Holyoak & Thagard, 1989), arches (which are often symmetric) (Winston, 1980), and series of turbines and batteries (Bhatta & Goel, 1997).

The problem of mapping commutative expressions is but one instance of a more general problem in analogical reasoning, that of mapping descriptions that themselves are self-similar. Descriptions of everyday events and objects often contain inner regularities and symmetries.

While some researchers have usefully noted the NP-hard nature of structure-mapping (Veale & Keane, 1997), an assessment of the effect of repeating relational structure may prove to be a more pragmatic result in the long run. Simply put, the complexity of structure mapping increases significantly as repeating structure is added, but the proper use of commutative expressions (and perhaps other grouping techniques as well!) can address that limitation.

Acknowledgements

Special thanks to Ken Forbus, with whom the SME commutatives algorithm was developed. Thanks also to Dedre Gentner and the Qualitative Reasoning and Analogy Groups at Northwestern University for many productive discussions on this topic, and to the anonymous reviewers. This research was supported by the Cognitive Science and Computer Science programs of the Office of Naval Research, and by the National Science Foundation under the Learning and Intelligent Systems program

References

Bhatta, S., & Goel, A. (1997). A functional theory of design patterns, *Proc. International Joint Conference on Artificial Intelligence (IJCAI-97)* (pp. 294-300). Nagoya, Japan.

Falkenhainer, B., Forbus, K. D., & Gentner, D. (1989). The Structure-Mapping Engine: Algorithm and examples. *Artificial Intelligence, 41*, 1-63.

Ferguson, R. W. (1994). MAGI: Analogy-based encoding using symmetry and regularity. In A. Ram & K. Eiselt (Eds.), *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society* (pp. 283-288). Atlanta, GA: Lawrence Erlbaum Associates.

Ferguson, R. W. (2000). Modeling orientation effects in symmetry detection: The role of visual structure, *Proceedings of the 22nd Conference of the Cognitive Science Society* (pp. 143). Hillsdale, NJ: Erlbaum.

Ferguson, R. W. (2001). *Symmetry: An Analysis of Cognitive and Diagrammatic Characteristics.* Unpubl. Ph.D., Northwestern University, Evanston, IL.

Ferguson, R. W. (in preparation). MAGI: A model of symmetry and repetition detection.

Ferguson, R. W., Aminoff, A., & Gentner, D. (1996). Modeling qualitative differences in symmetry judgments, *Proceedings of the Eighteenth Annual Conference of the Cognitive Science Society* (pp. 12). Hillsdale, NJ: Lawrence Erlbaum Associates.

Ferguson, R. W., & Forbus, K. D. (1998). Telling juxtapositions: Using repetition and alignable difference in diagram understanding. In K. Holyoak & D. Gentner & B. Kokinov (Eds.), *Advances in Analogy Research* (pp. 109-117). Sofia, Bulgaria: New Bulgarian University.

Forbus, K. D., Ferguson, R. W., & Gentner, D. (1994). Incremental structure mapping. In A. Ram & K. Eiselt (Eds.), *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society* (pp. 313-318). Atlanta, GA: Lawrence Erlbaum Associates.

Forbus, K. D., & Oblinger, D. (1990). *Making SME greedy and pragmatic.* Paper presented at the Conference of the Cognitive Science Society, Cambridge, MA.

French, R. M. (1995). *The Subtlety of Sameness: A Theory and Computer Model of Analogy-Making*. Cambridge, Massachusetts: Lawrence Erlbaum Associates.

Gentner, D. (1983). Structure-Mapping: A theoretical framework for analogy. *Cognitive Science, 7*, 155-170.

Hofstadter, D. L., & Mitchell, M. (1992). An overview of the Copycat project. In K. J. Holyoak & J. Barnden (Eds.), *Connectionist Approaches to Analogy, Metaphor, and Case-Based Reasoning*. Norwood, NJ: Ablex.

Holyoak, K. J., & Thagard, P. (1989). Analogical mapping by constraint satisfaction. *Cognitive Science, 13*, 295-355.

Hummel, J. E., & Holyoak, K. J. (1997). Distributed representations of structure: A theory of analogical access and mapping. *Psychological Review, 104*(3), 427-466.

Keane, M., & Brayshaw, M. (1988). The Incremental Analogy Machine: A computational model of analogy. In D. Sleeman (Ed.), *Proceedings of the Third European Working Session on Learning* (pp. 53-62). London: Pitman.

Kokinov, B. N., & Petrov, A. A. (2001). Integrating memory and reasoning in analogy-making: The AMBR model. In D. Gentner & K. J. Holyoak & B. N. Kokinov (Eds.), *The Analogical Mind: Perspectives from Cognitive Science* (pp. 59-124). Cambridge, MA: The MIT Press.

Lincoln, P., & Christian, J. (1989). Adventures in associative-commutative unification. *Journal of Symbolic Computation, 8*, 217--240.

Mitchell, M. (1993). *Analogy-Making as Perception*. Cambridge, MA: MIT Press.

Veale, T., & Keane, M. T. (1997). The competence of sub-optimal theories of structure mapping on hard analogies. IJCAI.

Winston, P. H. (1980). Learning and reasoning by analogy. *Communications of the ACM, 23*(12).