Forbus, K. D., Ferguson, R. W., Lovett, A., & Gentner, D. (in press). Extending SME to handle large-scale cognitive modeling. To appear in *Cognitive Science*.

Extending SME to Handle Large-Scale Cognitive Modeling

Kenneth D. Forbus
EECS Department, Northwestern University, 2133 Sheridan Road, Evanston, IL, 60208

Ronald W. Ferguson
Leidos, 4001 North Fairfax Drive, Arlington, VA, 22203

Andrew Lovett
Dedre Gentner
Psychology Department, Northwestern University, 2029 Sheridan Road, Evanston, IL, 60208

Abstract: Analogy and similarity are central phenomena in human cognition, involved in processes ranging from visual perception to conceptual change. To capture this centrality requires that a model of comparison must be able to integrate with other processes and handle the size and complexity of the representations required by the tasks being modeled. This paper describes extensions to SME since its inception in 1986 that have increased its scope of operation. We first review the basic SME algorithm, describe psychological evidence for SME as a process model, and summarize its role in simulating similarity-based retrieval and generalization. Then we describe five techniques now incorporated in the Structure-Mapping Engine (SME) that have enabled it to tackle realistic-scale modeling tasks: (1) *Greedy merging* rapidly constructs one or more best interpretations of a match in polynomial time ($O(n^2 log(n))$); (2) *Incremental operation* enables mappings to be extended as new information is retrieved or derived about the base or target, to model situations where information in a task is updated over time; (3) *Ubiquitous predicates* model the varying degrees to which items may suggest alignment; (4) *Structural evaluation* of analogical inferences models aspects of plausibility judgments; (5) *Match filters* enable realistic-scale task models to communicate constraints to SME to influence the mapping process. We illustrate via examples from published studies how these enable it to capture a broader range of psychological phenomena than before.

Contact author: Ken Forbus, forbus@northwestern.edu

Contents

LIST OF FIGURES

LIST OF TABLES

1 Introduction

Psychological studies of analogy and similarity suggest that there are core processes of comparison and analogical inference that enter into tasks ranging from visual perception to conceptual change (Gentner, Holyoak & Kokinov, 2001; Hofstadter & Sander 2013; Holyoak & Thagard, 1995). These comparison processes can be characterized by the principles of structure-mapping theory (Forbus, Gentner & Law, 1995; Gentner, 1983; Gentner & Markman, 1997). To be sure, some aspects of analogical processing seem to vary across different tasks: comparison in visual perception operates at a faster time scale than reasoning through an analogy in a debate, for example. However, many fundamental properties (for example, 1:1 mappings and systematicity) are preserved over a surprisingly broad range of cognitive processes (Gentner, 2003, 2010; Gentner & Markman, 1997; Krawczyk et al., 2004, 2005).

SME, the Structure-mapping Engine (Falkenhainer, Forbus & Gentner, 1986, 1989), embodies a process model of how structure-mapping takes place. A number of other computational models (including IAM (Keane & Brayshaw, 1988), ACME (Holyoak & Thagard 1989), COPYCAT (Mitchell, 1993), TABLETOP (French, 1995), LISA (Hummel & Holyoak, 1997), DRAMA (Eliasmith & Thagard, 2001), AMBR (Kokinov & Petrov, 2001), CAB (Larkey & Love, 2003), and DORA (Doumas et al., 2008)) also draw on structure-mapping theory, but use different processing algorithms. We briefly review some of these later in this paper. (For a more complete review of current simulations of analogical comparison, see Gentner & Forbus, 2011.)

SME's basic algorithm has held up quite well since its first publication in 1986. Nonetheless, it has evolved considerably. These extensions allow for what we term *realistic scale* analogical processing—processing that operates in concert with other cognitive processes in the kinds of complex tasks that occur in everyday reasoning. There has been considerable success using SME to model comparison processes in isolation (Gentner et al. 1993; Gentner et al. 2009; Loewenstein & Gentner, 2005; Markman & Gentner, 1993b). However, this work, along with most other work on analogical modeling, suffers

from three limitations: (1) restriction to small analogies; (2) the use of hand-coded representations; and (3) failure to integrate analogical processes with other parts of cognition. Our term 'realistic scale' is meant to convey an approach to simulation that addresses these three points.

Our goal is to make good on the claim that analogical comparison is a core process that cuts across different domains and tasks. We aim to capture the way analogy is used in tasks such as visual reasoning, understanding and solving textbook problems, and moral decision-making. These kinds of tasks have been largely neglected in the modeling literature. In part this is simply because modeling realistic-scale phenomena is hard—both because of the complexity of creating such software and because of the difficulty of gathering the necessary psychological data to guide the design and evaluate the results. But another reason is that most analogical matchers, including our own first-generation version of SME (Falkenhainer et al., 1986), cannot scale up to these challenges (Eliasmith & Thagard, 2001; Larkey & Love, 2003).

This paper describes a set of techniques we have incorporated into SME over the last two decades to improve its capacities and meet the challenge of simulating realistic-scale psychological phenomena involving comparison. The techniques are:

1. *Greedy merging* enables SME to rapidly construct up to three near-optimal global interpretations, guaranteeing polynomial-time operation.
2. *Incremental matching* enables SME to model tasks for which information is not all available at once.
3. *Ubiquitous predicates* enable SME to model the varying degrees to which items may suggest alignment.
4. *Structural evaluation of candidate inferences* enables SME to model judgments of the plausibility and interestingness of an analogical inference.

5. *Match filters* enable task models to communicate constraints to SME to influence the mapping process.

Section 2 briefly reviews the principles of structure-mapping and provides a high-level overview of how SME works to provide the necessary context for what follows. Section 3 discusses the issue of scale in analogical processing, and summarizes five investigations that provide evidence for the necessity of considering larger descriptions in analogical processing than most simulations have used. Section 4 discusses the techniques above in detail, showing how they work and why we believe they are psychologically plausible. Section 5 shows that the theoretical worst-case complexity of SME is $O(n^2 log(n))$, where n is the number of items in the base or target[1]. We also show, via an empirical complexity analysis over a large number (> 5,800) of examples from simulation studies, that the theoretical bound is a gross over-estimate of resource requirements in realistic situations, although it does capture growth in resource usage appropriately. Section 6 discusses related work not already brought up in earlier sections, and Section 7 discusses some broader issues. The current SME algorithm and the complexity analysis are described in Appendix A.

2 A brief summary of Structure-Mapping Theory and SME

Structure-mapping theory (Gentner, 1983, 1989, 2010; Gentner & Markman, 1997) postulates that analogy and similarity operate via the same structural alignment process, operating over structured representations. That is, the descriptions being compared are symbolic, including entities, attributes of those entities, and relationships between entities and other relationships.

[1] Big-O notation is standard in computer science as a way of describing how some property of a computation grows as a function of the size of its inputs, i.e. O(n) time means that as the number of input items, here, n, doubles, the run time of the computation will double, up to some multiplicative constant.

Formally, the elements of SME's representations are objects (or entities), object attributes (one-place predicates), relations (predicates that take two or more arguments), and functions. As predicates, attributes and relations express assertions with potential truth values, such as HOT(sun) or REVOLVE-AROUND(earth, sun)[2]. In contrast, functions map from a set of arguments onto another argument—typically, a dimension. Functions are chiefly used to express dimensional information, such as DIAMETER(earth)=12742 Km. Psychologically, functions capture the phenomenon that people fluently map relational structures across dimensions—e.g., space→time, as in "Exams are coming" and heat→anger, as in "It was a heated conversation." This pattern is captured by allowing functions to match non-identically within a mapping. By using functions to set up correspondences between dimensions—such as space→time—we can capture the fact that cross-domain mappings are often maintained consistently within a discourse, as in "Exams are coming"; "Yes, and then the holidays arrive" (Boroditsky, 2000; Gentner et al., 2001; Lakoff & Johnson, 1980; Thibodeau & Durgin, 2011). A final distinction is the *order* of a relation. The order of a relation is 1 plus the order of its highest-order argument. First-order relations are relations between objects. Higher-order relations are relations between other statements. Examples of higher-order relations include logical connectives (e.g. IMPLIES), causal relationships, and modal operators (e.g. BELIEVES).

The same structural alignment process is used whenever a comparison is to be done, whether it is an analogy or similarity. Moreover, the same process is used when contrasting two things, as discussed below. The results of a comparison can be classified based on the kind of overlap that the process finds. A match is considered to be *literal similarity* if both attributes and relations align, *analogy* if relations

[22] For this paper, we use infix notation when discussing examples, and use Lisp-style notation when displaying representations used in working systems.

align but attributes do not, *surface* similar if attributes align but relations do not, and *anomaly* if neither align. These are graded concepts, of course, since matches are rarely perfect.

According to structure-mapping theory, structural alignment 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* between items (i.e. entities and expressions) in the base and items in the target.
- A set of *candidate inferences*—surmises about the target made on the basis of common structure plus the base representation. Reverse candidate inferences can also be computed, from target to base, and these can give rise to *alignable differences* (Lovett et al., 2009; Markman & Gentner, 1996). These inferences can include *analogy skolems*, which represent a projected entity conjectured to exist in the other description. For example, in the historical heat/water analogy, a new entity, *caloric*, was conjectured to exist as one of the consequences of the analogy. Analogy skolems are defined as functions of the entity in the originating description, and can be read as "something like" the original entity.
- A *structural evaluation score* indicating the overall quality of the match. This is a purely structural score; other considerations, such as the relevance of the inferences, are computed outside the mapping engine.

Mappings are governed by the following constraints:

- *Structural consistency*: Structural consistency is defined by two constraints. The first, the *1:1 constraint*, requires that each item in the base maps to at most one item in the target and vice-versa. The second, the *parallel connectivity constraint*, requires that if a correspondence between two statements is included in a mapping, then so must correspondences between their arguments.

- *Tiered identicality*: Identical matches between predicates (relations and attributes) and functions are preferred. By default, relations must match identically, but non-identical functions can be aligned if such alignments would support a larger overlapping structure. A classic analogy from the history of science, for example, aligns *Pressure* with *Temperature*. Many conventional metaphors involve aligning nonidentical functions, such as height to intelligence ("Her I.Q is through the roof") or space to time ("Winter is behind us").

  Depending on task demands, the identicality constraint can be relaxed further to allow non-identical relations to correspond, if they are suggested by a larger structure and satisfy additional criteria. The most commonly used additional criterion is *minimal ascension* (Falkenhainer, 1987), whereby non-identical predicates are required to share a close superordinate.

- *Systematicity constraint*: Preference is given to mappings that align systems of relations in the base and target, especially including those involving nested expressions—that is, those involving higher-order relations.

Each of these theoretical constraints is motivated by the role analogy plays in cognitive processing. The 1:1 and parallel connectivity constraints ensure that the candidate inferences of a mapping are well defined. Tiered identicality is a strong semantic constraint, avoiding structurally isomorphic but nonsensical mappings. The systematicity constraint reflects a (tacit) preference for coherence and inferential power in analogical reasoning.

There is now widespread agreement on the importance of structural consistency constraints in analogical reasoning (cf. Eliasmith & Thagard, 2001, Kokinov & French, 2003; Hummel & Holyoak, 1997, Krawczyk et al. 2004, 2005; Larkey & Love, 2003). However, some computational models of analogy do not utilize systematicity, and there is no consensus on how central other constraints might be (e.g.,

pragmatics (Holyoak, 1985)) and on how these constraints should be expressed computationally (see Gentner and Forbus (2011) for a review).

Our own simulation, the Structure-Mapping Engine (SME) (Falkenhainer, Forbus, and Gentner, 1986, 1989) works roughly as follows: Given base and target descriptions, SME finds globally consistent interpretations via a local-to-global match process, which can be divided into three phases (see Fig. 1):

Phase One: Constructing the match hypothesis network. SME begins by proposing local correspondences, called *match hypotheses.* All possible local identity matches between expressions in the two representations are made in parallel[3], regardless of whether they are mutually consistent. Additional matches are made via local parallel connectivity—for example, if two relations are matched, then SME attempts to match their arguments. No attempt is made at this stage to enforce global consistency. Thus the initial network is inchoate, providing the material for potential mappings.

Phase Two: Parallel construction of structurally consistent kernels. At this point, SME starts building mappings by extracting structurally consistent sets from the forest of match hypotheses. To do so, SME does two things, again in parallel.

1. It marks as inconsistent those match hypotheses that violate parallel connectivity.
2. It marks as mutually inconsistent pairs of match hypotheses that would violate the 1:1 constraint.

It then coalesces the local matches into a set of structurally consistent connected structures, called *kernels.* Kernels are the grist from which mappings are created.

[3] In the current implementation of SME, this process is serial (but very fast). However, we posit that it may proceed in parallel in the brain.

Each kernel receives a structural evaluation. SME begins the structural evaluation process by first assigning a local score to each match hypothesis, then using a trickle-down process to propagate evidence downwards from a match hypothesis to match hypotheses between the arguments of the corresponding statements. This provides a local means of implementing the systematicity preference, since match hypotheses that participate in a large matching structure will receive a higher score.

Phase Three: Constructing mappings. SME uses a greedy merge algorithm to combine kernels into one or more global mappings. The basic idea is to start with the largest and deepest kernel (that is, the one with the highest structural evaluation) and serially add others that are structurally consistent with it. This results in one or a few large, structurally consistent *global mappings* between the representations. The global mapping reveals common structure between the two representations. At this stage, candidate inferences may be projected from one representation to the other and alignable differences emerge.

SME computes only a handful of mappings, based on the settings of two parameters. The *Max Limit* is an upper bound on the number of mappings produced, and defaults to three. The *Score Cutoff* is a drop in score below which further mappings are ignored, and defaults to 0.8, i.e. any mappings produced must be within 20% of the best mapping to be output. These parameters help model capacity limits in analogical mapping. For each mapping, SME computes the candidate inferences for the mapping and its structural evaluation score.

For concreteness, consider the simple example in Fig. 2, based on a commonly used analogy in physics instruction. It describes a cross-domain analogy between two systems that oscillate. The base is a classic spring-block oscillator. The attributes `spring`, `block`, and `system` describe the types of entities involved, while relationships such as `made-of` and `restoring-force` describe some of the basic physical properties of the system. There are two causal statements that describe relationships between

continuous parameters of the system, which are drawn from Qualitative Process theory (Forbus, 1984). The qprop+ statement says that the frequency of oscillation increases if the spring constant is increased, all else being equal. The qprop- statement says that the frequency of oscillation decreases if the mass of the block is increased, all else being equal. Finally, the restoring force is specified as the cause of the system's oscillation, using the cause relationship. The target is a pendulum. It, too, has a restoring force, and like the base, has the frequency of oscillation identified as an important aspect of the system. There is one causal relationship specified about frequency, namely that if the length of the string is longer, the frequency will decrease (i.e. the qprop- statement). Fig. 3 also shows a graphical visualization of these representations, with the height of each predicate indicating what order it is. Fig. 4 shows the match hypothesis forest that SME generates for this example, and Fig. 5 shows the kernels. The mapping SME constructs is shown in Fig. 6.

While abstract, this description of SME's operation is enough to ground the discussion of the advances described in Section 4. These advances interact – for instance, greedy merge implies the need to sometimes use automatically derived filters in the matching process – but to a first approximation they can be described independently. Appendix A describes the new SME algorithm in detail, illustrating how these techniques work together and analyzing its overall complexity. Complexity is a critical question: A central operation in cognitive processes must work in polynomial time, in order to account for the rapidity and scalability of human processing. We return to this issue in Section 5. The rest of this section summarizes some psychological evidence for SME, and outlines how SME is used as a component in models for analogical retrieval and category formation, to illustrate the explanatory power of this model.

SME's processing account has led to several predictions that have been borne out in psychology experiments. Table 1 shows a set of benchmark phenomena that characterize analogical mapping

(Gentner & Markman, 1995; Markman & Gentner, 2000). For example, there is considerable evidence that people prefer structural consistency, including 1-1 mappings, in analogical processing (Gentner & Markman, 2006; Krawczyk et al., 2004, 2005; Markman & Gentner, 1993b; Spellman & Holyoak, 1996). There is also evidence for systematicity: people prefer analogies that share deep systematic structure over matches that are otherwise identical, but that lack common higher-order relations linking the lower-order relations (Gentner, Rattermann & Forbus, 1993). Systematicity also influences analogical inferences: people draw inferences from the more systematic to the less systematic of two analogs (Bowdle & Gentner, 1997) by projecting predicates connected to the common structure (Clement & Gentner, 1991).

There is also evidence supporting SME's local-to-global matching algorithm—specifically, that the first stage of comparison processing is an initial symmetric alignment process. Wolff and Gentner (2011) tested comprehension of strongly directional metaphors, such as "Some suburbs are parasites." In untimed tasks, people strongly prefer "Some suburbs are parasites" to "Some parasites are suburbs." However, if participants had to answer under deadline pressure (within 600 ms), they did not exhibit a directional preference: they found both versions equally comprehensible (Importantly, the mean 'comprehensible' response was significantly greater than for scrambled metaphors, indicating that meaningful processing had been initiated.) By 1200 ms, there was a significant advantage for the forward direction. This is exactly what follows from an initially symmetric alignment process, assuming the deadline occurs during the first two phases of processing. During this early alignment stage, we conjecture that people can have the sense that something promising is happening, even in cases where they will ultimately reject the comparison.

Further psychological evidence for SME's processing algorithm comes from investigations of difference processing. SME predicts an empirical disassociation in the response times for two seemingly related

tasks: the same-different task, and a 'name the difference' task. A long-established finding is that in same-different tasks, people are faster to respond "different" for very dissimilar pairs than for similar (but nonidentical) pairs (Goldstone & Medin, 1994; Luce 1986; Posner & Mitchell, 1967). SME can capture this finding, as described below. But SME also makes a novel prediction: that if asked to state a difference between two things, people should be faster to do so for very similar pairs (Sagi et al 2012). This prediction rests on two prior findings. First, there is abundant psychological evidence that when people are asked to state differences between two things, they are likely to name *alignable differences*—differences that play the same role in the common structure[4] (Gentner & Gunn, 2001; Gentner & Markman, 1994; Kurtz & Gentner, 2013; Markman & Gentner, 1993a, 1996). By their nature, these differences emerge only after structural alignment is complete. Second, high-similarity pairs are faster to align than low-similarity pairs (Gentner & Kurtz, 2006). This also follows from SME's process. In high-similarity pairs, since most of the matches are compatible, there will be one or two large, dominant kernels. This means that the greedy merge process generally only needs to run once, since SME does not bother producing more than one mapping when there is little left over. For low-similarity pairs, there are typically many small kernels, and the final step may require comparing two or more different merges. Thus alignment takes longer for low-similarity than for high-similarity pairs (all else being equal). Thus, naming a difference should be faster for high-similarity pairs than for low-similarity pairs.

Now consider a same-different task. If two descriptions are completely different (i.e., the pair is very dissimilar), the size of the initial match hypothesis forest will be small compared to the size of the items. This means that the alignment process can be terminated at the first stage, resulting in an early

[4] If the pair cannot be aligned, people will give nonalignable differences, typically by stating a fact about one item and denying it for the other. For example, for shopping mall/traffic light, responses included "A traffic light tells you when to go, a shopping mall doesn't" and "You can go inside a shopping mall, you can't go inside a traffic light." We suggest that nonalignable differences (unlike alignable differences) do not naturally spring to mind in the course of comparison.

'different' response. But if the pair is highly (or even moderately) similar, such that the initial stage feels promising (i.e. the match hypothesis forest is large), then the alignment process cannot be aborted at Phase 1 and must be carried to the end. SME thus predicts opposite patterns for these two tasks: faster responding for very similar pairs in a name-a-difference task, and faster responding for very dissimilar pairs in a same-different task. Sagi, Gentner & Lovett (2012) found exactly this pattern. To our knowledge, SME is the only simulation of similarity or analogy that can predict this pattern.

SME has several attractive features as a cognitive model. As noted above, it operates in a parallel to serial manner[5], and can capture some essential properties of the comparison process, including abstraction, inference projection and alignable difference detection. It also matches findings on the ordinal time course of human comparison processes, as discussed above—such as the disassociation in the time course of difference processing (Sagi et al., 2012) and the fact that that high-similarity matches are processed faster than low-similarity matches (Gentner & Kurtz, 2006). Most importantly, SME operates without advance knowledge of the point of the comparison—capturing the fact that people can happen upon a comparison with no advance idea of what it will yield, and discover a hitherto unnoticed common structure or a new inference.

2.1 The Evolution of SME

SME has evolved considerably since 1986. For example, in the initial version of SME (Falkenhainer, Forbus & Gentner, 1989) we used different rule sets to process different match types—analogy, literal similarity, and mere-appearance (surface similarity). In terms of cognitive modeling, this amounts to the assumption that people process analogy with a different set (attending only to relations) than they use for literal similarity (in which attention goes to both relations and object attributes). But this has the disadvantage of having to postulate that people know in advance what kind of match they will be

[5] The current implementation is serial, but parallel issues are also discussed in Appendix A.

getting—violating a prime goal of capturing spontaneous discovery. Moreover, such rule sets turn out to not be necessary (Forbus et al., 1994). SME now runs in what used to be called literal similarity mode, in which it tries to match all types of predicates—attributes, functions and relations. Depending on what it finds to match, the comparison will be characterized as literal similarity, analogy, surface match, or anomaly, or as something intermediate.

We further note that SME can be used with cases from a variety of different sources; it can process cases whether they were originally presented perceptually, given in text, retrieved from long term episodic memory, produced dynamically from semantic memory (Mostek et al., 2000), or derived through reasoning and problem solving (as the examples in Section 3 illustrate). In larger models, it tends to be used in a map/analyze cycle (Falkenhainer, 1990), in which the results of a comparison are evaluated in a task-dependent way. The models in Section 3 illustrate these ideas in the context of larger-scale cognitive tasks.

A key principle in this work is that the basic similarity engine—SME—should be able to operate by default on its own, without external guidance. There are two reasons for this. First, comparison often occurs spontaneously. Although we sometimes compare things on command—for example, when we're told that two things are analogous—it's clear that the comparison process often occurs unbidden. For example, walking through a city, we generally aren't looking for twins, but if twins should appear, we will notice their similarity. This kind of "comparison-based interrupt" also happens at the conceptual level; noticing that two ideas are similar can be a source of insight. Indeed, there is evidence that comparison-based inference can happen even without our noticing it (Day & Gentner, 2007). A second argument for a relatively independent similarity engine is that comparison is ubiquitous in cognition, in arenas from decision-making to causal reasoning to categorization to perceptual learning. Although it's

theoretically possible that different comparison processes are called on in each arena[6], the many phenomenological similarities across areas suggest that the same basic comparison process is involved across a wide swath of cognition. This argues for a relatively modular similarity engine that can be used as a subprocess within more complex cognitive processes.

2.2 SME as a component in retrieval and generalization

Another piece of evidence that something like SME may operate as a general-purpose similarity operation is that it can be productively viewed as a sub-process in analogical retrieval and generalization. We illustrate by briefly summarizing how SME is used in MAC/FAC, a model of analogical retrieval, and SEQL (and its newer incarnation, SAGE), a model of analogical generalization.

MAC/FAC (Forbus et al., 1995) models similarity-based retrieval—the phenomenon by which a currently active representation reminds us of some prior similar situation. We model similarity-based retrieval as a fast, relatively indiscriminate process. This choice is motivated in part by information-level considerations (in Marr's (1983) sense): this process must by its nature operate over large swaths of LTM, suggesting that it should be computationally cheap to carry out. A second rationale is the empirical fact that similarity-based retrieval is a hit-or-miss phenomenon—the retrieved instances are sometimes relevant to the current situation, but often they simply share some surface features. Yet, once the instance is retrieved, people can often show considerable discernment, rejecting their own retrievals if they lack structural similarity (Gentner et al., 1993). MAC/FAC uses a two-stage retrieval process to capture these phenomena. The first stage (MAC) uses a vector representation automatically generated from the structured representations in working memory and in long-term memory as a crude, fast search. These content vectors consist of the relative frequency of occurrence of the predicates and functions in the structured representation. Psychologically, this captures the fact that memory retrieval

[6] This view is not unanimous: for example, Lee and Holyoak (2008) argue that causal analogies require a different mode of processing than other analogies.

is strongly influenced by content, and only weakly influenced by relational structure (Gentner et al., 1993; Holyoak & Koh, 1987; Trench & Minervino, 2014). It also captures the idea that deliberate indexing is not necessary in order for retrieval to occur. Computationally, these vectors have the property that their dot product provides an estimate for the size of the match hypothesis forest that SME would produce for the corresponding structural descriptions. We assume that the dot products are happening in parallel, and that the top few (up to three) structured representations corresponding to the winning vectors are passed on to the second stage. The second stage (FAC) runs SME in parallel, comparing these structured representations to the current situation, returning one or more mappings as the result. Studies comparing MAC/FAC's retrieval patterns with those of humans have shown good ordinal match (Gentner, Rattermann & Forbus, 1993; Gentner et. al, 2009).

SME is also used as a component process in models of analogical generalization. SEQL (Kuehne et al. 2000) used SME to compare incoming examples and assimilate them into ongoing generalizations. A later version, SAGE (McLure et al. 2010), keeps track of frequency information about alignable structures, enabling it to produce probabilistic generalizations. For example, if given a series of examples with the same category label, SAGE begins by storing the first input example. When the next example arrives, SAGE compares it to the first one, using SME. If there is sufficient overlap (that is, if SME's score is above a pre-set threshold) the common structure is stored as a generalization. If the similarity to the abstraction is below threshold, the example will be stored separately. This process continues as new examples arrive; if new examples are sufficiently similar to the ongoing generalization, they are assimilated into it and the generalization is updated. New examples that can't be assimilated into the main abstraction are compared to the set of examples; if a new example is very similar to a stored example, a new generalization is formed from their common structure. For example, suppose SAGE is given a set of items labeled 'birds.' If it receives the series *sparrow, thrush, starling, finch*, it will form an abstraction that we would recognize as *songbird*. If the next example of 'bird'—say, a *stork*—is

insufficiently similar to the abstraction, then it will be stored as a separate example. This process allows SAGE to form similarity-based subclusters; for example, if a *heron* and *egret* were added to the *bird* category, SAGE will form an abstraction of tall, long-legged birds subsuming *stork, heron* and *egret*.

The goal of using SME in models of realistic-scale cognitive processes brings up the crucial issue of scale, to which we turn next.

3 Representation and Scale in Analogical Processing: Evidence from Cognitive Models

One point of strong agreement among analogy researchers is the importance of representation in modeling analogy—in particular, the need for explicit representations of relations. Flat feature vectors, even if very large, have no way to capture the phenomena of relational matching and mapping (Gentner & Markman, 1993, 2006; Goldstone et al., 1991; Holyoak & Hummel, 2000; Markman, 1999; Sagi et al., 2012). In the early days of analogical research, this need was met by using hand-generated examples. Indeed, many prominent models of analogical processing have been tested only with hand-generated examples (e.g., ACME, LISA, DRAMA, CAB, DORA). SME has also been tested with such examples, especially in the early years (e.g., Gentner et al., 1993). While considerable insight can be gained from experiments using such materials, they raise significant methodological concerns. The most serious is *tailorability*—i.e., the degree to which a model's results depend on representation choices that are not theoretically constrained, and chosen by the modeler (often unknowingly) to make the simulation come out in the way that they desire. Tailorability is difficult to avoid in hand-coded representations, because many questions about the exact formats of mental representations are not yet strongly constrained by data. For example, even though there is agreement that relations are important in human cognition, exactly *which* relations people use to represent a given situation is still an open question. When hand-coded representations must be used, tailorability can be reduced by using independent evidence as to

the likely representations when it is available, and by using uniform attested representational conventions otherwise.

However, by far the best way to reduce tailorability is to use representations that are independently generated--either automatically generated (e.g., Falkenhainer 1990), and/or produced outside one's laboratory (Forbus et al 1995). In the last 20 years, although we have sometimes used hand-coded representations (e.g., Gentner et al., 2009), we have placed a high priority on testing SME with independently generated representations Automatically generated representations can be taken from other AI systems that are carrying out some task. In larger-scale systems that have SME as a component, SME's representations can be produced by another part of the system itself, through interpreting natural language text (e.g., Dehghani et al., 2008b) or sketches (e.g., Lovett et al., 2009). This has the advantage that in cognitive simulation, the materials given to the system are the same stimuli given to human subjects (Lovett et al., 2009; Sagi et al., 2012). Below we describe tests of SME using automatically generated representations, including representations derived from perceptual input (See Section 3.1.1 and 3.1.2), representations taken from other AI systems, including a natural language system (Sections 3.1.3 and 2.1.5) and representations provided by another institution (Section 3.1.4).

A further methodological concern is the adequacy of the representation—that is, whether it actually contains enough information to carry out the task(s) it is intended for. When carefully evaluated, a running program demonstrates that there is at least one set of representations and processes that can be used to carry out the task—that is, it shows sufficiency (though not necessity). Not all AI systems are intended as cognitive simulations, of course. However, even systems that are not designed as simulations can still provide evidence as to what information content, and how much of it, is required for particular tasks—that is, they can bear on Marr's (1983) information processing level of explanation. Another criterion for an adequate representation (beyond being able to support simulations that carry

out the task being modeled) is that it can be used in other tasks that could reasonably be supposed to draw on the same representation. For any single process, the representations can be chosen so as to create the desired outcome. Thus we agree with Cassimatis et al (2008) who argue that ability and breadth are crucial, but underutilized, criteria for evaluating cognitive simulations.

A third methodological point is the *Integration Constraint* (Forbus, 2001). This constraint arises from the claim that analogical mapping processes underlie many important cognitive processes. To make good on this claim, we need to test SME as a component of other realistic-scale cognitive processes, such as categorization or moral reasoning. The Integration Constraint states that a model of a cognitive process P should be usable as a component in models of larger-scale cognitive processes that are hypothesized to use P (Forbus, 2001). Embedding a model within a larger system also reduces tailorability; if the representations are automatically constructed by other processes, and the results of analogical comparison are used by yet other processes, there are many more constraints on the representations than would be found in isolation.

Thus, the properties of tasks impose constraints on models of component processes. In the case of analogical mapping, one implication of this principle is that a model of analogical mapping must be able to handle the kinds of representations that arise in the tasks in which people use analogy. Let us therefore look at representations used by larger-scale cognitive models that have used SME as a subprocess.

3.1 Evidence from Five Computational Investigations

Here we briefly summarize five previously published computational investigations that provide evidence about the kind of descriptions and number of relationships that models of analogy must handle. In each

case, SME was used as a component in a larger system. Data were collected by recording the contents of SME during the running of the larger model and saving this information to files[7].

3.1.1 Geometric Analogies

The first simulation of analogy was Evans' (1968) ANALOGY program. Fig. 7 shows two example problems from Evans' original corpus. These are problems of the form "A is to B as C is to…?" Running on an IBM mainframe, using punch cards as input, ANALOGY was able to automatically construct representations for half of the examples it operated over, a tour de force for that era. The program used a transformation-based model, with separate domain-specific comparison processes to compare stimuli and to compute transformations between them. However, despite multiple efforts to do so, until recently no model was built that could automatically encode these stimuli and successfully solve the same range of problems.

Lovett and his collaborators (Lovett et al. 2009) showed that, using automatically constructed inputs, structure-mapping could be used to perform this task. The figures for the problems were drawn using PowerPoint, and copy/pasted into CogSketch (Forbus et al. 2011), an open-domain sketch understanding system. CogSketch automatically produces structured, relational representations from digital ink, and these were used as inputs to the system. CogSketch computes a variety of qualitative spatial relationships, including positional relationships (e.g. `above`, `leftOf`), topological relationships (e.g. `partiallyOverlapping`, `inside`) and relative sizes. Recognizing resizing and rotation of shapes was automatically performed by using a model of mental rotation, which in turn uses SME. That is, SME was used to perform a qualitative comparison between two shapes, using an automatically-constructed edge level representation, from which quantitative scaling and rotation relations, if

[7] This kind of record (referred to as a "dehydrated SME file") contains all of definitions of the predicate vocabulary, base, target, and SME correspondences and mappings needed to "reconstitute" the original results. This corpus of matches will be made publicly available upon publication of this paper.

appropriate, could be derived. Thus even the original encoding of the stimuli involves the use of SME over automatically constructed representations.

The original model (Lovett et al. 2009) used a two-stage comparison process. That is, SME was used to compare A to B, and then SME was used to compare C to each possible solution in turn. SME's mappings were then compared to each other, again using SME (i.e. a *second-order comparison*). This involved automatically reifying mappings and candidate inferences as assertions, for input to the second order comparisons. This process also identifies reversals and dimensional changes (e.g. `leftOf` in the A/B pair, versus `above` in a C/X pair). Candidate inferences are computed in both directions, with the presence of analogy skolems indicating that an extra entity is in one of the descriptions. A score is computed for each answer, based on SME's structural evaluation score for the second order comparison, plus a penalty for extras, normalized to avoid size effects. The choice with the highest-scoring comparison with C is selected as the answer.

An interesting complication is that SME's initial mapping may not always be the best answer. An *executive* process is used to automatically look at mappings other than the best[8], if SME has produced more than one, and to consider whether to re-prioritize its interpretation of reflection versus rotation in comparing shapes. Its default strategy is to prefer identity matches, followed by reflections, and then rotations. But if a good solution is not found, it will also explore giving higher preference to reflections or rotations. Problems for which the default encoding strategy suffices to find an answer should be faster for people, since fewer comparisons are required, in contrast to problems which require backtracking and trying other strategies. As predicted, a laboratory study indicates that people do

[8] As discussed elsewhere, alternate mappings are only created if their structural evaluation is within 20% of that of the best mapping, with a maximum of three mappings.

indeed take longer on problems where the model predicts that backtracking is required (Lovett et al. 2009).

An extended model (Lovett & Forbus, 2012) added a second strategy based on constructing an answer by projecting the differences of the A/B comparison onto C, and then comparing the constructed answer to the alternatives. Fig. 7(a) shows an example for which the projection strategy works, and Fig. 7(b) shows an example where second-order comparison is required (because the constructed answer is not one of the valid responses). The combination of the model's predicted strategy shifts and working memory load together account for most of the variance in human reaction times on these problems (R^2 of 0.95). This model and the complete set of inputs is available for download[9]. Working memory load was coded for by counting the number of elements involved in the difference computed for two geometric descriptions. We return to the issue of working memory below.

3.1.2 Visual Oddity Task

To explore possible cultural differences involving geometric reasoning, Dehaene et al. (2006) used a visual oddity task (Fig. 8), in which participants are instructed to select the odd image from a group of six images. They gave this task to two distinct groups: North Americans and the Mundurukú[10] (a South American indigenous group). Overall, there was a high correlation in the performance of the groups, suggesting that some aspects of geometric reasoning may be universal. However, there were open questions about how representations and reasoning might vary across the groups. To address these questions, Lovett & Forbus (2011) modeled human performance on this task, using a combination of SME and qualitative visual representations.

[9] The model is embedded in the CogSketch executable. Two sketches bundled with the distribution include the original Evans problems, and how to run the model is described in the documentation.

[10] The Mundurukú differ markedly in language and culture from Americans. For example, their language appears to have only a rudimentary system of numbers (Pica et al., 2004), and they have few terms for geometric figures (Dehaene et al., 2006).

Again using stimuli copy/pasted from PowerPoint, CogSketch was used to automatically construct initial representations and do rerepresentation as required by the model. CogSketch incorporates three distinct levels of representation. The *object level* is the default, consisting of relationships between entities. The *group level* describes higher-level relationships between sets of objects, using gestalt principles of proximity and similarity (computed via SME). The *edge* level segments a shape into its component edges and computes qualitative spatial relationships between the edges. The model uses SME multiple times to determine what is in common with half of the images in a problem, and looks at the others to find if there is a noticeable dip in similarity. This can require shifting levels of representation, if the default object level does not lead to an answer. Fig. 8(a) shows a problem that can be solved at the object level, while Fig. 8(b) shows a problem requiring an edge-level representation.

Out of 45 problems, the model correctly solved 39 of them, and the problems that it failed to solve were among those hardest for human participants. Moreover, an ablation study suggested reasons for the differences between the two cultural groups found in the original study. The behavior of the Mundurukú is consistent with a stronger focus on the edge level representation, whereas North Americans tended to focus on objects or groups of objects – perhaps an outcome of schooling.

3.1.3 Solving textbook thermodynamics problems

Problem solving is one of the signature roles of analogy in cognition. To explore the importance of incremental mapping in problem solving, we built a very simple problem solver (MARS, Forbus et al 1994), which had no built-in knowledge of engineering thermodynamics, but was capable of using analogies with previously solved problems to solve new problems. The worked solutions were automatically generated by an AI system, CyclePad (Forbus et al. 1999), which has an expert-level model of the domain. CyclePad solutions include explanations, in terms of how each value is derived in terms of others, including what equations were used. These explanations were automatically translated to predicate calculus to be used as analogs for solving new problems. Given a new problem, MARS

compared the problem to the analog it was given, and performed an initial mapping. This initial mapping was used to import equations to be used in solving the new problem. As newly derived information is added to the problem being solved, the mapping is incrementally extended, providing new candidate inferences with further potentially relevant information.

We note that this is only one point in the spectrum of how analogy might be used in problem-solving (Gentner et al. 1997; VanLehn 1998). In this approach, the analogy is treated as a recipe for getting an onerous job done quickly. Although people do sometimes use analogy as a shortcut, this approach fails to take advantage of analogy as a learning mechanism. To capture the more ambitious use of analogy, we built a more extensive simulation of human textbook problem solving, which incorporated analogical retrieval to automatically find analogous worked solutions, first-principles qualitative reasoning to provide a more robust initial understanding of the problem and the ability to import control decisions as well as equations via analogy. Ablation experiments with this simulation that removed more expert-like aspects of performance (e.g. more rigorous encoding using qualitative models, validating analogy suggestions via reasoning, and using multiple retrievals) led to more novice-like behavior, providing evidence for these factors being among those underlying novice/expert differences (Ouyang & Forbus, 2006).

We include examples from the original model here, because it is sufficiently simple that its source code will run in any modern Common Lisp environment, enabling others to experiment with it more easily[11]. The more sophisticated model's worked solutions are more or less equivalent in structure.

3.1.4 Solving Advanced Placement Physics Problems

The Educational Testing Service (ETS) runs Advanced Placement exams for high-school students in the United States. These are high-stakes tests, since doing well on them can improve a student's chances of

[11] The complete source code for this model and representations will be made publicly available upon publication of this paper.

going to a top university. Fig. 9 shows a typical example. Students must ascertain which equations are relevant, solve them, and check whether their answer makes sense. As an experiment, ETS trained and tested a Companion (Forbus et al 2009) on part of the Dynamics component of the AP Physics examination. The relevance of this test for present purposes is that the Companion cognitive architecture[12] makes the structure-mapping processes of comparison, retrieval, and generalization central to its operations. The goal of the ETS experiment was to see if a Companion could, from worked solutions, rapidly transfer knowledge across the following six near-transfer conditions:

1. *Varying parameters*. Changing the numerical values, but in small ways that do not affect the qualitative outcome (e.g. changing the height of the building to be 81 meters).
2. *Extrapolation*. Changing the numerical values so much that the qualitative outcome changes (e.g. changing the height of the building to be 10 meters).
3. *Restructuring*. Asking for a different parameter (e.g. how high will the ball be?)
4. *Distractors*. Adding additional events that are irrelevant.
5. *Restyling*. Changing the kinds of everyday objects involved in problems.
6. *Composing*. Compound problems whose solution requires combining the results of solving two simpler kinds of problems.

The problems involved were drawn from the types of problems found in the Dynamics portion of the AP Physics exam, such as deriving numerical values for a situation, producing the correct prediction of qualitative outcomes in a situation based on numerical values, and producing symbolic equations to characterize a situation.

[12] More generally, the Companion cognitive architecture is exploring the hypothesis that analogical reasoning is central to cognition (Gentner 2003, 2010). It differs from other cognitive architectures in several ways, including that analogical matching, retrieval, and generalization are more primitive than back-chaining in its operations.

The representations for problems and worked solutions were generated by ETS. ETS, with the help of Cycorp, modified their problem generation process to produce problems in predicate calculus, using the Cyc ontology (Matuszek et al. 2006). In addition to producing problems, they also generated worked solutions, also in predicate calculus. The level of information in the worked solutions was similar to that found in textbook explanations of how to solve physics problems. Importantly, these explanations were not geared towards the workings of the Companions' problem solving mechanisms—indeed, ETS had no knowledge of how those mechanisms worked. Each step of a worked solution used the same general set of relationships—e.g. the relation `solutionStepUses` identifies assumptions and antecedents, `solutionStepResult` indicates the result of that step, and `priorSolutionStep` indicates the sequential relationship between two steps of the solution. Thus each step requires multiple relations to encode, in addition to the relations involved in the antecedents and the result. A typical problem can require eight or more steps. Thus, despite the relatively abstract nature of the ETS representations, the number of relationships needed to encode a worked solution is substantial.

To make this a clear test of transfer ability, the Companion had knowledge of how to solve equations and a set of problem-solving strategies, but no prior knowledge of the equations of physics. Its strategies included using MAC/FAC to retrieve prior analogs, looking first for a mapping between event structures similar to the problem it was facing. If a prior analog involved a different kind of event (e.g. throwing instead of dropping), it rejected that reminding and tried again. Candidate inferences were mined for equations and assumptions that could help it solve the current problem. If it could not retrieve a relevant example, it gave up, and did not guess.

All training and testing was done by ETS as follows. Two kinds of training sets were developed:

- Initial training set: Five quizzes, consisting of four problems each.

- Transfer training set: Four quizzes, consisting of four problems each, designed to vary from an initial training set according to which of the six transfer conditions was being tested.

After ETS quizzed a Companion, they gave it the worked solutions corresponding to those quiz problems. This built up the set of potential analogs that a Companion could draw upon. Learning curves across each set were constructed by measuring the number of problems correctly solved in each quiz. For each of the six conditions, the following protocol was used. First, they administered an initial training set followed by a transfer training set (i.e. nine quizzes, 36 problems). Then they wiped the Companion's memory of its recent experiences, and ran just the transfer training set, to provide a baseline. This was done five times for each transfer level, with novel problems used in each training set.

The results were encouraging (Klenk & Forbus, 2009), in that by using MAC/FAC and SME, Companions were able to quickly transfer knowledge across all six conditions. Moreover, their operation could be analyzed in terms of the analogy events (e.g. transferring a line from an example solution, checking transferred knowledge) that VanLehn (1998) found in human students (Klenk & Forbus, 2007).

3.1.5 Moral Decision Making

Cultural narratives have been identified as an important way in which people derive moral understanding about right and wrong behavior (Prasad, 2007; Weber & Hsee, 1999). The prevalence of such narratives suggests that they may serve as the basis for analogies to other moral situations. Indeed, Dehghani et al. (2009) have shown that analogical retrieval affects how moral stories are used across cultures. Based partly on prior research suggesting that decision-making often involves analogy (Markman & Medin, 2002), Dehghani's MoralDM system (Dehghani et al. 2008a) provides a computational model of moral decision-making that relies heavily on analogical retrieval and mapping. The MoralDM model includes protected values (Baron & Spranca, 1997), sometimes called sacred valued, as well as utilitarian concerns. Protected values are modeled via a qualitative order of

magnitude representation—that is, when protected values are present, normal utilitarian differences are lost in the noise. The stories used in psychological experiments were hand-translated into a simplified English syntax, which was then automatically translated into relational representations via a natural language system (Tomai & Forbus, 2009). While the stimuli are simplified in syntax, they involve quite subtle relationships. For example, one simplified English scenario (based on Ritov & Baron, 1999) reads as follows:

"Because of a dam on a river, 20 species of fish will be extinct. You can save them by opening the dam. The opening would cause 2 species of fish to be extinct."

Notice that this involves several kinds of events (e.g. extinction, opening), numerical quantification (e.g. that there are two groups of species with different cardinalities) and a counterfactual (e.g. "would cause"). A strictly utilitarian view would lead to opening the dam, but if directly causing extinction of a species by one's actions is a protected value, then inaction would be preferred. MoralDM uses a combination of first principles and analogical reasoning to work through moral dilemmas. If there are no protected values in a scenario, it looks at the relative utility between the two choices and makes the best choice (i.e. fewer people dying, fewer species of fish becoming extinct). On the other hand, if the highest-utility choice involves taking an action that violates a protected value (i.e. taking an action that will directly cause an extinction), consistent with most participants in these experiments, it prefers inaction to action. In addition to reasoning from first principles, MoralDM also uses SME to look for actions that violate protected values. These methods complement each other, since the rules used for first-principles reasoning are incomplete. MoralDM makes choices that are compatible with the majority choices made in multiple experiments (Dehghani et al. 2008a). Moreover, as the size of the case library grows, the proportion of correct responses improves (Dehghani et al. 2008b), demonstrating that the use of analogy matters in the model's performance.

3.2 Implications for Scale of Analogical reasoning

Table 2 summarizes the properties of the representations used in the experiments above. While this is only a small sample of the set of possible tasks, it includes very different kinds of tasks (visual reasoning, textbook problem solving, and moral reasoning), providing a look at how properties of representations might change across different domains. Table 3 summarizes the statistics of the representations[13].

One thing that stands out in Table 3 is the large size of the representations. These statistics suggest that in order to model human performance over this range of tasks, the analogical mapping process must be able to handle between 10 and 100 relations. Of course, these representations may be larger than is needed. Our priority has been to use automatically generated representations, from systems developed both in our laboratory and by others. Such systems tend to optimize for compact representations, given that large representations tend to raise the cost of other operations as well. However, there is no guarantee that they are the most compact possible representations. But even if our representations are two or three times larger than needed, this still leaves us with 20 or 30 relations for some kinds of problems. How do these figures square with current estimates of online processing capacity?

Let us first consider the case when external representations are present, such as in a geometric analogy problem. In these cases, people may match large descriptions incrementally, relying on the external representations to relieve the working memory burden. This would be consistent with the incremental matching techniques discussed above. Of course, even in this case, people would still have to internalize enough of the ongoing mapping to be able to accurately maintain structural consistency across the solution steps.

[13] For the AP Physics experiment, we include only a subset of the problems, i.e. those for one transfer condition. This is due to the large number of quizzes in that experiment.

But what about cases in which no external representation is present, as in comparing a current situation to a representation in LTM? This brings us to the critical issue of working memory. In some accounts, working memory (WM) is synonymous with short-term memory (STM), as assessed in tasks such as digit span. STM is primarily concerned with short-term information storage and is subject to temporal decay and capacity limits of around 4 chunks at best (Cowan, 2001). Clearly, limits implied by equating WM with STM are difficult to square with our representational assumptions. However, some recent theories have emphasized the role of WM in holding and manipulating information, including information from LTM (Baddeley, 2012; Ericsson & Kintsch, 1995). For example, Ericsson and Kintsch (1995) propose a distinction between ST-WM and LT-WM. The former is the traditional STM, characterized by strict temporal and chunk capacity limits. In contrast, LT-WM is concerned with accessing and manipulating information in LTM, and varies with people's knowledge and skill. Ericsson and Kintsch review evidence from studies of text comprehension that shows that readers are able to temporarily retain and use amounts of information that are far larger than typical STM limits. Moreover, this temporary store is durable in the face of brief interruptions. Baddeley (2012) agrees that interactions with LTM could boost WM performance, and maintains that this kind of interaction is compatible with current conceptions of WM[14].

In any case, the important point is that Ericsson and Kintsch's comprehensive review shows that domain knowledge is an important determinant of effective WM capacity. They review a wide array of evidence from studies of experts (in chess, bridge, abacus calculation and even waiting on tables), showing that the amount of material a person can retain and manipulate over short periods is far greater in their

[14] "Ericsson & Kintsch (1995) proposed this concept [long-term working memory] in explaining the superior performance of expert mnemonists, going on to extend it to the use of semantic and linguistic knowledge to boost memory performance. They argue that these and other situations utilize previously developed structures in LTM as a means of boosting WM performance. I agree, but I cannot see any advantage in treating this as a different kind of WM rather than a particularly clear example of the way in which WM and LTM interact." (Baddeley, 2012, p. 18).

domain of expertise than in other arenas. They also review evidence concerning "everyday expertise" — that is, people's ability to encode and retain meaningful text—which shows that the amount of information people can temporarily retain when processing meaningful text is far larger than would be expected from most estimates of capacity limits (and much larger than when processing scrambled text). As they put it, "Domain knowledge provides the retrieval structures that give readers direct access to the information they need when they need it. Given a richly interconnected knowledge net, the retrieval cues in the focus of attention can access and retrieve a large amount of information." (Ericsson & Kintsch, p. 231).

Ericsson and Kintsch's point that domain knowledge vastly influences the effective capacity of WM accords with work in analogy, which has shown repeatedly that the nature and quality of domain representations is a key determinant of how people process analogies. In particular, Ericsson and Kintsch's emphasis on "richly interconnected knowledge" is consistent with studies showing the importance of systematicity—the presence of higher-order relations[15] that connect lower-order relations—in allowing people to carry out large analogical mappings (e.g., Gentner & Toupin, 1986; Loewenstein & Gentner, 2005). A well-structured domain representation has at least two advantages for analogical processing: (a) it may be treated as a few higher-order nodes and unpacked into more detailed structures as needed; and (2) it may permit incremental processing, because the connecting relations allow the person to keep track of where they are in processing a large representation.

[15] The exact set of linking relations that provides coherence is an open question. Ericsson and Kintsch (1995) discuss six classes of coherence elements proposed by Givón (1992): referents, temporality, aspectuality, modality/mood, location, and action/script. Gernsbacher (1997) proposes the coherence can arise from (at least) referential, temporal, locational, and causal links. Researchers in analogy have emphasized causal relations (including PREVENT and ENABLE) as well as logical relations such as IMPLIES and higher-order spatial relations such as MONOTONIC-INCREASE.

Such higher-order structure might operate much like the large-scope chunks in cognitive architectures such as SOAR (Laird, 2012) and ACT-R (Anderson, 2007). If so, then the capacity limit may be the number of coherent substructures, rather than the number of individual predicates. The assumptions concerning working memory in the visual processing models, where the objects mentioned in the differences were counted as working memory, rather than the relations involving them, is compatible with this approach.

4 Techniques for Scaling Up Analogical Processing

This section describes five techniques that we have found crucial for scaling up analogical processing, in the investigations above as well as others: Greedy Merge, incremental operation, ubiquitous predicates, structural evaluation of candidate inferences, and match filters. We discuss each in turn.

4.1 The GreedyMerge Algorithm

Recall that in SME's local-to-global algorithm, local matches are first constructed in parallel, and kernels (structurally consistent combinations of matches starting from non-subsumed match hypotheses) are subsequently combined to form the correspondences of a global mapping. In our first two versions of SME, we used an exhaustive merge process to find all globally consistent solutions. While guaranteed to find optimal solutions, the worst-case performance of such an algorithm is factorial in the number of kernels. Clearly a more efficient search technique is necessary for structural alignment to be widely used as a process within realistic-scale tasks.

Our solution has been to trade off optimality for performance, by using a greedy algorithm for merging. The essence of a greedy algorithm is this: Suppose one has a set of partial solutions, only some of which are mutually consistent with each other, which must be combined into the best possible global solution. In general this is an NP-hard problem, i.e., requiring exponential growth in resources as the size of descriptions grows, because one must try all combinations of partial solutions together to guarantee

optimality. What a greedy algorithm does instead is to select the best partial solution, add to that the next-best partial solution consistent with it, then the next-best partial solution consistent with those two, and so on, until nothing else can be added. This algorithm is linear in the number of partial solutions, and yields optimal (or near optimal) answers surprisingly often, as discussed below.

The greedy idea is applied to building mappings in SME as follows: Kernels are the partial solutions, and are rank-ordered by their structural evaluation, i.e., the sum of the structural evaluations of the local match hypotheses included in them. The first interpretation is created by starting with the structurally largest kernel and going down the list, merging kernels with it that are not structurally inconsistent with the solution being generated. SME can generate more than one global mapping by selecting the highest-ranked remaining kernel and starting the process over again. The maximum number of interpretations returned is a parameter of the simulation, and defaults to three. There is also a cutoff percentage, such that if a subsequent kernel is going to be significantly smaller than previous solutions it is not generated. Thus when there is only one "obvious" mapping, only one is produced. But if there are competing interpretations, up to two additional mappings will be produced as well.

We have subsequently improved on the simple greedy merge algorithm reported in Forbus & Oblinger (1990) in two ways. First, we divided the merge process into two steps. The insight is that, since candidate inferences only arise when base statements are imported in the target, merging kernels that project to a common base root increases the likelihood of finding productive matches. Thus we first perform a greedy merge operation within kernels whose base projection shares a common root, and then a second round of greedy merging over those solutions to construct mappings. Second, we allow mappings to share kernels. The original greedy algorithm placed each kernel in at most one mapping. However, it is theoretically possible for distinct mappings to share some matches, and thus some kernels. To support this, after a mapping is greedily computed, the algorithm iterates over all kernels

from previously computed mappings. If any kernel is consistent with the current mapping, it is added to it. This allows secondary mappings to be better filled out, and because it requires only a single pass through the previous mappings, it adds minimally to the computational cost. Appendix A provides a complete description of the current GreedyMerge algorithm.

How good are the solutions produced by the greedy merge algorithm? As reported in Forbus and Oblinger (1990), a simple version of GreedyMerge was originally tested on 56 analogies, ranging from comparisons between physical phenomena, short stories, and object descriptions, drawn from a library of SME examples. We found that greedy merge produced the identical best mapping to the original exhaustive merge in 52 of these cases, i.e., 93% of the time. The few cases where its results were suboptimal involved a number of large and mutually inconsistent initial kernels. We discuss some subsequent similar analyses by other researchers in Section 6.

Why does GreedyMerge normally do so well? Typically, analogies over large descriptions have a few large kernels, only some of which are mutually inconsistent, and a much larger set of small kernels. Thus the first few decisions are the really critical ones, and they are relatively easy to make. When does GreedyMerge fail? There are two kinds of cases where it can fare poorly. The first is when there are many large kernels with a high degree of mutual inconsistency. In this case, a large number of decisions have to be correct, and hence the chance of error grows. This was the problem in the few cases in our original experiments (4 out of 56) in which a non-optimal solution was generated. We have also seen non-optimal solutions in practice, particularly when applying SME to large visual representations of diagrams (e.g. Ferguson & Forbus 2000). There are two solutions to this sort of problem. The first is to increase the number of interpretations one is willing to consider, or prune the set of possibilities via filter constraints, as discussed in Section 4.5. The second is to change the algorithms used to generate representations, to introduce more relevant structure.

Another kind of case where, in principle, Greedy Merge would do poorly is that in which an initial, large kernel is inconsistent with every member of a large set of small but mutually compatible kernels that together outweigh the initial one. We have not yet observed this phenomenon in natural representations. Fortunately, the ability to generate radically different interpretations provides the potential to recover from such problems.

The theory of *submodular functions* provides some useful insights. A submodular function can be thought of as an evaluation function such that the value of adding an additional component to a solution decreases as the size of the solution increases (Cormen et al. 2009). Greedy algorithms are optimal when their evaluation function is submodular, a global optimum can be arrived at by selecting a local optimum (the greedy choice property) and an optimal solution contains optimal solutions to subproblems (the optimal substructure property). Unfortunately, as the cases above illustrate, structural evaluation is not always submodular; hence we know that GreedyMerge cannot always be optimal. However, this can be turned around. It seems possible that human representations are tuned such that structural evaluation tends to be submodular, and when it isn't, that is a signal for rerepresentation. A related question is whether there is a set of broad representation conventions over which GreedyMerge is always optimal. These are currently interesting open questions.

4.2 Incremental Operation

Many perceptual and cognitive tasks involving structural alignment, including metaphor understanding, problem solving, and learning, require the ability to process information incrementally. For example, when processing an extended analogy or metaphor, readers often build up correspondences across several sentences. This means that each sentence must be linked to the relational structure that has been built up so far. One indication that this happens is that people experience a 'mixed metaphor' startle when the metaphoric mapping changes in midstream, as in "The ship of state is boiling over." To test the idea that people often maintain consistent mappings, Gentner et al. (2001) gave people

passages containing extended metaphors, one sentence at a time (See Table 4). The dependent measure was the time required to read each sentence. As predicted by the incremental mapping assumption, people took longer to read the final sentence in the Inconsistent case than in the Consistent case. (See also Gentner & Boronat, 1991; Gentner, Imai & Boroditsky, 2002; Thibodeau & Boroditsky, 2011; Thibodeau & Durgin, 2008).

In problem solving, students using a worked example to solve a related novel problem may go back and forth between them, seeking additional ways to interpret the new problem in light of the old. In conceptual change, new data can lead to analogies being modified or abandoned. Modeling these processes requires the ability to incrementally extend a match with new information. Burstein (1986) was the first to computationally model incremental processing in analogical learning, in a domain-specific system for modeling learning to program. Falkenhainer's (1987, 1990) PHINEAS demonstrated that SME could be used in a map/analyze cycle to model the incremental use of analogy in discovering physical theories, albeit with a number of external mechanisms.

The first general-purpose incremental analogical matcher was Keane's IAM (Keane & Bradshaw, 1988). A critical difference between IAM and the technique we have developed for SME concerns serial versus parallel processing. Recall that in SME, processing is essentially parallel within the first two stages: only the final step of constructing global mappings is serial. By contrast, IAM is serial throughout: Even decisions about local matches are made sequentially, so that exploring alternate interpretations requires backtracking. SME avoids backtracking by creating, in parallel, a network representing all local identity matches between items, followed by intermediate clusters (i.e., *kernels*, introduced in Section 2 and described formally in Appendix A).

We believe that a combination of initial parallel processing and later serial processing will best model human structural alignment processes. Some serial processing is essential: One cannot combine all

information in parallel when not all of it is yet available. However, we believe the fully serial approach of IAM would be difficult to scale up to cognitively plausible representations. Moreover, as noted above, there is now evidence that something like SME's initial symmetric alignment process occurs in human processing, even for strongly directional metaphors (Wolff & Gentner, 2011).

We suggest that the natural place for serial processing is in the Merge step. The kernels represent coherent, structurally-consistent collections of local matches, and therefore form a more appropriate unit of analysis for limited-resource serial processing than the individual local matches themselves. When base and target share large systematic structures, the number of kernels is small. Serial, capacity-limited merging of kernels could thus provide a plausible explanation for the "More is Faster" phenomenon whereby additional shared knowledge can improve both the rapidity and the accuracy of mapping (Gentner & Rattermann, 1991; Loewenstein & Gentner, 2005).

The constraint of incremental operation changes the processes of a matcher in several ways. The default operation becomes extending the current set of mappings when new information is added to the base and/or target, instead of starting from scratch. However, one possibility raised by extending existing mappings is that what looked promising initially might turn out to be sub-optimal as more information is known. Thus the matcher must have the ability to remap, that is, to take a fresh look at a comparison. SME does this by discarding the existing mappings and redoing the Merge operation from the set of kernels. We assume that the criterion for remapping is task-specific; hence there is no automatic criterion for remapping built into SME. The problem solver described in Section 3.1.3 uses incremental mapping, for example, since the problems can be quite large.

4.3 Ubiquitous Predicates

For purposes of matching, not all statements are created equal. Some researchers have suggested identifying specific "important" predicates, such as CAUSE, and focusing on those in matching (e.g.,

Winston, 1982). The problem with identifying certain predicates a priori as important is that it reduces the ability of a match process to be context-sensitive. While causality may be critical in some tasks, such as story understanding, other kinds of knowledge may be critical in different tasks, e.g. spatial configurations in navigation. Moreover, if CAUSE is given extra weight, would this also apply to PREVENT and ENABLE, not to mention AID, ABET, HAMPER and so on? Structure-mapping's systematicity principle suggests that importance arises from consideration of the size and depth of the overlapping structure between two descriptions, not as a purely local judgment. Moreover, encoding processes that produce the inputs to matching are presumably tuned to what is important in the current context, and what is stored in long term memory and subsequently retrieved is also. Thus we suggest that predicate-specific heuristics for judging a statement to be important are neither necessary nor sufficient to ensure optimal matching.

Perhaps surprisingly, the reverse turns out to sometimes be very useful: that is, using a local, predicate-level heuristic for judging a statement to be unimportant—that is, as unlikely to yield useful matches in itself, can greatly simplify the match process. In many domains, there are predicates that occur so frequently that the likelihood of any particular match involving them being useful is low. For example, using analogy to solve thermodynamics problems (Section 3.1.3) can involve matching descriptions that each contain many equations. Each equation could potentially match (because they all involve the predicate =), but most of them will be irrelevant. Another example is the conjunctive connective, and, which is crucial for bundling antecedents together, but is not, on its own, sufficient reason to attempt to form matches. Matches only based on = or and are unlikely to be useful. On the other hand, one cannot simply filter such statements out; they are central components of the domain knowledge. Indeed, matching the appropriate pair of equations can be essential for solving a problem.

In any local-to-global process, having a large number of irrelevant local matches reduces the likelihood of finding strong overall matches. The problem, of course, is that it is impossible to be certain when comparing two statements locally that they will not be useful, since (by systematicity) usefulness arises as a property of an overlapping system of relations.

Our solution to this dilemma is to declare certain predicates common in a domain to be *ubiquitous.* A *ubiquitous predicate* is a predicate which, by itself, does not constitute a sufficient condition to postulate a match between two statements that share it. Which predicates are to be treated as ubiquitous is provided as part of the input to the match process, automatically, by the larger-scale task model on a domain-wide basis. The only difference in how ubiquitous predicates are treated occurs in the initial match hypothesis construction step. In that step, every pair of expressions is compared and, if they satisfy tiered identicality, a match hypothesis is created – unless the predicates are ubiquitous. If the predicates are ubiquitous, the pair is ignored at this step. However, the pair can still appear in a match hypothesis if they are arguments of other statements that are matched. Thus, for example, the = relations in pairs of equations that play similar roles in an explanation or problem solution will have a match hypothesis created for them, by virtue of them being part of the aligned argument structures. Thus statements involving ubiquitous predicates cannot be matched on their own, but can be matched as part of a larger matching structure, thereby satisfying the demands of systematicity and parallel connectivity.

The choice of which predicates should be declared ubiquitous must be made with respect to the domain(s), independent of the demands of specific matching tasks or specific problems within the domain. This is an important point theoretically, since it reduces the potential for tailorability in modeling that would be introduced if such decisions were made on an example-specific basis. There are two general guidelines for using ubiquitous predicates in SME-based models:

1. Predicates that are basically "joints" in larger structures are not sufficient evidence, by themselves, to warrant matches. Examples include `and`, `TheSet`, and `TheList`. On the other hand, predicates like `cause` and `implies`, despite their high frequency, should never be declared ubiquitous because these form the potential backbone and connective tissue of relational structures. With them, the relevant joints can be found.
2. Attributes that are shared by most elements of two descriptions are also good candidates for being declared ubiquitous. For instance, in the representations for sketches used in (Forbus et al. 2011), every depicted object has an associated glyph. Consequently, the attribute of being a glyph is not diagnostic for matching two sketches, whereas spatial relationships involving the glyphs and domain attributes of the objects depicted typically are.

How important are ubiquitous predicates? We illustrate by examining the use of ubiquitous predicates in the five computational investigations outlined in Section 3. Table 5 describes what ubiquitous predicates were used in the SME comparisons performed in each domain. No ubiquitous predicates were used in the two visual tasks because the encoding processes for those models automatically filters out the `Glyph` attribute, which is reasonable because it plays no role in higher-order relations.

To see how ubiquitous predicates affect performance, Table 6 shows statistics on the increased size of the match hypothesis forest for each domain where ubiquitous predicates are used. While ubiquitous predicates had little effect in moral decision-making, they had significant impact on the size of the representations for the problem-solving experiments. This is because in the problem-solving domains most new steps introduce a new equation, so there are potentially many unproductive local matches there. On the other hand, in moral decision-making, the stories are short, making the representations smaller, and there is little repetition among the relations used.

4.4 Structural Evaluation of Candidate Inferences

Candidate inferences can be evaluated along several dimensions. One such dimension is structural quality, and since the comparison operation is based on structural properties, it is the natural place for such evaluations to be computed. (By contrast, utility or logical validity are properties that are task-specific, and therefore outside the comparison operation.) This section describes how SME performs structural evaluation of candidate inferences[16]. Structure-mapping theory defines analogical inferences as projections from the base to the target that are structurally supported by the correspondences of a mapping. It also defines reverse candidate inferences from the target to the base, which are also now supported by SME. The ideas in this section apply equally to forward and reverse candidate inferences.

To begin with, we need a little more terminology. Recall that candidate inferences for a mapping are generated by examining how either the base or target intersects the mapping. We call a statement a *root* in a description if it is not the argument of any other statement in that description. Recall that a match hypothesis indicates a potential correspondence between two statements or two entities in the descriptions being compared. The *arguments* of a match hypothesis are the match hypotheses that align the arguments of the statements which that match hypothesis aligns. Thus if we had

MH1: `(connectedTo A B)` ↔ `(connectedTo C D)`
MH2: `A` ↔ `C`
MH3: `B` ↔ `D`

Then MH1 would have two arguments, MH2 and MH3. By analogy with roots in a description, a match hypothesis is a root if it is not an argument in another match hypothesis.

Consider a statement that is a root of its description. If it participates in the mapping, i.e. there is a match hypothesis aligning it with a statement in the other description, then it is part of the overlap between the two descriptions, and can provide no additional new information. But if a statement that is

[16] This summarizes and extends the presentation in Forbus et al. (1997).

a root is not part of the mapping, but it has subexpressions that are, then a candidate inference is computed, to represent the projection of that potential new information into the other description. In Fig. 2, for example, there are two qualitative proportionality statements (`qprop+` and `qprop-`) which express causal relationships between continuous parameters. Both of these statements are roots of the base. The `qprop+` statement indicating that the frequency of the oscillation is positively affected by the spring constant participates in the mapping. By contrast, the `qprop-` statement indicating the causal connection between the frequency of oscillation and the mass of the block, being an unmapped root with sub-expressions covered in the mapping, serves as a starting point for generating a candidate inference. Similarly, since the `cause` statement, the `block` and `spring` statements, and one of the `part-of` statements are all roots, they too are used to create candidate inferences. The form of the inference is the root expression, with substitutions made as necessary from the correspondences, and with skolem functions introduced for base constants that do not have correspondences. This example has one skolem—namely, that the Earth is made of something like steel (i.e., `(:skolem steel)`, in Fig, 5).

Once candidate inferences are generated, how are they to be evaluated? One aspect of evaluating candidate inferences is validity—that is, using other types of reasoning to find out if they are in fact true or false in the target domain. However, this can be expensive, so it is worth providing some estimate of the properties of the inference based on the structural alignment, to serve as a guide to other processes.

The structural evaluation of a mapping provides an estimate of match quality, based on the nature of the overlap. We suggest that a similar structural evaluation occurs psychologically for candidate inferences. However, for candidate inferences we postulate two distinct dimensions:

1. *Support:* How much structural support does an analogical inference derive from the mapping that generated it? We estimate this by using the same trickle-down algorithm used in structural evaluation, but limited to the subset of the correspondences that support the inference in the mapping.
2. *Extrapolation:* How far does an analogical inference go beyond the support lent by the mapping? We estimate this by using the structural evaluation trickle-down algorithm within the candidate inference, taking the ratio of the score for the structure outside the mapping over the sum of the score for entire inference (i.e. inside and outside).

The methods for computing these scores are described in more detail in Appendix A.

We believe these two measures have significantly different functional roles. Support is like the structural evaluation of mappings: More is always better. Extrapolation is more complex: High extrapolation seems desirable in tasks like brainstorming or theory generation, but low extrapolation may be preferable for within-domain comparisons involving highly familiar situations.

That people are able to identify which inferences follow from a given set of correspondences has been demonstrated experimentally (Clement & Gentner 1991; Spellman & Holyoak 1996). For example, Markman (1997) found that analogical inferences follow structural consistency, even when there are multiple possible mappings (see also Clement & Gentner, 1991). Our model of candidate inferences, which only computes them from structurally consistent mappings, is consistent with these results.

Our definition of support score is consistent with several lines of evidence. Psychologically, matches involving larger systems of statements are viewed by subjects as more sound (Gentner et al., 1993). As noted above, Clement & Gentner (1991) showed that subjects made predictions based on statements connected to a common antecedent in the base, and that candidate inferences connected to systematic base structures are preferred to those which are not.

Similarity has been suggested as a central process in induction tasks (Heit & Rubenstein, 1994; Lassaline 1996; Osherson et al. 1990), so it is useful to see how this model fits with these studies. Lassaline (1996) asked subjects to rate the similarity of pairs of fictitious animals and the inductive strength of a property inference (i.e., if A has X, W, and Z, and B has X, Y, and Z, how likely is it that A has Y?, where A and B were fictitious animals and the rest of the variables were filled in with properties such as "dry flaky skin" or "attacks of paranoia"). Adding a relation in the base that explained the inferred property (i.e., telling the subjects that in B, X causes Y while leaving the description of A unchanged) increased inductive strength, but adding a relation that was not connected to the inference did not. A simple model of this task is to treat it as analogical mapping, with animal B serving as base and animal A as the target, and treating inductive strength as a function of the candidate inference support score. Using these assumptions, a simulation of her experiments using SME also yields this result (Forbus et al 1997).

Because of the systematicity principle, when there are multiple possible inferences from the base to the target, SME predicts that the inferences will be governed by where the greatest structural commonality can be found. This fits with findings by Heit and Rubenstein (1994), who found that people make stronger inferences about whether one animal has a property based on another animal's having it when the kind of property to be inferred (anatomical or behavioral) matches the kind of similarity between the animals (anatomical or behavioral). For instance, people judge the likelihood that whales travel shorter distances in extreme heat to be higher when told that tuna do, relative to when they are told that bears do, presumably because whales and tuna have greater behavioral overlap (both swim)--even though whales and bears match better anatomically (both mammals). These findings are consistent with the prediction that people prefer to make inferences with greater support from the common structure.

4.5 Match Filters

One strength of SME is that it can detect unanticipated structural correspondences. This is essential for capturing the generativity of human analogical reasoning. However, some tasks impose particular constraints on analogies. When reading an explanation of the greenhouse analogy for warming in the Earth's atmosphere, for example, the reader's interpretation of that analogy must include a correspondence between the greenhouse glass and the atmosphere. Particular correspondences are often specified in instructional analogies (Barbella & Forbus, 2011; Richland, Zur, & Holyoak, 2007). For example, when learners are given the hydraulic analogy for electricity, they are typically told the correspondences (water flow→electric current, pressure→voltage, etc.) Further, in learning a new domain by analogy, e.g. understanding how to solve rotational dynamics problems by analogy with linear dynamics problems, the mapping between the domains is often built up incrementally across multiple problems (Klenk & Forbus, 2013). These incrementally constructed domain mappings are then used in new analogies when solving problems that extend the system's understanding of the domain. In other words, the nature of some tasks require adding additional constraints that the matching process needs to respect.

Since SME is only generating one to three mappings, tasks need to be able to communicate constraints to SME, to influence it towards more useful matches. *Match filters* provide a restricted language for that communication. Match filters are local, based on structural properties of the representation system. By default, SME uses no filters. Importantly, when filters are used, they are automatically imposed by the task model, never by hand.

There are three kinds of match filters. The first is the *required correspondence* filter:

- `(required Bi Ti)`: Any mapping that includes a correspondence for either `Bi` or `Ti` must map them to each other.

The `required` filter is useful when the task imposes correspondences, as in the electricity-water analogy above, or when a speaker declares that "fume hoods are like vacuum cleaners" in an analogy intended to be used by learners (Barbella & Forbus, 2011). They are also used to establish persistent cross-domain mappings (Klenk & Forbus, 2013). We note that `required` constraints are sometimes used in large-scale models, including some of those in Section 3.

Something similar to the `required` filter was first used in PHINEAS (Falkenhainer 1987). PHINEAS learned new qualitative models for domains by first comparing a novel behavior to an understood behavior. The analogy between the behaviors introduced correspondences that were used in a second mapping, with the explanation of the first behavior as the base, and the (initially empty) explanation of the novel behavior as the target. The correspondences found in the first mapping were required to hold in the second mapping, which provided the necessary translation of terms to enable the importation of the explanation (as a set of candidate inferences) into the new domain.

The second kind of filter is excluding a pair of correspondences:

- `(excluded Bi Ti)`: No mapping can include a correspondence between `Bi` and `Ti`.

In many tasks, SME is used iteratively – a mapping is generated, inspected by the larger model, and if the mapping is unsuitable, it must look for another. For example, solving geometric analogy problems sometimes requires backtracking and looking for an alternate mapping between two images (Lovett et al., 2009). Hence `excluded` constraints serve a role analogous to nogoods in truth-maintenance systems (Forbus & de Kleer, 1993), encoding information about correspondences that, while structurally sound, have been found to be inappropriate for other reasons.

The final kinds of filters are predicate filters:

- `(identical-functions)`: No mapping can include correspondences between non-identical functions. This overrides the usual policy of letting non-identical functions match whenever suggested by a larger relational structure.

- `(require-within-partition-correspondences Att1 Att2)`: No mapping can include correspondences that map an entity with attribute `Att1` to an entity with attribute `Att2`.

The `identical-functions` filter supports a conservative strategy often used in routine problem-solving, and by learners in an unfamiliar domain, when they may reject all but the most certain mappings. In solving physics or thermodynamics problems, for instance, the circumstances under which one can substitute one type of parameter for another are strictly limited. For example, the analogical problem solver in (Ouyang & Forbus, 2006) used the identical-functions constraint when using SME to retrieve and apply plans from previously solved problems to handling new thermodynamics problems.

The `require-within-partition-correspondences` filter supports another conservative strategy that is valuable for within-domain comparisons involving complex examples. For instance, people and mountains can both be used in representations as ways of denoting locations, but people can be given tasks whereas mountains cannot. (This example comes from using SME to reason about military tactics problems (Forbus et al. 2003).) This class of constraint was also used in solving geometric analogy problems (Lovett et al 2009), to enforce human preferences for matching identical shape categories in this task.

5 Theoretical and Empirical Complexity Analysis of SME

Recall that the initial phase of SME's processing is assumed psychologically to occur in parallel, constructing a forest of local hypotheses about matches and constraints between them. This includes a local structural evaluation process that uses a propagation algorithm to compute scores for match hypotheses consistent with the systematicity principle. The maximally consistent single-root subsets of this forest are the kernels of the match, which are then combined via a greedy-merge process to create mappings. Assuming that the number of items in base and target is both n, then the worst-case size of the match hypothesis forest is bounded by n^2. This is also a worst-case bound on the number of kernels.

As Appendix A summarizes, the theoretical worst-case complexity for SME on a serial machine is $O(n^2 log(n))$.

Worst-case analyses are notorious for their pessimism, and this analysis is no exception. The best way to demonstrate this is empirically, using examples drawn from the modeling projects in Section 3. As noted earlier, we instrumented the simulations and dumped copies of the base, target, and enough of the vocabulary to enable that SME match to be duplicated later, apart from the rest of the simulation code, to provide more accurate timings. All runs were conducted on a cluster node running Linux, to minimize the number of other processes interfering. This gave us a total of 5,839 comparisons.

The size of the match hypothesis forest plays a central role in determining the overall effort needed in a comparison. If there is substantial overlap, the forest will be large. This may or may not lead to large matches, depending on the degree of structural consistency in the overlap, but certainly a large mapping cannot arise from a small forest. It is instructive to compare the actual size of match hypothesis forests to the theoretical worst-case. Since our worst-case analyses in Appendix A are all based on number of items (i.e. expressions plus entities), we can compare the square of this number, which is the worst-case complexity of this operation, to the actual number of match hypotheses and kernels in these mappings. Table 7 shows the results.

In general, the number of match hypotheses is typically well below the worst case, and the number of kernels (which governs the time for the serial processing phase) is even further below the worst case. In other words, on realistic descriptions, SME runs far faster than one would expect from the worst-case analysis. The extreme closeness of the statistics for the two visual reasoning tasks is almost certainly due to the use of CogSketch for automatic encoding in both tasks.

Can the size of descriptions be used to make any empirical prediction of run-time? The answer is no, because the size of the match hypothesis forest depends on the *overlap* between the base and target.

There is more potential overlap for large descriptions than small ones, but two medium sized descriptions with many of the same predicates can have more overlap than two larger descriptions whose predicates have little overlap. Thus there is no simple formula in terms of size alone for estimating run-time accurately. That said, the theoretical complexity analysis captures the growth in resource usage. Empirically, the correlation between $n^2\log(n)$ and run time on a serial machine is quite high for all of the domains ($R > 0.9$) except for moral decision-making, where it is uncorrelated ($R = -0.02$). We note that moral decision-making is the only experiment in Section 3 involving cross-domain matches involving simple stories, which we believe explains the difference in its growth curve. This suggests that the complexity analysis of Appendix A is a reasonable approximation in terms of growth of resources, but as Table 7 indicates, the actual numbers never get even close to the worst case, meaning that empirically performance is better than this analysis would suggest.

6 Related Work

Here we describe how our research fits into the broader picture of research on analogy and case-based reasoning. The success of the field has led to an increase in the number and variety of models of analogy, so this review is necessarily selective. (For more comprehensive reviews, see Gentner & Forbus, 2011; Kokinov & French, 2003). Chronologically, perhaps the earliest computational model of analogy was Evan's (1968) system for geometric analogy tests. Winston's (1980) pioneering work considered a broader range of analogies, including both perceptual and conceptual analogies. His work on "near miss" analogies dovetails with our work on alignable differences. While motivated by psychological concerns, these systems were rarely tested against empirical human data.

The original version of SME (Falkenhainer et al. 1986; 1989) was the first simulation of analogical matching shown to be consistent with human similarity ratings (Skorstad et al., 1987) while also being able to produce psychologically plausible candidate inferences. The current version of SME retains

those abilities, while adding the ability to generate alignable differences and abstractions and to handle incremental inputs, match filters, and scaling behavior that makes it more plausible as a human model.

Holyoak and Thagard's (1989) ACME followed closely after SME, and was partly based on SME. ACME was the first connectionist simulation of analogy. It utilized a multiconstraint approach to analogical mapping; the constraints were structural consistency; similarity between corresponding objects ("semantic similarity"); and pragmatic goal-consistency. ACME used a localist connectionist network to implement SME's match hypothesis forest, and a winner-take-all algorithm to compute the final mapping, taking all three constraints into account. However, ACME had some serious drawbacks (See Hummel & Holyoak, 1997). First, it allowed many-to-one matches, which led to structurally inconsistent mappings. Later studies confirmed structure-mapping's claim that many-to-one matches are psychologically implausible (Krawczyk et al., 2005; Markman, 1997). Second, people are capable of considering more than one mapping, and ACME's winner-take-all algorithm ruled this out. Third, ACME was not capable of generating novel candidate inferences. It could fill in a piece of structure given an explicit suggestion, but could not carry over inferences spontaneously as people routinely do (Falkenhainer, 1990; Markman, 1997). Thus it was unable to capture some of the key benchmark features of analogy summarized in Table 1.

Keane and Brayshaw's (1988) IAM operates in an incremental fashion. Like SME, it has explicit structural representations and maintains structural consistency. However, unlike SME, it can process 'unnatural analogies' that do not involve semantic commonalities—e.g., "Fido eats Kibbles" and "John loves New York". Because IAM hypothesizes matches sequentially, it is highly sensitive to the order in which information is presented. This strong dependence on order of matches is problematic in light of evidence that some of the object matches made early in processing are later overruled by structural consistency (Goldstone, 1994).

Larkey and Love's (2003) CAB model provides a parsimonious localist connectionist model of matching. CAB uses a simple iterative computation that matches elements in one representation with elements in the other. As in Goldstone's (1994) SIAM, initial matches are governed by local object matches, with structural constraints becoming more important over processing. Larkey and Love (2003 showed that CAB can capture some of the benchmark phenomena for analogical processing (see Table 1). However, it is unable to produce candidate inferences, one of the key benchmark phenomena.

Several early models focused on particular domains, on the assumption that analogical matching is tightly integrated with domain-specific processing. For example, COPYCAT (Hofstadter & Mitchell, 1995; Mitchell, 1993) constructed representations of strings of letters and solved analogies involving them. Similarly, TABLETOP (French, 1995) used processes of encoding and matching to capture how place settings at a table placed in analogy with each other. These models have some interesting features, particularly the attempt to model interleaved processes of encoding and matching. However, their mapping algorithms rely on domain-specific processes. The psychological evidence to date suggests that analogical mapping processes are domain-general. The current preponderance of domain-independent computational models of analogy matching suggest that models that focus on capturing a broader range of behavior in particular domains might do well to use a general-purpose matcher.

The closest problem-solving model to ours is CASCADE (VanLehn & Jones, 1993), which used analogy in problem-solving. When solving for a particular quantity in a new problem, CASCADE would look for a step in a prior solution that involved that quantity. We used the same type of strategy in the simple thermodynamics problem solver described earlier, but with a general-purpose model of analogical matching (SME) instead of the special-purpose matcher used in CASCADE. Another contrast is that in our AP Physics experiment, retrieval of prior problems was done using MAC/FAC, allowing us to capture the specific dynamics of analogical retrieval. Another interesting model was PRODIGY-ANALOGY (Veloso

& Carbonell, 1993). This was the first use of analogy in a cognitive architecture. PRODIGY treated analogy as a means of replaying problem-solving traces (i.e. *derivational analogy*), and its matcher and retrieval mechanism were specialized for that purpose. By contrast, SME has been shown to be useful for derivational analogy, but also for decision-making and visual problem solving, as the experiments summarized in Section 3 illustrate.

Grootswagers (2013) took a different approach to analogical modeling by conducting explorations of optimality. He constructed a generator that would produce pairs of structures that varied in degree and kinds of match, and used this to look at when a greedy algorithm would produce optimal results. He implemented his own version of the 1990 version of SME as well as an exhaustive version. His greedy code found the same solution as his exhaustive algorithm 87.5% of the time (page 27, ibid) and found the same solution 99.89% of the time when run on pairs of plays from the original ACME datasets (page 34, ibid.). We find these results encouraging, since they support our claim that greedy merge performs well most of the time. However, there is an important caveat: Our criteria was generating results identical to the exhaustive algorithm, whereas his optimality criteria was maximizing score. These are subtly different: Our greedy algorithm always produces structurally coherent candidate inferences, whereas Grootswagers' algorithm does not, since it uses as kernels non-maximal collections of match hypotheses. If two maximal kernels cannot be merged, it is because they are structurally inconsistent. So while including subcomponents of them might increase the match score, the resulting structure used to generate candidate inferences will be structurally inconsistent, which is psychologically implausible (Krawczyk et al. 2005; Markman 1997).

Grootswagers also suggested that two variants of the original exhaustive SME would be better models than our greedy algorithm. The specific models are (1) van Rooij et al. (2008) which searches every combination of sets of object matches, and (2) Wareham et al. (2011) which searches every combination

of statements. We disagree with this conclusion, since both of those algorithms have factorial complexity (the first in number of entities in the base and target, the second in the number of statements in the base and target). Those disagreements aside, we view Grootswagers' work as a valuable theoretical exploration of the computational properties of matching.

Like SME, the models discussed so far are focused on Marr's top two levels—the computational level (what does the system do) and the process level (what are the representations and algorithms). Some simulations of analogy also focus on Marr's implementation level—how one might carry out such computations in neural systems. Two prominent models of this type are LISA (Hummel and Holyoak, 1997) and DORA (Doumas et al. 2008). Both integrate retrieval with matching, and use a localist model of neural systems based on temporal binding. In both simulations, temporal binding imposes strong constraints on the number of relations that can be considered. LISA's initial representation scheme led to estimates of working memory involving at most two or three relations (Hummel & Holyoak, 1997, 2003). This is much lower than that suggested by the task constraints described in Section 3. Even if our estimates turn out to be too high, we note that ordinary everyday analogies often require more than three relations. A further problem is that, as pointed out by Eliasmith and Thagard (2001), the original version of LISA was incapable of handling matches involving higher-order relational structures, one of the benchmark properties of analogy. Even simple causal analogies require dealing with higher-order relations. To overcome this problem, LISA has been extended with a new mechanism, group units. Group units provide a representation of relational structure that is not subject to LISA's normal working memory limitations (Hummel et al. 2014). We view this evolution of LISA as recognizing that its prior working memory limitations were too restrictive to capture human abilities. Finally, LISA relies on serial ordering of activation of its units, and this activation order can potentially be manually specified by the experimenters for each example. Thus, LISA allows for a high degree of experimenter intervention in the internal operation of the system, making it highly tailorable.

DORA (Doumas et al., 2008) is closely related to LISA, but is aimed at modeling the discovery of relational representations during cognitive development. It represents binary relations via two unary predicates plus a linking connective. For example, `higher` is composed of the two unary predicates `high` and `low`, plus a unit that is active when both of them are active and which provides the connection between the two of them. DORA's process of connecting unary predicates into relations captures an important process in analogical development: namely, the relational shift, whereby children show the ability to match on object properties before the ability to match on relational structure (Gentner, 1988; Gentner & Rattermann, 1991; Gentner & Toupin, 1986; Richland et al., 2006).

DORA represents an important attempt to capture how relational representation begins, and it has been used to model a number of other developmental phenomena (Morrison et al., 2011). Still, there are some assumptions that can be questioned. First, DORA's constraint on the size of the representations seems unrealistic, even for children. In DORA, the capacity of WM is two or two-and-a-half role bindings, compared to LISA's original 4-5 role-filler bindings. Doumas and colleagues argue that this very low capacity fits with the finding that preschool children often fail to match on the basis of relational similarity. While this is an appealing feature of DORA, it is hard to see how to reconcile the assumption of a small fixed capacity with the many studies showing that children of the same age can show relational matching if they are led to have relational representations (Christie & Gentner, 2010; Gentner et al., 2011; Gentner & Rattermann, 1991; Kotovsky & Gentner, 1996; Loewenstein & Gentner, 2005; Son, Doumas & Goldstone, 2010).

DRAMA (Eliasmith & Thagard, 2001) constructs a localist network on top of fully distributed representations for base and target. DRAMA, unlike LISA, operates autonomously and can handle larger descriptions than LISA can—at least up to a dozen or so propositions. However, DRAMA currently has no mechanism for constructing candidate inferences—a critical benchmark feature for any model of

analogy. Further, the distributed representations it uses require a localist network instead to implement match hypotheses and structural consistency relationships.

Finally, we note that to date none of the neurally-inspired models has been used as a component in realistic-scale task models, as SME has, and all rely on hand-coded representations. Given their focus on attempting to work within constraints of biology as they are currently understood, this seems very reasonable. However, exactly how neural systems carry out their computations is still a matter of much debate. The information and process level constraints, on the other hand, are clearer, and provide evidence about what is sufficient and even to some degree what is necessary to carry out a range of human tasks. We hope that exploring how these two sets of constraints can be reconciled will be mutually productive.

7 Discussion

We have argued that structure-mapping is a core cognitive mechanism, used in a vast range of processes, from categorization to learning and transfer to problem-solving, and in everyday reasoning as well as scientific discovery. This means, first, that a simulation of analogical matching must be able to handle the size and scale of representations that are likely to be used in human processing. Second, the analogical matcher should be able to operate in concert with other processes to capture the use of structure-mapping in a range of cognitive arenas (the Integration Constraint (Forbus, 2001)). This paper describes the changes we have made to SME to meet this challenge. We described five extensions to SME itself that enable it to deal with analogies found in human tasks and to integrate with other processes:

1. Greedy merge enables SME to rapidly construct one or two near-optimal global interpretations, making it a polynomial-time algorithm. This allows SME to operate successfully on

representations that most other existing models cannot handle, and that are closer to the full range of representations likely to be used in human cognition.

2. Incremental matching enables SME to operate in models where information is not all available at once. This enables SME to better model tasks such as problem-solving, where analogies are incrementally elaborated.
3. Ubiquitous predicates enable SME to model the varying degrees to which items may suggest alignment. By marking some predicates as ubiquitous, and thereby not sufficient evidence by themselves to suggest a match, SME can handle complex representations, such as those found in problem solving.
4. Structural evaluation of candidate inferences provide quick plausibility estimates, enabling SME to model aspects of plausibility judgments in analogical inference, including aspects of category induction.
5. Match filters, automatically imposed by task models, enable SME to model the ability to respond to task demands in matching.

The examples described in Section 3 demonstrate that SME can be used as a module in building realistic-scale models that capture broader aspects of human behavior. We view task constraints as extremely important in evaluating cognitive models, since they bear directly on their ability to explain how people use comparison.

We see two main directions for future research. First, the large-scale simulations outlined in Section 3 are only a start: Exploring the roles of analogy and similarity in a broader range of cognitive processes via realistic-scale simulation is an enterprise that is just beginning. By making available a robust model

of analogical matching, we hope we can encourage others to join us in these investigations[17]. Second, the task demands on analogical matching pose a tough challenge for neural modeling, but such models must be constructed if we are to understand the phenomenon at all three of Marr's levels (computational, process, and implementation).

8 Acknowledgements

We thank Joe Blass, Morteza Dehghani, John Everett, Ben Jee, Mark Keane, Matthew Klenk, and Kate Lockwood for help with both experiments and the manuscript. This research was supported by the Office of Naval Research, the Defense Advanced Research Projects Agency, the Air Force Office of Scientific Research, and by an NSF Science of Learning Center grant, for the Spatial Intelligence and Learning Center.

9 References

Anderson, J. R. (2007) *How Can the Human Mind Occur in the Physical Universe?* New York: Oxford University Press.

Baddeley, A. (2012). Working Memory: Theories, Models, and Controversies. *Annual Review Psychology*, 63:1-29.

Barbella, D. & Forbus, K. (2011). Analogical Dialogue Acts: Supporting Learning by Reading Analogies in Instructional Texts. *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence* (AAAI 2011), San Francisco, CA.

Baron, J., & Spranca, M. (1997). Protected values. *Organizational Behavior and Human Decision Processes,* 70, pp. 1–16.

Boroditsky, L. (2000). Metaphoric structuring: Understanding time through spatial metaphors. *Cognition, 75* (1), 1-27.

Bowdle, B., & Gentner, D. (1997). Informativity and asymmetry in comparisons. *Cognitive Psychology*, 34, 244-286.

[17] The source code for this version of SME will be made available on publication of this article as well. There is already a system available, *CaseMapper*, which runs under Windows, Linux, and Macs which provides a cognitive-scientist friendly interface for using SME and MAC/FAC, as well as a number of classic examples.

Burstein, M. (1986). Concept formation by incremental analogical reasoning and debugging. In R.S. Michalski, J.G.Carbonell & J.M. Mitchell (Eds.), *Machine Learning II: An Artificial Intelligence Approach*. Los Altos, CA: Morgan-Kaufmann.

Cassimatis, N., Bello, P., & Langley, P. (2008) Ability, breadth, and parsimony in computational models of higher-order cognition. *Cognitive Science,* 32,1304-1322.

Christie, S. & Gentner, D. (2010). Where hypotheses come from: Learning new relations by structural alignment. *Journal of Cognition and Development*, 11 (3). 356-373.

Clement, C. A. & Gentner, D. (1991). Systematicity as a selection constraint in analogical mapping. *Cognitive Science*, 15, 89-132.

Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2009) *Introduction to Algorithms, 3rd edition*. MIT Press.

Cowan, N. (2001). The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences,* 24: 1–185.

Day, S. & Gentner, D. (2007). Nonintentional analogical inference in text comprehension. *Memory and Cognition*, 35, 39-49.

Dehaene, S., Izard, V., Pica, P., & Spelke, E. (2006). Core knowledge of geometry in an Amazonian indigene group. *Science*, 311, 381-384.

Dehghani, M., Tomai, E., Forbus, K., Iliev, R., & Klenk, M. (2008a). MoralDM: A computational model of moral decision-making. In B. C. Love, K. McRae, & V. M. Sloutsky (Eds.), *Proceedings of the 30th Annual Conference of the Cognitive Science Society.* Washington, D.C.: Cognitive Science Society.

Dehghani, M., Tomai, E., Forbus, K., & Klenk, M. (2008b). An Integrated Reasoning Approach to Moral Decision-Making. *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI).* Chicago, IL.

Dehghani, M., Sachdeva, S., Ekhtiari, H., Gentner D. & Forbus, K. (2009). The role of cultural narratives in moral decision making. In N.A. Taatgen & H. van Rijn (Eds.), *Proceedings of the 31st Annual Conference of the Cognitive Science Society.* Cognitive Science Society.

Doumas, L. A. A., Hummel, J. E., & Sandhofer, C. M. (2008). A theory of the discovery and predication of relational concepts. *Psychological Review*, 115, 1-43.

Eliasmith, C., & Thagard, P. (2001). Integrating structure and meaning: A distributed model of analogical mapping. *Cognitive Science*, *25*(2), 245-286.

Ericsson, K. A. & Kintsch, W. (1995). Long-term working memory. *Psychological Review*, *102*(2), 211.

Evans, T., (1968). A program for the solution of a class of geometric-analogy intelligence-test questions, In Minsky, M. (Ed.), *Semantic Information Processing*. MIT Press.

Falkenhainer, B. (1987). An examination of the third stage in the analogy process: Verification-based analogical learning. In *Proceedings of IJCAI-87*. Los Altos: Morgan-Kaufmann.

Falkenhainer, B. (1990). A unified approach to explanation and theory formation. In Shrager, J. & Langley, P. (Eds.), *Computational Models of Scientific Discovery and Theory Formation*. San Mateo, CA: Morgan Kaufmann.

Falkenhainer, B., Forbus, K.D., & Gentner, D. (1986). The Structure-Mapping Engine. *Proceedings of AAAI-86*. Los Altos, CA: Morgan-Kaufmann.

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

Ferguson, R. W. (2003). Mapping self-similar structure: Commutative expressions in structure mapping. *Proceedings of the 25th Annual Conference of the Cognitive Science Society*.

Forbus, K. D. (1984). Qualitative process theory. *Artificial intelligence*, *24*(1), 85-168.

Forbus, K. (2001). Exploring analogy in the large. In Gentner, D., Holyoak, K., and Kokinov, B. (Eds.) *Analogy: Perspectives from Cognitive Science*, pp 25-38. MIT Press.

Forbus, K. & de Kleer, J. (1993). *Building Problem Solvers*, MIT Press.

Forbus, K. D., Ferguson, R. W., & Gentner, D. (1994). Incremental structure-mapping. *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, 313-318. Hillsdale, NJ: Lawrence Erlbaum Associates.

Forbus, K., Gentner, D., & Law, K. (1995). MAC/FAC: A model of similarity-based retrieval. *Cognitive Science*, 19, 141-205.

Forbus, K. D., Gentner, D., Everett, J. O., & Wu, M. (1997). Towards a computational model of evaluating and using analogical inferences. *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society*, 229-234.

Forbus, K., Klenk, M., & Hinrichs, T. (2009). Companion cognitive systems: Design goals and lessons learned so far. *IEEE Intelligent Systems*, 24, 36-46.

Forbus, K.D. & Oblinger, D. (1990). Making SME greedy and pragmatic. *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*. Hillsdale: Erlbaum.

Forbus, K., Usher, J., & Chapman, V. (2003). Qualitative Spatial Reasoning about Sketch Maps. *Proceedings of the Intelligence User Interfaces Conference, 2003*, January: Miami, Florida.

Forbus, K.D., Whalley, P., Everett, J., Ureel, L., Brokowski, M., Baher, J. & Kuehne, S. (1999) CyclePad: An articulate virtual laboratory for engineering thermodynamics. *Artificial Intelligence*. 114, 297-347.

Forbus, K., Mostek, T. & Ferguson, R. (2002). An analogy ontology for integrating analogical processing and first-principles reasoning. *Proceedings of IAAI-02*, July.

Forbus, K., Usher, J., Lovett, A., Lockwood, K., & Wetzel, J. (2011). CogSketch: Sketch understanding for Cognitive Science Research and for Education. *Topics in Cognitive Science*. 3(4), pp 648-666.

French, R.M. (1995) *The subtlety of similarity*. MIT Press, Cambridge, MA.

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

Gentner, D. (1988). Metaphor as structure mapping: The relational shift. *Child Development*, 59, 47-59.

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.

Gentner, D. (2003). Why we're so smart. In D. Gentner and S. Goldin-Meadow (Eds.), *Language in mind: Advances in the study of language and thought* (pp.195-235). Cambridge, MA: MIT Press.

Gentner, D. (2010). Bootstrapping the mind: Analogical processes and symbol systems. *Cognitive Science*, 34 (5). 752-775.

Gentner, D., Anggoro, F. K., & Klibanoff, R. S. (2011). Structure-mapping and relational language support children's learning of relational categories. *Child Development*, 82(4). 1173-1188.

Gentner, D. and Boronat, C. (1991). Metaphors are (sometimes) processed as domain mappings. Paper presented at the Symposium on Metaphor and Conceptual Change, Meeting of the Cognitive Science Society of Chicago.

Gentner, D., Bowdle, B., Wolff, P., & Boronat, C. (2001). Metaphor is like analogy. In Gentner, D., Holyoak, K. J., & Kokinov, B. N. (Eds.), The analogical mind: Perspectives from cognitive science (pp. 199-253). Cambridge, MA: MIT Press.

Gentner, D., Brem, S., Ferguson, R. W., Markman, A. B., Levidow, B. B., Wolff, P., & Forbus, K. D. (1997). Analogical reasoning and conceptual change: A case study of Johannes Kepler. *The Journal of the Learning Sciences*, 6(1), 3-40.

Gentner, D., & Forbus, K. (2011). Computational models of analogy. *WIREs Cognitive Science*, 2. 266-276.

Gentner, D., & Gunn, V. (2001). Structural alignment facilitates the noticing of differences. *Memory and Cognition*, 29, 565-577.

Gentner, D., Holyoak, K. J., & Kokinov, B. (Eds.). (2001). *The analogical mind: Perspectives from cognitive science.* Cambridge, MA: MIT Press.

Gentner, D., Imai, M., & Boroditsky, L. (2002). As time goes by: Evidence for two systems in processing space-time metaphors. *Language and Cognitive Processes*, 17, 537-565.

Gentner, D., & Kurtz, K. (2006). Relations, objects, and the composition of analogies. *Cognitive Science*, 30, 609-642.

Gentner, D., Loewenstein, J., Thompson, L., & Forbus, K. (2009) Reviving inert knowledge: Analogical abstraction supports relational retrieval of past events. *Cognitive Science,* 3, 1343-1382.

Gentner, D., & Markman, A. B. (1994). Structural alignment in comparison: No difference without similarity. *Psychological Science, 5*, 152-158.

Gentner, D., & Markman, A. B. (1995). Analogy-based reasoning in connectionism. In M. Arbib (Ed.), *The handbook of brain theory and neural networks* (pp. 91-93). Cambridge, MA: MIT Press.

Gentner, D., & Markman, A.B. (1997). Reasoning and learning by analogy: Introduction. *American Psychologist*, 52, 32-34.

Gentner, D., & Markman, A. B. (2006). Defining structural similarity. *Journal of Cognitive Science*, 6, 1-20

Gentner, D. & Rattermann, M.J. (1991). Language and the career of similarity. In S.A.Gelman & J.P. Byrnes (Eds.), *Perspectives on thought and language: Interrelations in development*. (pp 225-277), London: Cambridge University Press.

Gentner, D., Rattermann, M.J., & Forbus, K.D. (1993). The roles of similarity in transfer: Separating retrievability from inferential soundness. *Cognitive Psychology* 25, 524-575.

Gentner, D., & Toupin, C. (1986). Systematicity and surface similarity in the development of analogy. *Cognitive Science*, 10, 277-300.

Gernsbacher, M. A. (1997). Two decades of structure-building. *Discourse Processes, 23*, 265-304

Givón, T. (1992). The grammar of referential coherence as mental processing instructions. *Linguistics*, *30*(1), 5-56.

Goldstone, R. L. (1994). Similarity, interactive activation, and mapping. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 20(1), 3-28.

Goldstone, R. L., & Medin, D. L. (1994). Time course of comparison. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 20 (1), 29–50.

Goldstone, R. L., Medin, D. L., & Gentner, D. (1991). Relational similarity and the non-independence of features in similarity judgment. *Cognitive Psychology*, *23,* 222-264.

Grootswagers, T. (2013) Having your cake and eating it too; Towards a fast and optimal method for analogy derivation. MSc Thesis, Department of AI and Psychology, Radboud University Nijmegen, The Netherlands.

Heit, E., & Rubinstein, J. (1994). Similarity and property effects in inductive reasoning. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 20, 411-422.

Hofstader, D., & Mitchell, M. (1995). The CopyCat Project: A model of mental fluidity and analogy-making. In Hofstader, D. and the Fluid Analogies Research Group (Eds.), *Fluid Concepts and Creative Analogies*. Basic Books. Chapter 5: 205-267.

Hofstader, D. , & Sander, E. (2013) *Surfaces and Essences: Analogy as the Fuel and Fire of Thinking*. Basic Books.

Holyoak, K.J. (1985). The pragmatics of analogical transfer. In G.H. Bower (Ed.), *The psychology of learning and motivation* , Vol 1,pp. 59-87. New York: Academic Press.

Holyoak, K. J., & Hummel, J. E. (2000). The proper treatment of symbols in a connectionist architecture. *Cognitive dynamics: Conceptual change in humans and machines*, 229-263.

Holyoak, K. J., & Koh, K. (1987). Surface and structural similarity in analogical transfer. *Memory & Cognition*, *15*(4), 332-340.

Holyoak, K.J., & Thagard, P. (1989). Analogical mapping by constraint satisfaction. *Cognitive Science*, 13(3), 295-355.

Holyoak, K.J. & Thagard, P. (1995). *Mental leaps: Analogy in creative thought*. Cambridge, MA: The MIT Press.

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

Hummel, J.E., Licato, J., & Bringsjord, S. (2014). Analogy, Explanation, and Proof. *Frontiers in Human Neuroscience*, 06 November 2014. doi: 10.3389/fnhum.2014.00867

Keane, M.T., & Brayshaw, M. (1988). The Incremental Analogy Machine: A computational model of analogy. In D. Sleeman (Ed.), *Third European Working Session on Machine Learning*. London: Pitman/San Mateo, CA:Morgan Kaufmann.

Klenk, M. & Forbus, K. (2007). Cognitive modeling of analogy events in physics problem solving from examples. In D. S. McNamara & J. G. Trafton (Eds.), *Proceedings of the Twenty-ninth Annual Meeting of the Cognitive Science Society*. Nashville, TN: Cognitive Science Society.

Klenk, M., & Forbus, K. (2009). Analogical model formulation for transfer learning in AP Physics. *Artificial Intelligence*, 173, 1615-1638.

Klenk, M., & Forbus, K. (2013). Exploiting persistent mappings in cross-domain analogical learning of physical domains. *Artificial Intelligence* 195, 398-417.

Kokinov, B. & French, R. M. (2003) . Computational Models of Analogy-making. In Nadel, L. (Ed.) *Encyclopedia of Cognitive Science*. Vol. 1, pp.113 - 118. London: Nature Publishing Group.

Kokinov, B., & Petrov, A. (2001). Integrating memory and reasoning in analogy-making: The AMBR model. In Gentner, D., Holyoak, K., & Kokinov, B. (Eds.) *The Analogical Mind: Perspectives From Cognitive Science*. Cambridge, MA: MIT Press, 161-196.

Kotovsky, L., & Gentner, D. (1996). Comparison and categorization in the development of relational similarity. *Child Development, 67,* 2797-2822.

Krawczyk, D., Holyoak, K., & Hummel, J. (2004). Structural constraints and object similarity in analogical mapping and inference*. Thinking and Reasoning*, 10, 85-104.

Krawczyk, D. C., Holyoak, K. J., & Hummel, J. E. (2005). The one-to-one constraint in analogical mapping and inference. *Cognitive Science*, 29, 797 – 806.

Kuehne, S., Forbus, K., Gentner, D., & Quinn, B. (2000). SEQL: Category learning as progressive abstraction using structure mapping. In L. R. Gleitman and J. K. Joshi (Eds.) *Proceedings of the 22nd Annual Meeting of the Cognitive Science Society*, pp. 770-775. Austin, TX: Cognitive Science Society.

Kurtz, K. J., & Gentner, D. (2013). Detecting anomalous features in complex stimuli: The role of structured comparison*. Journal of Experimental Psychology: Applied*, 19(3), 219-232.

Laird, J. E. (2012). *The Soar Cognitive Architecture*. MIT Press, Cambridge, MA.

Lakoff, G., & Johnson, M. (1980). *Metaphors we live by*. Chicago, IL: University of Chicago Press.

Larkey, L., & Love, B. (2003) CAB: Connectionist Analogy Builder. *Cognitive Science* 27:781-794.

Lassaline, M.E. (1996). Structural alignment in induction and similarity. *Journal of Experimental Psychology: Learning, Memory, and Cognition,* 22, 754-770.

Lee, H. S. & Holyoak, K. J. (2008). The role of causal models in analogical inference. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 34, No. 5, 1111–1122.

Loewenstein, J., & Gentner, D. (2005). Relational language and the development of relational mapping. *Cognitive Psychology*, 50, 315-353.

Lovett, A., Tomai, E., Forbus, K. & Usher, J. (2009). Solving geometric analogy problems through two-stage analogical mapping. *Cognitive Science* 33(7), 1192-1231.

Lovett, A., & Forbus, K. (2011) Cultural commonalities and differences in spatial problem solving: A computational analysis. *Cognition* 121, pp. 281-287.

Lovett, A., & Forbus, K. (2012). Modeling multiple strategies for solving geometric analogy problems. In N. Miyake, D. Peebles, & R. P. Cooper (Eds.) *Proceedings of the Thirty-fourth Annual Meeting of the Cognitive Science Society.* (pp. 34-35). Sapporo, Japan: Cognitive Science Society.

Luce, R. D. (1986). *Response times: Their role in inferring elementary mental organization*. New York: Oxford University Press.

Markman, A.B. (1997). Constraints on analogical inference. *Cognitive Science*, 21(4), 373-418.

Markman, A. B. (1999). *Knowledge representation*. Mahwah, NJ: Lawrence Erlbaum Associates.

Markman, A. B., & Gentner, D. (1993a). Splitting the differences: A structural alignment view of similarity. *Journal of Memory and Language, 32*, 517-535.

Markman, A. B., & Gentner, D. (1993b). Structural alignment during similarity comparisons. *Cognitive Psychology, 25*, 431-467.

Markman, A. B., & Gentner, D. (1996). Commonalities and differences in similarity comparisons. *Memory & Cognition*, 24(2), 235-249.

Markman, A. B., & Gentner, D. (2000). Structure-mapping in the comparison process. *American Journal of Psychology, 113*, 501-538.

Markman, A.B., & Medin, D.L. (2002). Decision Making. In D.L. Medin & H. Pashler (Eds.) *Stevens Handbook of Experimental Psychology (3rd Edition), Volume 2*. (pp. 413-466). New York: John Wiley and Sons.

Marr, D. (1983). *Vision*. W. H. Freeman.

Matuszek, C., Cabral, J., Witbrock, M. J., & DeOliveira, J. (2006). An introduction to the syntax and content ofCyc. In *AAAI Spring Symposium: Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*, 44-49.

McLure, M., Friedman, S., & Forbus, K. (2010). Combining progressive alignment and near-misses to learn concepts from sketches. *Proceedings of the 24th International Workshop on Qualitative Reasoning*. Portland, OR

Mitchell, M. (1993) *Analogy-making as perception: A computer model*. MIT Press, Cambridge, MA.

Morrison, R. G., Doumas, L. A., & Richland, L. E. (2011). A computational account of children's analogical reasoning: balancing inhibitory control in working memory and relational representation. *Developmental Science*, *14*(3), 516-529.

Mostek, T., Forbus, K, & Meverden, C. (2000). Dynamic case creation and expansion for analogical reasoning. *Proceedings of AAAI-2000,* pp. 323-329. Austin, TX.

Osherson, D.N., Smith, E.E., Wilkie, O., Lopez, A., & Shafir, E. (1990). Category-based induction. *Psychological Review*, 97, 185-200.

Ouyang, T. Y., & Forbus, K. D. (2006). Strategy variations in analogical problem solving. In *Proceedings of the National Conference on Artificial Intelligence*, 21, 446. Menlo Park, CA.

Pica, P., Lemer, C., Izard, V., & Dehaene, S. (2004). Exact and approximate arithmetic in an Amazonian indigene group. *Science*, *306*, 499-503.

Posner, M. I., & Mitchell, R. F. (1967). Chronometric analysis of classification. *Psychological Review*, 74 (5), 392–409.

Prasad, L. (2007). *Poetics of Conduct*. Columbia University Press, New York.

Richland, L. E., Morrison, R. G., & Holyoak, K. J. (2006). Children's development of analogical reasoning: Insights from scene analogy problems. *Journal of Experimental Child Psychology*, *94*, 249-271.

Richland, L. E., Zur, O., & Holyoak, K. J. (2007). Cognitive supports for analogies in the mathematics classroom. *Science*, 316, 1128.

Ritov, I. & Baron, J. 1999. Protected values and omission bias. *Organizational Behavior and Human Decision Processes*, 79, 79-94.

Sagi, E., Gentner, D. & Lovett, A. (2012). What difference reveals about similarity. *Cognitive Science*, 36 (6). 1019-1050.

Skorstad, J., Falkenhainer, B., & Gentner, D. (1987). Analogical processing: A simulation and empirical corroboration. *Proceedings of the Meeting of the American Association for Artificial Intelligence*, 322-326

Spellman, B. A. & Holyoak, K.J. (1996). Pragmatics in analogical mapping. *Cognitive Psychology*, 31, 307-346.

Son, J.Y., Doumas, L.A.A., & Goldstone, R.L. (2010). When do words promote analogical transfer? *Journal of Problem Solving*, 3, 52-92.

Thibodeau P.H., & Boroditsky L. (2011) Metaphors we think with: The role of metaphor in reasoning. *Plos ONE* 6(2): e16782.

Thibodeau, P., & Durgin, F. H. (2008). Productive figurative communication: Conventional metaphors facilitate the comprehension of related novel metaphors. *Journal of Memory and Language*, *58*(2), 521-540.

Thibodeau, P.H., & Durgin, F.H. (2011). Metaphor aptness and conventionality: A processing fluency account. *Metaphor and Symbol*. 26, 206-226.

Tomai, E. & Forbus, K. (2009). EA NLU: Practical Language Understanding for Cognitive Modeling. *Proceedings of the 22nd International Florida Artificial Intelligence Research Society Conference*. Sanibel Island, Florida.

Trench, M., & Minervino, R. A. (2014). The role of surface similarity in analogical retrieval: Bridging the gap between the naturalistic and the experimental traditions. *Cognitive Science*, 39, 1292-1319.

van Rooij, I., Evans, P., Muller, M., Gedge, J., & Wareham, T. (2008). Identifying sources of intractability in cognitive models: An illustration using analogical structure mapping. In B. C. Love, K. McRae, & V. M.

Sloutsky (Eds.), *Proceedings of the 30th Annual Conference of the Cognitive Science Society* (pp. 915-920). Washington, D.C.: Cognitive Science Society.

VanLehn, K. (1998). Analogy events: How examples are used during problem solving. *Cognitive Science*, 22(19), 347-388.

VanLehn, K. & Jones, R. (1993). Better learners use analogical problem solving sparingly. *Proceedings of the 10th International Conference on Machine Learning*

Veloso, M., & Carbonell, J. (1993). Derivational analogy in PRODIGY: Automating case acquisitions, storage, and utilization. *Machine Learning*, 10:249-278

Wareham, T., Evans, P., & Van Rooij, I., (2011). What does (and doesn't) make analogical problem solving easy? A complexity-theoretic perspective. *The Journal of Problem Solving*, 3(2), 30-71.

Weber, E., & Hsee, C. (1999). Cross-cultural differences in risk perception, but cross-cultural similarities in attitudes towards perceived risk. *Management Science*, 44, 9.

Winston, P. (1980). Learning and reasoning by analogy. *Communications of the ACM*, 23, 689-703.

Winston, P. (1982). Learning new principles from precedents and exercises. *Artificial Intelligence* 19, 321-350.

Wolff, P., & Gentner, D. (2011). Structure-mapping in metaphor comprehension. *Cognitive Science*, 35. 1456-1488.

10 Appendix A: The SME algorithm, version 4

This appendix describes the algorithms in Version 4 of the Structure Mapping Engine, including the details behind the theoretical complexity analysis whose results were summarized in Section 5. We begin with its inputs, outputs, and operations, then provide a step-by-step description of the algorithm, including the computational complexity of each step. Finally, we combine the complexity results for each step to yield the overall complexity.

We will continue to use Lisp syntax for representations, and infix mathematical syntax for procedures and algorithms, to make them easy to distinguish.

10.1 Inputs, outputs, and operations

The fundamental operation of SME is Comparison. The Comparison operation takes the following inputs:

- *Base:* A description, consisting of a set of statements in a structured representation language.
- *Target*: A description, also consisting of a set of statements in a structured representation language.
- *Output constraints*: These consist of two parameters:
    - The *Score Cutoff* indicates the range of mappings that SME should consider. A score cutoff of 0.8, for example, means that mappings whose structural evaluation fell below 20% of the best mapping would be pruned.
    - The *output limit* indicates the maximum number of mappings that SME should create. An output limit of three, for example, means that SME will never produce more than three mappings. This is its default setting.
- *Filters*: An optional set of additional constraints that mappings must satisfy. The set of filters allowed is described in Section 4.5.

As discussed above, the output constraints and filters provide ways for task models to automatically tune the matcher. The output constraints implement capacity limits, while filters provide ways of guiding the matcher using task-specific criteria. As described earlier, the language for describing filters is tightly constrained, since allowing them to be arbitrary computations violates the spirit of structure-mapping.

The Comparison operation produces as output a set of mappings. A mapping contains

- *Correspondences:* The alignment between base and target represented by this mapping, expressed as a set of pairings of items (statements, entities, and predicates) in the base with items in the target.
- *Structural evaluation*: A numerical estimate of the overall quality of the match.
- *Candidate inferences*: A set of analogical inferences suggested by this alignment, including structural evaluations of the degree to which the mapping supports them and how extrapolative they are (Section 4.4). Candidate inferences are computed by default from base to target, but reverse candidate inferences, from target to base, can also be computed on demand. Reverse candidate inferences are used when reasoning about differences, for example.

Mappings and candidate inferences are now first-class entities, i.e., they can be referred to by systems using SME. For example, a problem solver might have the explicit goal of extending a mapping or verifying a candidate inference. We believe that this is psychologically plausible, given the human ability to reason about analogies. An *analogy ontology* (Forbus et al. 2002) has been defined to enable analogical operations to smoothly interoperate with other kinds of reasoning when needed. However, SME itself only relies on very basic assumptions about the representation system it is being used with. The representation system must identify relations, attributes, and functions, and provide information

about their arity. If minimal ascension is to be used, the representation system must also supply superordinate relationships, which can be used by an optional procedure to evaluate it.

In addition to the Comparison operation, SME now supports two additional operations: *Extend* and *Remap*. The Extend operation is used to extend the results of a comparison when new information is added to the base or to the target. (We do not permit information to be removed from either base or target – if information is removed, a match must be started again from scratch. This greatly simplifies the algorithms.) Correspondences involving the new information are created and the existing mappings extended with this information as appropriate. Incremental mapping can require backtracking, since misleading early information can lead to mappings that are suboptimal when additional information becomes available. The Remap operation reconstructs the mappings, providing the same results that would have been found if the current state of the base and target had been available originally. As described below, both Extend and Remap algorithms are carefully organized so as to preserve previously computed results that are still valid, for maximal efficiency. We believe this is psychologically plausible – essentially, the initial stages remain unguided and parallel, while task-specific influences can operate at the later, serial phase of processing.

10.2 The SME Algorithm, Step by Step

Extend and Remap are almost entirely defined in terms of the same operations as Comparison, hence it simplest for exposition to describe Comparison and note along the way how Extend and Remap work. For the purposes of analyzing complexity, it is useful to decompose the phases from Section 2 further into the following steps:

Phase One: Constructing the Match Hypothesis Network

1. Finding match hypotheses. Local correspondences (match hypotheses) between items in the base and target are proposed in parallel.

Phase Two: Parallel Evaluation of the Match Hypothesis Network

2. Structural consistency filtering. Match hypotheses that violate structural consistency are removed from further consideration.
3. Structural evaluation propagation. Structural estimates of match quality are constructed for each correspondence based on a trickle-down algorithm that provides a local implementation of systematicity.

Phase Three: Constructing Mappings

4. Kernel creation. A kernel is a potential seed of a mapping. Identifying them and scoring them is the first step towards creating a global construal of a match.
5. (Optional) Filtering. Irrelevant kernels are filtered using automatically imposed constraints from task models, using strictly local criteria.
6. Greedy merge. A small number of global mappings are constructed from the kernels.
7. Candidate inference creation and evaluation. Analogical inferences are generated for each mapping and evaluated in structural terms.

We next describe how each step works in detail.

10.2.1 Finding Match Hypotheses

Match hypotheses are potential correspondences. A match hypothesis links an item in the base to an item in the target. Conceptually, this stage creates, in parallel, match hypotheses between all pairs of items in the base and target that could correspond. (By "item" we mean expressions, entities and functors.) The result is a network out of which mappings are constructed.

The structure of the match hypothesis network is motivated by the constraints of structure-mapping. The parallel connectivity constraint ties the structural consistency of a match hypothesis to the existence of match hypotheses for the corresponding arguments of the statements that it aligns. For

any statement S, we use `arguments(S)` to refer to its arguments. Similarly, given an item A, we use `parents(A)` to refer to the set of statements in which it appears as an argument. For example, in

```
S1: (contains (bloodstream animal12) seretonin)
arguments(S1) = {(bloodstream animal12), seretonin}
S1 ∈ parents(seretonin)
```

Parent and argument relationships are similarly defined for match hypotheses, based on the statements they align. That is, MH1 is in arguments(MH2) if the base and target items aligned by MH1 are corresponding arguments in the statements aligned by MH2, and conversely, MH2 is in parents(MH1). For example,

```
S2: (contains (bloodstream animal6) seretonin)
MH1: (bloodstream animal6) ↔ (bloodstream animal12)
MH2: S2 ↔ S1
MH1 ∈ arguments(MH2)
MH2 ∈ parents(MH1)
```

It is also useful to refer to the predicate, function, or connective involved in a statement. We use the function `functor` for this purpose. Thus

```
contains = functor(s2)
bloodstream = functor((bloodstream animal6))
```

A match hypothesis that has no parents is called a *root* match hypothesis. Similarly, statements in descriptions that are not themselves arguments of another statement are called root statements. We simply use the word "root" when context makes it clear which type we are talking about.

Here is the match hypothesis construction algorithm:

Match Hypothesis Network construction

Inputs: Base `B`, Target `T`, (optional) procedure `locally-alignable?`

Outputs: Network of match hypotheses `MHS`

1. *Initial network construction*: For each `Bi ∈ Expressions(Base)` and `Ti ∈ Expressions(Target)` such that `Functor(Bi) = Functor(Ti) & not(Ubiquitous(Functor(Bi)))`,

1.1. Create `MH(Bi, Ti)`

1.2. `Push(MH(Bi,Ti), MHS)`
1.3. For each corresponding pair of arguments `Bj`, `Tk` in `MH(Bi,Ti)`, `push(<Bj, Tk>, Queue)`
2. *Network growth:* Until `Queue` is empty, process each `<Bj, Tk>` as follows:
2.1. If `functor(Bj) = functor(Tk) & not(Ubiquitous(Functor(Bi)))`ignore.
;; Step 1 already handled this pair, but if ubiquitous, it didn't – part of larger structure, so
;; *worth binding*
2.2. If `Bj`, `Tk` are both entities, create `MH(Bj,Tk)`, `push(MH(Bj,Tk),MHS)`
2.3. If either `Bj` or `Tk` is an entity, ignore.
2.4. If `functor(Bj)` and `functor(Tk)` are functions and *identical-functions* constraint is false,
2.4.1. Create `MH(Bj,Tk)`, `push(MH(Bj,Tk),MHS)`
2.4.2. Push`(MH(Bj,Tk), Queue)`
2.5. If `locally-alignable?` is supplied and `locally-alignable?(Bj,Tk,MH(Bi,Ti))`,
2.5.1. Create `MH(Bj,Tk)`, `push(MH(Bj,Tk),MHS)`
2.5.2. Push`(MH(Bj,Tk), Queue)`

An initial set of match hypotheses is constructed based on purely local, structural grounds (Step 1), and then extended based on placing arguments of potentially corresponding statements into alignment (Step 2). The contents of the initial set of match hypotheses and the growth of the network are governed by the tiered identicality constraint[18]. Recall that tiered identicality by default requires relations to match identically. The initial set of match hypotheses is created by finding all pairs of statements Bi $\in$ Base and Ti $\in$ Target such that

`functor(Bi) = functor(Ti)`

and, if the functor is not a ubiquitous predicate, creating a match hypothesis for it.

This set is grown by propagating outward from the initial set, looking for matches between corresponding arguments of the statements aligned in the initial match hypothesis set. Weaker criteria are used for matching when alignment is suggested by other match hypotheses. Statements whose functors are ubiquitous predicates are matched, since the shared parent provides a reason to do so: not

[18] See Ferguson (2003) for the special case of creating match hypotheses over pairs of commutative expressions, such as matched group or set expressions. In this case, SME delays creating one-to-one matches until resolved by other non-commutative expression matches, during the merge process. It represents the set of potential matches between commutative expressions in a compact matrix called a *commutatives table*.

including it would violate parallel connectivity. By default, matches between non-identical functions are allowed when they would support a larger match, since these semantically correspond to cross-dimensional differences. Such cross-dimensional matches are not allowed if the identical-functions filter is in force. The optional procedure `locally-alignable?`, if supplied, implements the non-default cases of tiered identicality. For example, if minimal ascension is used, this procedure must use an appropriate knowledge representation system to ascertain if there is a close-enough common superordinate. All of the examples in this paper and the experiments in Section 3 use strict identicality or minimal ascension.

10.2.1.1 Complexity of Finding Match Hypotheses

The default test using tiered identicality is identity of functors, which can be considered a unit-time operation. Other techniques like similarity tables (Holyoak and Thagard, 1989) and minimal ascension (Falkenhainer, 1987) satisfy the unit-time operation assumption. We ignore potentially more complex tests here, since they lie outside the spirit of structure-mapping.

Finding the initial set of match hypotheses requires comparing every statement in the base with every statement in the target to see if their functors are identical, and if so, creating a match hypothesis. On a serial machine, this step is bounded above by $O(n^2)$. Since each comparison is independent, they can be done in parallel in unit time if there are at least n^2 processors available. We have found it useful to pre-sort expressions in the base and target into bins by functor, so that we can simply create match hypotheses between expressions in corresponding bins. In the worst case, where every expression had the same functor, this would still be $O(n^2)$, but in the best case, where every statement within the base and target had a different functor, this would reduce to the complexity of the sort, i.e. $O(n\log(n))$.

The filling out of the match hypothesis forest by generating match hypotheses between corresponding arguments (when possible), is a function of the number of match hypotheses found in the initial step

and the depth of each tree of arguments. We ignore the depth-related costs for two reasons. First, argument trees are typically much smaller compared to the total number of statements in the description. Second, all match hypotheses between statements with identical functors have already been found in the initial step, so only entities plus statements involving non-identical functions or ubiquitous predicates will cause new match hypotheses to be created. This means the complexity of the filling in the match hypothesis forest is bounded by $O(n^2)$.

10.2.2 Structural Consistency Filtering

The collection of match hypotheses as generated provides the threads out of which mappings are woven. However, at this stage in processing it is still inchoate. Local application of structure-mapping constraints prunes all match hypotheses that could never be part of a consistent mapping. Such hypotheses are marked as structurally inconsistent by this stage of processing and subsequently ignored.

Recall that parallel connectivity states that the arguments of a pair of aligned statements must also be aligned. Match hypotheses that violate parallel connectivity are said to be *incomplete*. The first part of detecting incomplete match hypotheses occurs during the construction of the match hypothesis forest, since failure during the attempt to align the arguments of two matching statements indicates that that match hypothesis is incomplete. However, parallel connectivity also implies that all parents of that match hypothesis are also incomplete. This implication is enforced by propagating incompleteness markers upwards to all parents from incomplete match hypotheses once the forest has been finished.

The 1:1 constraint is enforced by propagating information through the argument relations about structural dependencies of match hypotheses. The `descendants` of a match hypothesis is the set of match hypotheses that it structurally depends upon. For example,

```
MH1: (above triangle32 circle6) ↔ (above triangle18 circle3)
MH2: triangle32 ↔ triangle18
```

```
MH3: circle6 ↔ circle3
MH4: above ↔ above
descendants(MH1) = {MH2, MH3, MH4}
```

`descendants` is the transitive closure of the `arguments` relation. We use `descendants` to define the set of `nogoods` for a match hypothesis, i.e., those match hypotheses which, combined with it, would lead to a structurally inconsistent result[19].

We define nogoods recursively as follows:

```
nogoods(MH) = { MHi | MH ≠ MHi
                ∧ ([BaseItem(MH) = BaseItem(MHi)
                    ∨ TargetItem(MH) = TargetItem(MHi)]
                   ∨ [∃ MHj (MHj ∈ descendants(MH))
                       ∧ MHi ∈nogoods(MHj)])}
```

That is, two match hypotheses are together structurally inconsistent if either they directly map the same base item to different target items (or the same target item to different base items) or if corresponding descendants do.

Since descendants and nogoods are heavily used in creating mappings, we compute these sets explicitly and cache them with each match hypothesis. Structurally inconsistent match hypotheses are detected during this process, i.e., when the intersection of a match hypotheses' descendants and nogoods is non-empty.

To support incremental operation, the descendants and nogoods sets are recalculated whenever new match hypotheses are added to the forest. (The same calculation is used when a match is started, with every match hypothesis being new.) Perhaps surprisingly, the structural consistency computations are monotonic with respect to the addition of new items to the base and target. That is, the sets of descendants and nogoods can only grow, not shrink. This is easier to see if one remembers that

[19] The term nogoods is an analogy with truth maintenance systems, in which nogoods are either sets of inconsistent assumptions or clauses that generate such sets (Forbus & de Kleer, 1993).

statements can be added but not modified. This means for any statement its arguments remain constant. (Tweaking a representation to improve the match is carried out by adding redundant items to base or target, but these only give rise to new match hypotheses, rather than replacing or mutating existing ones.) Updates occur by propagating upwards from the lowest-order newly added match hypotheses. The descendants are simply the union of the descendants of the arguments, plus the match hypothesis between the corresponding functors (when the match hypothesis is an expression). The nogoods are simply the union of the nogoods of the arguments with the set of match hypotheses that directly conflict with it (i.e., that satisfy the first disjunct in the definition above).

10.2.2.1 Complexity of Structural Consistency Filtering

Suppose there are *m* match hypotheses. Organizing the propagation step as an iteration that proceeds from entity matches up through the `parents` relations can be done in such a way that each match hypothesis is processed exactly once, using standard tree traversal algorithms, hence this step is *O(m)*. Again we treat set operations as constant-time, since they can be implemented using bit vectors (or run-length encoded bit vectors) to minimize cost.

10.2.3 Structural Evaluation Propagation

Structural evaluation implements the systematicity preference. Ultimately, structural evaluation scores will be assigned to each mapping by adding up the scores of the match hypotheses that comprise the mapping. Structural evaluation needs to be done early, since its results are used in guiding the greedy merge process described below. Thus as soon as structural inconsistencies have been removed, a numerical propagation step is used to compute scores for each match hypothesis.

The score of a match hypothesis is computed in two parts. First, there is a local component, a starting score given to every match hypothesis. There are two parameters here: `same-functor` is the score given if the match hypothesis involves statements with identical functors or involves entities, and `different-functor` otherwise. The default values for same-functor and different-functor are $5x10^{-4}$

and $2x10^{-4}$ respectively. Note that their exact values do not matter, only their values in relation to each other. The second part of the score is computed by the *trickle-down rule*: Given `MHa` with parents `MHS`,

```
Score(MHa) ← Score(MHa)+ trickle-down * Sum(Scores(MHS)]
```

where `trickle-down` is a constant indicating the strength of the systematicity constraint. The default value for `trickle-down` is 8. Notice that this algorithm results in high scores for entity match hypotheses that support large structurally consistent matches. Thus the global preference for systematicity is computed by a propagation algorithm operating on local evidence.

In some cases, a match hypothesis can receive trickle-down from parents that are structurally inconsistent with each other. For example,

```
MH1: (above triangle32 circle6) ↔ (above triangle18 circle3)
MH2: (above triangle32 circle6) ↔ (above triangle18 square19)
MH3: triangle32 ↔ triangle18
```

Both `MH1` and `MH2` are parents of `MH3`, but a final mapping could not contain both of them. To avoid inflating the value of a match hypothesis during trickle-down, the algorithm greedily selects a match hypothesis' parents, beginning with the highest-scoring parent, and skipping over any parent that is structural inconsistent with parents already selected. It only applies trickle-down from these selected parents (in this example, either `MH1` or `MH2`).

Earlier versions of SME normalized the score of each node to be a maximum of 1.0. This proved to be problematic for large, deeply nested representations, since many of the lower-level nodes would max out to 1.0. Consequently, we eliminated this limit, and now normalize at the level of the mapping, as discussed below.

10.2.3.1 Complexity of Structural Evaluation Propagation

Local scores are initialized when match hypotheses are created, so the only cost is that of applying the trickle-down rule. On a serial machine, applying trickle-down can be done by iterating over the match

hypotheses, starting at the roots of the match hypothesis forest and working downwards. First, suppose each match hypothesis has only a single parent, so greedily selecting among parents is not a factor. Since this only requires processing each match hypothesis once, the complexity is *O(m)*. On a data-parallel machine, assuming at least *m* processors, the time required will be proportional to the maximum depth of expressions being processed. In the worst case this would be *O(m)*, in the (extremely unnatural) case when the base and target were single expressions with extremely deep nesting, i.e.,

```
(P (P (P … (P e)…)))
```

More typically, there is a small integer d that can be found as an upper bound on the depth of expressions, in which case the data-parallel time required will be constant independent of *m*.

Now, consider the case where match hypotheses have multiple parents. Each match's parents must be sorted by score, so that they can be greedily selected for trickle-down. If a match has p parents, this will require *O(plog(p))* time, however, *p* in our experience is always small, so we ignore the cost of this operation.

10.2.4 Kernel Creation

Kernels form an important intermediate representation in the creation of a mapping. They represent the place where we believe that the shift from parallel processing to serial processing in analogical matching occurs, and where we suspect that penetrability can begin to occur.

A kernel consists of the union of a structurally consistent root match hypothesis with its descendants. The structural evaluation score of a kernel is the sum of the structural evaluation scores for the match hypotheses that comprise it. The nogoods of a kernel is the union of the nogoods of the match hypotheses that comprise it.

Kernels are found by the following algorithm:

1. For each root `R` in the match hypothesis forest,
   a. If `R` is structurally consistent, then create kernel `K` consisting of `R` ∪ `descendants(R)`.
   b. Otherwise, recurse on `arguments(R)`

It is important to note that match hypotheses can be in multiple kernels. Merging kernels with overlapping base structure can lead to candidate inferences, if the base statements corresponding to the match hypothesis roots are not themselves roots of the base description, as Fig. 6 illustrates.

10.2.4.1 Complexity of Kernel Creation

An extreme upper bound for the complexity of this step is *O(m)*, where *m* is the number of match hypotheses. This could in theory occur when the match hypothesis forest is extremely shallow, consisting of correspondences between distinct unary predicates with an entity as their argument, leading to *m/2* kernels. The best case would be when the base and target were completely isomorphic structures with a single root, with each subexpression having a unique functor. In that case there would be exactly one kernel. In practice the number is somewhere in between, with good matches having a small number of large kernels and poor matches having a lot of small ones.

10.2.5 Match Filters

The greedy merge algorithm used to combine kernels into mappings provides a good approximation to the best mapping, in terms of structural consistency. We believe that this is both psychologically accurate and useful computationally. However, in cases where task demands impose additional constraints on mappings, a carefully constrained set of filters, automatically constructed by the larger task model, can be provided as one of the inputs to the match process.

Filters work by eliminating kernels from further consideration, weeding them out before they are used in the merge phase. Here is how they operate:

- `(Excluded bi tj)` ⇒ remove kernel if it contains a match hypothesis which maps bi to tj.
- `(Required bi tj)` ⇒ remove kernel if it contains a match hypothesis which maps bi to some tk, tk ≠ tj or if there is a match hypothesis that maps tj to some bl, bl ≠ bi.

- `(Identical-functions)` ⇒ remove kernel if it contains a match hypothesis that maps a functor F which is a function to some function G, G ≠ F.
- `(require-within-partition-correspondences Att1 Att2)` ⇒ remove kernel if it contains a match hypothesis that maps an entity with attribute `Att1` to an entity with attribute `Att2`.

It is important to notice that filtering only determines whether or not kernels are considered in the merge phase, i.e., kernels inconsistent with the filters are not destroyed. This provides the ability to rapidly explore alternate interpretations, since the only work that needs to be re-done when exploring alternate constraints is re-doing the merge process itself.

10.2.5.1 Complexity of Match Filters

All of the tests for filter constraints are strictly local and rely only on structural properties of the expressions themselves, with the exception of the partition constraints. There, category membership is tested by the existence of attribute statements explicitly within the descriptions, which again is a local operation. Thus applying any of these filters to a single kernel can be considered a constant-time operation, so the complexity of applying them to the set of kernels is simply *O(k)*, where k is the number of kernels.

10.2.6 Greedy Merge

A mapping is a structurally consistent set of correspondences that is maximal, i.e., adding more match hypotheses would make it structurally inconsistent. Importantly, a comparison can have multiple maximal mappings, due to some interpretations of that comparison being structurally inconsistent with each other.

Mappings are created by merging kernels. As Section 4.1 outlined, we now use a greedy merge algorithm instead of an exhaustive algorithm, trading guarantees of optimal outputs for efficiency. We believe that this kind of approximation is psychologically plausible.

Our algorithm proceeds in two phases:

1. For each base root that participates in a kernel, greedily merge the kernels that project to it. This step improves the likelihood of candidate inferences by pre-combining kernels that could lead to them.
2. Greedily merge the solutions found for each base root to form a handful of global mappings.

We give the formal description of the entire algorithm below. We start with the crucial core algorithm combining partial mappings, which we call `GreedyMerge`. Greedy algorithms combine local solutions to form global solutions. They require a notion of solution quality that can be used to impose a preference ordering on local solutions. For constructing interpretations of a match, the local solutions are the kernels and the quality metric is their structural evaluation scores. The core `GreedyMerge` algorithm is:

Algorithm: `GreedyMerge`
Input: A set of partial mappings `PMAPS`, a score cutoff `S`, and a maximum number of desired interpretations `N`
Output: Up to `N` combined mappings, `MAPPINGS`

1. Sort `PMAPS` into a list in descending order, based on their structural evaluation scores.
2. `INTERPS` ← {}; `MAX` ← 0
3. Until `PMAPS` = {}
   - 3.1. `INTERP` ← pop(`PMAPS`)
   - 3.2. For each `K` in `PMAPS`,
     - 3.2.1. If ¬`Nogood(INTERP,K)` then
       - 3.2.1.1. `INTERP` ← `INTERP` ∪ {K}
       - 3.2.1.2. `PMAPS`←`PMAPS` – K
   - 3.3. For each `INTERP-B` in `INTERPS`,
     - 3.3.1. For each `K` in `INTERP-B`,
       - 3.3.1.1. If ¬`Nogood(INTERP,K)` then
         - 3.3.1.1.1. `INTERP` ← `INTERP` ∪ {K}
   - 3.4. If `score(INTERP)` > `MAX` then `MAX` ←`score(INTERP)`
   - 3.5. `If score(INTERP)` < `S*MAX` then go to 4.
   - 3.6. `INTERPS` ←`INTERPS` ∪ {`INTERP`}
   - 3.7. If |`INTERP`| = `N` then go to 4.
4. `Mappings` ← `map(CreateMapping,INTERPS)`

Note that step 3 allows an interpretation to also include kernels from previously-found interpretations. This step is solely used in the second phase, in which global mappings are discovered.

Algorithm: `CreateMapping`
Input: An interpretation `INTERP`, consisting of a set of partial mappings
Output: A mapping `M`

1. Let M be a new mapping
2. Correspondences(M) ←apply(∪,map(correspondences, INTERP))
3. Score(M) ← apply(+,map(score,Correspondences(M))
4. CandidateInferences(M) ← FindCandidateInferences(M)

Finding candidate inferences is discussed in the next section.

Intuitively, the GreedyMerge algorithm selects the largest partial mapping and merges into it everything that is structurally consistent. Each partial mapping added to the interpretation can rule out others, since it imposes new structural consistency constraints. (The nogoods for a set of partial mappings is simply the union of the nogoods for the partial mappings.) By starting with the largest we improve our chances of getting the best solution. By starting subsequent solutions with the largest remaining partial mapping, we improve our chances of getting a different yet still good solution, since to be still available it must be structurally inconsistent with earlier solutions. We view the ability to generate multiple interpretations of an analogy as critical. Even with a firm goal in mind, there can still be several ways to interpret an analogy (e.g. the Contras example in Holyoak and Thagard (1989)).

With the GreedyMerge algorithm in hand, now we can define GreedyMap:

Algorithm: GreedyMap
Inputs: A list of kernels KERNELS, a set of match constraints CC, a score cutoff S, and a maximum number of desired interpretations N
Outputs: Up to N global mappings, MAPPINGS

1. Let BASE-PARTITIONS = CalculateBasePartitions(KERNELS).
2. Let CANDIDATES = apply(∪, map(GreedyMerge, BASE-PARTITIONS)).
3. GreedyWeave({},CANDIDATES, S, N).

CalculateBasePartitions involves sorting the kernels into equivalences classes according to what base root(s) they project onto, and performing GreedyMerge within each equivalence class. This step is useful because candidate inferences arise from common base structure, hence pre-merging kernels that project onto the same base increases the likelihood of good candidate inferences.

GreedyWeave simply calls GreedyMerge on each element of REQUIRED in turn, with PMAPS = a required seed ∈ CANDIDATES, if any, using this output recursively for each subsequent candidate. The

process stops when either N solutions are generated or a new solution drops below the score cutoff of the previous solution.

Recall that N, the maximum number and S, the score cutoff, are psychologically motivated. The limited number of mappings that can be produced respects constraints on memory resources, and the score cutoff implements the intuition that an overwhelmingly better mapping swamps consideration of any alternatives. The default value for N is 3, and the default value for S is 0.8.

10.2.6.1 Complexity of Greedy Merge

Let us begin with analyzing the `GreedyMerge` algorithm, since it is at the core of the process. The first step, sorting the kernels, is $O(klog(k))$ in the number of kernels k. Each mapping is found in linear time $O(k)$ because it requires considering first the unused kernels and then the kernels in previous mappings. At most, N mappings are considered (where $N = 3$ by default). Therefore, the overall complexity is $O(klog(k))$. This is assuming that the cost of structural consistency tests can be ignored, which is reasonable given fast set intersection/union techniques involving bit-vectors.

Recall that the number of kernels k is worst-case $O(n^2)$ in the size of the base and target. In terms of base and target sizes, then, the complexity of `GreedyMerge` is thus worst case $O(n^2log(n^2))$, i.e., $O(n^2log(n))$. In practice the number of kernels is typically much smaller, since the worst-case presumes that every statement is independent. Rich, structured representations with substantial overlap tend to provide fewer and larger kernels, leading to better performance in such situations. This is unlike many match algorithms, where matching larger structures always leads to worse performance.

`CalculateBasePartitions` involves doing a `GreedyMerge` step for each of the base roots. The number of base roots as a function of the size of the base can range anywhere from 1 to $n/2$, the former when the entire base is a coherent focused argument, and the latter where the base consists of a disconnected set of entities, each with a single attribute known about it. (It can never be larger than $n/2$

because entities without any attribute or relational information will not participate in any match hypotheses, and hence will be irrelevant for the algorithm.) Therefore in the worst case, the complexity of `CalculateBasePartitions` is $O(n^2 log(n))$ in the case of isolated entities, with typical case complexity being much lower than that. Again, with this algorithm, growth in base and target sizes does not always result in growth in processing time or memory – it can actually decrease if the growth makes the relational structures more connected!

Given the tight bounds imposed by `N` and `F`, the complexity of `GreedyWeave` is simply that of `GreedyMerge`, $O(n^2 log(n))$, since the number of times it is executed depends on them instead of the sizes of the base and target. The worst-case complexity for the `GreedyMap` step is simply the complexity of its most expensive step, `CalculateBasePartitions`, and hence is $O(n^2 log(n))$.

10.2.7 Generating Candidate Inferences

The algorithm for computing candidate inferences is:

Algorithm: `FindCandidateInferences`
Input: A mapping `M`
Output: the set of candidate inferences `CandidateInferences(M)`

1. `CandidateInferences(M)` ← {}
2. For each `R` ∈ `Roots(Base(M))`,
   2.1. When ∃ `MH` ∈ `Correspondences(M)` | `Root(BaseItem(MH))` = `R`,
      2.1.1. `CandidateInferences(M)` ← `CandidateInferences(M)` ∪ `ConstructCI(R, M, {})`
3. For each `CI` ∈ `CandidateInferences(M)`, `CIStrucuturalEvaluation(CI)`

That is, each root expression of the base that intersects the subset of the base that is mapped by `M` gives rise to a candidate inference. Reverse candidate inferences are computed via the same algorithm, using the target as the starting point instead of the base.

There is a subtle issue here concerning how much overlap between the base and the mapping is needed to suggest a candidate inference. The most conservative criterion requires overlapping statements, the

most liberal criterion requires only overlapping entities. The conservative criterion limits candidate inferences to filling in causal, inferential, or other higher-order structure involving overlapping statements. The liberal criterion enables candidate inferences to import whole new structures into the target, based on entity overlaps established by other parts of the base. Given the need to evaluate candidate inferences in any case, the default mode of operation is the liberal criterion. However, SME includes a switch for enforcing the conservative criterion, which is implemented by an extra condition in line 2.1 above.

Recall that candidate inferences can introduce new entities into the target, called *skolem entities*. The `ConstructCI` algorithm must do a tree walk through the base expression, introducing such skolems as necessary. We say that a base item `B` is mapped in mapping `M` if there is some match hypothesis in the correspondences of `M` which has `B` as its base item. If `B` is mapped, then its correspondent is the target item for the match hypothesis which mentions `B`. Similarly, if a target item `T` is mapped in `M`, then its correspondent is the base item for the match hypothesis that involves `T`. (That correspondent, when defined, is a function follows directly from the 1:1 constraint of structure-mapping.) Thus we must introduce skolems for each entity in the base expression that does not have a correspondent. Moreover, we must introduce the same skolem for each occurrence of the entity in the base expression, since an entity can occur multiple times in the same expression. This is what makes `ConstructCI` a bit complex, since it must maintain a table of bindings.

Algorithm: `ConstructCI`
Input: A base item `B`, a mapping `M`, and a set of bindings `Skolems`
Outputs: An expression representing a candidate inference and a set of skolem entities for base items that have no correspondent in the target.

1. If `B` is an entity,
   1.1. If `Mapped(B,M)` then return `Correspondent(B,M)` and `Skolems`
   1.2. If lookup(`B`,`Skolems`) then return value(lookup(`B`,`Skolems`)) and `Skolems`
   1.3. Let `Sk(B)` be a new skolem constant. Return `Sk(b)` and `Skolems` ∪ Bind(`B`,`Sk(B)`)
2. If `B` is a functor, if `Mapped(B,M)` then return `Correspondent(B,M)` and `Skolems`, otherwise return `B` and `Skolems`

3. Otherwise B is an expression.
   3.1. If Mapped(B,M) then return Correspondent(B,M) and Skolems
   3.2. Otherwise, let cargs = empty list
      3.2.1.Let cfunctor, skolems = ConstructCI(functor(B),M, skolems)
      3.2.2.For each A ∈ arguments(B),
         3.2.2.1. Let carg, newskolems = ConstructCI(A,M,skolems)
         3.2.2.2. Let cargs = cargs ∪ {carg}
         3.2.2.3. Let skolems = newskolems
      3.2.3.Return MakeExpression(cfunctor, cargs)

A subtle issue in ConstructCI is that it assumes that functors lying outside the mapping should be brought over intact. This design choice reflects our intuition that one purpose of analogical matching is to help regularize, and thus extend, one's knowledge. The alternative would be to always create a skolem constant for the functor, and then attempt to replace it with the functor from the base as a separate step. Since candidate inferences always need to be checked for validity in any case, inappropriate carryover of functors will be detected during this process anyway. Our choice to carry them over intact biases the process towards accepting the carryover by default.

The structural evaluation algorithm for computing support and extrapolation scores (CIStructuralEvaluation above) is a variation of the algorithm used for mappings. To compute the support score of a candidate inference, the initial bias plus trickle-down algorithm is executed on just the subset of the correspondences that support it in the mapping and adding up the results.

The extrapolation score of an analogical inference is, roughly, the size of the new information over the total size of the inference. Consider two limiting cases, neither of which can actually occur. If there were no support (i.e., a hallucination), all the information would be new, so the extrapolation score would be 1. If there were nothing new (everything was there already), then the score would be 0. Any real candidate inference will be somewhere in between these two values.

The algorithm for computing extrapolation scores is

1. Apply the trickle-down algorithm to the structure of the inference itself, i.e., as if we were matching the inference to itself

2. The extrapolation score is

$$\frac{score\,(Outside\,)}{score\,(Outside\,) + score\,(Inside\,)}$$

where *Inside* refers to the items in the candidate inference that are part of the mapping and *Outside* refers to the items in the candidate inference that are being projected. Using the trickle-down algorithm provides a more conservative score than simply counting items would, since the existence of large structures outside the mapping will lead to higher scores inside the mapping due to trickle-down.

10.2.7.1 Complexity of Candidate Inference Generation

Since the size of statements is typically small compared to the number of statements in the descriptions, we ignore all variability in cost of recursive traversal of statements, treating it as a constant, and focus instead on the number of candidate inferences there can be, since that can vary as a function of the size of the input descriptions. Because of structural consistency, each base root can participate in at most one candidate inference per mapping. Consequently the growth of candidate inferences is bounded by *O(n)*.

10.2.8 Extending and Remapping

The intuition behind incremental mapping is that normally people first try to incorporate new information into an ongoing mapping (`Extend`), but that they can reinterpret the analogy if necessary (`Remap`).

Extending a mapping occurs when new information is added to the base or target of a match. Basically, the new information is matched against the other representation, leading to new match hypotheses and new kernels. Notice that since the information is new by assumption, there cannot already be any correspondences pertaining to it in the match. Therefore either some new kernels will be formed, or the new information does not match at all to the other description in its current state. New kernels are

then added to existing mapping(s) if they are structurally consistent with them. This can lead to the elimination of candidate inferences, if the new information is about the target and "fills in" the missing structure.

The worst-case complexity of extending a mapping is the same as the Comparison algorithm, since in the worst case one is adding all of the information to an empty base and target. The typical case complexity is of course much lower, since adding one new item to the base (or target) only requires checking it against all of the items in the target (or base), not re-checking any previous base (target) statements.

The `Remap` algorithm simply destroys the existing mappings and re-performs the Greedy Merge algorithm based on the full set of kernels from scratch. Thus the complexity of the Remap operation is simply that of the Greedy Merge algorithm. Psychologically, we believe that the remapping criteria people use is task-specific. Consequently, SME does not automatically remap: The decision to remap must be taken by some external model or system. To help external systems make such decisions SME does provide an estimate of what fraction of the total possible structural evaluation the current mappings represent. When this fraction gets low, it suggests that remapping might lead to a better global interpretation.

10.2.8.1 Complexity of Extending a Match and Remapping

Extending a match with new items in the base and/or target requires extending the match hypothesis forest and adding the kernels (if any) to the existing mappings. The candidate inferences for the mappings need to be recomputed. This is clearly bounded above by the complexity of computing the match from scratch, although in practice it is typically far less. Remapping simply re-runs `GreedyMap`, which is $O(n^2 log(n))$, as per the analysis above.

10.3 Complexity of the SME algorithm

If, as we believe, comparison comprises one of the core processes of cognition, then it is crucial for its computational complexity to be low. SME's computational complexity is in fact quite low. Recall that if the number of items in the base and target is *n*, the number of kernels *k* is bounded by n^2. The results of the complexity analysis can be summarized as follows:

Operation	Worst-case time, serial processing
Finding match hypotheses	$O(n^2)$
Structural consistency filtering	$O(n^2)$
Structural evaluation propagation	$O(n^2)$
Kernel creation	$O(n^2)$
Filtering and pragmatic marking	$O(n^2)$
Greedy merge	$O(n^2 log(n))$
Candidate inference construction	$O(n)$
Extending/Remapping	$O(n^2 log(n))$

Thus the SME algorithm is worst-case $O(n^2 log(n))$ on a serial processor. Most of SME's processing can be done in parallel, as our analyses of individual steps noted. Assuming a data-parallel machine with at least n^2 processing elements to handle match hypothesis networks, the processing for the overall algorithm would be between log and linear, depending on specific assumptions about the parallel architecture.

11 Figures

Figure 1: The Three phases of the SME algorithm

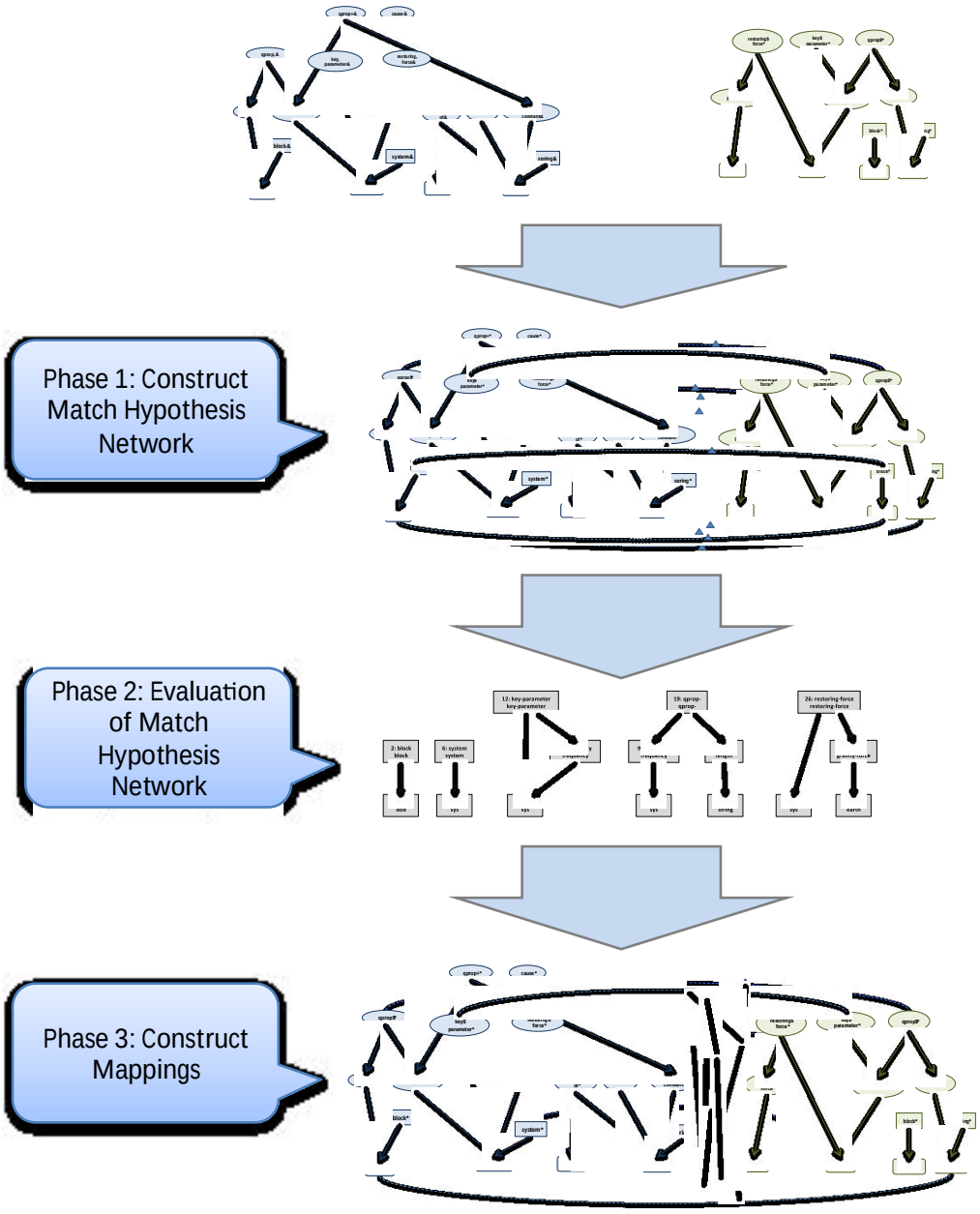


Figure 2: Analogous oscillators

(a) A spring-block oscillator

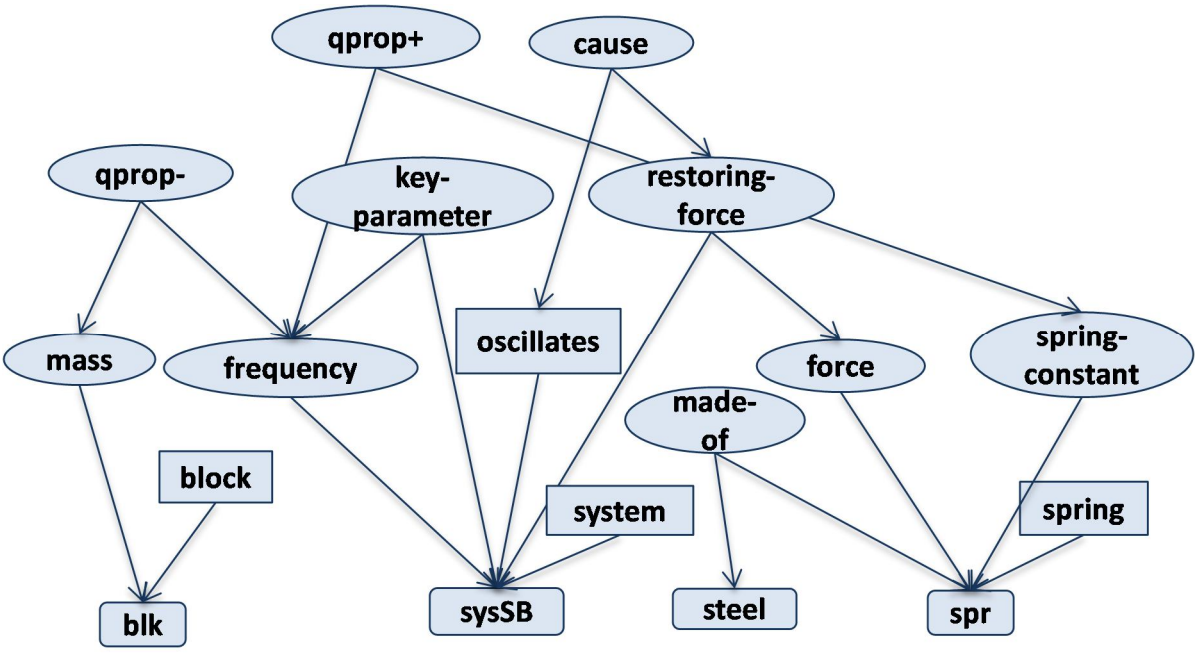
(spring spr)
(block blk)
(system sysSB)
(made-of spr steel)
(key-parameter sysSB (frequency sysSB))
(qprop+ (frequency sysSB) (spring-constant spr))
(qprop- (frequency sysSB) (mass blk))
(restoring-force sysSB (force spr))
(cause (restoring-force sysSB (force spr))
(oscillates sysSB))

(b) A pendulum

(string str)
(block bob)
(system sys)
(key-parameter sys (frequency sys))
(qprop- (frequency sys) (length str))
(restoring-force sys
(gravity-force earth))))

Figure 3: Graphical depiction of the base and target for the oscillators example

Base:



Target:

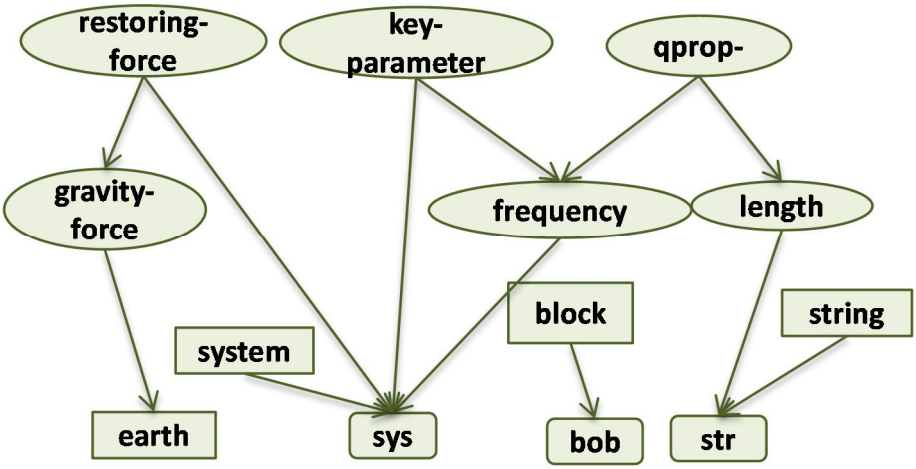


Figure 4: Match hypothesis forest for the oscillators

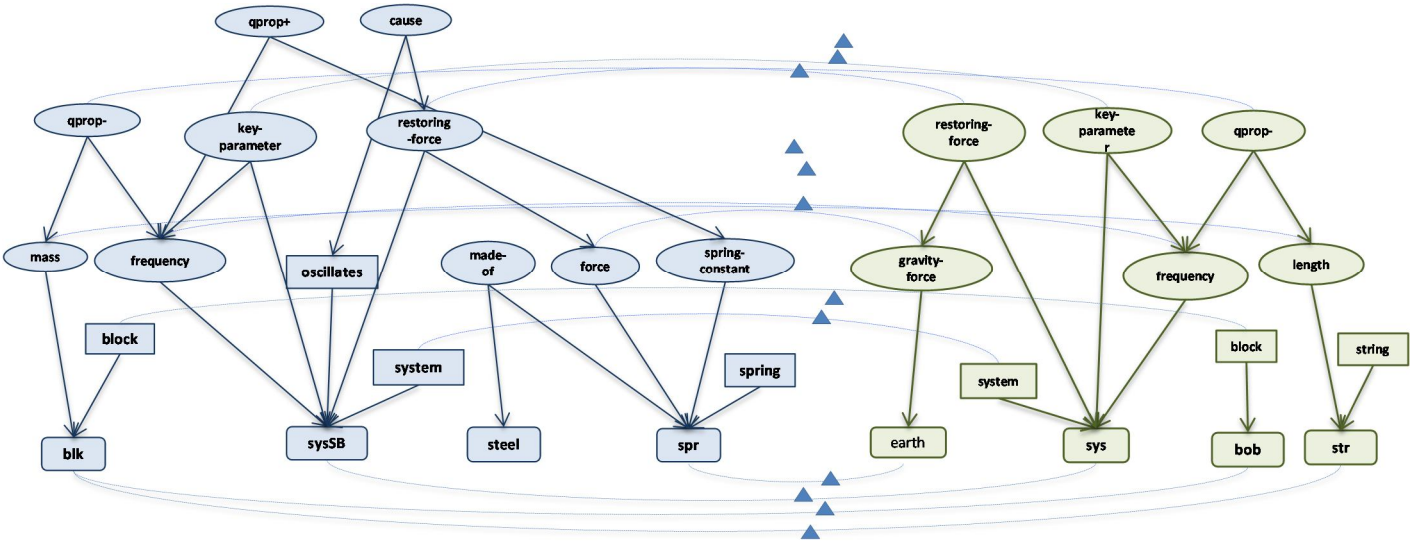


Figure 5: Kernels for the oscillators comparison

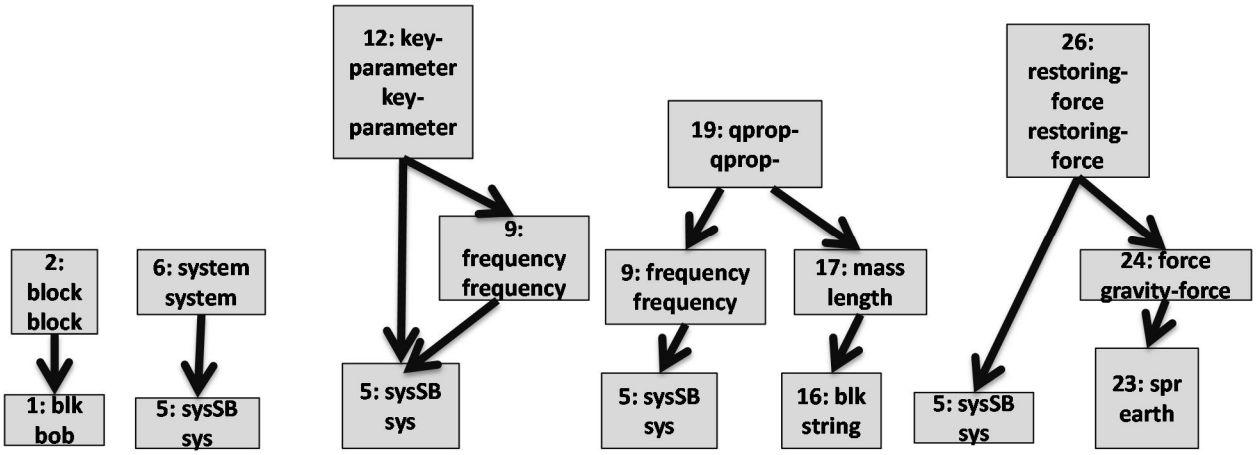


Figure 6: The mapping SME constructs between the oscillators

Mapping 8 SME #1

Score: 0.1661

Base: spring-block-oscillator

Target: pendulum

Support	Base Item	Target Item	MH Score
(4)	sysSB	sys	0.0800
(1)	spr	earth	0.0336
(1)	blk	str	0.0336

- 5 candidate inferences
- No reverse candidate inferences
- 7 expression correspondences
- 7 functor correspondences

Legend:

= Match Hypothesis Details = Expression Details = Candidate Inference Explanation

Expression Correspondences
[7 items]

SME #1
Mapping 8

	Base Item		Target Item		Score
★	✱	(frequency sysSB)	✱	(frequency sys)	0.0085
★	✱	(force spr)	✱	(gravity-force earth)	0.0042
★	✱	(mass blk)	✱	(length str)	0.0042
★	✱	(system sysSB)	✱	(system sys)	0.0005
★	✱	(restoring-force sysSB (force spr))	✱	(restoring-force sys (gravity-force earth))	0.0005
★	✱	(qprop- (frequency sysSB) (mass blk))	✱	(qprop- (frequency sys) (length str))	0.0005
★	✱	(key-parameter sysSB (frequency sysSB))	✱	(key-parameter sys (frequency sys))	0.0005

Legend:

★ = Match Hypothesis Details ✱ = Expression Details ? = Candidate Inference Explanation

Candidate Inferences

[5 items]

SME #1
Mapping 8

	Inference	Support	Extrapolation
?	(spring earth)	0.0005	0.9000
?	(block str)	0.0005	0.9000
?	(made-of earth (:skolem steel))	0.0005	0.9444
?	(qprop+ (frequency sys) (spring-constant earth))	0.0050	0.6296
?	(cause (restoring-force sys (gravity-force earth)) (oscillates sys))	0.0090	0.4857

Legend:

= Match Hypothesis Details

= Expression Details

? = Candidate Inference Explanation

Figure 7: Examples of geometric analogies. "A is to B as C is to...?"

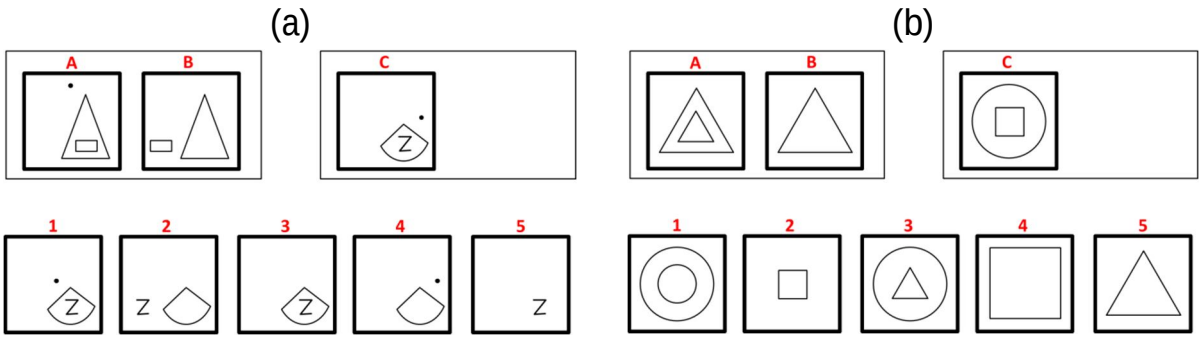
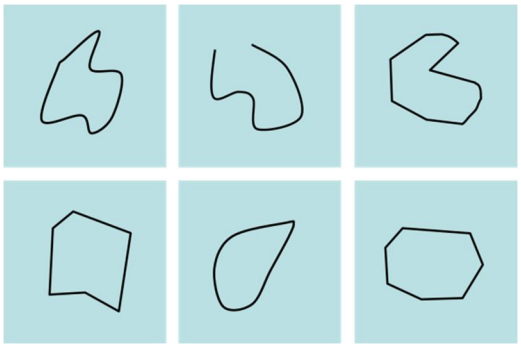


Figure 8: Examples of a visual oddity task. "Pick the image that doesn't belong."

(a)



(b)

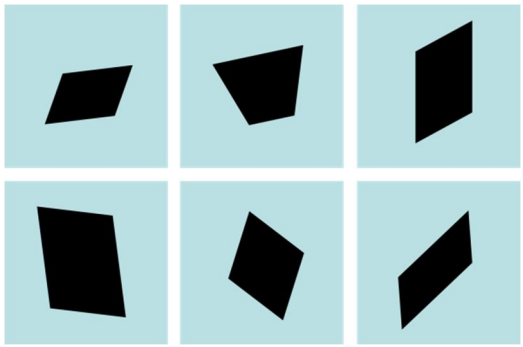
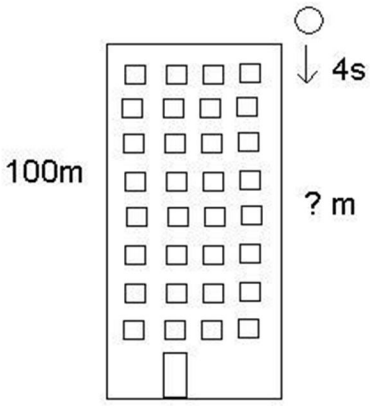


Figure 9: An example AP Physics problem



A ball is dropped from a 100 meter tall building. How far will it have fallen after four seconds?

A) 20m

B) 40m

C) 80m

D) 120m

12 Tables

Table 1: Benchmark Phenomena of Analogy

Adapted from Gentner & Markman, 2005; Markman & Gentner, 2000)

Relational similarity	Analogies involve relational commonalities; object commonalities are optional.
Structural consistency	Analogical mapping involves one-to-one correspondence and parallel connectivity.
Systematicity	In analogical mapping, connected systems of relations governed by higher-order constraining relations are preferred over isolated relations.
Candidate inferences	Analogical inferences are generated via structural completion.
Alignable differences	Differences that are connected to the commonalities of a pair (and not unconnected differences) are rendered more salient by a comparison.
Interactive interpretation	Analogy interpretation depends on both terms; the same term yields different interpretations in different comparisons.
Multiple interpretations	Analogy allows multiple interpretations of a single comparison.
Cross-mapping	Though difficult, cross-mappings are generally interpreted relationally although the competing object similarities are perceived.

Table 2: Experiments with sources of representations.

#Cases indicates the total number of cases generated, across all problems in that task. #Comparisons indicates the total number of comparisons made in each of those tasks. This includes comparisons made in service of analogical retrieval, by the second stage of MAC/FAC.

Task	# Cases	# Comparisons	Source of Cases
Geometric Analogies	299	873	CogSketch
Oddity Task	446	3,409	CogSketch
Thermodynamics Problems	8	4	AI System
Physics Problems	253	1,137	Educational Testing Service
Moral Decision-Making	41	420	Natural Language System

Table 3: Representation Statistics from Computational Investigations

This table shows the mean, minimum, and maximum number of entities, expressions, and relations in the descriptions given to SME in the computational experiments summarized here. Expressions include both relations and non-atomic terms, i.e. entities denoted via functional expressions.

Task	Entities			Expressions			Relations		
	Mean	Min	Max	Mean	Min	Max	Mean	Min	Max
Geometric Analogies	2.5	1	8	24	3	136	16	0	112
Oddity Task	3	1	16	29	2	216	20	0	168
Thermo Problems	15	8	31	147	57	400	89	25	282
Physics Problems	32	4	71	129	5	431	87	2	298
Moral Reasoning	16	13	21	45	31	62	31	18	46

Table 4: Sample materials from Gentner et al. (2001). Passages were presented sentence by sentence and time to read each sentence was timed. The key result was that people took longer to read the final sentence in the Inconsistent case than in the Consistent case.

Consistent: *A Debate is a Race*
Dan saw the big debate as a *race:* he was determined to win it. *He* knew that he had to *steer his course* carefully in the competition. His strategy was to go *cruising through* the initial points and then make his move. After months of debating practice, Dan knew how to present his conclusions. If he could only *keep up the pace,* he had a good chance of winning. Before long, he felt the audience was receptive to his arguments. Then, he *revved up* as he made his last key points. His skill left his opponent *far behind him* at the *finish line.*
Inconsistent: *A Debate is a War*
Dan saw the big debate as a *war:* he was determined to be victorious. He knew that he had to use every *weapon* at his command in the competition. He mapped out his strategy to insure he established a *dominant position.* After months of debating practice, Dan knew how to present his conclusions. If he could only *marshal his forces,* he had a good chance of winning. Before long, he felt the audience was receptive to his arguments. Then, he *intensified the bombardment* as he made his last key points. His skill left his opponent *far behind him* at the *finish line.*

Table 5: Ubiquitous predicates by domain

Domain	Ubiquitous predicates
Geometric Analogies	None
Oddity Task	None
Thermodynamics Problems	=, and, nvalue, equation, the-set
Physics Problems	and, multipleChoiceSingleOptionList, testAnswers-SingleCorrectMultipleChoice, TheList, TheSet
Moral Decision-Making	and

Table 6: Match hypothesis forest growth without ubiquitous predicates

Domain	% matches bloated	% increase in # of match hypotheses	
		Mean Bloat	Max Bloat
Thermodynamics Problems	100%	183%	329%
Physics Problems	99%	34%	51%
Moral Decision-Making	2.3%	2.7%	3.5%

Table 7: Statistics on sizes of match hypothesis forests and kernels, compared to worst-case analysis

Mean % and Max % are the percentages of data structure sizes computed by SME over the computational experiments, compared to the theoretical worst case sizes, to illustrate how pessimistic the worst case analysis can be in practice.

Experiment	Actual match hypotheses vs. worst case		Actual kernels vs. worst case	
	Mean %	**Max %**	**Mean %**	**Max %**
Geometric Analogies	21.3%	55.5%	10.1%	22.2%
Oddity Task	20.28%	55.5%	9.9%	22.2%
Thermodynamics Problems	2.1%	2.6%	0.56%	0.7%
Physics Problems	1.1%	17.3%	0.32%	6.2%
Moral Decision-Making	2.3%	18%	1.1%	8.5%