Identifying Sources of Intractability in Cognitive Models: An Illustration using Analogical Structure Mapping

Iris van Rooij (i.vanrooij@nici.ru.nl)
Nijmegen Institute for Cognition and Information
6525 HR Nijmegen, The Netherlands

Patricia Evans (pevans@unb.ca)
Faculty of Computer Science
New Brunswick, E3B 5A3 Canada

Moritz Müller (moritz.muller@math.uni-freiburg.de)
Department of Mathematics
79104 Freiburg, Germany

Jason Gedge (gedge@cs.mun.ca) and Todd Wareham (harold@cs.mun.ca)
Department of Computer Science
St. John's, NL A1B 3X5 Canada

Abstract

Many computational models in cognitive science and artificial intelligence face the problem of computational intractability when assumed to operate for unrestricted input domains. Tractability may be achieved by restricting the input domain, but some degree of generality is typically required to model human-like intelligence. Moreover, it is often non-obvious which restrictions will render a model tractable or not. We present an analytical tool that can be used to identify sources of intractability in a model's input domain. For our illustration, we use Gentner's Structure-Mapping Theory of analogy as a running example.

Keywords: computational complexity; intractability; parameterized complexity; analogy; structure mapping

Introduction

Computational intractability is a problem that plagues many models of human and artificial intelligence. If such models are assumed to apply to inputs of real-world size and complexity, then they require more computational resources (e.g., time or memory) than can be reasonably attributed to any computing machine, whether human or artificial. The problem seems to often arise from the rich, combinatorial structure of the representations posited by our theories of cognition, but typically it is hard to tell what it is exactly about the structure that makes the computations defined over these structures computationally so expensive. If cognitive modelers have a means of identifying structural properties that—either combined or in isolation—are responsible for a model's intractability, then they could use this knowledge for coming up with informed hypotheses of how an intractable cognitive theory can be rendered tractable.

Take, for example, the influential theory of Dedre Gentner (1983) of analogy, called Structure-Mapping Theory (SMT). According to this theory, humans form analogies by mapping relations in one predicate structure (the base) to relations in another (the target). One can think of predicate structures as directed acyclic graphs with the nodes in the graph labelled by predicates and objects, and arcs directed from higher-order predicates to the lower-order predicates and objects that are their arguments (see Figure 1 for an illustration). Because both the base and the target can have many nodes and arcs, with complex connectivity, there exist many possible mappings to choose from. To be precise, if both base and target have n nodes, then there exist $\sum_{k=0}^{n} \binom{n}{k}^2 \cdot k!$ possible mappings. For networks with 6 nodes this already leads to 13,327 possible mappings and for networks with 18 or more nodes the number of mappings exceed the seconds since the birth of the universe. Exhaustively searching such a (super-polynomially sized) search space is unfeasible even for intermediate problem sizes. Yet the proposal that SMT may model the human ability for analogizing—as well as the desire to emulate this ability in artificial systems—raises the question if there exist algorithms that can find the right mapping without having to perform such an exhaustive search.

A finding that seems to bear on this question is that structure-mapping as defined by SMT is an NP-hard problem (Evans, Gedge, Müller, van Rooij, & Wareham, 2008; Veale & Keane, 1997). This means that *all* algorithms solving the problem are of super-polynomial complexity.[1] It also means that there is only one way to ensure SMT is a computationally feasible model of analogy-making:[2] The structure-mapping

[1] This interpretation of the NP-hardness of SMT holds under the assumption that P $\neq$ NP, a mathematical conjecture that is unproven but has strong empirical support. The interested reader is referred to Garey and Johnson (1979) and Arora and Barak (in press) for more details.

[2] The problem of intractability is so familiar that many cognitive scientists may instantaneously have ideas about how this problem could be solved, but the fact of the matter is that all such solutions either restrict the domain of inputs for which the theory is believed to hold or the theory is revised so as to allow for (slightly) different outputs than the one specified by the original theory (van Rooij, in press). Since our purpose is to present analytical tools for identifying sources of intractability in a *given* theory, we focus on the first option

processes must be assumed to operate for a restricted domain of input structures, where those structures have special properties that can be exploited in the tractable computation of analogies.

How can we find out what these special properties are? One way of approaching this question (though not one that we recommend) is to implement an algorithm that computes structure-mappings and investigate how long it runs for different input structures (e.g., Falkenhainer, Forbus, & Gentner, 1989). By systematically varying structural aspects of the input one may then discover that even though the algorithm runs slow for many input structures, it runs relatively fast for some. By comparing the "easy" and "hard" inputs one may observe that they differ in several respects, e.g., certain structural aspects may be relatively small, relatively large, or otherwise special in the "easy" inputs. One may then be led to believe that it is the *absence* of these special properties in the "hard" inputs that makes structure-mapping hard in general.

Granting that such an approach may overcome the practical obstacle that a systematic search of the space of inputs is itself computationally expensive (to our knowledge, so far only unsystematic searches have been performed for SMT), the more important theoretical obstacle remains that we cannot infer from the slow running of an algorithm that the structure-mapping problem is intractable for the same domain of inputs. There could always exist a different algorithm for the structure-mapping problem that runs fast for those same inputs.[3] In other words, an algorithmic simulation approach can perhaps tell us something about the computational efficiency of particular structure-mapping *algorithms*, but it need not tell us anything about the complexity inherent in the structure-mapping *problem*. It is for this reason that we propose to use a different approach.

The approach that we investigate in this paper adopts the analytical tools of computational complexity theory. We will first explain how these tools can be used to identify what we call 'sources of complexity' in an intractable problem (i.e., problem aspects that can confine the super-polynomial time complexity inherent in a problem). We then use these tools to test if aspects that have been proposed to be responsible for the intractability of structure-mapping are indeed sources of complexity in SMT. We show that none of the conjectured aspects are—by themselves or in combination—responsible for the intractability of SMT. We furthermore show that some previously unidentified aspects *are* so responsible. The non-obvious nature of these theoretical results illustrates the utility of the analytical tools that we describe.

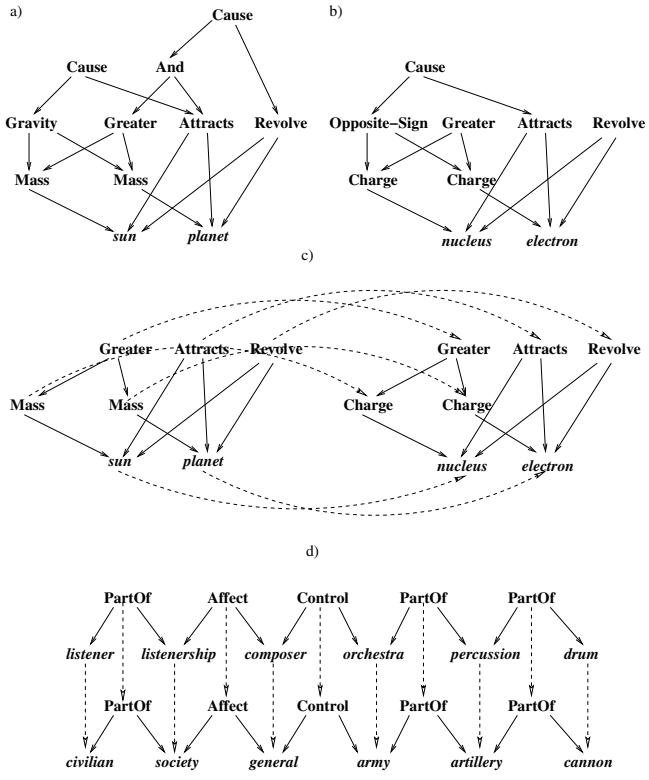

Figure 1: Illustrations of graph representations of predicate-structures and analogy-mappings as defined by SMT. (a) Solar system predicate-structure. (b) Rutherford atom predicate-structure. (c) Analogy-mapping between (a) and (b). (d) Analogy-mapping between Composer and General predicate-structures. Parts (a) and (b) are adapted from (Falkenhainer et al., 1989, Fig. 9) and part (d) is adapted from (Veale et al., 1999, Fig. 8).

A Method for Identifying Sources of Complexity

Computational complexity theory actually refers to a whole family of mathematical theories developed with the purpose of classifying problems according to their inherent complexity. Of particular relevance for our purposes is a relatively recent variant called *parameterized complexity theory*, founded by Downey and Fellows in the 90s and currently the topic of many new complexity results and techniques (see the special issue edited by Downey, Fellows, & Langston, 2008). Parameterized complexity theory is motivated by the observation that many NP-hard problems can be computed by algorithms whose running time is polynomial in the overall input size n and non-polynomial only in one or more small aspects of the input. These aspects are called *parameters*. As the main part of the input contributes to the overall complexity in a "good" way, and only the parameters contribute to the overall complexity in a "bad" way, the problem is well-solved even for

here (but see Hamilton, Müller, van Rooij, and Wareham (2007) and van Rooij and Wright (2006) for discussions of the second option).

[3] In general, if A_1 and A_2 are two algorithms that compute intractable problem $P : I \rightarrow O$. Then there can exist two distinct input domains $I_1, I_2 \subset I$ such that A_1 is a tractable algorithm for I_1 but not for I_2, and A_2 is a tractable algorithm for I_2 but not for I_1. If so, then the problem P is computationally tractable for $I_1 \cup I_2$.

large inputs provided only that the parameters remain small. This intuitive characterization is captured by the formal notion of fixed-parameter tractability (see also Downey & Fellows, 1999).

Definition 1. Fixed-parameter tractability. Let $P: I \rightarrow O$ be a problem with input parameters $k_1, k_2, ..., k_m$. Then P is said to be *fixed-parameter tractable* for parameter set $K = \{k_1, k_2, ..., k_m\}$ if there exists at least one algorithm that computes P for any input of size n in time $f(k_1, k_2, ..., k_m)n^c$, where $f(.)$ is an arbitrary computable function and c is a constant. If no such algorithm exists then P is said to be *fixed-parameter intractable*.

We note the following observation, which follows from Definition 1.

Observation 1. If P is fixed-parameter intractable for parameter set K then P is also fixed-parameter intractable for any subset of parameters $K' \subset K$. If P is fixed-parameter tractable for parameter set K then P is also fixed-parameter tractable for any superset of parameters $K'' \supset K$.

Given the notion of fixed-parameter (in)tractability and Observation 1 we can derive a natural candidate for what defines a source of complexity in an intractable problem (see also van Rooij, Stege, & Kadlec, 2005; van Rooij & Wareham, in press).

Definition 2. Source of complexity. Let $P: I \rightarrow O$ be an intractable problem. Then parameter set $K = \{k_1, k_2, ..., k_m\}$ is said to be a *source of complexity* in P, if P is fixed-parameter tractable for parameter set K and fixed-parameter intractable for all subsets $K' \subset K$.

In other words, a parameter set K is considered a source of complexity in an intractable problem if it is sufficient for capturing the non-polynomial complexity inherent in the problem and it does not contain any unnecessary elements. The notion of a source of complexity, so defined, expresses the intuitive idea that a parameter is a source of complexity if, all else being equal,[4] high values of the parameter cause the problem to be hard, and low values of the parameter cause it to be easy. In accordance, we judge an input aspect (in this case, elements of K being large) to be responsible for the intractability of the problem if its *absence* (in this case, elements of K being small) renders the problem tractable.

It remains to be explained how one can determine if a problem P is fixed-parameter (in)tractable for some parameter set K. Proving fixed-parameter tractability may be technically challenging but is conceptually straightforward: It suffices to produce just one algorithm that computes the problem in fixed-parameter tractable time (see, e.g., Sloper & Telle, 2008, for a review of generic techniques for building such algorithms). Fixed-parameter intractability can be established by proving the problem W[1]-hard (the parameterized analog of NP-hard).[5] To prove this it suffices to construct a *parameterized reduction* from a known W[1]-hard problem P' to the problem P.

Definition 3. Parameterized reduction. Let $P_1: I_1 \rightarrow O_1$ and $P_2: I_2 \rightarrow O_2$ be two problems with parameter set K_1 and K_2 respectively. Then a *parameterized reduction* from P_1 to P_2 consist of two algorithms, A_i and A_o, that are fixed-parameter tractable relative to K_1, such that

- A_i transforms any input $i_1 \in I_1$ (with associated values for parameters in K_1) into an input $i_2 \in I_2$ (with the elements in K_2 bounded by some computable function of K_1), and
- A_o transforms any output $o_2 = P_2(i_2)$ into an output $o_1 = P_1(i_1)$.

Note that if there exists a parameterized reduction from a problem P' to another problem P, then P is fixed-parameter tractable only if P' is too. After all, in that situation P' can be computed in fixed-parameter tractable time by first transforming its input into the corresponding input for P, using algorithm A_1, then solving P, and finally transforming the output of P back to the corresponding output for P' using algorithm A_o. This implies that, if P' is fixed-parameter intractable, then so is P.

Candidate Sources of Complexity in Structure-mapping

The structure-mapping problem can be informally stated as follows (see Evans et al. (2008) for a formalization of this definition).

STRUCTURE-MAPPING
Input: Two directed acyclic graphs $G_1 = (V_1, A_1)$ and $G_2 = (V_2, A_2)$, each encoding a predicate structure.
Output: The best of all structurally consistent mappings from G_1 to G_2.

Here a mapping from G_1 and G_2 is *structurally consistent* if the following two conditions are met: (1) a vertex in V_1 is mapped to at most one vertex in V_2 and (2) for every predicate $v \in V_1$ that is mapped to a predicate $v_2 \in V_2$ also all the arguments of v_1 are mapped to the arguments of v_2. Further, a mapping m_1 is considered *better* than another mapping m_2

[4] We add the phrase 'all else being equal' because there may exist some other parameter set K' distinct from K for which P may also be fixed-parameter tractable. If parameters in that set are small, then even if the elements of K are large, P will be tractable. This means that a problem P need not have one unique source of complexity. We believe that this does not undermine the intuitive interpretation of large values for parameters in K being responsible for the intractability in P, even if large values for parameters in K' are as well.

[5] We will work under the assumption W[1] $\neq$ FPT (here FPT the class of fixed-parameter tractable problems). Like P $\neq$ NP, this mathematical conjecture is unproven but has strong empirical support. The interested reader is referred to Downey and Fellows (1999) and Flum and Grohe (2006) for more details.

if m_1 maps relatively more higher-order predicates than m_2. SMT defines the *order* of a predicate to be the maximum order of its arguments *plus* 1, with objects being order 0.

Now note that every instance of the structure-mapping problem will have values for specific problem parameters, each such parameter constituting a potential source of complexity. Table 1 lists the parameters considered in this paper. For simplicity, in the remainder of this paper we will drop the subscript for a parameter x_i and write x to refer to either x_1 or x_2 (depending on which happens to be largest).

Our choice of parameters is motivated in part by speculations in the literature about aspects of predicate structures that may be responsible for the computational intractability of SMT.

Based on the finding that their implemented structure-mapping algorithm ran slower for the Composer-General example than for the Rutherford example, Falkenhainer et al. (1989) conjectured that the complexity of structure-mapping may depend not so much on the number of vertices (n), but more on the height (h) of the predicate structures. Specifically, these researchers suggested that worst-case times occur when relatively 'flat' predicate structures (i.e., structures with minimal or no predicate-nesting (see also Veale & Keane, 1997, p. 1). To investigate this possibility using our proposed tool of parameterized complexity analysis, we need to define a parameter that is large for 'flat' predicate structures and prove that it is fixed-parameter (in)tractable. We propose to use $\frac{n}{h}$ as a measure of the relative 'flatness' of the input structures. With this measure we intend to capture the idea that flatness ($\frac{n}{h}$) is large when height (h) is small, relative to the overall size of the graphs (n).

We noticed that the Composer-General predicate structures, besides being relatively flat, also have relatively many root predicates r and relatively many objects o, especially when compared to the Rutherford example. To investigate if they could serve as alternative explanations of the apparent hardness of Structure mapping for these types of predicate structures we included these parameters in our analysis. For completeness, we also include the total number of predicates (p) and the number of non-root predicates (s) in our analysis, to investigate if possibly they could be responsible for the intractability of Structure mapping for other types of inputs.

It has also been proposed that the "performance [of algorithms for SMT] is more a function of representation and repetitiveness rather than ... size [of the predicate structures]" (Falkenhainer et al., 1989, p. 47). One possible measure of repetitiveness is the number of occurrences of the same predicate (f) in a given predicate structure. To investigate if this form of repetitiveness indeed is a cause for difficulties for Structure-mapping algorithms we included f in the set of considered parameters.

Lastly, we included arity of predicates (a) in our analysis, because there exists empirical evidence that this parameter is naturally kept small in human mental representations by cognitive processing (e.g., working memory) limitations (Halford, Wilson, & Phillips, 1998). It is of interest to see if a bound on the arity of mentally represented relations helps keep analogical mapping computational tractable for human minds.

Table 1: Overview of parameters for structure-mapping, with the corresponding value of each parameter in the Rutherford example in Figure 1a/b, and the Composer-General example in Figure 1d. Without loss of generality, we assume that G_1 is larger than G_2.

Name	Definition	Fig. 1a/b	Fig. 1d
n_1	number of vertices in G_1	11	11
n_2	number of vertices in G_2	9	11
h_1	maximum order of predicates in G_1	4	1
h_2	maximum order of predicates in G_2	3	1
$\frac{n_1}{h_1}$	measure of 'flatness' of G_1	2.75	11
$\frac{n_2}{h_2}$	measure of 'flatness' of G_2	3	11
p_1	number of predicates in G_1	9	5
p_2	number of predicates in G_2	7	5
r_1	number of root predicates in G_1	2	5
r_2	number of root predicates in G_2	3	5
s_1	number of non-root predicates in G_1	7	0
s_2	number of non-root predicates in G_2	5	0
o_i	number of objects in $G_{i=1,2}$	2	6
f_i	frequency of a given predicate label in $G_{i=1,2}$	2	3
a_i	number of arguments per predicate in $G_{i=1,2}$	2	2

Results and Discussion

We next present a list of fixed-parameter (fp-) tractability and intractability results for sets of parameters selected from Table 1. Proofs of all these results can be found in *Supplementary Materials* published online.[6] We start with the results for general input structures. Subsequently we also consider

[6] http://www.nici.ru.nl/~irisvr/supplement08.pdf

results for predicate structures consisting of ordered[7] predicates only. It will become clear why this has relevance for SMT.

Structure mapping in general is

1. fp-intractable for parameter set $\{h, a, f, s\}$
2. fp-intractable for parameter set $\{\frac{n}{h}\}$
3. fp-tractable for parameter set $\{n_1\}$
4. fp-intractable for parameter set $\{n_2, r, h, a, p\}$

Result (1) means that the height of the predicate structures (h), the arity of predicates (a), the frequency of predicate labels (f), and the number of non-root predicates ($s = p - r$, where p is the total number of predicates and r is the number of root predicates) are neither individually, nor combined in any way, a source of complexity for Structure mapping. In other words, even if all these parameters are small then, all else being equal, Structure mapping remains computationally unfeasible for all but small inputs. We particularly note two important implications for the SMT literature: First, even if repetitiveness in predicate structures introduces difficulties for structure-mapping, as proposed by Falkenhainer et al. (1989, p. 47), then this will not be due to the repetitiveness of predicate labels alone, and second, the natural bound on arity in human mental representations assumed by Halford et al. (1998) is insufficient to yield analogical mappings computationally tractable for human minds, at least for analogical mappings as construed by SMT.

Interestingly, we have also Result (2), which shows that contrary to the conjecture of Falkenhainer et al. (1989), the flatness of a predicate structure (measured by $\frac{n}{h}$) is *not* a source of complexity for Structure mapping. Also, contrary to the conjecture that the number of vertices do not matter much, we have Result (3), showing that the number of vertices—at least in the larger of the two predicate structures, i.e., n_1—is a source of complexity. We should qualify, however, that it is impossible to have large input if parameter n_1 is small, since the whole input size $n_1 + n_2$ is bounded by $2 \times n_1$. Since intractability is only an issue for non-small inputs, the observation that n_1 is a source of complexity is more or less redundant. We see that if we switch from n_1 to n_2 (i.e., the number of vertices in the smaller of the two predicate structures) then the ability to confine the non-polynomial complexity in Structure mapping to the parameter is lost (Result (4)). Also, Result 4 shows that the number of root predicates (r) is not a source of complexity, not individually nor combined with any of the parameters h, a, n_2, p. In other words, like its relative flatness, the large number of roots in the Composer-General example in Figure 1d fails to explain why inputs of this type (or any type) are "hard" for Structure mapping. We next present a result that *can* explain the apparent hardness of this type of input.

[7] A predicate is said to be ordered if the order of its arguments matter, otherwise it is said to be unordered. For example, the predicate AND(X,Y) is unordered, but GREATER(X,Y) is ordered.

Structure mapping for predicate structures with ordered predicates is

5. fp-tractable for parameter set $\{o\}$

To interpret Result (5), first observe that the Composer-General example in Figure 1d contains only ordered predicates. This means that if we want to explain why this type of input is hard we may assume w.l.o.g. that we are dealing only with predicate structures with ordered predicates. Then Result (5) naturally explains why Structure mapping algorithms run long for this type of input. After all, Result (5) shows that the parameter o (the number of objects in the predicate structures) is a source of complexity for Structure mapping of predicate structures with ordered predicates, and o is relatively large in the Composer-General predicate structures.

At this point, the reader may wonder if perhaps the parameters shown not to be sources of complexity for Structure mapping in general (Results (1), (2), and (4)), may also turn out to be sources of complexity if inputs are constrained to predicate structures with ordered predicates only. This is not the case, however, as is evidenced by Result (6).

6. Results $1-4$ hold even if the predicate structures contain ordered predicates only

From Result (6) we conclude that of all the parameters that we have considered in this paper, the relatively large size of only one of them (viz., o) yields a parsimonious explanation of the apparent 'hardness' of Structure mapping for inputs of Composer-General type. Admittedly, the parameter n_1 could in principle be used to explain 'hardness' of this type (or any type!) of input as well, but it would hardly be parsimonious, because $n_1 \geq o + p$ and courtesy of Result (5) we know that o already suffices to capture the non-polynomial complexity inherent in the Structure mapping problem for inputs like the Composer-General example.

We remark that Result (5) does not yet explain why Structure mapping for inputs of the type shown in Figure 1a/b (the Rutherford example) is "easy", because these predicate structures contain *un*ordered predicates (viz., AND(X,Y)). Therefore it is of interest to note that we have the following result which establishes that Result (5) also holds for general inputs with both ordered and unordered predicates.

Structure mapping in general is

7. fp-tractable for parameter set $\{o\}$.

Result (7) yields a natural explanation of why inputs like the Rutherford example make for easy structure-mapping, viz., because the number of objects in base and target is small (in this case, $o_1 = o_2 = 2$).

In sum, with our analyses we have shown that several (intuitively plausible) conjectures about what makes structure-mapping computationally difficult are incorrect. In addition, our results show that the relative difficulty of the Composer-General example compared to the Rutherford examples can be parsimoniously explained by the difference in number of objects in the predicate structures.

Conclusion

Often algorithmic simulations of computational-level theories give good, first guesses about input aspects that cause the computational problem defined by a cognitive theory to be computationally intractable. However, to validate those guesses we need evidence that we have actually identified sources of intractibility in the computational *problem*, rather than artifacts of an inadvertently inefficient implementation of the theory.

In this paper, we have illustrated how parameterized complexity theory provides some useful analytical tools that can help substantiate claims or intuitions about what makes a given problem hard or easy. The same tools also can help us discover when our intuitions about sources of intractability are in fact mistaken. That such intuitions can be mistaken, even after considerable simulation tests, is illustrated by our results for SMT and how they bear on existing conjectures in the literature about potential sources of intractability in this theory.

Intuitions about sources of intractability may be more often mistaken than we realize. Not only are people poor at intuiting the speed of combinatorial expansion,[8] but to pinpoint exactly which aspects of representational structures are responsible for (or contribute to) computational intractability one needs to understand the subtle *interaction* between a combinatorially complex domain and the problem to be solved for that domain. This is a highly non-trivial task. It is not for nothing that a whole branch of mathematics is devoted to building tools and concepts for performing exactly this task.

We think that cognitive scientists can greatly benefit from adopting the tools of parameterized complexity theory, as many cognitive theories are known to face computational intractability for unrestricted domains. If a computational-level theory can be shown to be tractable under certain input constraints and there is empirical evidence that inputs are indeed so constrained for human cognizers, then the theory can maintain a status of psychological and computational plausibility, despite its intractability for unrestricted domains.

References

Arora, S., & Barak, B. (in press). *Computational complexity: A modern approach*.

Downey, R. G., & Fellows, M. R. (1999). *Parameterized complexity*. Berlin: Springer.

Downey, R. G., Fellows, M. R., & Langston, M. A. (2008). The Computer Journal special issue on parameterized complexity: Foreword by the guest editors. *Computer Journal*, *51*, 1–6.

Evans, P. A., Gedge, J., Müller, M., van Rooij, I., & Wareham, T. (2008). *On the computational complexity of analogy derivation in structure-mapping theory* (Tech. Rep. No. 2008-003). Department of Computer Science: Memorial University of Newfoundland.

Falkenhainer, B., Forbus, K. D., & Gentner, D. (1989). The structure-mapping engine: Algorithm and examples. *Artificial Intelligence*, *41*, 1–63.

Flum, J., & Grohe, M. (2006). *Parameterized complexity theory*. Berlin: Springer.

Forbus, K. D., & Gentner, D. (1989). Structural evaluation of analogies: What counts? In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 341–348). Mahwah, NJ: Erlbaum.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. San Francisco, CA: W.H. Freeman.

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

Halford, G. S., Wilson, W. H., & Phillips, W. (1998). Processing capacity defined by relational complexity. *Behavioral & Brain Sciences*, *21*, 803–831.

Hamilton, M., Müller, M., van Rooij, I., & Wareham, T. (2007). Approximating solution structure. In E. Demaine, G. Z. Gutin, D. Marx, & U. Stege (Eds.), *Structure Theory and FPT Algorithmics for Graphs, Digraphs, and Hypergraphs.* Schloss Dagstuhl, Germany: Internationales Begegnungs- und Forschungszentrum fur Informatik.

Sloper, C., & Telle, J. A. (2008). An overview of techniques for designing parameterized algorithms. *Computer Journal*, *51*, 122–136.

Tversky, A., & Kahneman, D. (1973). Availability: A heuristic for judging frequency and probability. In D. Kahneman, P. Slovic, & A. Tversky (Eds.), *Judgment under uncertainty: Heuristics and biases.* Oxford University Press.

van Rooij, I. (in press). The tractable cognition thesis. *Cognitive Science*.

van Rooij, I., Stege, U., & Kadlec, H. (2005). Sources of complexity in subset choice. *Journal of Mathematical Psychology*, *49*, 160–187.

van Rooij, I., & Wareham, T. (in press). Parameterized complexity in cognitive modeling: Foundations, applications and opportunities. *Computer Journal*.

van Rooij, I., & Wright, C. D. (2006). The incoherence of heuristically explaining coherence. In *Proceedings of the 28th Annual Conference of the Cognitive Science Society* (p. 2622).

Veale, T., & Keane, M. T. (1997). The competence of sub-optimal theories of structure mapping on hard analogies. In *Proceeding of the 1997 International Joint Conference on Artificial Intelligence.*

Veale, T., O'Donoghue, D., & Keane, M. T. (1999). Computability as a limiting cognitive constraint: Complexity concerns in metaphor comprehension about which cognitive linguists should be aware. In E. M. Hiraga, C. Sinha, & S. Wilcox (Eds.), *Cultural, psychological and typological issues in cognitive linguistics.* Amsterdam: John Benjamins.

[8] For example, Tversky and Kahneman (1973) found that people estimate $1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 \times 8$ to be about 500, while it is more than 40,000.