

Computational Approaches to Analogical Reasoning
Current Trends

Henri Prade and Gilles Richard (Eds.)
University of Toulouse, France
IRIT-CNRS
118 Route de Narbonne
31062 Toulouse Cedex 9 (France)

Acknowledgements

This book originates from an international workshop on Similarity and Analogy-based Methods in AI (SAMAI'12), co-located with the 20th European Conference on Artificial Intelligence (ECAI'12), held in Montpellier, France on August 27, 2012.

The contributions of this book include revised and extended versions of nine of the ten papers that were presented at the workshop, together with five invited contributions in order to improve its overall coverage.

We are grateful to all the authors for their willingness to participate to this book, and for the time and efforts they have spent to prepare their contribution.

All contributions have benefited of useful comments which had helped the authors improving their papers substantially through one or two rounds of revision. The reviewers were primarily contributors of the other chapters in the book.

Moreover, the volume has also greatly benefited from the useful comments and suggestions of a number of additional reviewers: Stergos D. Afantinos, Danushka Bollegala, Antoine Cornuéjols, Jean Paul Delahaye, Olivier Gasquet, Bruno Gaume, Lluis Godo, Eyke Hüllermeier, Mark Keane, Gabriele Kern-Isberner, Matt Klenk, Maithilee Kunda, Arun Majumdar, Ramon Lopez de Mantaras, Andrew Lovett, Diarmuid O'Donoghue, Yoshiaki Okubo, Praveen Paritosh, Eyal Sagi. We are especially grateful to all of them.

We are grateful to Janusz Kacprzyk for hosting this volume in the Springer-Verlag series: Studies in Computational Intelligence.

The editors, Toulouse, Dec. 2013.

Contents

A short introduction to computational trends in analogical reasoning
Henri Prade, Gilles Richard

Part 1: Analogy in action

Analogies between binary images: Application to Chinese characters
Yves Lepage

Issues in analogical inference over sequences of symbols: A case study on proper name transliteration
Philippe Langlais, François Yvon

Analogy as an organizational principle in the construction of large knowledge-bases
Tony Veale, Guofu Li

Analogy in automated deduction: A survey
Thierry Boy de la Tour, Nicolas Peltier

Part 2: Modeling analogy

Applying belief revision to case-based reasoning
Julien Cojan, Jean Lieber

Heuristic-driven theory projection: An overview
Martin Schmidt, Ulf Krumnack, Helmar Gust, and Kai-Uwe Kühnberger

Completing rule bases using analogical proportions
Steven Schockaert, Henri Prade

From analogical proportion to logical proportions: A brief survey
Henri Prade, Gilles Richard

Analogical proportions in a lattice of sets of alignments built on the common subwords in a finite language
Laurent Miclet, Nelly Barbot, and Baptiste Jeudy

Part 3: From cognition to computational experiments

How similar is what I get to what I want : Matchmaking for mobility support
Ute Schmid, Lukas Berle, Michael Munz, Klaus Stein, and Martin Sticht

A computational strategy for fractal analogies in visual perception
Keith McGreggor, Ashok K. Goel

An information-processing theory of interactive analogical retrieval
Swaroop S. Vattam, Ashok K. Goel

Analyzing Raven's intelligence test: Cognitive model, demand, and complexity
Marco Ragni, Stefanie Neubert

Perceptual similarity and analogy in creativity and cognitive development
Georgi Stojanov, Bipin Indurkhyā

List of Authors

Nelly Barbot
IRISA-Cordial
Lannion, France
barbot@enssat.fr

Lukas Berle
Faculty Information Systems
and Applied Computer Science
University of Bamberg, Germany
lukas-florian-markus.berle@stud.uni-bamberg.de

Thierry Boy de la Tour
University of Grenoble, CNRS/LIG, France
thierry.boy-de-la-tour@imag.fr

Julien Cojan
Université de Lorraine, LORIA, UMR 7503
54506 Vandœuvre-lès-Nancy, France
CNRS - 54506 Vandœuvre-lès-Nancy, France
INRIA - 54602 Villers-lès-Nancy, France
julien.cojan@gmail.com

Baptiste Jeudy
Laboratoire Hubert Curien
Université de Saint-Etienne, France
baptiste.jeudy@univ-st-etienne.fr

Ashok Goel
Design & Intelligence Laboratory
School of Interactive Computing
Georgia Institute of Technology
Atlanta, GA 30332, USA
goel@cc.gatech.edu

Helmar Gust
Institute of Cognitive Science
University of Osnabrück, Germany
Helmar.Gust@uni-osnabrueck.de

Bipin Indurkhy
Computer Science department
AGH University of Science and Technology
Cracow, Poland
bipin@agh.edu.pl

Ulf Krumnack
Institute of Cognitive Science

University of Osnabrück, Germany
Ulf.Krumnack@uni-osnabrueck.de

Kai-Uwe Kühnberger
Institute of Cognitive Science
University of Osnabrück, Germany
kai-uwe.kuehnberger@uni-osnabrueck.de

Philippe Langlais
Département d'informatique et
de recherche opérationnelle
Université de Montréal, Canada
felipe@iro.umontreal.ca

Yves Lepage
Graduate school of Information
Production and Systems
Waseda University
808-0135 Hibikino 2-7
Wakamatsu-ku, Kitakyushu-shi
Fukuoka-ken, Japan
yves.lepage@waseda.jp

Guofu Li
School of Computer Science and Informatics
University College Dublin
Belfield, Dublin D2, Ireland
li.guofu@gmail.com

Jean Lieber
Université de Lorraine, LORIA, UMR 7503
54506 Vandœuvre-lès-Nancy, France
CNRS - 54506 Vandœuvre-lès-Nancy, France
INRIA - 54602 Villers-lès-Nancy, France
Jean.Lieber@loria.fr

Keith McGregor
Design & Intelligence Laboratory
School of Interactive Computing
Georgia Institute of Technology
Atlanta, GA 30332, USA
keith.mcgregor@gatech.edu

Laurent Miclet
IRISA-Dyliss
Rennes, France

miclet@enssat.fr

Michael Munz
Faculty Information Systems
and Applied Computer Science
University of Bamberg, Germany
Michael.Munz@uni-bamberg.de

Stefanie Neubert
Ludwig-Maximilians-Universität
München, Germany
stefanie.neubert@campus.lmu.de

Nicolas Peltier
University of Grenoble, France
CNRS / LIG
Nicolas.Peltier@imag.fr

Henri Prade
University of Toulouse, France
IRIT-CNRS
prade@irit.fr

Marco Ragni
University of Freiburg
Center for Cognitive Science
Friedrichstr. 50
79098 Freiburg, Germany
ragni@informatik.uni-freiburg.de

Gilles Richard
University of Toulouse, France
IRIT-CNRS
richard@irit.fr

Ute Schmid
Faculty Information Systems
and Applied Computer Science
University of Bamberg, Germany
Ute.Schmid@uni-bamberg.de

Martin Schmidt
Institute of Cognitive Science

University of Osnabrück, Germany
Martin.Schmidt@uni-osnabrueck.de

Steven Schockaert
School of Computer Science & Informatics
Cardiff University, 5 The Parade, CF24 3AA
Cardiff, United Kingdom
s.schockaert@cs.cardiff.ac.uk

Klaus Stein
Faculty Information Systems
and Applied Computer Science
University of Bamberg, Germany
Klaus.Stein@uni-bamberg.de

Martin Sticht
Faculty Information Systems
and Applied Computer Science
University of Bamberg, Germany
Martin.Sticht@uni-bamberg.de

Georgi Stojanov
The American University of Paris
147, rue de Grenelle
75007 Paris, France
gstojanov@aup.edu

Swaroop S. Vattam
Design & Intelligence Laboratory
School of Interactive Computing
Georgia Institute of Technology
Atlanta GA 30308, USA
svattam@cc.gatech.edu

Tony Veale
Web Science and Technology Division
KAIST
Yuseong, Korea
tony.veale@gmail.com

François Yvon
Université Paris Sud & LIMSI/CNRS, France
yvon@limsi.fr

A short introduction to computational trends in analogical reasoning

Henri Prade and Gilles Richard

University of Toulouse, IRIT-CNRS, France, email: {prade, richard}@irit.fr

For a long time, identifying analogies and reasoning by analogy have been recognized as major cognitive capabilities of human mind. As such, analogy has been widely studied and discussed, in particular by philosophers. More recently, it has attracted a lot of attention in cognitive sciences, and in computer sciences, especially artificial intelligence. The aim of this volume is to report on current advances regarding the computational aspects of analogy. Before presenting the structure and the contents of the book, we provide a short historical account of researches on analogy, focusing on a modeling perspective.

1 The emergence of computational models

Roughly speaking, the first formal attempts at modeling analogy seem to appear in the second half of the XX^e century. Before reporting on formal models and computational applications, we give a brief overview of works in philosophy.

1.1 From philosophical roots to cognitive concerns

Aristotle seems to be the first philosopher, at least in the western world¹, to discuss analogical reasoning in a substantial way. In different places [9, 10], Aristotle makes use of quantitative (geometric) proportions, and also considers comparative relations between four terms expressing an identity of relation between two ordered pairs, which form in modern words an *analogical proportion* (i.e., statements of the form “*A* is to *B* as *C* is to *D*”). For instance, in the Book 1 (Part 17) of his *Topics* [9], when discussing the idea of “likeness”, Aristotle provides examples of proportions between things belonging to different genera; see [100] for a modern account on Aristotle’s view of analogy. In medieval Scholastic philosophy [12], the studies of analogy have continued and, in particular, Thomas Aquinas used the analogy of proportion (see, e.g., [138, 156]) in metaphysical and theological contexts. After Aristotle, different forms of analogy have been identified (see [174, 173]), and further discussed by the theologian and philosopher Cajetan just after the end of the Middle Ages [104].

From the beginning, analogical proportions were understood not only as a matter of things having common properties, but also as a matter of similarity between relations. This was made clear, for instance in Diderot and D’Alembert’s *Encyclopédie*, where, in the article “Analogie”, one can read “Les scholastiques définissent l’analogie,

¹ Indeed, it would be fair to mention at least the Chinese philosopher, Meng Tzeu (Mencius, in English), who lived in the same century as Aristotle.

une ressemblance jointe à quelque diversité” [58]; however, the qualitative use of the word “proportion” for objects, is only briefly mentionned in the article “Proportion” [57], beside its use in mathematics or aesthetics. This point is also emphasized by Kant when he wrote [121] (pp.146-7): “This type of cognition is cognition according to analogy, which surely does not signify, as the word is usually taken, an imperfect similarity between two things, but rather a perfect similarity between two relations in wholly dissimilar things”. Most books on logic in the 19th century include a section on analogical reasoning, which already contains the main basic ideas on the topic, see, e.g., [163].

The idea of analogy has also been extensively discussed by modern time philosophers, highlighting the notions of analogy relation between objects and of analogical proportions, as well as their use in reasoning, see, e.g., [105, 59, 82, 196, 206]. The merits of analogical reasoning in physics (see e.g., [177]), already mentioned in [58], and more generally in sciences [101, 102, 1, 64] have been particularly pointed out, with its limits, as in [33, 28] for instance. Furthermore, the links between analogy, generalization or induction, and confirmation theory have been investigated in formal philosophy [36, 5, 6, 99, 238], thus in relation with a probabilistic counting. Initially discussed in the perspective, just mentioned, of scientific discovery and confirmation theory, the idea of argument from analogy also considered in the studies of human thought and writings [150, 175, 39], has been widely discussed in argumentative reasoning [207, 23, 241, 16, 231]. The role of analogy has been especially studied in legal reasoning and argumentation [176, 26, 77, 230, 232] as well as in social and human sciences [56, 60]. In this view, as emphasized by C. Perelman, “the nature of analogy is not logical but rhetorical” [77], while for J. B. Grize [92] it has an important place in “natural logic”, quite different from formal logic.

Besides, human ability for analogy making has been a topic of interest in psychology for a long time, as well as in linguistics, e.g., [67, 66]. In linguistics and rhetorics, analogy comes close to metaphor as a figure of speech that transfers the meaning of one thing directly on another one [20]. In psychology, it has inspired different tests aiming at evaluating IQ or general cognitive ability, such as Raven’s progressive matrices [194, 193] or Miller analogies test [164]. Besides, recognized as a tool for discovery in science, analogy has also been advocated as an explanatory device in knowledge acquisition [85], and as such is an important topic of research in instructional science, see e.g., [50, 63, 73, 89, 31, 95, 204, 87, 227, 137, 119, 88]. Beyond these practical applications, analogical reasoning has become an area of research in itself in cognitive sciences [86, 81, 110, 109], with a special emphasis on its role in problem solving [83, 42], in schema induction, prediction, and creativity [84, 116, 117], or on the links between analogy and categorisation [199].

1.2 Modeling analogy

Analogical reasoning is often regarded as a heuristic device, which may explain that the number of attempts at modeling it in a formal way is comparatively smaller than the number of works in logical reasoning. Still there have been different types of approaches to the formal modeling of analogical reasoning. First, it has been early recognized that analogical proportions enjoy symmetry and central permutation as characteristic properties (see, e.g. [59]), while its transitivity is more debatable (see, e.g. [228]). One of

the first proposals in analogy-based reasoning is probably due to G. Polya, who emphasized and analyzed the role of analogy in the solving of mathematical problems [181] and proposed patterns of plausible reasoning involving analogy relations between statements [182, 118, 186]. Another early attempt at providing a semantical discussion of analogy, in a quite different perspective, has been proposed by the logician J. Bochen-ski [22]. This author introduces a formal logic language for clarifying issues in the traditional theory of analogy developed by Thomas Aquinas, Cajetan and their followers, where identities and differences between two structured situations represented by ‘semantic complexes’ (involving ‘names’, ‘languages’, ‘contents’ and ‘things’) are instrumental. Such a view has been rediscussed in [224] by quantifying the extent to which two items are analogous in terms of the number of dimensions where they are identical. Such a view then becomes closer to the notion of similarity, which was fully formalized later by A. Tversky [222]. See [229] for a collection of papers dealing with similarity and / or analogy. An original view of analogical proportions was proposed by the philosopher M. Hesse who outlined a lattice-based setting for computing the fourth term of such a proportion from the three others [98], anticipating a formal view which was reintroduced much later [214, 15].

A special mention is also due to the psychologist J. Piaget. He was the first to provide a logical counterpart to the idea of numerical proportion. Indeed in the annex of [179] and also in [180] (pp. 35–37), the following definition of a so-called *logical proportion* is given: four propositions A , B , C , and D make a logical proportion if the two following conditions hold $A \wedge D \equiv B \wedge C$ and $A \vee D \equiv B \vee C$. However, this logical proportion, which turns to be one of the possible (equivalent) definitions of an analogical proportion $A : B :: C : D$ in a Boolean setting [162] is introduced in these books without any reference to analogy².

Formal anthropology is another area of research where attempts at formalizing analogy were presented and discussed, especially at a time where the structuralism view was prominent [151, 65, 125, 129, 127, 128, 242, 191]. In this research trend, often ignored in the analogy literature, especially remarkable is the work of S. Klein [125–127], a computer scientist with a background in anthropology and linguistics, who introduced a so-called ATO operator (where ATO stands for “Appositional Transformation Operator”). This Boolean operator, which is based on the logical equivalence connective, amounts to compute the 4th argument of an analogical proportion between Boolean vectors (describing the items in terms of binary features) by applying $D = C \equiv (A \equiv B)$ componentwise (even if it was discovered later [162] that this calculation does not always fit with the correct definition of an analogical proportion).

An important renewal in the modeling of analogy has come from cognitive science in the last three decades. At the forefront of these proposals, three leading approaches should be especially mentioned: the structure mapping theory (SMT) proposed by D. Gentner [79, 80], the analogical constraint mapping approach proposed by K. Holyoak and P. Thagard [112, 218]), and the model of analogy making based on the idea of the

² However, in a previous book [178] (pp. 97–99), Piaget informally investigates a similar idea, even using an example of linguistic analogical proportion, still without explicitly mentioning analogy.

parallel terraced scan developed by D. Hofstadter and M. Mitchell [108, 165, 166]; see [4] for a comparative view in a scientific modeling and discovery perspective.

Structure mapping theory views an analogy as a mapping between a source and a target domain. The associated structure-mapping engine (SME) returns the correspondences between the constituents of the base and target descriptions (expressed in terms of relations, properties, and functions), a set of candidate inferences about the target according to the mapping, and a structural evaluation score. Such a view is closely related to the idea of structural similarity [215], and has been also advocated in artificial intelligence [239, 240, 70, 44, 45]; see also [235] for a discussion about the interest of isomorphic structures when comparing situations. Besides, the view of analogy as a constraint satisfaction process, also defended in [115, 223], is at work in the analogical constraint mapping engine (ACME), which represents constraints by means of a network of supporting and competing hypotheses regarding what elements to map, and where an algorithm identifies mapping hypotheses that collectively represent the overall mapping that best fits the interacting constraints [112, 111].

Roughly speaking, following [75], one may distinguish between three broad groups: i) the symbolic models that establish a structural similarity between the source and target descriptions generally expressed in formal logic terms, as SME ; ii) the connectionist models well suited for representing relational structures with nodes and links between nodes, as in ACME from using a kind of neuron network-like structure, or in LISA [114] the strong constraint of pairwise connection between objects is relaxed to partial and dynamic connections (see [75] for other references) ; iii) the hybrid models relying on a combination of the previous approaches. These latter models generally use constraint satisfaction networks, explore competing hypotheses and are stochastic in nature. They rather focus on the optimization process at work to extract the most plausible solution. Moreover, this type of approach naturally embeds a graded view of similarity, while symbolic approaches have generally difficulties to handle similarity beyond mere identity. The COPYCAT project [108, 165] is probably one of the most well-known attempt of analogy-making program falling in the hybrid category. Based on a similar paradigm, let us also mention Tabletop [76, 74], and NARS [233].

In a similar spirit, Cornuejols [46] is the first to establish a link between analogical reasoning and Kolmogorov complexity (see also [184, 17]), by relying on a minimum description length principle when evaluating the complexity of candidate solutions for completing analogical proportions. This approach is in complete line with the works in [40] in which “choose the pattern that provides the briefest representation of the available information” is the rational basis of a wide range of cognitive processes, including similarity-based reasoning. The idea of distance minimization was already proposed in [197], where an analogical proportion is supposed to hold if the vector distance between the first two components is the same as the distance between the last two ; then given the three first elements of an incomplete proportion, there exists an ideal solution, and the choice of the best solution for completing it amounts to minimize the distance of a candidate solution to the ideal one ; see also [21].

On the symbolic side, a logical view of analogical inference based on first order logic has been investigated in [55, 198], although analogical reasoning is not amenable to a formal logic framework in a straightforward manner due to the brittleness of its

conclusions. Considering two particular items s and t that share a common property P (i.e. $P(s)$ and $P(t)$ hold), where moreover s satisfies an additional property Q (i.e. $Q(s)$ holds), then, by an analogical jump, t should satisfy Q (i.e. $Q(t)$ should hold). The legitimacy of this conclusion cannot be insured without some external background knowledge (which alone should not be sufficient to entail the property $Q(t)$, i. e. without the use of $P(s)$ and $Q(s)$). The weakest external conditions of this kind making the inference scheme valid have been identified. Still they remain extremely strong, and are not far from expressing a functional dependency relation. Of a different nature, and more in line with the structure mapping theory is the Heuristic-Driven Theory Projection (HDTP) framework [201, 93, 236, 134], where higher-order features and anti-unification are added to the classical first order logic framework. In this approach, all the items are represented as first order terms, and an analogical proportion $A : B :: C : D$ holds if and only if we can find two patterns (i.e. terms) P and Q such that $P\sigma_1 = A$, $P\sigma_2 = C$ (i.e. σ_1 and σ_2 denote higher order substitutions and P generalizes A and C), and $Q\sigma_1 = B$, $Q\sigma_2 = D$ (i.e. Q generalizes B and D). Then solving an analogical proportion equation, where D is unknown, is based on an anti-unification process for extracting P and Q from A, B, C . Another approach that refers to unification is the one proposed in [53] whose starting point is a general theory of pattern perception. Besides, the completion of the fourth term of an analogical proportion can be seen as a particular instance of matrix abduction problems [3, 187].

A formal study of analogical proportions, in a set-theoretic perspective, obeying the three basic postulates, symmetry, central permutation, and unicity of the solution of $A : B :: A : x$, has been first proposed in [144, 145] for computational linguistics purposes, and further developed in [244, 246, 213, 245]. These authors have proposed different definitions, which turn out to be equivalent. The main one, due to [213], reads as follows. Let A, B, C, D be subsets of features, the analogical proportion $A : B :: C : D$ holds if and only if there exist four subsets U, V, W and Z , such that $A = U \cup V$, $B = U \cup W$, $C = Z \cup V$, $D = Z \cup W$. An algebraic, factorization-based, generalization of this definition of analogical proportion has been proposed and applied to various structures ranging from semi-groups to lattices, including words over finite alphabets and finite trees [212, 213, 161, 14, 19]. Besides, the above set-theoretic definition has been later restated in a different, simpler but equivalent way, as $A \setminus B = C \setminus D$ and $B \setminus A = D \setminus C$, and the propositional logic counterpart of this definition has also been studied in [162], then leading to the study of the general notion of logical proportions that encompasses analogical proportions [185, 187, 188].

1.3 Computational applications

With the progress in the understanding of the mechanisms underlying it, analogy has occupied an important place in artificial intelligence, as a tool for problem solving, automated deduction and learning, especially in the 1970's and 1980's (see [94, 96] for a survey and a collection of representative papers). Later, the interest for analogy in artificial intelligence has particular focused on case-based reasoning, has also been acknowledged in computational linguistics, and in other domain-oriented fields.

The interest for the heuristic use of analogy in problem solving [41, 157, 158], as emphasized in the last reference, relies on the fact that it is “particularly valuable in ill-

defined, complex or new situations because it is easier to adapt a source such that it fits the conditions of the target than to solve a problem from first principles”. It may also help to overcome an often explosive search complexity, in model building and theory formation [78, 52].

The role of analogy as an essential tool for creativity also in mathematics [171] has long been stressed by Polya [181], but the first computerized attempt to use analogy for automatic theorem proving dates from the work of [131]. It has been followed by a long list of developments, among which we can cite [32, 167, 29, 30, 91], and more recently [27, 159, 237], even leading to complete implementation in a logic programming framework as in [216, 48]. The interested reader is referred to [120] for a detailed survey.

A system capable of assimilation and accommodation by means of analogy was early proposed in [155] for coping with new situations. Case-based reasoning, e.g. [132, 143, 133, 142, 2, 148, 51], has become a popular form of analogical reasoning, where the solving of a new problem is based on the solutions of similar past problems stored in a repertory. In a transformational analogy process, solutions to closely related problems are retrieved and adapted to satisfy the requirements of the new problem, while derivational analogy is a reconstructive method that takes advantage of past problem-solving experience [35, 200]. With the latter strategy, a target problem is solved by modifying and composing past successful plans, rather than by transferring a complete solution. By extracting knowledge from past successful problem solving situations that bear strong similarity to the current problem, an analogical transformation process may acquire generalized plans from solutions attempts to similar problems, thus leading to a learning process [34]. Besides, fuzzy set-based methods have been advocated for handling similarity in case-based reasoning and decision, as well as adaptation [61, 62, 113, 24], while fuzzy set-based approximate reasoning has been itself analyzed as an analogical reasoning process [25]. Besides, relations and complementarities between non monotonic and analogical forms of reasoning have also been investigated [190, 124, 186].

Since analogy has strongly been advocated as an explanatory device for human learning, its use for machine learning purposes has also been a subject of interest in diverse ways. Analogy-based learning pioneered by [18], ranges from classification tasks as in [212, 214, 160, 189] (including the handling of numerical features in the latter reference), to more symbolic learning tasks where, for instance, a system acquires unknown rules as in [123, 103, 183], or even produces images [153].

Considered as a powerful heuristic device, analogical reasoning has been investigated in artificial intelligence for a long time in relation with the solving of IQ tests. Evans’ “Analogy” program [68, 69] is the first attempt at implementing an analogy-solving program using of a knowledge representation language. It was designed for solving a class of IQ puzzles of the form “ A is to B as C is to which of D_1 , or D_2 or ... D_n ?” involving simple geometric figures. The program, starting from a representation of A and B , first looks for transformation rules starting from A and leading to B , and from C to each D_i . Then it selects the D_i associated with the set of rules that are the most similar to the ones linking A and B . This was the starting point of a research line about reasoning with geometric analogies [169, 168]. More sophisticated IQ

tests, the Raven's progressive matrices [193], have motivated successful proposals for the automatic solving of such tests, either by using rules elicited from people succeeding in the tests [37], by applying the structure mapping approach [154, 152], by taking advantage of the cognitive architecture ACT-R [8] (and then producing predictions of human performance) [211, 192], or by implementing the logical view of analogical proportions [47] (a set of candidate solutions is not required in this latter case). Moreover, let us mention the Affine and Set Transformation Induction (ASTI) model [136, 135] that uses transformations on (pixel-based) imagistic representations to compare images (on the basis of a similarity metric), induce transformations, and make predictions as the best fit among candidate answer images. Besides, Bennett mechanical comprehension tests have also been considered [130].

When it comes to natural language processing, the role of analogy is manyfold. Approaches to the problems of identifying lexical analogies [221, 220], or of generating them [43] have been proposed ; see for discussions [106, 107]. Within an intra-domain context, analogical matching can be used to create new categories within a given taxonomy or semantic network [225, 226, 210]. When dealing with different languages, the use of analogical proportion shows promising results in the difficult task of language translation [144, 146, 219, 71, 139, 141, 140]. Generally speaking, the usage of analogy in computational linguistics, initiated in [72, 243, 145], is a lively domain, and can still lead to challenging view as in [122]. Instead of dealing with raw texts, the Vivomind engine of [209], relying on a high level representation (conceptual graphs), allows language understanding and scene recognition, and even supports classical inferences as soon as analogies have been identified.

With the emergence of huge natural language corpora, new issues arise which cannot only be dealt via standard computational linguistics techniques. Still here analogy has its own role: for instance, in the frequent cases where there is no exact match or even no match at all for a given query, it can be used to suggest approximate answers [234, 149]. More generally, wherever we have to compromise on a final answer, analogical reasoning can bring new solutions [208]. Ultimately, it may also help to complete missing values by suggesting plausible options [11].

Vision is also a crucial part of human perception where analogy plays its own role. This leads to another way of thinking about cognitive models of analogy by distinguishing models using (i) logical-style representations (e.g., SMT), (ii) imagistic representations [217, 49, 135] acknowledging the fact that mental imagery is also an important part of the analogy-making process, or (iii) mixed representations as in [54, 116]. Diverse applications of visual analogy have to be mentioned: as a low level image processing tool [97] for design or architecture as in [90, 38]), or still to help solving geometric problems using existing models as in [205, 168, 172].

Finally, in topics such as software engineering, for instance, analogy is used to design comparative statements between past and present projects. This helps estimating project performance, providing better indicators than classical methods [13, 147]. Quantitative estimation has also been considered [170], while a qualitative approach close to analogical reasoning, which has been recently proposed, enables the extrapolation of rough estimates [203, 202].

This brief panorama highlights a large range of current applications for analogical reasoning, supporting once again, the claim that analogy is not only a crucial cognitive ability of human being, but also has to play an important role in the design of a computational intelligence.

2 Contents of the volume

This volume originates in an international workshop on Similarity and Analogy-based Methods in AI (SAMAI'12), co-located with the 20th European Conference on Artificial Intelligence, and held in Montpellier on August 27, 2012 [195]. Almost two thirds of the research articles gathered in this book are revised and expanded versions of all the workshop papers (with the exception of [7]), while the remaining third are made of invited articles. All contributions have benefited of useful comments which had helped the authors improving their papers substantially through one or two rounds of revision.

This volume contains 14 contributions which have been organized within 3 groups.

In the first main section of this volume “Analogy in action”, we have gathered recent works related to the application of analogy to diverse fields of research. In the first paper of this group, Y. Lepage, taking into account the inherent complexity of the analogy-making process, designs a new method to extract analogies, applying it to sets of binary images representing Sino-Japanese or Chinese characters. Overcoming this complexity, Lepage shows that the analogical proportions holding between images lead to rediscover the coherent structure of these characters, paving the way for other natural language processing tasks.

The second article by P. Langlais and F. Yvon investigates some of the issues behind the scene of analogical learning, other than its well known high complexity. Still, the authors suggest the use of analogical proportions as a powerful tool for natural language processing tasks, provided that satisfactory solutions to the raised issues are found.

In some sense, the paper by T. Veale and G. Li supports the previous one. It relies on the assumption that analogy is an accurate way to build up or to extend large knowledge bases. Analogy-guided acquisition is shown as a serious candidate tool for enriching a knowledge base, while maintaining well-balance, coherence and wide coverage.

In another area, the different attempts which have been done in order to guide automatic theorem proving via analogical reasoning are surveyed in the last paper of this section, by N. Peltier and T. Boy de la Tour. Despite the fact that everybody agrees about the importance of analogy-making in the process of building proofs, it appears that it is a very tough task to implement it in a formal way.

As briefly sketched at the beginning of this preface, analogy and analogical proportions have been modeled in several ways in the literature. In the second section of this volume “Modeling analogy”, we gather recent papers contributing to this aim.

Focused on the particular case of case-based reasoning, the article by J. Cojan and J. Lieber provides a formal framework where they recast the necessary case adaptation step as a specific form of knowledge revision.

In the article by M. Schmidt, U. Krumnack, H. Gust, and K. Kühnberger, a higher-order logic framework, named ”Heuristic-Driven Theory Projection” (HDTTP for short),

is recast together with its set-theoretic semantics, highlighting the fact that analogy-making process is mainly based on diverse types of substitution. Moreover, similar to many other approaches, HDTp has to use heuristics in order to reduce the computational complexity.

The paper by S. Schockaert and H. Prade provides a new way to interpolate or extrapolate in order to complete a set of rules in a qualitative setting, using analogical proportions and the concepts of betweenness and parallelism in change.

The paper by H. Prade and G. Richard discusses a Boolean modeling of analogical proportion which appears to be a specific case of a more general concept of logical proportion, leading for some of them to applications, from automatic solving of IQ tests to classification.

Starting from the study of the lattice structure of the sets of alignments of sequences, the article of L. Miclet, N. Barbot, and B. Jeudy establishes a connection to the calculus of analogical proportions. Finally, they sketch potential machine learning applications, highlighting once again the predictive power of analogical proportions.

The third section of this volume “From cognition to computational experiments” deals with cognitive aspects of similarity judgments and analogical jumps, reports on implementations, or gives hints for advanced models.

In the first paper, by U. Schmid, L. Berl, M. Munz, K. Stein, and M. Sticht, the authors emphasize the fact that the process of matching distinct entities is not a symmetric process, and as such has to take into account not only the similarities of the two entities, but also the way they differ. This fact relates the proposed view of matchmaking to analogy where dissimilarities play also a key role. Their approach is implemented as a part of a web-based service system.

The second paper, by K. McGregor and A. Goel, deals with perceptual similarity and analogy. The authors revisit visual analogical proportions, using fractal representations of images, and then apply their framework to the solving of the “Odd One Out” intelligence test.

In the third paper, the authors, S. Vattam and A. Goel, study analogical retrieval in the context of biologically inspired design that is the retrieval of a source analogue to the problem at hand. Starting from a field study devoted to bio-design where students were asked to solve engineering problems taking inspiration from a biological source, the authors describe a cognitive architecture for analogical retrieval.

The article by M. Ragni and S. Neubert presents a computational approach for solving Raven’s progressive matrices tests, based on the ACT-R cognitive architecture, which also enables the authors to evaluate a meaningful measure of problem difficulty. Very good experimental results are reported for the approach.

The final paper, by G. Stojanov and B. Indurkhy, brings together a large body of literature on analogy, from philosophy to cognitive sciences and artificial intelligence, emphasizing that analogy represents the core mechanism in human cognition. They stressed that an analogy-making process involves lower-level perceptual features for similarity matching as well as dynamic and flexible representations constructs when needed.

The research and survey articles gathered in this volume illustrate current trends of research about the use of analogy in computational tasks from an artificial intelligence

perspective. It shows a renewal of interest in analogical reasoning and highlights innovative viewpoints. We hope that this volume contributes to the development of further research in the field.

References

1. *Analogie dans les Sciences - Fécondités et Obstacles*. Univ. des Sciences et Techniques de Lille, 2003-2004. Contents: L'analogie dans les sciences du végétal : méthode heuristique ou obstacle (G. Denis), Le modèle du vivant dans la chimie à l'âge classique (B. Joly), L'analogie dans le médioplatonisme (J. Delattre), La fonction de l'analogie dans la Dynamique de Leibniz (A.-L. Rey), Analogies et lois chez Christiaan Huygens (F. Chareix) Maxwell et les “analogies physiques” (S. Devernay and B. Maitte), Analogie et solutions graphiques (D. Tournès), Ressemblance entre objets (J.-P. Delahaye), Analogie entre le calcul différentiel et le calcul algébrique (1ère partie) (J.-P. Lubet and A.-M. Marmier), Analogie entre le calcul différentiel et le calcul algébrique (2e partie) (R. Bkouche), L'analogie dans la théorie des fonctions algébriques de Dedekind et Weber (S. Couche).
2. A. Aamodt and E. Plaza. Case-based reasoning; foundational issues, methodological variations, and system approaches. *AICOM*, 7(1), pages 39–59, 1994.
3. M. Abraham, D. M. Gabbay, and U. J. Schild. Analysis of the talmudic argumentum a fortiori inference rule (kal vachomer) using matrix abduction. *Studia Logica*, 92(3):281–364, 2009.
4. P. Abrantes. Analogical reasoning and modeling in the sciences. *Foundations of Science*, 4:237–270, 1999.
5. P. Achinstein. Variety and analogy in confirmation theory. *Philosophy of Science*, 30 (3):207–221, 1963.
6. J. Agassi. Analogies as generalizations. *Philosophy of Science*, 31 (4):351–356, 1963.
7. L. Amgoud, Y. Ouannani, and H. Prade. Arguing by analogy - towards a formal view. a preliminary discussion. In G. Richard, editor, *Working Papers 1st Int. Workshop on Similarity and Analogy-based Methods in AI (SAMAI'12)*, Aug.27, pages 65–69. Tech. Rep. IRIT/RR-2012-20, 2012.
8. J. R. Anderson. *How Can the Human Mind Occur in the Physical Universe?* Oxford University Press, New York, 2007.
9. Aristotle. *Organon: Categories; On Interpretation; Prior Analytics; Posterior Analytics; Topics; Sophistical Refutations*. Harvard Univ. Pr., 1938. Transl. by H. P. Cook and H. Tredennick.
10. Aristotle. *Nicomachean Ethics*. Univ. of Chicago Press, 2011. Trans. by R. C. Bartlett and S. D. Collins.
11. I. Arrazola, A. Plainfossé, H. Prade, and C. Testemale. Extrapolation of fuzzy values from incomplete data bases. *Information Systems*, 14(6):487 – 492, 1989.
12. E. J. Ashworth. Medieval theories of analogy. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Winter 2013 edition, 2013.
13. M. Azzeh, D. Neagu, and P. Cowling. Improving analogy software effort estimation using fuzzy feature subset selection algorithm. In *Proc. of the 4th Int. Workshop on Predictor Models in Software Engineering*, PROMISE'08, pages 71–78, New York, NY, 2008. ACM.
14. N. Barbot and L. Miclet. La proportion analogique dans les groupes: applications aux permutations et aux matrices. Technical Report 1914, IRISA, July 2009.
15. N. Barbot, L. Miclet, and H. Prade. Analogical proportions and the factorization of information in distributive lattices. In M. Ojeda-Aciego and J. Ostrata, editors, *CEUR Workshop Proc. 10th Int. Conf. on Concept Lattices and their Applications (CLA'13)*, La Rochelle, Oct. 15-18, pages 175–186, 2013.

16. P. F. A. Bartha. *By Parallel Reasoning: The Construction and Evaluation of Analogical Arguments*. Oxford University Press, 2009.
17. M. Bayoudh, H. Prade, and G. Richard. Evaluation of analogical proportions through Kolmogorov complexity. *Knowledge-Based Systems*, 29:20–30, 2012.
18. J. D. Becker. The modeling of simple analogic and inductive processes in a semantic memory system. *Proc. 1st Int. Conf. on AI (IJCAI'69), Washington DC*, pages 655–668, 1969.
19. A. Ben Hassena and L. Miclet. Analogical learning using dissimilarity between tree-structures. In *Proc. 19th European Conf. on Artificial Intelligence (ECAI'10), Lisbon*, volume 215 of *Frontiers in AI and Applications*, pages 1039–1040. IOS Press, 2010.
20. M. Black. *Models and Metaphors: Studies in Language and Philosophy*. Cornell Univers. Press, Ithaca, 1962.
21. C. Bocérén. La relation de similarité à la base du choix de distracteurs : contribution à la généralisation du modèle de Rumelhart et Abrahamson. *L'Année Psychologique*, 101(1):33–63, 2001.
22. I. M. Bochenski. On analogy. *The Thomist*, 11:424–447, 1948. Reprinted in “Logico-Philosophical Studies” (A. Menne, ed.) D. Reidel, 1962, pp. 97–117. Review by R. Feys, *The Journal of Symbolic Logic*, 14(4), 265–266, 1950.
23. A. Bohan and M. T. Keane. Boosting analogical arguments: The effects of goodness & complexity on everyday arguments. In *Proc XXVIIth Ann. Conf. Cognitive Sci. Soc. (CogSci'05)*, pages 304–309, Stresa, Italy, July 21–23, 2005.
24. B. Bouchon-Meunier, C. Marsala, and M. Rifqi. Fuzzy analogical model of adaptation for case-based reasoning. *Fuzzy Sets and Systems*, pages 1625–1630, 2009.
25. B. Bouchon-Meunier and L. Valverde. A resemblance approach to analogical reasoning functions. In T. P. Martin and A. L. Ralescu, editors, *Fuzzy Logic in Artificial Intelligence Towards Intelligent Systems*, volume 1188 of *Lecture Notes in Computer Science*, pages 266–272. Springer Berlin Heidelberg, 1997.
26. D. Bourcier. Analogie. In A.J. Arnaud, editor, *Dictio. Encycl. de Théorie et de Sociologie du Droit*, Collection Droit et société. Lib. Gén. de Droit et de Jurisprudence, 1993.
27. C. Bourely, G. Défourneaux, and N. Peltier. Building proofs or counterexamples by analogy in a resolution framework. In *Proc. of JELIA 96*, LNAI 1126, pages 34–49. Springer, 1996.
28. J. Bouveresse. *Prodiges et Vertiges de l'Analogie. De l'Abus des Belles-Lettres dans la Pensée*. Raison d'Agir Éditions, 1999.
29. T. Boy de la Tour and R. Caferra. Proof analogy in interactive theorem proving: A method to express and use it via second order matching. In *Proc. 6th National Conf. on Artificial Intelligence (AAAI'87)*, pages 95–99. Morgan Kaufmann, 1987.
30. B. Brock, S. Cooper, and W. Pierce. Analogical reasoning and proof discovery. In Ewing Lusk and Ross Overbeek, editors, *9th International Conference on Automated Deduction*, volume 310 of *Lecture Notes in Computer Science*, pages 454–468. Springer Berlin Heidelberg, 1988.
31. D. E. Brown. Using examples and analogies to remediate misconceptions in physics: Factors influencing conceptual change. *J. of Research in Science Teaching*, 29(1):17–34, 1993.
32. R. Brown. Use of analogy to achieve new expertise. Technical report, AI Lab., MIT, AI-TR-403, 1977.
33. M. Bunge. Analogy in quantum theory: From insight to nonsense. *British Journal for the Philosophy of Science*, 18(4):265–286, 1968.
34. J. G. Carbonell. Learning by analogy: Formulating and generalizing plans from past experience. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine learning*, pages 137–161. Tioga Pub. Corp., Palo Alto, 1983.
35. J. G. Carbonell. Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In T. M. Mitchel, J. G. Carbonell, and R. S. Michalski, editors, *Machine*

- Learning - A Guide to Current Research, Vol. II*, pages 371–392. Morgan Kaufmann Inc., Los Altos, 1986.
36. R. Carnap. Variety, analogy and periodicity in inductive logic. *Philosophy of Science*, 30(3):222–227, 1963.
 37. P. A. Carpenter, M. A. Just, and P. Shell. What one intelligence test measures: A theoretical account of the processing in the raven progressive matrices test. *Psychological Review*, 97(3):404–431, 1990.
 38. H. P. Casakin and G. Goldschmidt. Reasoning by visual analogy in design problem-solving: the role of guidance. *Environment and Planning B: Planning and Design*, 27(1):105–119, 2000.
 39. J. Charbonnet. *Analogie et Logique Naturelle. Une étude des traces linguistiques du raisonnement analogique à travers différents discours*. Peter Lang, Bern, 2003.
 40. N. Chater. The search for simplicity: A fundamental cognitive principle? *The Quarterly Journal of Experimental Psychology*, 52(2):273–302, 1999.
 41. D. T. W. Chen and N. V. Findler. Toward analogical reasoning in problem solving by computers. *J. of Cybernetics*, 19(4):369–397, 1979.
 42. Z. Chen. Analogical transfer: From schematic pictures to problem solving. *Memory & Cognition*, 23(2):255–269, 1995.
 43. A. Chiu, P. Poupart, and C. DiMarco. Generating lexical analogies using dependency relations. In *Proc. Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL'07), June 28-30, Prague*, pages 561–570. ACL, 2007.
 44. E. Chouraqui. Construction of a model for reasoning by analogy. In *Proc. 5th Europ. Conf. in Artificial Intelligence (ECAI'82), Orsay, France*, pages 48–53, 1982.
 45. E. Chouraqui and C. Inghilterra. Résolution par analogie de problèmes géométriques dans une perspective tutorielle. In C. Frasson, G. Gauthier, and G. I. McCalla, editors, *Intelligent Tutoring Systems, Proc. 2nd Int. Conf., ITS '92, Montréal, June 10-12*, volume 608 of *LNCS*, pages 156–163. Springer, 1992.
 46. A. Cornuéjols. Analogy as minimization of description length. In G. Nakhaeizadeh and C. Taylor, editors, *Machine Learning and Statistics: The interface*, pages 321–336. Wiley and Sons, 1996.
 47. W. Correa, H. Prade, and G. Richard. When intelligence is just a matter of copying. In *Proc. 20th Eur. Conf. on Artificial Intelligence, Montpellier, Aug. 27-31*, pages 276–281. IOS Press, 2012.
 48. S. Costantini, G. A. Lanzarone, and L. Sbarbaro. A formal definition and a sound implementation of analogical reasoning in logic programming. *Ann. Math. Artif. Intell.*, 14:17–36, 1995.
 49. D. Croft and P. Thagard. Dynamic imagery: A computational model of motion and visual analogy. In L. Magnani and N. J. Nersessian, editors, *Model-Based Reasoning*, pages 259–274. Springer US, 2002.
 50. R. V. Curtis and C. M. Reigeluth. The use of analogies in written text. *Instructional Science*, 13:99–117, 1984.
 51. M. D'Aquin, J. Lieber, and A. Napoli. Adaptation knowledge acquisition: A case study for case-based decision support in oncology. *Computational Intelligence*, 22(3-4):161–176, 2006.
 52. L. Darden. Artificial intelligence and philosophy of science: Reasoning by analogy in theory construction. *Proc. Biennial Meeting of the Philosophy of Science Association*, pages 147–165, 1982.
 53. M. Dastani, B. Indurkhy, and R. Scha. Analogical projection in pattern perception. *J. of Experimental and Theoretical Artificial Intelligence*, 15(4):489–511, 2003.

54. J. Davies, N. J. Nersessian, and A. K. Goel. Visual models in analogical problem solving. *Foundations of Science*, 10(1):133–152, 2005.
55. T. R. Davies and Russell S. J. A logical approach to reasoning by analogy. In J. P. McDermott, editor, *Proc. of the 10th International Joint Conference on Artificial Intelligence (IJCAI'87), Milan*, pages 264–270. Morgan Kaufmann, 1987.
56. M. de Coster. *L'Analogie en Sciences Sumaines*. PUF, 1978.
57. L. De Jaucourt. Proportion. In D. Diderot and J. le Rond D'Alembert, editors, *Encyclopédie ou Dictionnaire Raisonné des Sciences, des Arts et des Métiers*. 1751.
58. U. de Vandenesse. Analogie. In D. Diderot and J. Le Rond D'Alembert, editors, *Encyclopédie ou Dictionnaire Raisonné des Sciences, des Arts et des Métiers*. 1751.
59. M. Dorolle. *Le Raisonnement par Analogie*. PUF, Paris, 1949.
60. F. Douay-Soublin. La contre-analogie. Réflexion sur la récusation de certaines analogies pourtant bien formées cognitivement. *Texto !*, 1999.
61. D. Dubois, F. Esteva, P. Garcia, L. Godo, R. López de Mántaras, and H. Prade. Fuzzy modelling of case-based reasoning and decision. In D. B. Leake and E. Plaza, editors, *Proc. 2nd Int. Conf. on Case-Based Reasoning (ICCBR'97), Providence, RI, July 25-27*, volume 1266 of *LNCS*, pages 599–610. Springer, 1997.
62. D. Dubois, E. Hüllermeier, and H. Prade. Fuzzy set-based methods in instance-based reasoning. *IEEE Trans. on Fuzzy Systems*, 10(3):322–332, 2002.
63. R. Duit. On the role of analogies and metaphors in learning science. *Science Education*, 75 (6):649–672, 1991.
64. M. J. Durand-Richard, editor. *L'Analogie dans la Démarche Scientifique, Perspective Historique*. L'Harmattan, 2008. Contents: Introduction (M. J. Durand-Richard), Sur la conception aristotélicienne de l'analogie (P. Huneman), Le raisonnement par analogie dans les mathématiques chinoises du premier millénaire de notre ère (A. Volkov), Analogie entre théorie des nombres et théorie des fonctions (C. Houzel), L'analogie algébrique au fondement de l'Analysis Situs (A. Herreman), De l'algèbre à la théorie des modèles : Structuration de l'analogie comme méthode démonstrative (M. J. Durand-Richard), Les analogies mathématiques au sens de Poincaré et leur fonction en physique (M. Paty), Le pouvoir heuristique de l'analogie en physique (C. Comte), L'analogie dans les sciences du végétal : À propos des positions de F. Fontana et A. P. de Candolle sur les maladies des plantes (G. Denis), Analogie et métaphore mécaniciste en biologie moléculaire (P. Bromberg).
65. P. Durrenberger and J. W. Morrison. A theory of analogy. *J. of Anthropological Research*, 33(4):372–387, 1977.
66. K. Duvignau, O Gasquet, and B. Gaume, editors. *Regards Croisés sur l'Analogie*. Revue d'intelligence artificielle, vol.17 (5-6), 701-951, 2003. Contents: Towards the categorization of conceptual theories (F. Cordier), Analogie et catégorisation (E. Sander), L'analogie entre catégorisation et expression (M. Prandi), Vers une révision de la notion de lexicalisation. Contribution à une vision dynamique du lexique mental : stock lexical, catégories vs réseau lexico-sémantique (J. Nespolous, J. Virbel), Useful examples and a new model of the immune system (U. Hershberg), Le rôle de l'analogie dans l'acquisition de la langue maternelle (G. Hilaire-debove, S. Kern), Acquisition des verbes et reformulations (C. Martinot), Processus cognitifs de la construction du lexique verbal dans l'acquisition (L1 et L2) (C. Noyau), Analogy and transfer of learning in syntactic development (A. Ninio), Symbolic rule versus analogy in the processing of complex verbal morphology (K. Gor), Métaphore, analogie et syntaxe (J. Gardes Tamine), La polysémie comme source d'analogie (J. Meunier), Métaphore verbale et approximation (K. Duvignau), Ressemblance entre objets (J. Delahaye), Analogie et équivalence dans la traduction (I. Tamba), Distance sémantique et degrés d'analogie dans les matrices analytiques définitoires (A. Ibrahim), L'analogie, un moyen de croiser les contraintes et les paradigmes (N. Hathout), Analogie et proxémie dans les réseaux petits mondes (B. Gaume).

67. K. Duvignau, O. Gasquet, B. Gaume, and M. D. Gineste. Categorisation of actions by analogy: From the analysis of metaphoric utterances to a computational model. In S. M. Haller and G. Simmons, editors, *Proc. of the 15th Florida Artificial Intelligence Research Soc. Conf. (FLAIRS), Pensacola Beach, May 14-16*, pages 143–147. AAAI Press, 2002.
68. T. G. Evans. A heuristic program to solve geometry-analogy problems. In *Proc. A.F.I.P. Spring Joint Computer Conf.*, volume 25, pages 5–16, 1964.
69. T. G. Evans. A program for the solution of a class of geometric-analogy intelligence-test questions. In M. L. Minsky, editor, *Semantic Information Processing*, pages 271–353. MIT Press, Cambridge, Ma, 1968.
70. H. Farreny and H. Prade. About flexible matching and its use in analogical reasoning. In *Proc. 5th Europ. Conf. in Art. Int. (ECAI'82), Orsay, France*, pages 43–47, 1982.
71. S. Federici, S. Montemagni, and V. Pirrelli. Inferring semantic similarity from distributional evidence: an analogy-based approach to word sense disambiguation. In *Proc. of Workshop Automatic information extraction and building of lexical semantic resource for NLP applications(EACL'97)*, pages 90–97, 1997.
72. S. Federici, V. Pirrelli, and F. Yvon. A dynamic approach to paradigm-driven analogy. In S. Wermter, E. Riloff, and G. Scheler, editors, *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*, volume 1040 of *LNCS*, pages 385–398. Springer, 1996.
73. L. Flick. Analogy and metaphor: Tools for understanding inquiry science methods. *J. of Science Teacher Education*, 2(3):61–66, 1991.
74. R. M. French. *The Subtlety of Sameness. A Theory and Computer Model of Analogy-Making*. MIT Press, 1995.
75. R. M. French. The computational modeling of analogy-making. *Trends in Cognitive Sciences*, 6(5):200 – 205, 2002.
76. R. M. French and D. Hofstadter. Tabletop: An emergent, stochastic model of analogy-making. In *Proc. of the 13th Annual Conf. of the Cognitive Science Society*, pages 175–182. Lawrence Erlbaum, Hillsdale, NJ, 1991.
77. B. Frydman. Les formes de l’analogie. *Revue de la Recherche Juridique - Droit Prospectif, Cahiers de méthodologie juridique*, pages 1053–1064, 1995.
78. M. R. Genesereth. Metaphors and models. In R. Balzer, editor, *Proc. 1st Annual National Conf. on Artificial Intelligence. Stanford, Aug.18-21*, pages 208–211. AAAI Press/MIT Press, 1980.
79. D. Gentner. Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2):155–170, 1983.
80. D. Gentner. The mechanisms of analogical learning. In S. Vosniadou and A. Ortony, editors, *Similarity and Analogical Reasoning*, pages 197–241. Cambridge University Press, New York, 1989.
81. D. Gentner, K. J. Holyoak, and B. N. Kokinov. *The Analogical Mind: Perspectives from Cognitive Science*. Cognitive Science, and Philosophy. MIT Press, Cambridge, MA, 2001.
82. M. C. Ghyska. *Sortilèges du Verbe*. Gallimard, Paris, 1949. Chap. VIII “Métaphore et analogie, pp.165-188.
83. N. L. Gick and K. J. Holyoak. Analogical problem solving. *Cognitive Psychology*, 12 (3):306–335, 1980.
84. N. L. Gick and K. J. Holyoak. Schema induction and analogical tranfer. *Cognitive Psychology*, 15:1–38, 1983.
85. M. D. Gineste. Les analogies : Modèles pour l’appréhension de nouvelles connaissances. *L’Année Psychologique*, 84(3):387–397, 1984.
86. M. D. Gineste. *Analogie et Cognition - Etude Expérimentale et Simulation Informatique*. Presses Universitaires de France, 1997.

87. S. Glynn. Conceptual bridges: using analogies to explain scientific concepts. *Science Teacher*, 62(9):24–27, 1995.
88. S. Glynn. Methods and strategies: The Teaching-With-Analogies model. *Science and Children*, 44(8):52–55, 2007.
89. S. M. Glynn. Explaining science concepts: A Teaching-With-Analogies mode. In S. M. Glynn, R. H. Yeany, and B. K. Britton, editors, *The Psychology of Learning Science*, pages 219–239. Erlbaum, Hillsdale, NJ, 1995.
90. G. Goldschmidt. Visual analogy - A strategy for design reasoning and learning. In A. Koutamanis, H. Timmermans, and I. Vermeulen, editors, *Visual Databases in Architecture: Recent Advances in Design and Decision Making*, pages 53–76. Aldershot, Avebury, 1995.
91. R. Greiner. Learning by understanding analogies. *Artificial Intelligence*, 35(1):81–125, 1988.
92. J. B. Grize. Logique, analogie et identité. In D. Miéville, editor, *La Logique Naturelle. Enjeux et Perspectives, Actes du colloque de Neuchâtel, 12-13 Sept. 2008*, pages 91–98. Droz, Genève, 2010. Travaux du Centre de Recherches Sémiologiques 68, Université de Neuchâtel.
93. H. Gust, K. Kühnberger, and U. Schmid. Metaphors and heuristic-driven theory projection (HDTP). *Theoretical Computer Science*, 354(1):98 – 117, 2006.
94. R. P. Hall. Computational approaches to analogical reasoning: A comparative analysis. *Artificial Intelligence*, 39:39–120, 1989.
95. A. G. Harrison and D. F. Treagust. Teaching with analogies: A case study in grade-10 optics. *J. Res. Sci. Teach.*, 30:1291–1307, 1993.
96. D. H. Helman, editor. *Analogical Reasoning: Perspectives of Artificial Intelligence*. Cognitive Science, and Philosophy. Kluwer, Dordrecht, 1988.
97. A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *Proc. 28th Annual Conf. on Computer Graphics and Interactive Techniques, SIGGRAPH01*, pages 327–340, New York, NY, 2001. ACM.
98. M. Hesse. On defining analogy. *Proceedings of the Aristotelian Society*, 60:79–100, 1959.
99. M. Hesse. Analogy and confirmation theory. *Philosophy of Science*, xxxi:319–327, 1964.
100. M. Hesse. Aristotle's logic of analogy. *The Philosophical Quarterly*, 15:328–340, 1965.
101. M. Hesse. *Models and Analogies in Science*. 1st ed. Sheed & Ward, London, 1963; 2nd augmented ed. University of Notre Dame Press, 1966.
102. M. Hesse. Models and analogy in science. In P. Edwards, editor, *The Encyclopedia of Philosophy*, vol. v, pages 354–359. Macmillan, New York, 1967.
103. E. Hirowatari and S. Arikawa. Incorporating explanation-based generalization with analogical reasoning. *Bulletin of Informatics and Cybernetics*, 26:13–33, 1994.
104. J. P. Hochschild. The rest of cajetan's analogy theory. *International Philosophical Quarterly*, 45(3):341–356, 2005.
105. H. Höffding. *Le Concept d'Analogie*. Vrin, Paris, 1931. Original title: Der Begriff der Analogie, Reisland, Leipzig, 1924.
106. R. R. Hoffman. Monster analogies. *The AI Magazine*, 16(3):11–35, 1995.
107. R. R. Hoffman and T. C. Eskridge. Varieties of analogical reasoning. In *Proc. 9th Bi-annual Int. Conf. on Naturalistic Decision Making (NDM'09), London, UK, June 23-26*, pages 60–66, 2009.
108. D. Hofstadter and M. Mitchell. The Copycat project: A model of mental fluidity and analogy-making. In D. Hofstadter and The Fluid Analogies Research Group, editors, *Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought*, pages 205–267, New York, NY, 1995. Basic Books, Inc.
109. D. Hofstadter and E. Sander. *Surfaces and Essences: Analogy as the Fuel and Fire of Thinking*. Basic Books, 2013. French version: L'Analogie, Cœur de la Pensée - Odile Jacob, Paris.

110. K. J. Holyoak. Analogy. In K. J. Holyoak and R. G. Morrison, editors, *The Cambridge Handbook of Thinking and Reasoning*, chapter 6, pages 306–335. Cambridge University Press, 2005.
111. K. J. Holyoak, L. R. Novick, and E. R. Melz. Component processes in analogical transfer: Mapping, pattern completion, and adaptation. In K. J. Holyoak and J. A. Barnden, editors, *Advances in Connectionist and Neural Computation Theory, Vol. 2: Analogical Connections*, pages 113–180. Ablex Publ., Westport, CT, 1994.
112. K. J. Holyoak and P. Thagard. Analogical mapping by constraint satisfaction. *Cognitive Science*, 13:295–355, 1989.
113. E. Hüllermeier. *Case-Based Approximate Reasoning*. Springer-Verlag, Berlin, 2007.
114. J. E. Hummel and K. J. Holyoak. Distributed representations of structure: a theory of analogical access and mapping. *Psychological Review*, 104(3):427–466, 1997.
115. B. Indurkhy. Approximate semantic transference: A computational theory of metaphors and analogies. *Cognitive Science*, 11:445–480, 1987.
116. B. Indurkhy. Modes of analogy. In *Analogical and inductive inference*, volume 397 of *LNCS*, pages 217–230. Springer, 1989.
117. B. Indurkhy. Predicative analogy and cognition. In *Analogical and Inductive Inference*, volume 642 of *LNCS*, pages 214–231. Springer, 1992.
118. E. Ippoliti. Demonstrative and non-demonstrative reasoning by analogy. In C. Cellucci and P. Pecere, editors, *Demonstrative and non-demonstrative reasoning in mathematics and natural science*, pages 309–338. Edizioni dell’Universia’ di Cassino, 2006.
119. M. C. James and L. C. Scharmann. Using analogies to improve the teaching performance of novice teachers. *J. of Research in Science Teaching*, 44(4):565–585, 2007.
120. M. Jamnik. Analogy and automated reasoning. Technical Report CSRP-99-14, University of Birmingham, 1999.
121. E. Kant. *Prolegomena to Any Future Metaphysics That Will Be Able to Present Itself as a Science*. 2002. Trans. G. Hatfield.
122. M. T. Keane and F. Costello. Setting limits on analogy: Why conceptual combination is not structural alignment. In D. Gentner, K. J. Holyoak, and B. N. Kokinov, editors, *The Analogical Mind: Perspectives from Cognitive Science*. MIT press, Cambridge, MA, 2001.
123. S. Kedar-Cabelli. Purpose-directed analogy: A summary of current research. In T. M. Mitchel, J. G. Carbonell, and R. S. Michalski, editors, *Machine Learning - A Guide to Current Research*, pages 123–126. Kluwer, 1986.
124. M. Kerber and E. Melis. Two kinds of non-monotonic analogical inference. In D. M. Gabbay and H. J. Ohlbach, editors, *Practical Reasoning, Proc. International Conference on Formal and Applied Practical Reasoning (FAPR'96) Bonn, Germany, June 3-7*, volume LNAI 1085, pages 361–374. Springer, 1996.
125. S. Klein. Whorf transforms and a computer model for propositional/appositional reasoning. In *Proc. of the Applied Mathematics colloquium*, Univ.of Bielefeld, West Germany, 1977.
126. S. Klein. Culture, mysticism & social structure and the calculation of behavior. In *Proc. 5th Europ. Conf. in Artificial Intelligence (ECAI'82), Orsay, France*, pages 141–146, 1982.
127. S. Klein. Analogy and mysticism and the structure of culture (and Comments & Reply). *Current Anthropology*, 24 (2):151–180, 1983.
128. S. Klein. The analogical foundations of creativity in language, culture & the arts: the upper paleolithic to 2100ce. In P. McKevitt, C. Mulvihill, and S. O’ Nuallin, editors, *Language, Vision & Music*, pages 347–371. Amsterdam: John Benjamin, 2002.
129. S. Klein, D. A. Ross, M. S. Manasse, J. Danos, M. S. Bickford, K. I. Jensen, W. A. Burt, G. D. Blank, and W. T. Blanks. Propositional & analogical generation of coordinated verbal, visual & musical texts: U. of Wisconsin. *SIGART Bull.*, 79:104, Jan 1982.

130. M. Klenk, K. D. Forbus, E. Tomai, and H. Kim. Using analogical model formulation with sketches to solve Bennett mechanical comprehension test problems. *J. Exp. Theor. Artif. Intell.*, 23(3):299–327, 2011.
131. R. Kling. A paradigm for reasoning by analogy. *Artificial Intelligence*, 2:147–178, 1971. Also Proc. of IJCAI’1971, London, pp. 568–585.
132. J. L. Kolodner. Reconstructive memory: A computer model. *Cognitive Science*, 7(4):281–328, 1983.
133. J. L. Kolodner. *Case-Based Reasoning*. Morgan Kaufmann, 1993.
134. U. Krumnack, H. Gust, K. Kühnberger, and A. Schwering. Re-representation in a logic-based model for analogy making. In W. Wobcke and M. Zhang, editors, *AI 2008: Advances in Artificial Intelligence*, volume 5360 of *LNCS*, pages 42–48. Springer, 2008.
135. M. Kunda, K. McGreggor, and A. Goel. A computational model for solving problems from the Raven’s progressive matrices intelligence test using iconic visual representations. *Cognitive Systems Research*, 22-23:47–66, 2013.
136. M. Kunda, I. Soulières, A. Rozga, and A. K. Goel. Methods for classifying errors on the Raven’s standard progressive matrices test. In *Proc. Annual Meeting of the Cognitive Science Society (COGSCI’13)*, pages 2796–2801, 2013.
137. K.J. Kurtz, C. Mao, and D. Gentner. Learning by analogical bootstrapping. *The Journal of the Learning Sciences*, 10(4):417–446, 2001.
138. B. Landry. L’analogie de proportion chez Saint Thomas d’Aquin. *Revue Néo-scolastique de Philosophie*, 24, 1922.
139. P. Langlais and A. Patry. Translating unknown words by analogical learning. In *Joint Conference on Empirical Methods in Natural Language Processing (EMNLP) and Conference on Computational Natural Language Learning (CONLL)*, pages 877–886, Prague, 2007.
140. P. Langlais, F. Yvon, and P. Zweigenbaum. Analogical translation of medical words in different languages. In B. Nordström and A. Ranta, editors, *Advances in Natural Language Processing*, volume 5221 of *LNCS*, pages 284–295. Springer, 2008.
141. J. F. Lavallée and P. Langlais. Moranapho: un système multilingue d’analyse morphologique basé sur l’analogie formelle. *TAL*, 52(2):17–44, 2011.
142. D. B. Leake, editor. *Case-Based Reasoning: Experiences, Lessons and Future Directions*. AAAI Press / MIT Press, 1993.
143. M. Lebowitz. Memory-based parsing. *Artificial Intelligence*, 21(4):363–404, 1983.
144. Y. Lepage. Analogy and formal languages. *Electr. Notes Theor. Comput. Sci.*, 53, 2001.
145. Y. Lepage. De l’analogie rendant compte de la commutation en linguistique. *Habilit. à Diriger des Recher.*, Univ. J. Fourier, Grenoble, 2003.
146. Y. Lepage and E. Denoual. Purest ever example-based machine translation: Detailed presentation and assessment. *Machine Translation*, 19(3-4):251–282, 2005.
147. J. Li and G. Ruhe. A comparative study of attribute weighting heuristics for effort estimation by analogy. In *Proc. 2006 ACM/IEEE Int. Symp. on Empirical Software Engineering*, ISESE ’06, pages 66–74, New York, NY, 2006. ACM.
148. J. Lieber and A. Napoli. Correct and complete retrieval for case-based problem-solving. In *Proc. 13th Europ. Conf. on Artificial Intelligence (ECAI’98)*, pages 68–72. Wiley UK, 1998.
149. H. Lieberman and A. Kumar. Providing expert advice by analogy for on-line help. In *Proc. 2005 IEEE/WIC/ACM Int. Conf. on Web Intelligence*, WI ’05, pages 26–32, Washington, DC, 2005. IEEE Computer Society.
150. G. E. R. Lloyd. *Polarity and Analogy, Two Types of Argumentation in Early Greek Thought*. Cambridge University Press, 1966.
151. F. Lorrain. *Réseaux Sociaux et Classifications Sociales. Essai sur l’Algèbre et la Géométrie des Structures Sociales*. Hermann, Paris, 1975.

152. A. Lovett, K. Forbus, and J. Usher. A structure-mapping model of Raven’s progressive matrices. In *Proc. 32nd Annual Conf. of the Cognitive Science Soc., Portland, OR*, 2010.
153. A. Lovett, K. Lockwood, M. Dehghani, and K. Forbus. Modeling human-like rates of learning via analogical generalization. In *Proceedings of Analogies: Integrating Multiple Cognitive Abilities*, 2007.
154. A. Lovett, E. Tomai, K. Forbus, and J. Usher. Solving geometric analogy problems through two-stage analogical mapping. *Cognitive Science*, 33(7):1192–1231, 2009.
155. J. McDermott. Learning to use analogies. In *IJCAI*, pages 568–576, 1979.
156. R. M. McInerny. *The Logic of Analogy: An Interpretation of St. Thomas*. Martinus Nijhoff, The Hague, 1961.
157. E. Melis. A model of analogy-driven proof-plan construction. In *Proc. 14th Int. Joint Conf. on Artificial Intelligence (IJCAI’95)- Volume 1*, pages 182–188, 1995.
158. E. Melis and M. Veloso. Analogy in problem solving. In *Handbook of Practical Reasoning: Computational and Theoretical Aspects*. Oxford Univ. Press, 1998.
159. E. Melis and J. Whittle. Analogy in inductive theorem proving. *Journal of Automated Reasoning*, 22(2):117–147, 1999.
160. L. Miclet, S. Bayoudh, and A. Delhay. Analogical dissimilarity: definition, algorithms and two experiments in machine learning. *JAIR*, 32, pages 793–824, 2008.
161. L. Miclet and A. Delhay. Relation d’analogie et distance sur un alphabet défini par des traits. Technical Report 1632, IRISA, July 2004.
162. L. Miclet and H. Prade. Handling analogical proportions in classical logic and fuzzy logics settings. In *Proc. 10th Eur. Conf. on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU’09), Verona*, pages 638–650. Springer, LNCS 5590, 2009.
163. J. Stuart Mill. *A System of Logic, Ratiocinative and Inductive, being a connected view of the principles of evidence and the methods of scientific investigation*. 1843. Book III, chap. XX: “Of Analogy”.
164. W. S. Miller. *Technical Manual for the Miller Analogies Test*. The Psychological Corporation, New York, 1960.
165. M. Mitchell. *Analogy-Making as Perception: A Computer Model*. MIT Press, Cambridge MA, 1993.
166. M. Mitchell. Analogy-making as a complex adaptive system. In L. Segel and I. Cohen, editors, *Design Principles for the Immune System and Other Distributed Autonomous Systems*. Oxford University Press, 2001.
167. J. C. Munyer. *Analogy as a means of discovery in problem solving and learning*. PhD thesis, University of California, 1981.
168. D. P. O’Donoghue, A. J. Bohan, and M. T. Keane. Seeing things: Inventive reasoning with geometric analogies and topographic maps. *New Generation Comput.*, 24(3):267–288, 2006.
169. S. O’Hara. A model of the ‘redescription’ process in the context of geometric proportional analogy problems. In K. P. Jantke, editor, *Analogical and Inductive Inference, Proc. Int. Workshop (AII92), Dagstuhl, Oct. 5-9*, volume 642 of *LNCS*, pages 268–293. Springer, 1992.
170. P. K. Paritosh and M. E. Klenk. Cognitive processes in quantitative estimation: Analogical anchors and causal adjustment. In *Proc. 28th Annual Conf. of the Cognitive Science Society, Vancouver*, 2006.
171. A. Pease, M. Guhe, and A. Smaill. Using analogies to find and evaluate mathematical conjectures. In *Proc. 1st Int. Conf. on Computational Creativity, Lisbon*, 7-9 Jan., pages 60–64. 2010.
172. R. Pedone, J. E. Hummel, and K. J. Holyoak. The use of diagrams in analogical problem solving. *Memory and Cognition*, 29:214–221, 2001.

173. Y. Pelletier. La division de l'analogie. *Peripatikoss*, pages 8–14, 1995.
174. Y. Pelletier. La doctrine aristotélicienne de l'analogie. *Philosophia perennis*, 2(1):3–44, 1995.
175. C. Perelman. Analogie et métaphore en science, poésie et philosophie. *Revue Internationale de Philosophie, fasc. 1*, (87):3–15, 1969.
176. C. Perelman. *Ethique et Droit*. Edition de l'Univ. Libre de Bruxelles, 1990.
177. M. Petrovitch. *Mécanismes communs aux phénomènes disparates*. Félix Alcan, Paris, 1921.
178. J. Piaget. *Classe, Relations et Nombres. Essai sur les Groupements de la Logistique et sur la Réversibilité de la Pensée*. Vrin, Paris, 1942.
179. J. Piaget. *Essai sur les Transformations des Opérations Logiques: les 256 Opérations Ternaires de la Logique Bivalente des Propositions*. Presses Univ. de France, Paris, 1952.
180. J. Piaget. *Logic and Psychology*. Manchester Univ. Press, 1953.
181. G. Polya. *How to Solve It*. Princeton University Press, 2nd ed. 1957, 1945.
182. G. Polya. *Mathematics and Plausible Reasoning-Vol.1: Induction and analogy in Mathematics, Vol.2: Patterns of Plausible Inference*. Princeton Univ. Press, 2nd ed. 1968, 1954.
183. V. Poprcova, G. Stojanov, and A. Kulakov. Inductive logic programming ilp and reasoning by analogy in context of embodied robot learning. *Int. J. Agent Technol. Syst.*, 2(2):64–73, April 2010.
184. H. Prade and G. Richard. Testing analogical proportions with google using Kolmogorov information theory. In *Proc. of the 22nd Florida Artificial Intelligence Research Soc. Conf. (FLAIRS)*, Fort Myers, pages 272–277. AAAI Press publisher, 2009.
185. H. Prade and G. Richard. Reasoning with logical proportions. In *Proc. 12th Int. Conf. on Principles of Knowledge Representation and Reasoning, KR 2010, Toronto, May 9-13, 2010 (F. Z. Lin, U. Sattler, M. Truszcynski, eds.)*, pages 545–555. AAAI Press, 2010.
186. H. Prade and G. Richard. Cataloguing/analogizing: A non monotonic view. *Int. J. Intell. Syst.*, 26(12):1176–1195, 2011.
187. H. Prade and G. Richard. Homogeneous logical proportions: Their uniqueness and their role in similarity-based prediction. In G. Brewka, T. Eiter, and S. A. McIlraith, editors, *Proc. 13th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'12), Roma, June 10-14*, pages 402–412. AAAI Press, 2012.
188. H. Prade and G. Richard. From analogical proportion to logical proportions. *Logica Universalis*, 7(4):441–505, 2013.
189. H. Prade, G. Richard, and B. Yao. Classification by means of fuzzy analogy-related proportions. A preliminary report. In *Proc. 2d IEEE Int. Conf. Soft Comp. & Pattern Recog. (SocPar'10)*, Evry. 2010, 297-302.
190. G.y. Qiu. Nonmonotonic logic for analogical reasoning. In R. Greiner and D.Subramanian, editors, *Working Notes of the 1994 AAAI Fall Symposium on Relevance, New Orleans, Louisiana, Nov. 4-6*, volume AAAI Tech. Rep.94-02, pages 161–164, 1994.
191. L. Racine. Du modèle analogique dans l'analyse des représentations magico-religieuses. *L'Homme*, 29(109):5–25, 1989.
192. M. Ragni and S. Neubert. Solving Raven's IQ-tests: An AI and cognitive modeling approach. In L. De Raedt, C. Bessière, D. Dubois, P. Doherty, P. Frasconi, F. Heintz, and P. J. F. Lucas, editors, *Proc. 20th Europ. Conf. on Artificial Intelligence (ECAI'12) Montpellier, Aug. 27-31*, pages 666–671, 2012.
193. J. Raven. The Raven's progressive matrices: Change and stability over culture and time. *Cognitive Psychology*, 41(1):1 – 48, 2000.
194. J. C. Raven. *Progressive Matrices*. The Psychological Corporation, New York, 1965.
195. G. Richard (ed.). *Working Papers of the 1st International Workshop on Similarity and Analogy-based Methods in AI (SAMAI 2012)*. Tech. Rep. IRIT/RR-2012-20, 2012. Co-located with the 20th European Conference on Artificial Intelligence Montpellier, France, August 27, 2012.

196. J. F. Ross. *Portraying Analogy*. Cambridge University Press, 1981.
197. D. E. Rumelhart and A. A. Abrahamson. A model for analogical reasoning. *Cognitive Psychol.*, 5:1–28, 2005.
198. S. J. Russell. *The Use of Knowledge in Analogy and Induction*. Pitman, UK, 1989.
199. E. Sander. *L'Analogie, du Naïf au Créatif : Analogie et Catégorisation*. L'Harmattan, 2000.
200. S.-E. Schelhorn, J. Griego, and U. Schmid. Transformational and derivational strategies in analogical problem solving. *Cognitive Processing*, 8(1):45–55, 2007.
201. U. Schmid, H. Gust, K. Kühnberger, and J. Burghardt. An algebraic framework for solving proportional and predictive analogies. *Eur. Conf. Cogn. Sci.* 295-300, 2003.
202. S. Schockaert and H. Prade. Interpolation and extrapolation in conceptual spaces: A case study in the music domain. In S. Rudolph and C. Gutierrez, editors, *Proc. 5th Int. Conf. on Web Reasoning and Rule Systems (RR'11), Galway, Ireland, Aug. 29-30*, volume 6902 of *LNCS*, pages 217–231. Springer, 2011.
203. S. Schockaert and H. Prade. Interpolative and extrapolative reasoning in propositional theories using qualitative knowledge about conceptual spaces. *Artificial Intelligence*, 202:86–131, 2013.
204. D. L. Schwartz. The construction and analogical transfer of symbolic visualizations. *J. of Research in Science Teaching*, 30(10):1309–1325, 1993.
205. A. Schwering, K. Kühnberger, U. Krumnack, and H. Gust. Spatial cognition of geometric figures in the context of proportional analogies. In K. S. Hornsby, Ch. Claramunt, M. Dennis, and G. Ligozat, editors, *Spatial Information Theory*, volume 5756 of *Lecture Notes in Computer Science*, pages 18–35. Springer, 2009.
206. P. Secretan. *L'Analogie*. Que Sais-Je ? Presses Universitaires de France, 1985.
207. C. Shelley. Analogy counterarguments and the acceptability of analogical hypotheses. *The British Journal for the Philosophy of Science*, 53(4):477–496, 2002.
208. S. Simoff, C. Sierra, and R. López de Mántaras. Mediation = information revelation + analogical reasoning. In J.-J. Meyer and J. Broersen, editors, *Knowledge Representation for Agents and Multi-Agent Systems*, volume 5605 of *LNCS*, pages 145–160. Springer, 2009.
209. J. F. Sowa and A. K. Majumdar. Analogical reasoning. In *Proc. Inter. Conf. on Conceptual Structures*, pages 16–36. Springer, LNAI 2746, 2003.
210. R. Speer, C. Havasi, and H. Lieberman. AnalogySpace: reducing the dimensionality of common sense knowledge. In *Proc. 23rd National Conf. on Artificial Intelligence (AAAI'08)*, pages 548–553, 2008.
211. P. Stahl and M. Ragni. Complexity in analogy tasks: An analysis and computational model. In *Proc. 19th European Conference on Artificial Intelligence (ECAI'10)*, pages 1041–1042, Amsterdam, 2010. IOS Press.
212. N. Stroppa and F. Yvon. An analogical learner for morphological analysis. In *Online Proc. 9th Conf. Comput. Natural Language Learning (CoNLL-2005)*, pages 120–127, 2005.
213. N. Stroppa and F. Yvon. Analogical learning and formal proportions: Definitions and methodological issues. Technical report, June 2005.
214. N. Stroppa and F. Yvon. Du quatrième de proportion comme principe inductif : une proposition et son application à l'apprentissage de la morphologie. *Traitemen Automatique des Langues*, 47(2):1–27, 2006.
215. J. Syrovatka. Analogy and understanding. *Theoria. Revista de Teor?a, Historia y Fundamentos de la Ciencia*, 15 (3):435–450, 2000.
216. B. Tausend and S. Bell. Analogical reasoning for logic programming. In Y. Kodratoff, editor, *Machine Learning (EWSL'91)*, volume 482 of *LNCS*, pages 391–397. Springer, 1991.
217. P. Thagard, D. Gochfeld, and S. Hardy. Visual analogy mapping. *Proc of 14th Annual Conf. of the Cognitive Science Society*, pages 522–527, 1992.

218. P. Thagard, K. J. Holyoak, G. Nelson, and D. Gochfeld. Analog retrieval by constraint satisfaction. *Artificial Intelligence*, 46(3):259–310, 1990.
219. P. D. Turney. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416, 2006.
220. P. D. Turney. A uniform approach to analogies, synonyms, antonyms, and associations. In *Proc. 22nd Int. Conf. on Computational Linguistics - Vol. 1 (COLING '08)*, pages 905–912. Association for Computational Linguistics, 2008.
221. P. D. Turney and M. L. Littman. Corpus-based learning of analogies and semantic relations. *Mach. Learn.*, 60(1–3):251–278, 2005.
222. A. Tversky. Features of similarity. In *Psychological Review*, number 84, pages 327–352. American Psychological Association, 1977.
223. J. Van Dormael. The emergence of analogy. Analogical reasoning as a constraint satisfaction process. *Philosophica*, 46:157–177, 1990.
224. F. Vandamme. An explication of the concept “analogy”. *Studia Philosophica Gandensia*, 4:157–177, 1966.
225. T. Veale. An analogy-oriented type hierarchy for linguistic creativity. *Knowledge-Based Systems*, 19(7):471 – 479, 2006.
226. T. Veale. Re-representation and creative analogy: A lexico-semantic perspective. *New Generation Computing*, 24(3):223–240, 2006.
227. G. J. Venhille and D. F. Treagust. Analogies in biology education: A contentious issue. *The American Biology Teacher*, 59(50):282–287, 1997.
228. D. Vernant. Métaphore, analogie et iconicité. *Revue Francophone d'Esthétique, La pensée plastique*, 3:117–137, 2005.
229. S. Vosniadou and A. Ortony, editors. *Similarity and Analogical Reasoning*. Cambridge University Press, New York, NY, 1989.
230. D. Walton. Similarity, precedent and argument from analogy. *Artificial Intelligence and Law*, 18 (3):217–246, 2010.
231. D. Walton. Story similarity in arguments from analogy. *Informal Logic*, 32 (2):190–221, 2012.
232. D. Walton. Argument from analogy in legal rhetoric. *Artificial Intelligence and Law*, 21(3):279–302, 2013.
233. P. Wang. Analogy in a general-purpose reasoning system. *Cognitive Systems Research*, 10(3):286 – 296, 2009.
234. S.-L. Wang, T.-P. Hong, and W.-Y. Lin. Answering null queries by analogical reasoning on similarity-based fuzzy relational databases. *J. of Advanced Computational Intelligence*, 5:163–171, 2001.
235. J. S. Weitzenfeld. Valid reasoning by analogy. *Philosophy of Science*, 51(1):137–149, 1984.
236. S. Weller and U. Schmid. Solving proportional analogies by E-generalization. In C. Freksa, M. Kohlhase, and K. Schill, editors, *KI 2006: Advances in Artificial Intelligence*, volume 4314 of *LNCS*, pages 64–75. Springer, 2007.
237. J. Whittle, A. Bundy, and R. Boulton. Proofs-as-programs as a framework for the design of an analogy-based ml editor. *Formal Aspects of Computing*, 13(3-5):403–421, 2002.
238. P. R. Wilson. On the argument by analogy. *Philosophy of Science*, 31(1):34–39, 1964.
239. P. H. Winston. Learning by creatifying transfer frames. *Artificial Intelligence*, 10(2):147–172, 1978.
240. P. H. Winston. Learning and reasoning by analogy. *Commun. ACM*, 23(12):689–703, 1980.
241. M. J. Wreen. A second form of argument from analogy. *Theoria*, 73 (3):221–239, 2007.
242. A. Wylie. The reaction against analogy. *Advances in Archaeological Method and Theory*, 8:63–111, 1985.

243. F. Yvon. Pronouncing unknown words using multi-dimensional analogies. In *Proc. 6th Eur. Conf. on Speech Communication and Technology, (EUROSPEECH99), Budapest, September 5-9*. ISCA, 1999.
244. F. Yvon. Finite-state transducers solving analogies on words. Technical report, Ecole Nationale Supérieure des Télécommunications, 2003.
245. F. Yvon and N. Stroppa. Formal models of analogical proportions. Technical report, Ecole Nationale Supérieure des Télécommunications, no D008, 2006.
246. F. Yvon, N. Stroppa, A. Delhay, and L. Miclet. Solving analogical equations on words. Technical report, Ecole Nationale Supérieure des Télécommunications, 2004.

Part1: Analogy in action

Analogies between Binary Images: Application to Chinese Characters

Yves Lepage

Graduate school of Information, Production and Systems, Waseda university,
808-0135 Hibikino 2-7, Wakamatsu-ku, Kitakyushu-shi, Fukuoka-ken, Japan
yves.lepage@waseda.jp
http://www.waseda.jp/ips/english/kyoin/01_8.html

Abstract. The purpose of this paper is to show how it is possible to efficiently extract the structure of a set of objects by use of the notion of proportional analogy. As a proportional analogy involves four objects, the very naïve approach to the problem, has basically a complexity of $O(n^4)$ for a given set of n objects. We show, under some conditions on proportional analogy, how to reduce this complexity to $O(n^2)$ by considering an equivalent problem, that of enumerating analogical clusters that are informative and not redundant. We further show how some improvements make the task tractable. We illustrate our technique with a task related with natural language processing, that of clustering Chinese characters. In this way, we re-discover the graphical structure of these characters.

Keywords: Analogy; Analogical clusters; Binary images; Chinese characters

1 Introduction

Proportional analogy is defined in various ways by different recent authors [1, 2, 3]. Referring back to the most ancient definitions, one can reach an agreement on the following definition:

Four objects, A , B , C and D , form a proportional analogy if the first object is to the second object in the same way as the third object is to the fourth object.
A proportional analogy is noted $A : B :: C : D$.

In all generality, if the relation between two objects (noted by the colon :) is called a *ratio* and the relation between the two pairs of objects (noted by the two colons ::) is called a *conformity*, then a proportional analogy is a conformity of ratios between two pairs of objects.

Proportional analogy can be seen between words on the level of form or on the level of meaning or on both at the same time (see [4] for abnormal cases). Table 1 gives examples of analogies that hold on these two levels at the same time or only on one of these two levels.

Proportional analogies on the levels of form and meaning at the same time are called true analogies. Between chunks or short sentences, their number has been shown

to be quite important [5, 6, 7]. Many studies, too many to cite here, address the efficiency of analogy for segmenting words or grouping them according to word families (as for Chinese, see for instance [8]). Forms which depart from declension or conjugation paradigms (groups of proportions in [9]) were called anomalies in Classical grammar [10]. Recently, analogies between word meanings (*water : riverbed :: traffic : road*) have been shown to be reproducible on computers using large corpora and vector space models [11, 12, 13].

Table 1: Examples of proportional analogies between words.

| Proportional analogy | Levels on which the analogy holds | |
|---|-----------------------------------|---------|
| | Form | Meaning |
| <i>to walk : walked :: to work : worked</i> | Yes | Yes |
| <i>wings : fins :: a wing : a fin</i> | Yes | Yes |
| <i>to walk : walked :: to be : was</i> | No | Yes |
| <i>wings : fins :: bird : fish</i> | No | Yes |
| <i>to walk : walked :: to me : need</i> | Yes | No |
| <i>wings : fins :: winged : fined</i> | Yes | No |

1.1 Proportional Analogies between Binary Images

Proportional analogies are not only verbal. They may hold between any kind of objects provided, generally, that the objects be of the same kind, a point Aristotle, among other ancient and recent authors, insists on. The general principle, viewed as a cognitive process, is based on iconicity [14]. Taking the term to its restrained graphical sense, the example in Fig. 1 illustrates a proportional analogy between binary images made of black and white pixels.

This analogy makes sense for human beings. Two oppositions are seen in this analogy. The first opposition is between a happy and a sad expression. The second opposition is between two faces (of two people). The fact that the two faces may be interpreted as belonging to a male and a female character may depend on cultural, social or even individual judgements.

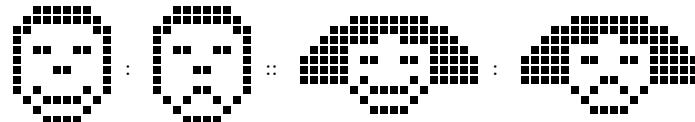


Fig. 1: An analogy between binary images made of black and white pixels.

1.2 Proportional Analogies between Chinese Characters

The example in Fig. 1 is a purely graphical example. The example in Fig. 2 is also a graphical proportional analogy but the images represent Chinese characters written in some given font. As is true with the analogy in Fig. 1, this analogy is understandable by anybody, i.e., the oppositions expressed by the analogy can be extracted by anybody, even people without a prior knowledge of the Chinese writing system. Indeed, it is understandable that the left parts (not printable in isolation) and the right parts (隹, 吉) of the characters can be exchanged to give rise to the four different characters present in the analogy: 维, 结, 谁 and 诘. In the same way as with the but last example of Table 1, *to walk : walked :: to me : need*, this is only an analogy of form. Indeed, this analogy does not apply on the level of meaning, as the meanings of these characters are unrelated: ‘maintain, preserve’, ‘knot, tie’, ‘who’ and ‘question, interrogate’. It does not constitute an analogy on the level of pronunciation either: /wéi/, /jié/, /shuí/ and /jíe/.

In [15], where the goal is to compute similarity between Chinese characters, this kind of decomposition of characters into elementary parts is performed thanks to prior knowledge about the Chinese writing system, so that the elements into which the decomposition is performed are not accessible to anybody, let alone a machine that would only be given the binary images as input.

The particular and practical problem which we tackle in a broader research concerned with the ease of learning the Chinese characters for human beings [16], is to re-discover the graphical structure of Chinese characters in a fully automatic way by relying on the notion of proportional analogy, without any prior knowledge of the Chinese writing system. This graphical structure can be made explicit by finding analogies between characters, as exemplified by Fig. 2. In a first step, we are only concerned with gathering such graphical analogies, possibly, all the possible ones. Intersecting graphical analogies with analogies on the levels of pronunciation and meaning is performed only in subsequent steps of our research not reported in this paper.

The problem tackled in the present paper is thus only that of recognizing graphical analogies between binary images made of black and white pixels, but our data will be Chinese or Sino-Japanese characters. In other terms, the general and theoretical problem that we tackle in the following sections is to rely on the properties of proportional analogies so as to automatically visualize the structure of a set of objects, binary images in general, Chinese characters in particular.

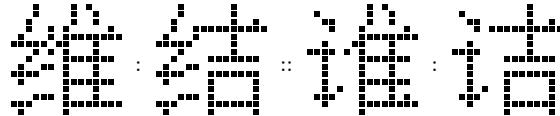


Fig. 2: Another analogy between binary images made of black and white pixels.

1.3 Structure of the Paper

The paper is structured as follows: Section 2 introduces the general problem of enumerating all analogies between objects contained in a set of objects and its complexity. Section 3 shows how the problem can be transformed into a problem of quadratic complexity and introduces the notion of analogical clusters for this purpose. Section 4 gives our proposed method to output analogical clusters. Section 5 mentions some improvements that can reduce computational time. Section 6 describes the application of the proposed method to the problem of structuring Chinese characters, and describes the results obtained in our experiments.

2 Enumerating All Analogies between Objects in a Set of Objects

The naïve approach to the general problem of the enumeration of all proportional analogies between a set of n objects consists in examining all possible quadruples of objects and checking for analogy. This naïve approach has a complexity of $O(n^4)$.

Without changing the complexity, the computation time may be reduced. For a given proportional analogy, there exists seven other equivalent forms (see Theorem 2.1 in [17]). This is implied by the basic properties of exchange of the means (exchanging objects B and C in the analogy $A : B :: C : D$, second line below) and symmetry of conformity (exchanging the terms on both sides of the :: sign, sixth line below). By applying these basic properties in any order iteratively, the following eight analogies are shown to be equivalent:

$$\begin{aligned} A : B :: C : D \\ A : C :: B : D & \text{ exch. means} \\ B : A :: D : C & \text{ exch. means + sym. :: + exch. means} \\ B : D :: A : C & \text{ exch. means + sym. ::} \\ C : A :: D : B & \text{ sym. :: + exch. means} \\ C : D :: A : B & \text{ sym. ::} \\ D : B :: C : A & \text{ sym. :: + exch. means + sym. ::} \\ D : C :: B : A & \text{ exch. means + sym. :: + exch. means + sym. ::} \end{aligned}$$

Because of these eight equivalent forms, the enumeration time can be divided by a factor of 8, but the complexity remains $O(n^4)$.

To make our point clear, consider the following naïve estimation. In a preliminary experiment, we estimated the average time needed for to verify one analogy between four Chinese characters using 36 features (see Section 6.3 for a description of the features). An average time of 0.8 ms was measured. For almost five thousand Chinese characters (see Section 6.3 for a description of the data), and knowing that there are a

little bit less than 3.2×10^7 seconds in a year, verifying all possible analogies would take almost four hundred years.¹

In order to reduce the complexity of this problem, we propose to modify our goal. Rather than aiming at individual analogies, we compute all possible ratios between all possible objects at hand. This computation is basically $O(n^2)$. The result of this computation allows us to cluster pairs of objects according to their ratios. These clusters summarize all possible analogies between all objects in a non-redundant way that still provides the total amount of information (see Section 3). The sequel of the paper shows how to compute such clusters and presents some of the actual results of such a computation on a set of Chinese characters.

3 Analogical Clusters

3.1 Objects as Feature Vectors

In this work, we represent an object by a vector of features with numerical values. We also impose that the feature space be the same for all objects, so that it is trivially possible to define a ratio between two objects as the vector of their difference. In such a setting, conformity is trivially reduced to equality between vectors, more precisely equality between difference vectors.

The following equation illustrates a possible case of proportional analogy between vectors in a four-dimensional space.

$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 3 \\ 2 \\ 2 \\ 1 \\ 2 \end{pmatrix} - \begin{pmatrix} 2 \\ 2 \\ 2 \\ 0 \\ 2 \end{pmatrix}$$

Vector difference as a ratio, and equality between vectors as conformity, consistently define analogies that meet the intuitive notions about proportional analogies. Among other properties, the eight forms of equivalence for the same proportional analogy (see above Section 2) always hold.

¹

$$\begin{aligned} 5,000^4 \times 0.8 \text{ ms} / 8 &> 5^4 \times 10^9 \times 0.1 \text{ s} \\ &> 125 \times 10^8 \text{ s} \\ &> 1250 \times 10^7 / (3.1563 \times 10^7) \text{ years} \\ &> 394.2 \text{ years} \end{aligned}$$

3.2 Transitivity of Conformity: Analogical Clusters

It is not always the case that conformity verifies transitivity. For instance, [18] shows that the intuitive notions of proportional analogy between strings of characters imply that there is no transitivity for conformity.²

In our setting with conformity being an equality, i.e., an equivalence relation, transitivity naturally holds in addition to reflexivity and symmetry. For proportional analogies, transitivity of conformity implies that:

$$A : B :: C : D \text{ and } C : D :: E : F \Rightarrow A : B :: E : F$$

For our present task of enumerating all possible proportional analogies between all objects in a given set, a transitive conformity can lead to an enormous economy in representation. To illustrate this point, consider the following three proportional analogies.

$$\begin{aligned} A : B :: C : D \\ C : D :: E : F \\ A : B :: E : F \end{aligned}$$

They can be represented in a more economical way by the following list of equal ratios:

$$\begin{aligned} A : B \\ C : D \\ E : F \end{aligned}$$

All ratios being equal, any possible proportional analogy formed by taking any two ratios holds.

From the above example, it is clear that, provided conformity is transitive, a list of n pairs of objects with the same ratio stands for a list of $n \times (n - 1) / 2$ non-trivial proportional analogies (see Section 3.5 for trivial analogies). Consequently, under the assumption of transitivity for conformity, the problem of enumerating all possible proportional analogies between all possible objects in a given set can be transformed into a problem of enumerating all possible pairs of objects with the same ratio. The former problem has a complexity of $O(n^4)$ while the latter one has a complexity of $O(n^2)$.

From now on, we shall call a list of ratios of objects with the same value, an *analogical cluster*.

3.3 Equivalent Forms of Analogy: Redundancy of Clusters

Each analogical cluster stands for a different ratio, i.e., a vector that represents the difference between any two feature vectors each representing an object.

Because the order in analogical clusters is not relevant, an analogy extracted from an analogical cluster stands for two equivalent forms, obtained by symmetry of conformity (sixth line in the eight equivalent forms of proposition analogy in Section 2).

² This comes from the fact that some analogies between strings of characters admit multiple solutions. When this is the case, then, there is not transitivity for $::$ in the general case for the objects considered (see [18, p. 113]).

$$A : B :: C : D \Leftrightarrow C : D :: A : B$$

By inversion of ratios (third line in the eight equivalent forms of proposition analogy in Section 2), a proportional analogy involves two different ratios. And by exchange of the means, (second line in the eight equivalent forms of proposition analogy in Section 2), another two different ratios.

$$A : B :: C : D \Leftrightarrow B : A :: D : C \Leftrightarrow A : C :: B : D$$

Consequently, in total, the eight different forms of the same proportional analogy are to be found in four different analogical clusters (and only four clusters) among all the possible clusters that are output by a method yielding all the possible clusters standing for differences between all feature vectors representing all the objects in a given set.

Figure 3 shows such four analogical clusters for the proportional analogy $A : B :: C : D$. These four clusters are redundant because of the eight equivalent forms for the same proportional analogy, as we have just stated:

- In any cluster, the order of appearance of the pairs of objects being irrelevant, each cluster encapsulates two equivalent forms of the same proportional analogy. This is symmetry of conformity.
- Analogical clusters (1) and (2) together contain the same information as clusters (3) and (4) together. The relation between (1) and (2) (and between (3) and (4)) is the exchange of the means.
- Analogical clusters (1) and (3) are indeed the same up to an exchange of the objects on the left and the right of the : sign. This is actually inversion of ratios. The same is also true for clusters (2) and (4).

It is trivially possible to eliminate the redundancy between clusters (1) and (2) and clusters (3) and (4). This can be done by avoiding the computation of the difference between two vectors and its opposite value (the same two vectors in the reverse order). For that, it suffices to sort all the vectors in some predefined increasing order, and to compute only the differences between two vectors ranked in that order. In this way, a particular proportional analogy will appear in two, and only two, different analogical clusters among the set of all clusters. As a result, globally, the set of all clusters will contain no redundant information.

3.4 Equality of Feature Vectors: Separation of Space

By definition of the ratio as a vector difference, the case where $A : B$ and $A : C$ belong to the same analogical cluster is only possible if the vectors representing B and C are the same. This can only happen if the feature vectors do not separate the space of objects into each individual object. Reciprocally, if the feature vectors are unique for each different objects in the given set, the two ratios $A : B$ and $A : C$ for different B and C will be different. For our proposed method, this implies to check for the *separation of the space of objects* before proceeding to clustering.

| | | Cluster number | | | |
|----------|----------|----------------|----------|--|--|
| (1) | (2) | (3) | (4) | | |
| $A : B$ | \vdots | \vdots | \vdots | | |
| \vdots | $B : D$ | $B : A$ | $C : A$ | | |
| \vdots | \vdots | \vdots | \vdots | | |
| $C : D$ | $A : C$ | \vdots | $D : B$ | | |
| \vdots | \vdots | $D : C$ | \vdots | | |

Fig. 3: For a given proportional analogy $A : B :: C : D$, the set of analogical clusters output by a method that looks for all possible vector differences between all possible feature vectors representing objects in a given set should include four clusters.

3.5 Trivial Analogies: Informativity of Clusters

Finally, we must mention a particular case of no interest as it does not bring any information. This is the special case of the cluster for the null vector; i.e., null ratio. It has the following form.

$$\begin{aligned} &A : A \\ &B : B \\ &C : C \\ &\vdots \end{aligned}$$

It represents the set of all *trivial proportional analogies*, i.e., proportional analogies of the form: $A : A :: B : B$. As our interest is the enumeration of informative analogical clusters we simply avoid to produce this cluster.

By exchange of the means, trivial analogies are equivalent to analogies of the form $A : B :: A : B$. Enumeration of pairs of objects in a predefined sorting order trivially ensures that the difference between two objects is never computed twice. However, it does not prevent from outputting clusters that would contain only one pair of objects. This happens when two objects have a unique vector difference. This problem will be tackled in Section 5.1.

4 Informative and Non-redundant Enumeration of Analogical Clusters

4.1 Feature Tree and Quadratic Exploration of the Feature Tree

Each object is represented by a vector of features. An order can be imposed on the features. In this way, each vector is considered as a list with a recursive structure of a head (the first feature value) and a tail (the remaining features). The lexicographic

order, relying on the order on integers, can be applied to such a set of lists. In this way, it is possible to sort the feature vectors representing all objects.

A tree structure underlies such an ordered list. For the first feature, each different value can be encoded in one node. Each such node can be assigned the interval that represents the span over the sorted list of objects. This can be recursively applied to each interval with the tail of the feature vectors considered as lists (thus for the second feature and so on) to build a tree structure where the levels stand for each different feature and where each node holds the interval of the sorted objects with the same value for that feature, all values above being equal. On the last level, each interval should be reduced to one object if the space is well separated. Such a tree structure can be traversed in breadth-first order. Figure 4b illustrates such a data structure for a set of 5 feature vectors given in Fig. 4a.

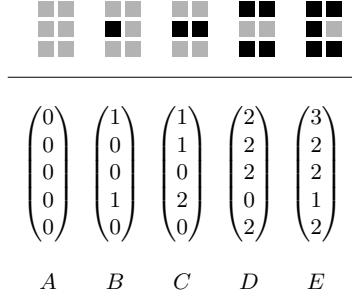
This tree structure is the same as the one we used in two of our previous works: for the complete enumeration of all analogies between sentences contained in corpora of 100,000 short sentences in Chinese, Japanese and English [5] (with sequences of bits as features and various ratios for various features and automatic sorting of the features for early detection of useless zones in the cluster space so as to speed up the overall process); and for the enumeration of clusters reflecting linguistic oppositions among 40,000 short sentences in English and Japanese [19]. In these two works, respecting the equality of edit distance for analogies of commutation between strings of characters implied extra processing.

This tree structure is quite different from the one used in [20] to search a space of strings of characters for analogies. Most importantly, our goal is different as we aim at a complete enumeration of all possible ratios, which compelled the design of the proposed data structure. In [20], the goal is more specific as it consists in retrieving all the possible pairs of objects that would be in analogical relation with a given pair of objects. This implies the two main differences given hereafter.

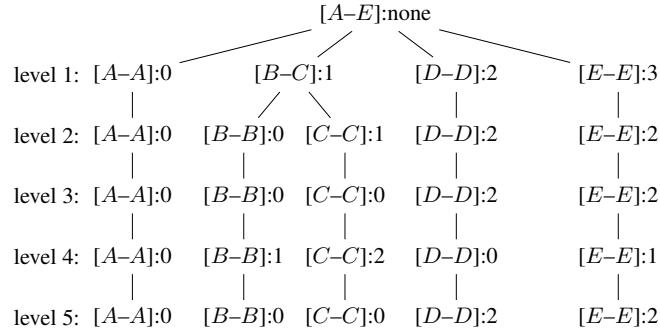
Firstly, the geometry is different. In [20], the nodes on the same level may correspond to different characters (i.e. features). This is not the case here. Each level must correspond to exactly the same feature. Consequently, on the contrary to the structure in [20] no intermediate node can stand for an object. All the objects are to be found on the leaves.

Secondly, the labels borne by the nodes are different. In our proposed data structure, the nodes bear the spanning interval in the sorted list of objects and the value of the feature; the name of the feature, being useless, is forgotten. This is of primary importance for the parallel traversal in sorted order of objects with the first interval never overtaking the second interval so as to avoid redundancy (see Section 3.3 and see below).

The computation of all ratios between all feature vectors simply consists in traversing the same tree in parallel in breadth-first order (a kind of a Cartesian self-product), and computing the differences between the values on each pair of nodes. For the same difference at a given local level, all the pairs of intervals are memorized as a block. This procedure is recursively applied down to the last level for each different value at a local level. Figure 5 illustrates this process for the feature tree given in Fig. 4.



(a) A set of five objects. Each object is a binary image of size 2 columns \times 3 rows. For each object a feature vector of size 5 can be built by counting the number of black pixels on each column and on each row. The objects are sorted according to a given sorting procedure, here, the lexicographic order based on the natural order on numbers; vectors are read top-down. One analogy can be seen among these objects: $\begin{array}{c} \text{Object } A \\ \text{Object } B \\ \text{Object } C \\ \text{Object } D \\ \text{Object } E \end{array} :: \begin{array}{c} \text{Object } A \\ \text{Object } B \\ \text{Object } C \\ \text{Object } D \\ \text{Object } E \end{array} :: \begin{array}{c} \text{Object } A \\ \text{Object } B \\ \text{Object } C \\ \text{Object } D \\ \text{Object } E \end{array}$, i.e., $A : B :: D : E$. This analogy is equivalent to the analogy between vectors given in Section 3.1.



(b) The feature tree corresponding to the set of five objects represented as feature vectors given in Fig. 4a. The intervals, noted with brackets (e.g., $[B-C]$), are followed by the value of the feature on that level (e.g., 1 for $[B-C]$). The letters in the intervals refer to the naming of the objects given at the bottom of Fig. 4a.

Fig. 4: A set of objects represented as feature vectors and its corresponding feature tree.

| | [A-A]:0 | [B-C]:1 | [D-D]:2 | [E-E]:3 |
|---------|---------|---------|---------|---------|
| [A-A]:0 | | 1 | 2 | 3 |
| [B-C]:1 | | 0 | 1 | 2 |
| [D-D]:2 | | | | 1 |
| [E-E]:3 | | | | |

- (a) Computation of the value differences on the first level for the feature tree of Fig. 4b (refer to that figure for the values on this level). The blocks with the same values are the seeds of possible clusters of same ratio.

| | [A-A]:0 | [B-B]:0 | [C-C]:1 | [D-D]:2 | [E-E]:2 |
|---------|---------|---------|---------|---------|---------|
| [A-A]:0 | | 1,0 | 1,1 | 2,2 | |
| [B-B]:0 | | | 0,1 | 1,2 | 2,2 |
| [C-C]:1 | | | | 1,1 | 2,1 |
| [D-D]:2 | | | | | 1,0 |
| [E-E]:2 | | | | | |

- (b) Recursive computation of the value differences on the second level (refer to Fig. 4b for the values on the second level; the first numbers before the comma in the cells are inherited from the first level in Fig. 5a). The blocks with the same values on the first level are further decomposed into sub-blocks. E.g., the block in Fig. 5a with value 1 (3 cells) gives birth to two sub-blocks, one with value of 1,0 and the other one with value 1,1. All other blocks with different values will not be worth further exploring as they all consist of one cell of intervals reducing to one object.

| | [A-A]:0 | [B-B]:0 | [C-C]:0 | [D-D]:2 | [E-E]:2 |
|---------|---------|---------|---------|---------|---------|
| [A-A]:0 | | 1,0,0 | 1,1,0 | 2,2,2 | |
| [B-B]:0 | | | | | 2,2,2 |
| [C-C]:0 | | | | 1,1,2 | |
| [D-D]:2 | | | | | 1,0,0 |
| [E-E]:2 | | | | | |

- (c) Recursive computation of the value differences on the third level (same principle as Fig. 5b). The block with value 1,0,0 corresponds to the two cells $[A-A] \times [B-B]$ and $[D-D] \times [E-E]$; further exploration on the next lower levels will check for equality of differences of values until the last level; thus, these two cells represent the analogy $A : B :: D : E$. The same thing happens for the two cells $[A-A] \times [D-D]$ and $[B-B] \times [E-E]$ standing for the analogy $A : D :: B : E$ (exchange of the means of the previous one). The two cells with values 1,1,0 and 1,1,2 will not be further explored as they consist of one cell of intervals reducing to one object; they would give rise to degenerated clusters.

Fig. 5: Illustration of the computation of all ratios for the set of objects given in Fig. 4a. In each cell, the value inherited from the previous level is concatenated with the difference: value in column minus value in row on the current level. Empty white cells are not computed either to avoid redundancy (opposite values) or because they correspond to a ratio that is unique. Black cells are not considered because they would give rise to trivial analogies: they are located on the diagonal and their corresponding interval is reduced to one object.

4.2 Sketch of the Method

The following gives a sketch of the proposed method.

- Convert each object into a feature vector;
- check for separation of space;
- define an order on the feature vectors (we use least correlations of values among features);
- sort the feature vectors according to lexicographic order in the defined order of features;
- build a feature tree for the sorted feature vectors;
- traverse the feature tree in parallel in breadth-first order to compute the differences between the feature vectors by blocks;
- output the list of pairs of intervals (on the last level, each interval should be reduced to one object if the space is well separated) that corresponds to each vector difference.

By construction and by definition, each list of pairs of objects, that share the same feature vector difference, is an analogical cluster.

The computation in Fig. 5 already illustrates the following improvements that can be made to the computation.

Sections 5.1 and 5.2 show that it is possible to terminate the exploration by checking for some structural conditions on the list of pairs of intervals memorized.

In the parallel traversal, we impose that for two lists of pairs of intervals to be processed, the first list be strictly before the second list. This is tantamount to explore only the upper corner of a matrix excluding its diagonal. This avoids redundancy and non-informativity when computing all possible analogies: the ratio of two vectors is computed once, its opposite is not (Section 3.3); intervals that are reduced to one object on the diagonal are checked to avoid trivial clusters (Section 3.5).

The effect of these improvements is an important practical reduction in computation. For the example in Fig. 5, the number of cells computed to output the only analogy that can be found in the set of objects is:

$$\begin{array}{r} \text{7 cells on the 1st level} \\ + \text{ 9 cells on the 2nd level} \\ + \text{ 6 cells on the 3rd level} \\ + \text{ 4 cells on the 4th level} \\ + \text{ 4 cells on the 5th level} \\ \hline = \text{ 30 cells} \end{array}$$

out of all the possible 125 cells ($5 \text{ objects} \times 5 \text{ objects} \times 5 \text{ levels}$). This is a reduction of 75% of the computation.

5 Improvements

5.1 Elimination of Clusters Reduced to one Pair of Objects

We call degenerated clusters those clusters which contain only one pair of objects, $A : B$. Obviously, such clusters do not give rise to any analogy other than the triv-

ial analogy $A : B :: A : B$ and are thus not worth to output. An early detection of such cases leads to an important reduction in processing time.

The implementation of the early detection of such degenerated clusters relies on the data structure of feature tree. After the computation of all possible differences between all possible vectors down to a certain level in the tree, it is easy to scan all the differences and look at the intervals they represent. If a set of pairs of intervals contains only one pair of intervals, each of which being reduced to one object, this is a case of a degenerated cluster. Such a cluster may be immediately deleted so as to stop any further computation on the lower levels.

A comparison of two runs of the programs with or without early detection of degenerated clusters is given in Table 2. It shows that, for our special case of structuring Sino-Japanese characters, a reduction of one third of the computational time can be achieved. There exists some overhead as is shown by the fact that an increase of 55% in computational time is observed for 1,000 characters.

Table 2: Comparison of runtimes with or without detection of degenerated clusters (clusters reduced to one pair of objects). The computation runtimes without detection of degenerated clusters are obtained for the technique described in Sections 4.1 and 4.2.

| number of chars processed | runtimes in seconds | | time reduction in percentage |
|------------------------------|---------------------|------|---------------------------------|
| | without | with | |
| 1,000 | 9 | 14 | +55 % |
| 2,000 | 39 | 36 | -7 % |
| 3,000 | 92 | 82 | -10 % |
| 4,000 | 173 | 142 | -17 % |
| 5,000 | 277 | 219 | -20 % |
| 6,000 | 426 | 313 | -26 % |
| 7,000 | 605 | 438 | -27 % |
| 8,000 | 739 | 557 | -24 % |
| 9,000 | 944 | 702 | -25 % |
| 10,000 | 1204 | 836 | -30 % |
| 11,000 | 1517 | 1123 | -25 % |
| 12,000 | 1864 | 1302 | -30 % |
| 13,000 | 2265 | 1342 | -40 % |
| 14,000 | 2646 | 1791 | -32 % |
| 14,655 | 2873 | 1889 | -34 % |

5.2 Conditional Elimination of Clusters Reduced to one Proportional Analogy

In Section 3.5, it was shown that an analogy appears in only two analogical clusters. For economy of description, we would like to eliminate redundant information as most as possible. When an analogy belongs to two clusters that contain a large number of pairs of objects, it is *a priori* impossible, without loss of information, to remove those lines

that correspond to this analogy from one of the cluster. This is not the case when one of the analogy is reduced to a cluster that contains only one analogy, i.e., exactly those two lines corresponding to the analogy at hand. This situation is illustrated below:

$$\begin{array}{ccc}
 & \vdots & \\
 & A : B & \\
 & \vdots & \\
 E : F & & A : C \\
 & \vdots & B : D \\
 & C : D & \\
 & \vdots &
 \end{array}$$

Here the analogy $A : B :: C : D$ can be read in two clusters. One on the left, under the form $A : B :: C : D$, and another one on the right, under the equivalent form $A : C :: B : D$. On the right, the cluster does not contain any other pair of objects than the two ones constituting the analogy in question. On the left, the two pairs $A : B$ and $C : D$ are just two pairs among many other pairs (noted by enumeration dots).

In such a case, the cluster on the right does not bring any additional information, as an equivalent form of the analogy is already present in the cluster on the left. Consequently, it is possible to delete the cluster reduced to one analogy, on the left, without deleting any information.

This can be performed during the enumeration of analogical clusters, level by level, using the feature tree. In this case, clusters reduced to one analogy should be memorized on each level. At the end of the exploration of each level of the feature tree, such clusters can be removed from the list of clusters to explore further. This should lead to a reduction in the total computational time. Our current implementation does not make use of this possibility and performs the deletion of clusters reduced to one analogy after complete enumeration of analogical clusters, in a post-processing phase, which induces some additional computation cost.

6 Experiments

In the frame of a larger study concerned with measuring the ease with which learners can remember Chinese characters along with their pronunciation, we are interested in studying the regularities and the correspondences between the Chinese graphical forms of characters and their pronunciation.

It is known that Chinese characters exhibit some structure and are made of graphical elements which reflect either some iconic meaning or some pronunciation. As a first step in this study, we extracted all the possible analogies between Chinese characters using a fixed-sized font. We report hereafter some of the results obtained.

6.1 The Structure of Chinese Characters

Based on a classification given in Xǔ Shèn's book 說文解字 /shuōwén jiězì/ 'Explaining and Analyzing Characters', written in the second century, the tradition considers 6 main categories (六書 /lùshū/) for the constitution of individual Chinese characters:

- pictograms (象形): representation of the object itself, e.g. 月 'moon';
- symbols (指事): abstract representation of the notion, e.g. 上 'above';
- radical-pronunciation (形聲): combination of a meaning part and a pronunciation part, e.g. 河 'river', pronounced /hè/, is written as 氵, the pronunciation of which is similar: /kè/, with the addition of a semantic graphical part for the notion of water: 氵;
- composition (會意): e.g. 林 'woods', 森 'forest' are obtained by reduplication of 木 'tree';
- modification (轉注): a character is adopted for another word with a similar pronunciation and meaning, e.g. 長 'long' (adj.) or 'to grow' (verb) depending on tone;
- borrowing (假借): a character used by extension to write another word with a similar pronunciation is specialized for this other word usually because of a higher frequency; the older word is then written with another character; e.g. the character for 'leather clothing', 衣, being also used more and more for 'to seek' (same pronunciation), was replaced by 衣 = same character + 衣 'clothing'.

A large number of Chinese characters are obviously decomposable into components and it is usually claimed that almost 80% of the Chinese characters can be considered as belonging to the 3rd category of radical-pronunciation characters. The most frequent structure of such characters consists of two elements, one being a pronunciation clue and the second one being a meaning clue, usually called semantic key, hence the name of phono-semantic characters. An illustration is given in Fig. 6.

This structure, although being quite common, is not valid for all characters. It is also believed that, because of phonetic changes, many characters that exhibited such kind of structure in ancient times cannot be interpreted in this way anymore.

In this paper, we are not interested with the relationship between graphical form and pronunciation. Our goal is limited to the extraction of the graphical structure of Chinese characters by automatic means.

To perform our experiments, we use two sets of characters. The first one is a set of almost 15,000 Sino-Japanese characters and the second one is a set of 5,000 Chinese simplified characters. To compute the analogical structure of both sets, we use monospace (or fixed-width or fixed-size) fonts. Monospace fonts are lists of characters described as binary images of fixed height and width.

6.2 15,000 Sino-Japanese Characters

The font we use in our first experiment is knj10B.bdf³. It contains 14,655 Sino-Japanese characters in the range between the Unicode codepoints 13,312 (—) and 40,891 (諱).

³ Font designed by Nagao Sadakazu (snagao@tkb.att.ne.jp). We use version 1.1 of 1999.

| | | | |
|-----|---------------------------------------|-----|--|
| 京:先 | identical left part (semantic key) | 江:亻 | identical right part (pronunciation clue) |
| 涼:洗 | 氵 [water] | 泮:伴 | 半 pàn |
| 涼:洗 | 冫 [ice] | 涼:惊 | 京 liàng |
| 惊:侁 | 亻 [human] | 洗:侁 | 先 ēn |

Fig. 6: On the left, on each line, the characters share the same semantic key. On the right, the characters share a same pronunciation indicated by the right part. Only four of the above characters are in use nowadays. The pronunciation clue refers to old pronunciations for some of these characters and is not necessarily valid for modern days Chinese pronunciation.

Figure 7 shows a sample of these characters. As shown in this figure, the characters in this font have a fixed height of 18 lines and a fixed width of 24 pixels. The actual width used is 18 pixels, so that this font is a 18×18 pixel font, the value of 24 comes from the encoding of each line of the icons on 3 bytes that we use without change. Figure 7 visualizes the representation of three randomly selected characters from this font.

| Unicode codepoint: | u24381 | u40863 | u40865 |
|--------------------|--|--|---|
| Bitmap data: | <pre>.....0..0..0. .0000..0..0..0.0.....00000.0..0..0..0..0.0..0..0..0..0. .0000..0..0..0000. .0.....0. 0.....00000000. .0000..0.0.....00000000.0.....00000000.0.....00000000.0.....00000000.0.....00000000.0.....00000000.0.....00000000.</pre> | <pre>.....0.0.0.0.0.0.0.0.0.0.0.0.0.</pre> | <pre>.....0..0.0..0.0..0.0..0.0..0.0..0.0..0.0..0.0..0.0..0.</pre> |
| Binary image: | | | |

Fig. 7: Visualization of three Sino-Japanese characters (木, 龟 and 父) from the font knj10B as binary images. The second one is not used in Japanese but belongs to the font. White pixels are visualized as a dot, black pixels as a small circle.

Features Used The features we used are simply the number of black pixels on each row and each column. We use 18 lines and 18 columns; making up a sum of 36 features.

per characters. As an illustration, the feature vector for the leftmost character in Fig. 7 is given in Fig. 8.

We checked that the space of objects is completely separated by these features, *i.e.*, each vector of feature values represents only one object in the set of objects. The above 36 features discriminate each character among the 14,655 characters used in our experiments.

Analogical Clusters Obtained We applied our method to extract the graphical structure of the 14,655 characters in our selected fixed-point font. The program, written in Python, needed less than 30 minutes to terminate⁴.

Figure 9 shows a sample of 15 clusters output by the method. Visual inspection reveals the typical structure of characters decomposed into a left and a right part. This is in no way surprising given the features that we used and the frequency of this structure.

Figure 10 shows two examples of less usual patterns that consist in the addition of some elements in the middle of characters or a longer stroke in the central part of the characters.

Experiments performed with a lesser number of features show examples of clusters where the differentiation between the characters is not sufficient. An example of this is given in Fig. 11. In this example, only the numbers of black pixels on each row have been used (18 features). In this case, the vertical directions of the strokes in the left characters on the first and second lines cannot be distinguished so that the method concluded to a proportional analogy that may be questionable or not (for reasons of equivalence between various forms of writing).

The distribution of clusters by number of pairs of objects (36 features) is plotted on Fig. 12. This distribution exhibits a long tail. Few clusters are very large while short clusters are more numerous, which sounds intuitive. The fact that the number of clusters with 3 pairs of characters (451) is greater than the number of clusters with only two pairs of objects (216) is explained by the elimination of redundant clusters reduced to one analogy. The largest cluster contains 55 pairs of characters.

Number of Clusters per Character The number of different characters that appear in at least one analogical cluster was 5,982. This represents 41% of the total number of characters used (14,655). We *a priori* expected a higher number.

We also measured the number of (non-redundant) clusters each character appears in. Figure 13 plots the distribution of characters per number of clusters they appear in. The use of logarithmic scales suggests a Zipfian distribution that needs more enquiry. This measure gives an estimation of the complexity of a character by the number of oppositions it has with all other characters. This reflects its degree of freedom in the overall graphical system. A character which does not appear in any cluster is somehow free relatively to the overall system. From the point of view of acquisition, we hypothesize that characters that appear in more clusters may be easier to remember if the learner has access to the global view given by the clusters. In addition, of course, the number of strokes should be taken into consideration.

⁴ We use a machine with 4 Gb memory equipped with an Intel Core i5 processor at 1.7 GHz.

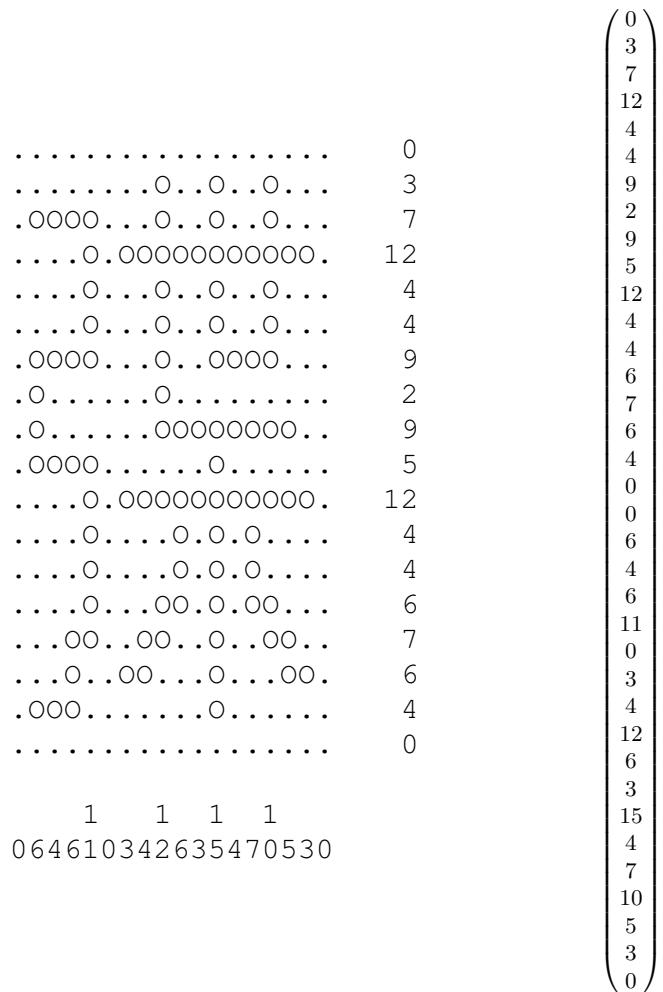


Fig. 8: Values of features for the leftmost character of Fig. 7. The features used here are the number of black pixels (represented by a small circle) on each row and each line. Eighteen rows and eighteen columns are used, hence a total of 36 values. The values for each row are given to the right of the bitmap. The values for each column are given below the bitmap. The vector of feature values for this character is given on the right. This vector lists the values for each row followed by the values for each column.

| | | | | |
|-----|-----|-----|-----|-----|
| 謁:鐸 | | | | |
| 譚:鍾 | | | | |
| 証:鉉 | | | | |
| 謫:鎬 | 棵:稞 | 練:鯀 | 詖:棟 | 課:稞 |
| 諦:錠 | 忼:秎 | 結:鮀 | 洛:格 | 詞:桐 |
| 誕:鋌 | 𠂇:祇 | 綰:鮀 | 詭:榦 | 謫:稿 |
| 讀:饋 | 忄:杭 | 紹:鮀 | 諧:檜 | 諗:稔 |
| 論:鑰 | 忊:粘 | 純:鮀 | 誠:械 | 詔:稻 |
| 談:銕 | 惄:稽 | 紺:鮀 | 議:儀 | |
| 諧:錯 | 悄:稍 | | | |
| (1) | (2) | (3) | (4) | (5) |

| | | | | |
|-----|-----|-----|-----|------|
| 涼:倞 | 玿:璣 | 冷:泮 | 滲:沼 | 悍:犴 |
| 泮:伴 | 沼:壽 | 冷:泮 | 璆:玿 | 俎:俎 |
| 津:律 | 招:囂 | 冷:泮 | 璆:召 | 狴:狴 |
| 洗:侁 | 召:囂 | 扠:拌 | 𢃥:召 | |
| (6) | (7) | (8) | (9) | (10) |

| | | | | |
|------|------|------|------|------|
| 謫:談 | 璣:玿 | 暭:曄 | 找:括 | 鎮:稹 |
| 槁:棖 | 壽:滔 | 澤:潼 | 耽:耽 | 鎬:稿 |
| 鎬:銕 | 囂:招 | 擇:撞 | | |
| (11) | (12) | (13) | (14) | (15) |

Fig. 9: A sample of 15 analogical clusters output by our clustering method on 14,655 characters from the font knj10B. For Clusters (1) to (6), Cluster (10) and Cluster (15), both characters on the same line share the same right part (radical). The left parts (keys) are different but common to all lines. Conversely, in the other clusters the right part of the characters is the same in each column of the cluster. These clusters show commutations of various keys with only two different radicals.

| | |
|-----|-----|
| 口:回 | 另:另 |
| □:回 | 余:余 |
| 匚:匱 | |

Fig. 10: Two examples of less typical cases of analogical clusters output by our clustering method. The first one on the left shows the insertion of a square in the middle of the character. The second one captures a longer stroke in the center of the characters on the right.

| | |
|-----|-----|
| : | : |
| 消:洸 | 俏:恍 |
| : | : |

Fig. 11: A cluster output by our clustering method using only the number of black pixels on each row as features. Taking into account the columns eliminates this analogy.

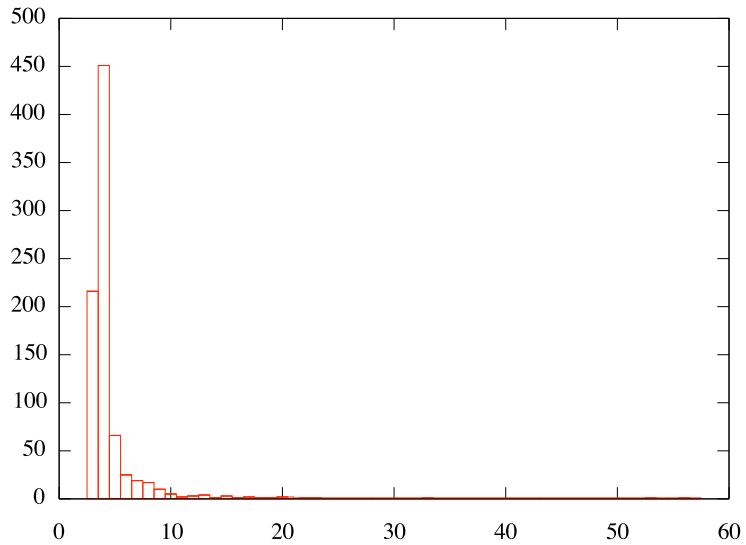


Fig. 12: Distribution of clusters by number of pairs. In abscissae, number of pairs in the clusters. In ordinates, number of clusters with the same number of pairs. The largest cluster contains 55 pairs. There are only 16 clusters between 13 and 55 on the horizontal axis.

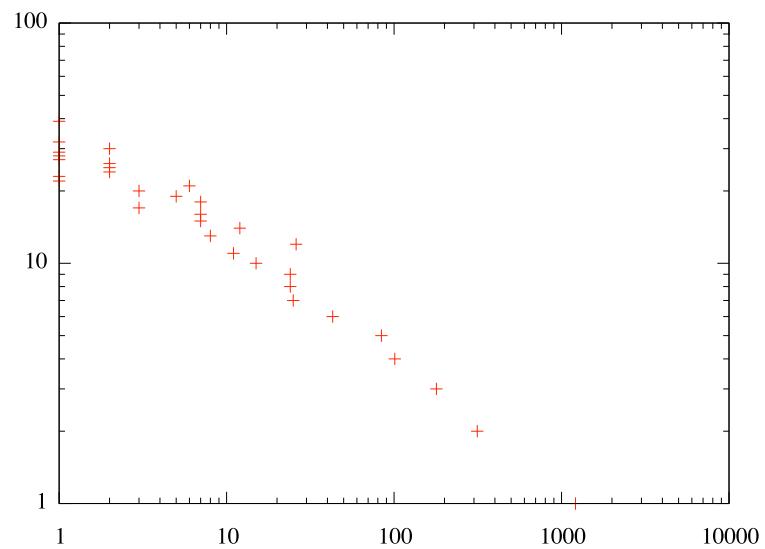


Fig. 13: Distribution of characters by number of clusters they appear in. In abscissae, number of characters that appear in the same number of clusters. In ordinates, number of clusters. Logarithmic scales.

6.3 5,000 Most Frequent Simplified Chinese Characters

The font we use in this second series of experiments is FireflyR16⁵. The characters in this font have a fixed height of 16 rows and a fixed width of 16 pixels. Figure 14 visualizes the representation of three randomly selected characters from this font. This font contains traditional and simplified Chinese characters.

From this font, we extracted 4,997 simplified Chinese characters by intersection with a list of the most frequent 5,000 simplified Chinese characters (three characters absent from the font). The frequencies come from the usage on Usenet and the information is available through Unihan data⁶. The 4,997 characters extracted are scattered inside the range of the Unicode codepoints between 19,968 (一) and 40,866 (龠).

Fig. 14: Visualization of three simplified Chinese characters (伴, 暖 and 纱) from the font FireflyR16 as binary images. In the bitmap data, white pixels are visualized as a dot, black pixels as a small circle.

Features Used In this second series of experiments, we explore a certain number of features. The set of features is presented in Table 3. We use the same basic features as in the first experiments with the number of black pixels on each row or column (features “lin”, “col” and “lincol”). A feature vector for a specific character and the features “lin” is given in Fig. 15. In addition, we tested the use of more elaborated features inspired by classical processing of binary images: number of pixels with a given contextual configuration (“nei”) and number of pixels with the same gray level for its neighborhood (“ngr”). A classical data structure on images is that of a quadtree [21]. We use this data structure and count the number of black pixels in each node of

⁵ See http://wiki.debian.org/w/Make_Debian_support_Chinese.

⁶ See <http://www.unicode.org/reports/tr38/tr38-10.html>

Table 3: Name and description of the different features used.

| Name | Number of dimensions of the vectors representing the objects | Possible values |
|--------|---|--|
| col | 16 = Number of columns. | Number between 0 and 16 of black pixels on each column. |
| lin | 16 = Number of rows. | Number between 0 and 16 of black pixels on each row. |
| lincol | $32 = 16 + 16 =$ Number of rows and columns. | Number between 0 and 16 of black pixels on each row or column. |
| nei | $512 = 2^9$ = Possible number of possible exact configurations around a pixel, this pixel included. | Number of pixels, between 0 and $16 \times 16 = 256$, in this configuration. |
| ngr | $9 = 8+1$ = Possible number of black pixels around a pixel, the pixel included. | Number of pixels, between 0 and $16 \times 16 = 256$, with this number of black pixels around it, itself included. |
| pix | 256/n pixel positions selected at random depending on the value of the parameter n: – 1: 256 = all pixels, – 2: 128 = half of the pixels at random, – 3: 85 = one third of the pixels, – 4: 64 = a quarter of the pixels, – 5: 51 = one fifth of the pixels, – etc. | 0 or 1 |
| pyx | 256 × (n – 1)/n pixel positions selected at random according to the value of the parameter n: – 2: 128 = pix 2 (thus, not used), – 3: 171 = two thirds of the pixels, – 4: 192 = three quarters of the pixels, – etc. | 0 or 1 |
| qua | Number of nodes in the quadtree according to the number of levels in the quadtree: – 0: 1 (too rough, not used here), – 1: $1 + 4 = 5$ (too rough, not used here), – 2: $1 + 4 + 16 = 21$, – 3: $1 + 4 + 16 + 64 = 85$, – 4: $1 + 4 + 16 + 64 + 256 = 341$. | Number of black pixels in each node of the quadtree, i.e., a number between 0 and 256 for nodes of level 0, between 0 and 64 for nodes of level 1, between 0 and 16 for nodes of level 2, etc. |

the quadtree to form a certain number of sets of features according to the number of levels in the quadtree (“qua”). As intuitively not all pixels are necessary to get an image of the characters represented, we also created a certain number of sets of features by selecting pixel positions at random (“pix” and “pyx”). The actual value of the pixel, black or white, is used as the value of such features.

For each of the features used, we checked whether the space of objects is completely separated by these features, *i.e.*, whether each vector of feature values represent only one object in the set of objects. This is not the case for each set of features and the number of indistinguishable characters, *i.e.*, characters with the same feature vector as some other one, is given in Table 4 for each of set of features. A maximum of 35 indistinguishable characters is observed for “pix 7” which stands for a set of 37 pixel positions selected at random out of 256 possible positions. This shows that the binary images representing these simplified Chinese characters are highly graphically redundant.

Table 4: Number of characters indistinguishable from another one (doubles) according to each set of features and number of analogies output for each set of features. The number of analogies is given here as the analogies (not the clusters themselves) serve as comparison across the experiments with different features (see Table 5).

| | col | lin | lincol | nei | ngr | pix 1 | pix 2 | pix 3 | pix 4 | pix 5 | pix 6 | pix 7 |
|---|-----|-----|--------|-----|------|-------|-------|-------|-------|-------|-------|-------|
| # of doubles | 1 | 1 | 0 | 1 | 2 | 0 | 2 | 1 | 8 | 16 | 31 | 35 |
| # of analogies | 14 | 40 | 8 | 0 | 6042 | 8 | 42 | 43 | 115 | 308 | 4770 | 5016 |
| pyx 2 pyx 3 pyx 4 pyx 5 pyx 6 pyx 7 qua 2 qua 3 qua 4 | | | | | | | | | | | | |
| # of doublons | 3 | 2 | 0 | 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| # of analogies | 40 | 14 | 15 | 10 | 12 | 9 | 22 | 14 | 8 | | | |

Analogical Clusters Obtained We applied our method to extract the analogical graphical structure from the selected 5,000 simplified Chinese characters represented in our selected monospace font. The same program as in the first experiment, run on the same machine used in the first experiment, needed around five minutes for each set of features.

A summary of the number of analogies is given in Table 4. The maximum is obtained for the feature set “pix 7”. A visual inspection shows that a lot of analogies obtained do not meet intuition. For an example of such counter intuitive analogy, but obtained with the feature set “pix 5”, see Fig. 16.

The best set of clusters according to human judgment is the one obtained with the feature set “pix 5”. A sample of the clusters obtained is given in Fig. 16. Again, the typical structure of characters with a left and a right part is revealed.

In order to estimate how similar or different the clusters obtained using the different sets of features deliver are, we performed a systematic comparison. The results of this comparison are given in Table 5. In this table, the sets of clusters obtained by different

| Quadtree decomposition | | | | | | Feature vector | |
|------------------------|-------------|----------|---------|---------|--------|----------------|----|
| Level 0 | | | Level 1 | | | Level 2 | |
|O.....O..... |O..... | O..... | | O..... | O..... | | |
|O.....O..... |O..... | O..... | | O..... | O..... | | |
|O.O...O..OO.. |O.O. | O..OO.. | | O.O. | O.. | OO.. | |
|O...O.O..O.... |O...O | O..O... | | O...O | O.. | O... | |
|O...O.O.O..... |O...O | O..O.... | | O...O | O..O | | |
|O...O.O.O..... |O...O | O..O.... | | O...O | O..O | | |
|O.....O..... |O..... | O..... | | O..... | O.. | | |
|OO...OOOOOOOO.. |OO..OO | OOOOOO.. | | OO..OO | OOOO | OO.. | |
|O.O....O..... |O.O... | O..... | | O.O | | O.. | |
| O...O.....O..... | O...O..... | O..... | O...O | | O.. | | |
|O.....O..... |O..... | O..... |O | | O.. | | |
|O.OOOOOOOOOOO.. |O.OOO | OOOOOO.. |O |OOO | OOOO | OO.. | |
|O.....O..... |O..... | O..... |O | | O.. | | |
|O.....O..... |O..... | O..... |O | | O.. | | |
|O.....O..... |O..... | O..... |O | | O.. | | |
|O.....O..... |O..... | O..... |O | | O.. | | |
|O.....O..... |O..... | O..... | | | | | |
| 56 | 15 | 17 | 1 | 5 | 4 | 3 | 56 |
| | 11 | 13 | 6 | 3 | 8 | 2 | 15 |
| | 11 | 13 | 5 | 3 | 7 | 3 | 17 |
| | | | 3 | 0 | 3 | 0 | 11 |
| | | | | | | | 13 |
| | | | | | | | 1 |
| | | | | | | | 5 |
| | | | | | | | 4 |
| | | | | | | | 3 |
| | | | | | | | 6 |
| | | | | | | | 3 |
| | | | | | | | 8 |
| | | | | | | | 2 |
| | | | | | | | 5 |
| | | | | | | | 3 |
| | | | | | | | 7 |
| | | | | | | | 3 |
| | | | | | | | 3 |
| | | | | | | | 0 |
| | | | | | | | 3 |
| | | | | | | | 0 |
| | | | | | | | 3 |
| | | | | | | | 0 |

Fig. 15: An example of features used: number of black pixels in the quadtree down to level 2 ('qua 2' in Table 3). The leftmost character of Fig. 14 is used. The quads are enumerated one level after another. To build the vector, on each level, the quads are read in the usual reading fashion, i.e., from left to right and from top to bottom. The resulting feature vector is given on the right.

| | | | | |
|------|------|------|------|------|
| 倔:掘 | | | | |
| 恨:恨 | | | | |
| 怕:拍 | 诘:结 | 梧:梧 | 偏:惆 | |
| 惜:措 | 调:绸 | 抗:杭 | 编:调 | 诅:祖 |
| 快:抉 | 谝:编 | 拮:桔 | 编:绸 | 诈:祚 |
| 怜:怜 | 谁:维 | | | |
| 惦:掂 | | | | |
| 俸:捧 | | | | |
| (1) | (2) | (3) | (4) | (5) |
| 铂:珀 | 湟:徨 | 技:扛 | 钡:锂 | 捅:括 |
| 锂:理 | 注:往 | 肢:胚 | 狈:狸 | 诵:话 |
| (6) | (7) | (8) | (9) | (10) |
| 扭:抄 | 跑:跃 | 咩:啼 | 烘:供 | 锟:钝 |
| 纽:紗 | 袍:袄 | 详:谛 | 煌:惶 | 锟:饨 |
| (11) | (12) | (13) | (14) | (15) |

Fig. 16: A sample of 15 analogical clusters output by our clustering method on 5,000 characters from the font FireflyR16 using 51 pixel positions selected at random as features (pix 5 in Table 3). These clusters show commutations of various supposed phonetic keys with two different supposed semantic radicals. Cluster (8) is not acceptable for the human eye. Similarly, the second line in Cluster (1) should have been left out.

Table 5: Similarity between clusters obtained using different features. Null values are replaced with a dot. Each cell in the table contains three numbers: the first one is the the percentage of analogies obtained using the set of features given on the row that are included in the set of analogies obtained using the set of features given on the column; the second one is the same percentage, but with row and column exchanged; the third one is the similarity between the two sets of analogies obtained using the two sets of features, computed as their Jaccard index. For readability, the cells on the diagonal are left blank but should contain: 100%, 100%, 1.0. For compactness, the set of features which were observed identical to another one (common cell containing 100%, 100%, 1.0) are not mentioned as independent lines and columns, but by equalities in the first line and the first column.

| | col | lin | nei | ngr | pix 1 = lincol | pix 2 = pyx 2 | pix 3 | pix 4 | pix 5 | pyx 3 | pyx 4 | pyx 5 | pyx 6 | pyx 7 | qua 2 | qua 3 |
|-------|-----|--|-----|-----|-------------------|------------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| | | | | | = lincol | = pyx 2 | | | | | | | | | | |
| | | | | | = qua 4 | | | | | | | | | | | |
| col | | | | | | | | | | | | | | | | |
| lin | | 57% 20% 0.17 | | | | | | | | | | | | | | |
| nei | | . | . | . | | | | | | | | | | | | |
| ngr | | . | . | . | | | | | | | | | | | | |
| pix 1 | | 57% 100% 0.5720% 100% 0.20 | . | . | | | | | | | | | | | | |
| pix 2 | | 57% 19% 0.1725% 23% 0.14 | . | . | | 100% | | | | | | | | | | |
| pix 3 | | 57% 18% 0.1635% 32% 0.20 | . | . | | 18% | 54% | | | | | | | | | |
| pix 4 | | 57% 6% 0.0730% 10% 0.08 | . | . | | 100% | 38% | 55% | | | | | | | | |
| pix 5 | | 64% 2% 0.0337% 4% 0.05 | . | . | | 100% | 78% | 67% | 24% | | | | | | | |
| pyx 3 | | 57% 57% 0.4020% 57% 0.17 | . | . | | 100% | 28% | 27% | 9% | 4% | | | | | | |
| pyx 4 | | 57% 53% 0.3820% 53% 0.17 | . | . | | 100% | 23% | 20% | 12% | 3% | 64% | | | | | |
| pyx 5 | | 57% 80% 0.5020% 80% 0.19 | . | . | | 100% | 23% | 20% | 7% | 3% | 57% | 53% | | | | |
| pyx 6 | | 57% 66% 0.4425% 83% 0.24 | . | . | | 100% | 23% | 20% | 7% | 3% | 57% | 60% | 80% | | | |
| pyx 7 | | 57% 88% 0.5320% 88% 0.20 | . | . | | 100% | 19% | 18% | 7% | 2% | 57% | 53% | 80% | 66% | | |
| qua 2 | | 57% 36% 0.2942% 77% 0.38 | . | . | | 100% | 19% | 23% | 7% | 4% | 57% | 53% | 80% | 83% | 88% | |
| qua 3 | | 57% 57% 0.4032% 92% 0.32 | . | . | | 100% | 19% | 20% | 6% | 3% | 57% | 53% | 80% | 66% | 88% | 63% |

features are expanded as sets of analogies, i.e., each cluster is expanded into the set of analogies it represents. The results of the experiments are then compared in terms of analogies in common, as the analogies are the final product of the clustering process and as such constitute the final units to compare. The number of clusters would not give such a meaningful comparison. In Table 5, each cell contains three values. The first value gives the percentage of analogies obtained using the set of features given on the row that are included in the set of analogies obtained using the set of features given on the column. The second value gives the same percentage, but with row and column exchanged. The third value is the similarity between the two sets of analogies obtained using the two sets of features, computed as their Jaccard index. For instance, the cell on the line “lincol” and column “col” contains the three values 57%, 100% and 0.57. These values should be read as follows: 57% of the analogies obtained with the “col” features are found using the “lincol” features. All the analogies (100%) found using the “lincol” features are found using the “col” features. As a result, logically, as this is a case of inclusion of one set of analogies in the other one, the Jaccard index (0.57) is equal to the inclusion percentage 57% of the smaller set in the larger one.

Different feature configurations were found to deliver the same results. For instance, the features for “pix 1” and “qua 4” had a similarity of 1. This means that both methods find exactly the same analogies. The theoretical explanation for this experimental result is as follows: the comparison in quadtree decomposition of binary images of 16×16 pixels down to four levels ($2^4 = 16$), noted by “quad 4”, is equivalent to the comparison at each pixel position, noted by “pix 1”. Similarly, although theoretically not equivalent in the general case, “lincol” and “pix 1” were found to be equivalent. The same was found for “pix 2” and “pyx 2”. These results are mentioned in the first column of Table 5 by equalities.

A negative but interesting result is that the features using the neighbors of a pixel, either by configuration (“nei”) or by level of grey (“ngr”) are totally unable to deliver any cluster. We thought that these features would grasp some global view of the characters, by capturing some information about strokes. Indeed a vertical (or horizontal) stroke includes many black pixels with the same environment of only two black pixels around vertically (or horizontally). Intuitively, such information should be important for Chinese characters. A possible explanation of the failure is that the equality of difference vectors may be too strong a constraint. Relaxing the equality to some approximation could supposedly help, but approximations imply thresholds that are sometimes difficult to justify in all generality.

7 Related Works and Conclusion

This paper presented a method to automatically extract all possible proportional analogies from a given set of objects represented by feature vectors. The sets of objects we applied this method on were sets of binary images representing Sino-Japanese or Chinese characters in monospace fonts.

The present work is part of a more general research program that attempts at exploring the ways proportional analogy structures language units, following linguistic insights (Varro, Paul, Saussure, Bloomfield, Itkonen). In previous works we have re-

ported on the use of proportional analogy between strings of symbols for various purposes:

- estimation of the number of true analogies, i.e., in form and meaning, between short sentences contained in a corpus from the tourism domain [5];
- estimation of the number of true analogies between chunks in languages of different typologies, Japanese and English [6, 7];
- measure of the proximity of 11 European languages by commonality in the structure of their vocabularies [22];
- use of analogy in machine translation and paraphrasing [23, 24].

The work reported in this paper is not directly concerned with the linguistic implications of the findings. But it represents one of the extensions of our research program to other types of data structures than strings of symbols, on datasets still in relationship with language. This is in agreement with the first of the two desirable extensions from the data structure of strings of symbols that we have already mentioned in [18].

Strings of symbols being a data structure with only one dimension, a first extension concerns the number of dimensions. The work presented here on images of black and white pixels is an answer to this problem, as binary images are two-dimensional objects on a vocabulary reduced to two symbols: the black pixel and the white pixel. The second extension, not addressed in this paper, concerns the discrete number of symbols used in strings of symbols. The use of continuous values instead of a discrete number of values is certainly desirable as, for one dimension, strings of continuous values in relation with language are sound waves of spoken utterances.

The object of the first part of the present paper was to introduce a technique that we have already used in previous works. But we did not give an in-depth description of this technique. In these works [5, 19], we applied the method presented in this paper to short sentences, in complement to the computation of the edit distance constraint which is necessary for proportional analogies of commutation between strings of symbols⁷. This method allowed us to visualize linguistically relevant oppositions between short sentences in English and Japanese, and constructions in line with construction grammars (CxG, [26]). The clusters that we obtained on these data sets illustrated a range of phenomena of different order, like:

- orthographical variations (e.g.: centre : center);
- fronting of interjections (e.g.: Do ..., please. : Please do...);
- exchange of place names, document names, item names etc.;
- positive vs comparative forms of adjectives (e.g.: green : greener);
- structural transformations, like interrogative vs affirmative;
- exchange of predicates in the same grammatical subject and object context (as illustrated by Table 6);
- questions in different levels of politeness (as illustrated by Table 7);
- etc.

⁷ [25] is the first mention of the edit distance constraint in terms of similarities; [2] gives the equivalent expression with edit distances; [17] is the published form of the proceedings in which [2] appeared, with few years delay. The edit distance constraint is necessary between strings of symbols to avoid many spurious analogies that would be formed without it.

Table 6: A cluster of short sentences that illustrates the exchange of predicates: *keep this baggage* vs *draw me a map*. The sentences have been tokenized.

| Pairs of sentences |
|---|
| could you keep this baggage ? : could you draw me a map ? |
| keep this baggage , please . : draw me a map , please . |
| will you keep this baggage ? : will you draw me a map ? |
| please keep this baggage . : please draw me a map . |

Table 7: A cluster of short sentences that illustrates the structural transformation of *i 'd like to ...* into *can i ... here ?* The first form is an understatement for a request. The second form is more direct. The sentences have been tokenized.

| Pairs of sentences |
|---|
| i 'd like to cash this traveler 's check . : can i cash this traveler 's check here ? |
| i 'd like to make a hotel reservation . : can i make a hotel reservation here ? |
| i 'd like to make a reservation . : can i make a reservation here ? |
| i 'd like to check my baggage . : can i check my baggage here ? |
| i 'd like to leave my baggage . : can i leave my baggage here ? |
| i 'd like to leave my luggage . : can i leave my luggage here ? |
| i 'd like to reserve a room . : can i reserve a room here ? |
| i 'd like to have dinner . : can i have dinner here ? |
| i 'd like to check in . : can i check in here ? |
| i 'd like to swim . : can i swim here ? |

In the present paper, relying on specific properties of our formalization of objects as feature vectors, we defined the ratio between objects as a difference between vectors, and conformity as equality between difference vectors. This particular setting allowed us to reformulate the problem, which has a complexity of $O(n^4)$, in an equivalent problem with a quadratic complexity, that of enumerating analogical clusters, i.e., lists of pairs of objects with the same ratio (Section 4).

We proposed an adequate data structure for this problem (Section 3) and, by further exploiting the properties of proportional analogies, we showed how to avoid redundancy in the enumeration and non-informative clusters (Section 5). With all this, we showed that the problem becomes tractable so as to solve our problem at hand: extracting all analogies between Sino-Japanese characters in their graphical form, in a reasonable amount of time (Section 6).

Although the iconicity of proportional analogies [14, 27] has already been stressed in a broader sense of the word, all the previous approaches to the problem of analogies between images [28, 29, 30] are, to our knowledge, based on some high level descriptions of the images, i.e., they rely on a coding of the images. It is usually left unclear how this encoding could be achieved automatically. Thus the previous approaches to the problem are not fully automatized: they do not process the images directly. To our knowledge, this paper is the first attempt at solving proportional analogies between images directly, under their actual form of binary images of black and white pixels, using their graphical form directly, i.e., without any recourse to high level abstract representations. Our approach, its application to Chinese characters, and our results, show that an explicit description of characters in terms of their constituents; i.e., keys or radicals, as used in [15], can be avoided.

Our proposed method does not exhaust the subject of proportional analogies between binary images made of black and white pixels. There remains a number of problems. One problem is the necessarily fixed size of the icons to compare, hence our use of monospace fixed-size fonts. Firstly, any shift of a character by one or several rows (or columns) would disrupt the analogical relations that are made possible to compute with our feature vectors because almost all characters are well lined up with the first line and column. Secondly, analogical relations between characters of different sizes cannot obviously be captured with the method proposed here. These two reasons lead to the fact that, for the same fixed-size font, our method is not yet able to identify analogies of the form: 木 : 沐 :: 十 : 汗. This pleads for the application of scale invariant feature transform techniques, something that we intend to test in the future.

The work presented here is a preprocessing step in a larger study of proportional analogies of graphical form and pronunciation among Chinese characters. We are also performing the same kind of analogical clustering on the level of pronunciation and compute the intersection between analogical clusters on the graphical and on the pronunciation levels. We hypothesized that knowing analogical correspondences between the graphical and pronunciation levels of Chinese or Sino-Japanese characters would ease their memorization by learners. We already performed tests on part of this hypothesis with subjects. We confirmed that the presence of graphical analogies between characters in sets of characters to remember, greatly increases their memorization, when compared with sets of characters containing no analogy [31].

References

- [1] Gentner, D.: Structure mapping: A theoretical model for analogy. *Cognitive Science* **7**(2) (1983) 155–170
- [2] Lepage, Y.: Analogy and formal languages. In: Proceedings of FG/MOL 2001, Helsinki (August 2001) 1–12
- [3] Yvon, F., Stroppa, N., Miclet, L., Delhay, A.: Solving analogical equations on words. *Rapport technique ENST2004D005*, ENST (July 2004)
- [4] Hoffman, R.R.: Monster analogies. *AI Magazine* **11** (1995) 11–35
- [5] Lepage, Y.: Lower and higher estimates of the number of “true analogies” between sentences contained in a large multilingual corpus. In: Proceedings of COLING-2004. Volume 1., Geneva (August 2004) 736–742
- [6] Lepage, Y., Migeot, J., Guillerm, E.: A corpus study on the number of true proportional analogies between chunks in two typologically different languages. In: Proceedings of the seventh international Symposium on Natural Language Processing (SNLP 2007), Pattaya, Thailand, Kasetsart University, ISBN 978-974-623-062-9 (December 2007) 117–122
- [7] Lepage, Y., Migeot, J., Guillerm, E.: A measure of the number of true analogies between chunks in Japanese. *Lecture Notes in Artificial Intelligence* **5603** (2009) 154–164
- [8] Veale, T., Chen, S.: Learning to extract semantic content from the orthographic structure of Chinese words. proceedings of the 17th Irish conference on Artificial Intelligence and Cognitive Science (AICS2006) (sept 2006)
- [9] Paul, H.: *Prinzipien der Sprachgeschichte*. Niemayer, Tübingen (1920)
- [10] Varro, M.T.: *De lingua latina*. Coll. Belles-lettres, Paris (1954) Trad. J. Collart.
- [11] Turney, P.D., Littman, M.L.: Corpus-based learning of analogies and semantic relations. *Machine Learning* **60**(1–3) (2005) 251–278
- [12] Turney, P.D.: Similarity of semantic relations. *Computational Linguistics* **32**(2) (2006) 379–416
- [13] Turney, P.: A uniform approach to analogies, synonyms, antonyms, and associations. In: Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008), Manchester, UK, Coling 2008 Organizing Committee (August 2008) 905–912
- [14] Itkonen, E.: Iconicity, analogy, and universal grammar. *Journal of Pragmatics* **22**(1) (1994) 37–53
- [15] Yencken, L., Baldwin, T.: Measuring and predicting orthographic associations: Modelling the similarity of Japanese kanji. In: Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008), Manchester, UK, Coling 2008 Organizing Committee (August 2008) 1041–1048
- [16] Matsushita, K., Lepage, Y.: Rediscovering the structure of Chinese characters using analogy-based methods (in Japanese). In: Proceedings of the 18th Japanese National Conference in Natural Language Processing, Nagoya (March 2013) 438–441
- [17] Lepage, Y.: Analogy and formal languages. *Electronic notes in theoretical computer science* **53** (April 2004) 180–191
- [18] Lepage, Y.: Of that kind of analogies capturing linguistic commutations (in French). Habilitation thesis, Joseph Fourier Grenoble University (May 2003)
- [19] Lepage, Y., Goh, C.: Towards automatic acquisition of linguistic features. In Jokinen, K., Bick, E., eds.: *Proceedings of the 17th Nordic Conference on Computational Linguistics (NODALIDA 2009)*, Odense (May 2009) 118–125
- [20] Langlais, P., Yvon, F.: Scaling up analogical learning. In: *Coling 2008: Companion volume: Posters*, Manchester, UK, Coling 2008 Organizing Committee (August 2008) 51–54
- [21] Finkel, R., Bentley, J.: Quad trees: a data structure for retrieval on composite keys. *Acta Informatica* **4**(1) (1974) 1–9

- [22] Lepage, Y., Gosme, J., Lardilleux, A.: Estimating the proximity between languages by their commonality in vocabulary structures. Lecture Notes in Artificial Intelligence **Human Language Technology – Challenges for Computer Science and Linguistics** (2011) 127–138
- [23] Lepage, Y., Denoual, E.: Purest ever example-based machine translation: detailed presentation and assessment. *Machine Translation* **19** (2005) 251–282
- [24] Lepage, Y., Denoual, E.: Automatic generation of paraphrases to be used as translation references in objective evaluation measures of machine translation. In: Proceedings of the third international workshop on Paraphrasing (IWP 2005), Jeju (October 2005) 57–64
- [25] Lepage, Y.: Languages of analogical strings. In: Proceedings of COLING-2000. Volume 1., Saarbrücken (July-August 2000) 488–494
- [26] Croft, W.: Radical construction grammar: syntactic theory in typological perspective. Oxford Linguistics. Oxford University Press (2001)
- [27] Itkonen, E.: Analogy as structure and process: Approaches in linguistics, cognitive psychology and philosophy of science. In Dascal, M., Gibbs, R.W., Nuyts, J., eds.: Human cognitive processing. Volume 14. John Benjamins Publishing Company, Amsterdam / Philadelphia (2005) 250 p.
- [28] Hofstadter, D., the Fluid Analogies Research Group: Fluid Concepts and Creative Analogies. Basic Books, New York (1994)
- [29] Lepage, Y.: Solving analogies on words: an algorithm. In: Proceedings of COLING-ACL'98. Volume I., Montréal (August 1998) 728–735
- [30] Correa, W., Prade, H., Richard, G.: When intelligence is just a matter of copying. In: Proceedings of the 20th European Conference on Artificial Intelligence (ECAI 2012). (August 2012) 276–281
- [31] Matsushita, K.: Data processing of Chinese Hanzi by proportional analogy and verification of learning efficiency by subjects (in Japanese). Master's thesis, Graduate school of Information, Production and Systems, Waseda University (March 2013)

Issues in Analogical Inference over Sequences of Symbols: a Case Study on Proper Name Transliteration

Philippe Langlais†, François Yvon‡

†Université de Montréal, Canada, email: felipe@iro.umontreal.ca

‡Université Paris Sud & LIMSI/CNRS, France, email: yvon@limsi.fr

Abstract. Formal analogies, that is, proportional analogies involving relations at a formal level (e.g. *cordially* is to *cordial* as *appreciatively* is to *appreciative*) have a long history in Linguistics. They can accommodate a wide variety of linguistic data without resorting to *ad hoc* representations and are inherently good at capturing long range dependencies between data. Unfortunately, applying analogical learning on top of formal analogy to current Natural Language Processing (NLP) tasks, which often involve massive amount of data, is quite challenging. In this paper, we draw on our previous works and identify some issues that remain to be addressed for formal analogy to stand by itself in the landscape of NLP. As a case study, we monitor our current implementation of analogical learning on a task of transliterating English proper names into Chinese.

1 INTRODUCTION

A proportional analogy is a relationship between four objects, denoted $[x : y :: z : t]$, which reads as “*x is to y as z is to t*”. While some studies have been devoted to handle semantic relationships [1, 2], we focus in this work on *formal* proportional analogies (hereafter formal analogies or simply analogies), that is, proportional analogies involving relationships at the formal level, such as $[miracle : miraculous :: fable : fabulous]$. This is different from analogies considered by computational models of analogy-making such as Copycat [3] and Metacat [4, 5]. Those systems do learn to solve puzzles on letters such as $[abc : abd :: mrrj : ?]$. In these studies, letters are nothing more than abstractions with a few meaningful relations (i.e., successorship, predecessorship and sameness) with which the system reasons.

Early work on formal analogies for Natural Language Processing (NLP) was devoted to propose computational definitions of proportional analogies. Yvon [6] proposed a definition where a prefixation or a suffixation operation was allowed between forms. In [7], Lepage proposed a richer model allowing at the same time, prefixation, suffixation, as well as infixation operations. His model is characterized in terms of the edit-distance that must verify 4 entities in (formal) analogical relation. Later on, Yvon et al. [8, 9] proposed a model of analogy

which generalizes the model of [7] thanks to finite-state machines. In particular, this model can account for inversions (i.e. *Paul gave an apple to Mary* is to *Mary received an apple from Paul* as *Paul gave a letter to Mary* is to *Mary received a letter from Paul*). Stroppa and Yvon [10] further extended this model to various algebraic structures, notably trees, which are ubiquitous in NLP. Also, Miclet et al. [11] built on the definition of [9] and defined the notion of *analogical dissimilarity* on forms. Presumably, allowing near analogies might be of interest in several AI applications. An extension of analogical dissimilarity to tree structures has been recently proposed in [12].

Another thread of studies is devoted to applications of analogical learning to NLP tasks. Lepage [7] early proposed an analogical model of parsing which uses a treebank (a database of syntactically analyzed sentences). He conducted proof-of-concept experiments. Yvon [6] addressed the task of grapheme-to-phoneme conversion, a problem which continues to be studied thoroughly (e.g. [13]). In [14], the authors address the task of identifying morphologically related word forms in a lexicon, the main task of the MorphoChallenge evaluation campaign [15]. Their approach, which capitalizes on formal analogy to learn relations between words proved to be competitive with state-of-the-art approaches (e.g. [16]) and ranked first on the Finnish language according the EMMA metric (see [17]) which is now the official metric since Morphochallenge 2010. Stroppa and Yvon [18] applied analogical learning to the computation of morphosyntactic properties associated with a word form (lemma, part-of-speech, and additional features such as number, gender, case, tense, mood, etc.). The performance of the analogical learner on the Dutch language was as good as or better than the one reported in [19].

Lepage and Denoual [20] pioneered the application of analogical learning to Machine Translation. Different variants of the system they proposed have been tested in a number of evaluation campaigns (see for instance [21]). Langlais and Patry [22] investigated the more specific task of translating unknown words, a problem simultaneously studied in [23]. In [24], the authors applied analogical learning to translating terms of the medical domain in different language directions, including some that do not share the same scripts (e.g. Russian/English). The precision of the analogical engine was higher than the one of a state-of-the-art phrase-based statistical engine [25] trained at the character level, but the recall was lower. A simple combination of both systems outperformed significantly both engines. See [26] for a technical discussion of those works. Very recently, Gosme et Lepage [27] studied the use of formal analogy for smoothing n-gram language models.

Analogical learning has also been applied to various other purposes, among which terminology management [28], query expansion in Information Retrieval [29], classification of nominal and binary data, handwritten character recognition [11], as well as for solving Raven IQ tests [30]. All these studies witness that analogical learning based on formal analogies can lead to state-of-the-art performance in a number of applications. Still, it encompasses a number of issues that seriously hinder its widespread use in NLP [26]. This motivates the present paper.

In the remainder of this paper, we first describe in Section 2 the principles behind analogical learning. Then, we provide in Section 3 an account of the issues involved in deploying analogical inference over strings in typical NLP tasks. Section 4 reports our experiments in applying analogical learning on the task of transliterating English proper names into Chinese. In Section 5, we conclude our work and identify a number of avenues that deserve further investigations.

2 PRINCIPLES

In order to understand the methodology, we first clarify the process of analogical learning. Let $\mathcal{L} = \{(i(x_k), o(x_k))_k\}$ be a training set gathering pairs of input $i(x_k)$ and output $o(x_k)$ representations for instances x_k . We call *input set*, denoted $\mathcal{I} = \bigcup_k i(x_k)$, the set of input-space representations in the training set. Given an element t for which only $i(t)$ (or alternatively $o(t)$) is known, analogical learning works by:

1. building $\mathcal{E}_i(t) = \{(x, y, z) \in \mathcal{L}^3 \mid [i(x) : i(y) :: i(z) : i(t)]\}$, the set of triplets in the training set that stand in analogical proportion with t in the input space,
2. building $\mathcal{E}_o(t) = \{u \mid [o(x) : o(y) :: o(z) : u] \text{ and } (x, y, z) \in \mathcal{E}_i(t)\}$, the set of solutions to the *analogical equations* obtained in the output space,
3. aggregating the solutions in $\mathcal{E}_o(t)$ in order to select $o(t)$.

In this description, $[x : y :: z : t]$ is our notation for a (formal) proportional analogy,¹ and $[x : y :: z : ?]$ is called an analogical equation and represents the set of its solutions. In the sequel, we call *x-form*, *y-form*, *z-form* and *t-form* the first, second, third and fourth forms respectively of $[x : y :: z : t]$. Also, we sometimes refer the two first steps of the inference procedure as the *generator*, while we call the third one the *aggregator*.

Let us illustrate this on a tiny example where the task is to associate a sequence of part-of-speech (POS) tags to any given sentence, viewed as a sequence of words. Let $\mathcal{L} = \{(he \text{ loves } her, \text{PRP VBZ PRP}), (she \text{ loved } him, \text{PRP VBD PRP}), (he \text{ smiles at } her, \text{PRP VBZ IN PRP})\}$ be our training set which maps sequences of words (input) to sequences of POS tags (output). Tagging a (new) sentence such as *she smiled at him*, involves: (i) identifying analogies in the input space: $[he \text{ loves } her : she \text{ loved } him :: he \text{ smiles at } her : she \text{ smiled at } him]$ would be found, (ii) solving the corresponding equations in the output space: $[\text{PRP VBZ PRP} : \text{PRP VBD PRP} :: \text{PRP VBZ IN PRP} : ?]$ would be solved, and (iii) selecting the solution. Here, *PRP VBD IN PRP* would be the only solution produced.

There are three important aspects to consider when deploying the above learning procedure. First, the search stage (step-1) has a time complexity which is prohibitive in most applications of interest (cubic in the size of \mathcal{I}). Second, the aggregation (step-3) of the possibly numerous spurious² solutions produced during step-2 is difficult. Last, it might happen that the overall approach does not

¹ We also use $[x : y :: z : t]$ as a predicate.

² A solver typically produces several analogical solutions, among which a few are valid.

produce any solution at all, simply because no input analogy is identified during step-1, or because the input analogies identified do not lead to analogies in the output space (failure of the inductive bias).

3 ISSUES WITH ANALOGICAL LEARNING

From now on, we focus our discussion to the case of formal analogies over strings. Section 5 expands the discussion to (formal) analogies over other structures, such as trees.

3.1 Formal Analogy

We mentioned that several definitions of formal analogy have been proposed. Two of them seem to stand above the others, in the sense that they can account for a larger variety of relations than the others: the proposal of [7] and the one introduced in [8], then generalized in [9]. The latter, which encompasses the former, is defined in terms of *d*-factorization. A *d*-factorization of a string x over an alphabet Σ , noted f_x , is a sequence of *d* factors $f_x \equiv (f_x^1, \dots, f_x^d)$, where $f_x^i \in \Sigma^*$ for all i , and such that $f_x^1 \odot f_x^2 \odot \dots \odot f_x^d \equiv x$; \odot denotes the concatenation operator. In [9], the authors define an analogy as follows.

Definition: $\forall x, y, z$ and $t \in \Sigma^*$, $[x : y :: z : t]$ iff there exists a 4-uple of d -factorizations (f_x, f_y, f_z, f_t) of x, y, z and t respectively, such that $\forall i \in [1, d]$, $(f_y^i, f_z^i) \in \{(f_x^i, f_t^i), (f_t^i, f_x^i)\}$. The smallest d for which this definition holds is called the *degree* of the analogy.

For instance, [*This guy drinks too much* : *This boat sinks* :: *These guys drank too much* : *These boats sank*] is true, because of the following 4-uple of 6-factorizations, whose factors are aligned column-wise for clarity, and where spaces (underlined) are treated as regular characters:

$$\begin{array}{llllllll} f_x & \equiv & (& This, & _guy, & \epsilon, & _dr, & inks, & _too_much \\ f_y & \equiv & (& This, & _boat_\!, & \epsilon, & s, & inks, & \epsilon \\ f_z & \equiv & (& These, & _guy, & s, & _dr, & ank, & _too_much \\ f_t & \equiv & (& These, & _boat_\!, & s, & s, & ank, & \epsilon \end{array})$$

There is no 4-uple of d -factorizations, with d smaller than 6. Therefore, the degree of this analogy is 6. Note that there are many 4-uple of 6-factorizations that verify the definition, as well as many 4-uple of d -factorizations for d greater than 6.

Although the choice of the definition to work with has some practical impact (the more general the definition, the more complex the machinery to recognize an analogy), we will use the definition given above. Still, the discussion in this paper generally applies for all sensible definitions we know.

3.2 Searching in the Input Space

Identifying analogies in the input space (step-1) has a cubic complexity with respect to the size of \mathcal{I} . Clearly, a brute-force approach would be manageable for toy problems only. This explains why several authors have worked out alternative strategies, as briefly discussed in this section. We refer the reader to [31] for a comparison of those strategies.

A quadratic search procedure The search for input analogies can be transformed into a quadratic number of equation solving [20] thanks to the symmetry property of analogical relations ($[x : y :: z : t] \Leftrightarrow [y : x :: t : z]$). Unfortunately, this solution barely scales to sets of a few thousands of representatives and is not practical: performing analogies on words for realistic NLP applications typically requires to work with vocabularies in the order of 10^5 words. A possible work around is to use sampling techniques.

More precisely, when performing inference for t , we solve analogical equations $[y : x :: i(t) : ?]$ only for some pairs $\langle x, y \rangle$ belonging to the neighborhood of $i(t)$. Solutions belonging to the input space are the z -forms we are looking for. This strategy reduces the search procedure to the resolution of a number of analogical equations which grows quadratically with the size of the neighbor set \mathcal{N} :

$$\mathcal{E}_{\mathcal{I}}(t) = \{ \langle x, y, z \rangle \mid \langle x, y \rangle \in \mathcal{N}(i(t)) \times \mathcal{N}(i(t)), \\ [y : x :: i(t) : z] \}$$

For instance, in [22] the authors deal with an input space in the order of tens of thousand forms by sampling x and y among the closest neighbors of t , where neighborhood are defined in terms of the standard edit-distance.

Exhaustive tree-count search In [31], the authors developed an alternative approach for scaling up the search procedure. The main idea is to exploit a property of formal analogies [7]:

$$[x : y :: z : t] \Rightarrow |x|_c + |t|_c = |y|_c + |z|_c \quad \forall c \in \Sigma \quad (1)$$

where Σ is the alphabet on which the forms are built, and $|x|_c$ stands for the number of occurrences of character c in x . In the sequel, we denote $\mathcal{C}(\langle x, t \rangle) = \{ \langle y, z \rangle \in \mathcal{I}^2 \mid \forall c \in \Sigma, |x|_c + |t|_c = |y|_c + |z|_c \}$ the set of pairs satisfying the count property with respect to $\langle x, t \rangle$.

Their strategy consists in first selecting an x -form in the input space. This enforces a set of necessary constraints on the counts of characters that any two forms y and z must satisfy for $[x : y :: z : t]$ to hold. By considering all forms x in turn,³ one can collect a set of candidate triplets for t . A verification of those triplets that actually define an analogy with t must then be carried out.

³ Anagram forms do not have to be considered separately.

Formally, they build:

$$\begin{aligned}\mathcal{E}_{\mathcal{I}}(t) = \{ & \langle x, y, z \rangle \mid x \in \mathcal{I}, \\ & \langle y, z \rangle \in \mathcal{C}(\langle x, i(t) \rangle), \\ & [x : y :: z : i(t)] \} \end{aligned}$$

This strategy will only work if (i) the number of quadruplets to check is much smaller than the number of triplets we can form in the input space (which happens to be the case in practice), and if (ii) we can efficiently identify the pairs $\langle y, z \rangle$ that satisfy a set of constraints on character counts. To this end, the authors proposed to organize the input space thanks to a data structure they call a *tree-count* (hence the name of the search procedure), which is easy to build and supports efficient runtime retrieval.⁴

Sampled tree-count search The tree-count search strategy enables to *exhaustively* solve step 1 for reasonably large input spaces (tens of thousands of forms). However, computing analogies in very large input spaces (hundreds of thousand of forms) remains computationally demanding, as the retrieval algorithm must be carried out $o(\mathcal{I})$ times. In this case, in [31], the authors proposed to sample the x -forms:

$$\begin{aligned}\mathcal{E}_{\mathcal{I}}(t) = \{ & \langle x, y, z \rangle \mid x \in \mathcal{N}(i(t)), \\ & \langle y, z \rangle \in \mathcal{C}(\langle x, i(t) \rangle), \\ & [x : y :: i(t) : z] \} \end{aligned}$$

The sampling strategy selects x -forms that share with t some sequences of symbols. To this end, input forms are represented in a κ -dimensional vector space, whose dimensions are frequent symbol n -grams, where $n \in [\text{min}; \text{max}]$. A form is thus encoded as a binary vector of dimension κ , in which the i th component indicates whether the form contains an occurrence of the i th n -gram. At runtime, sampling selects the η forms that are the closest to t , according to some distance measure (e.g. the cosine).⁵

Checking for analogies All the aforementioned search strategies require to verify whether each quadruplet in the candidate lists actually defines an analogical relation. Stroppa [32] proposed a dynamic programming algorithm for checking $[x : y :: z : t]$ when the definition in [9] is used. The complexity of this algorithm is in $o(|x| \times |y| \times |z| \times |t|)$. Since a large number of calls to the analogy checking algorithm must be performed during step 1 of analogical learning, the following property may come at help [31]:

$$\begin{aligned}[x : y :: z : t] \Rightarrow & \\ (x[1] \in \{y[1], z[1]\}) \vee (t[1] \in \{y[1], z[1]\}) & \\ (x[\$] \in \{y[\$], z[\$]\}) \vee (t[\$] \in \{y[\$], z[\$]\}) & \end{aligned} \tag{2}$$

where $s[\$]$ indicates the last symbol of s , and $s[1]$ the first. A simple trick consists in checking the proportionality condition only for quadruplets that pass this test.

⁴ Possibly involving filtering.

⁵ Typical values are $\text{min}=\text{max}=3$, $\kappa=20\,000$, and $\eta=5\,000$.

Open issues One can already go a long way with the sampled tree-count approach described above. Still, it is unclear which sampling strategy should be considered for a given application. The vector space model proposed in [31] seems to work well in practice, but more experiments are needed to confirm this.

More fundamentally, none of the search procedures proposed so far takes into account the fact that many analogies might be redundant. For instance, to relate the masculine French noun *directeur* to its feminine form *directrice*, it is enough to consider [*recteur* : *rectrice* :: *directeur* : *directrice*]. Other analogies (i.e. [*fondateur* : *fondatrice* :: *directeur* : *directrice*]) would simply confirm this relation. In [32], Stroppa formalizes this redundancy by the concept of *analogical support set*. Formally, A is an analogical support set of E iff:

$$\{[x : y :: z :?] : \langle x, y, z \rangle \in A^3\} \supseteq E$$

This raises the question of whether it would be possible to identify a minimal subset of the training set, such that analogical learning would perform equally well in this subset. Determining such a subset would reduce computation time drastically. Also, it would be invaluable for modelling how forms in an input system are related to forms in an output one. We are not aware of studies addressing this issue.

3.3 Solving Equations

Algorithms for solving analogical equations have been proposed for both definitions of interest mentioned above. For the definition of [9], it can be shown [8] that the set of solutions to an analogical equation is a rational language:

Theorem 1 *t* is a solution to $[x : y :: z :?]$ iff *t* belongs to $\{y \circ z\} \setminus x$.

The *shuffle* of two strings w and v , noted $w \circ v$, is the rational language containing all strings obtained by selecting (without replacement) sequences of characters in a left-to-right manner alternatively in w and v . For instance, spondyondontilalgiatis and ondspondonyalaltitisgia are two strings in the set spondylalgia \circ ondontitis). The *complementary set* of w with respect to v , noted $w \setminus v$, is the set of strings formed by removing from w , in a left-to-right manner, the symbols in v . For instance, *spondylitis* and *spydoniltis* are belonging to *spondyondontilalgiatis* \setminus *ondontalgia*. This operation can be straightforwardly extended to complement a rational set. The pseudo-code of our implementation of these two operations is provided in Algorithm 1.

Since these two operations preserve rationality, it is possible to build a finite-state machine for encoding those solutions. In practice however, the automaton is highly non deterministic, and in the worst case, enumerating the solutions can be exponential in the length of the sequences involved in the equation. The solution proposed in [24] consists in sampling this automaton without building it. The more we sample this automaton the more solutions we produce. In our implementation, we call *sampling rate* (ρ) the number of samples in $y \circ z$ that are considered. This is formalized in Algorithm 2.

```

function shuffle( $y, z$ )
Require:  $y$ , and  $z$  two forms
Ensure: a random word in  $y \circ z$ 
  if  $y = \epsilon$  then
    return  $z$ 
  else
     $n \leftarrow \text{rand}(1, |y|)$ 
    return  $y[1:n] . \text{shuffle}(z, y[n+1:])$ 

function complementary( $m, x$ )
Require:  $m$  and  $x$ , two forms
Ensure: the set  $m \setminus x$ 
  return complement( $m, x, \epsilon$ )

function complement( $m, x, r$ )
  if ( $m = \epsilon$ ) then
    if ( $x = \epsilon$ ) then
      return  $\{r\}$ 
    else
       $s_1 \leftarrow \text{complement}(m[2:], x, r.m[1])$ 
      if  $m[1] = x[1]$  then
         $s_2 \leftarrow \text{complement}(m[2:], x[2:], r)$ 
      return  $s_1 \cup s_2$ 
  return  $\phi$ 

```

Algorithm 1: Simulation of the two rational operations required by the solver. $x[a : b]$ denotes the sequence of symbols x starting from index a to index b inclusive. $x[a :]$ denotes the suffix of x starting at index a . $\text{rand}(a, b)$ returns a random integer between a and b (included).

```

function solver( $\langle x, y, z \rangle, \rho$ )
Require:  $\langle x, y, z \rangle$ , a triplet,  $\rho$  the sampling rate
Ensure: a set of solutions to  $[x : y :: z : ?]$ 
   $sol \leftarrow \phi$ 
  for  $i \leftarrow 1$  to  $\rho$  do
     $\langle a, b \rangle \leftarrow \text{odd}(\text{rand}(0, 1)) ? \langle z, y \rangle : \langle y, z \rangle$ 
     $m \leftarrow \text{shuffle}(a, b)$ 
     $c \leftarrow \text{complementary}(m, x)$ 
     $sol \leftarrow sol \cup c$ 
  return  $sol$ 

```

Algorithm 2: A Stroppa&Yvon flavored solver. $\text{rand}(a, b)$ returns a random integer between a and b (included). The ternary operator $? :$ is to be understood as in the C language.

It is important to note that typically, a solver produces several solutions to an equation, most of them spurious, which means that even though they obey the definition of formal analogy, they are not linguistically valid. To illustrate this, Figure 1 reports solutions produced to the equation $[even : usual :: unevenly : ?]$ by our implementation of Algorithm 2. As can be observed, most solutions are not valid forms in English: indeed, this definition recognizes no less than 72 different legitimate solutions, which we were able to produce with enough sampling ($\rho \geq 2000$) in less than a few tenth of a millisecond.⁶

The problem of multiple solutions to an equation is exacerbated when we deal with longer forms. In such cases, the number of spurious solutions can become quite large. As a simple illustration of this, consider the equation $e =$

⁶ The time measurements reported in this study have been made on an ordinary desktop computer, and are provided for illustration purposes only.

Fig. 1. 3-most frequent solutions to [even : usual :: unevenly : ?] along with their frequency, as produced by our solver, as a function of the sampling rate ρ . nb stands for the total number of solutions produced.

| ρ | nb | solutions |
|--------|------|---|
| 20 | 12 | usuaually (3) unusually (2) usunually (2) |
| 100 | 34 | unusually (6) usuaually (6) uunsually (4) |
| 1000 | 67 | unusually (57) uunusually (23) usuunally (19) |
| 2000 | 72 | unusually (130) uunusually (77) usunually (43) |

[this guy drinks too much : this boat sinks :: those guys drink too much : ?] where forms are considered as strings of characters (the space character does not have any special meaning here). Figure 2 reports the number of solutions produced as a function of the sampling rate. For small values of ρ , the solution might be missed by the solver (i.e. $\rho \leq 20$). For larger sampling rates, the expected solution typically appears (with frequent exceptions) among the most frequently generated ones. Note that the number of solutions generated also increases drastically. Clearly, enumerating all the solutions is not a good idea (too many solutions, too time consuming).

Fig. 2. 3-most frequent solutions produced by our solver at different sampling rates for the equation e . r indicates the position of the expected solution in the list if present (ϕ otherwise). nb indicates the number of solutions produced, and t the time counted in seconds taken by the solver. For readability, spaces are represented with the symbol $_$.

| | | | |
|-----------------------------|------------|-------------------------|----------------|
| $\rho = 20$ | $nb = 8$ | $\rho = 100$ | $nb = 28$ |
| $t = 0.0003$ | $r = \phi$ | $t = 0.001$ | $r = 13$ |
| thos_boate_sinks (2) | | thoboatse__sinks (2) | |
| tho_boatse_sinks (2) | | tho_boatse_sinks (2) | |
| thboatse__sinks (2) | | those_sboat_sink (2) | |
| <hr/> | | | |
| $\rho = 1000$ | $nb = 28$ | $\rho = 10^6$ | $nb = 19\,796$ |
| $t = 0.009$ | $r = 2$ | $t = 3.82$ | $r = 10$ |
| those_boat_ssink (5) | | thoes_boat_sinks (2550) | |
| those_boats_sink (5) | | thoses_boat_sink (1037) | |
| thoes_tboa_sinks (5) | | those_boat_ssink (999) | |

The fact that a solver can (and typically does) produce spurious solutions means that we must devise a way to distinguish "good" solutions from spurious ones. We defer this issue to the next section. Yet, we want to stress that currently, our sampling of the automaton that recognizes the solutions to an equation is completely random. It would be much more efficient to learn to sample the automaton, such that more likely solutions are enumerated first. Several algorithms

might be applied for this task, among which the Expectation-Maximization algorithm for transducers described in [33].

3.4 Aggregating Solutions

Step-3 of analogical learning consists in aggregating all the solutions produced. We saw in the previous section that the number of solutions to an analogical equation can be quite large. Also, there might be quite a large number of analogical equations to solve during step-2, which simply increases the number of solutions gathered in $\mathcal{E}_o(t)$. In many works we know, this problem is not discussed; yet, our experiments indicate that this is an important issue. In [34], Lepage and Lardilleux filter out solutions containing sequences of symbols not seen in the output space of the training set. This typically leaves many solutions alive, including spurious ones. In [20], Lepage and Denoual propose to keep the most frequently generated solutions. The rationale being that forms that are generated by various analogical equations are more likely to be good ones. Also, Ando and Lepage [35] show that the closeness of objects in analogical relations is another interesting feature for ranking candidate solutions.

In [24], the authors investigate the use of a classifier trained in a supervised fashion to recognize correct solutions from bad ones. This approach improved the selection mechanism over several baselines (such as selecting the most frequently generated solution), but proved to be difficult to implement, in part because many examples have to be classified, which is time consuming, but also because most of the solutions in $\mathcal{E}_o(t)$ are spurious ones, leaving us with a very unbalanced task, which is challenging. Last but not least, the best classifiers trained were using features computed on the whole set $\mathcal{E}_o(t)$, such as the frequency with which a solution is proposed. This means that it cannot be used to filter the unlikely solutions generated for a test form t in the early stages.

Improving the classifier paradigm deserves further investigations. Notably, in [24], only a small number of features have been considered. Better feature engineering, as well as more systematic tests on different tasks must be carried out to better understand the limitations of the approach.

As discussed in [35], it is intuitively more suited to see the problem of separating correct from spurious solutions as a ranking problem. Ranking is an active research topic in machine learning. We refer the reader to the LETOR (LEarning TO Rank) website for an extensive list of resources on this subject.⁷ Ranking the solutions proposed by the two-first steps of analogical learning must be investigated as a replacement of the classification solution proposed in [24].

3.5 Dealing with Silence

In most experiments we conducted, we faced the problem that the learning mechanism might fail to produce any solution for a given test form. This can happen because no input analogy is found, or because the input analogies identified yield

⁷ <http://research.microsoft.com/en-us/um/beijing/projects/letor/>

output equations that have no solution. Depending on the nature of the input space and the training material available, this problem can be rather important. On a task of translating medical terms [24], the authors submitted the silent cases to another approach (in their case a statistical translation engine). Combining analogical learning with statistical machine translation has also been investigated in [36]. In [20], the authors proposed to split the form to treat in two parts and apply analogical learning to solve those two subforms. This raises a number of issues which do not seem to have received a lot of attention. Knowing where to split the input form in order to maximize the chance of being able to solve the two new sub-problems is one of those.

4 CASE STUDY

4.1 Settings

In order to illustrate some of the elements discussed in previous sections, we applied analogical learning to the task of transliterating English proper names into Chinese. The task we study is part of the NEWS evaluation campaign conducted in 2009 [37]. Transliteration is generally defined as the phonetic transcription of names across languages, and is often thought as a critical technology in many domains, such as machine translation and cross-language information retrieval or extraction [37].

Table 1. Examples of transliterations of English proper names into Chinese taken from the NEWS 2009 English-Chinese corpus. The segmentations (shown with + signs) are ours, and are intended to illustrate the matching between English and Chinese sequences of characters.

| | |
|-------------------------|-------------|
| <i>A+co+ca</i> | 阿+克+卡 |
| <i>A+f+ri+ca+nu+s</i> | 阿+弗+里+卡+纳+斯 |
| <i>A+lessan+d+ro+ni</i> | 亚+历山+德+罗+尼 |
| <i>Bo+lan+d</i> | 伯+兰 |
| <i>Bo+ree+1</i> | 伯+雷+尔 |
| <i>B+ra+d+fo+rd</i> | 布+拉+德+福+德 |
| <i>C+ro+sse+tt</i> | 克+罗+西+特 |
| <i>Na+v+ra+ti+lo+va</i> | 纳+夫+拉+蒂+洛+娃 |
| <i>Ne+me+ro+v</i> | 内+梅+罗+夫 |

Examples of transliterations from English proper names into Chinese are reported in Table 1. The segmentations (represented by + signs) are ours, and were produced by inspection of the English-into-Chinese transcription table available on Wikipedia.⁸ As can be observed, the transcription is written with mono-syllabic characters which may not correspond exactly to syllables in English. A

⁸ http://en.wikipedia.org/wiki/Template:Transcription_into_Chinese.

We could not segment *lessan* with this table.

Chinese character may correspond to different English ones, and vice versa. For instance, the character *A* is transcribed into 阿 in *Acoca*, but not in *Alessendroni* (where the sound "ya" is assumed). Note also that special rules are used to encode initial characters, therefore, it is not advisory to convert English names into lower case, as was done in [38]. Also, a transcription sometimes reflects the meaning as well as the sound of the transcribed word. For instance, the common ending -va in Slavic female family names is usually transcribed as 娃 (girl), as in *Navratilova*.

The organizers of the NEWS campaign kindly provided us with the data that was distributed to the participants of the English-into-Chinese transliteration task. Its main characteristics are reported in Table 2. The distribution of Chinese characters is typically Zipfian, and 116 out of the 370 different characters seen in the training set appear less than 10 times (39 characters appear only once).

Table 2. Main characteristics of the English-into-Chinese data provided in NEWS 2009.

| | <i>train</i> | <i>dev</i> | <i>train + dev</i> | <i>test</i> |
|----------------|--------------|------------|--------------------|-------------|
| examples | 31 961 | 2 896 | 34 857 | 2 896 |
| EN symbols | 52 | 51 | 52 | 51 |
| CH symbols | 370 | 275 | 371 | 283 |
| avr. EN-length | 6.8 | 6.8 | 6.8 | 6.9 |
| min EN-length | 2 | 3 | 2 | 3 |
| max EN-length | 15 | 15 | 15 | 13 |
| avr. CH-length | 9.5 | 9.5 | 9.5 | 9.6 |
| min CH-length | 3 | 3 | 3 | 3 |
| max CH-length | 21 | 21 | 21 | 21 |

In order to transliterate the English proper names of the test set, we gathered a training set $\mathcal{L}_1 = \textit{train} + \textit{dev}$ by concatenating the training set and the development set that were released, that is, 34 857 pairs of English and Chinese proper names. Including the development set in the training material is fine, since there is no training involved when generating the set of solutions. In parallel to this, we also generated solutions for the development set (*dev*), using the training material only ($\mathcal{L}_2 = \textit{train}$); the solutions produced were used for training a classifier to recognize good from spurious solutions. This classifier was then applied to the solutions produced for the test set (*test*) thanks to \mathcal{L}_1 .

We mainly ran two configurations of our *generator*. The first one, named FULL-TC, corresponds to the exhaustive tree-count setting described in Section 3.2. The second one, named SAMP-TC, corresponds to the sampled version described in Section 3.2, where the $\eta = 1000$ closest input forms to each English test form were considered, based on a vector space representing the $\kappa = 5000$ most frequent 3-grams of characters observed in \mathcal{I} , and the cosine distance. In both

cases, the solver was run with a sampling rate of $\rho = 400$. We decided to keep up to the 100-most frequently generated solutions for a given test form. A solution is typically generated several times per equation (the frequency of which depends of ρ), and by several equations.

Regarding the classifier, we followed [24] and trained a voted-perceptron [39]. We computed a total of 33 (partly redundant) features including the frequency of a solution, its rank in the list, average degrees of the input and output analogies leading to a solution, likelihoods of several n-gram language models trained at the (Chinese) character level, etc. We trained the classifier over 5 000 epochs. A greedy search over the feature set revealed that half of these features are enough for optimal performance.

Table 3. Details of the two configurations tested.

| | FULL-TC | | | SAMP-TC | | |
|----------------------|---------|-----------|---------|------------|------|--------|
| | avr. | min. | max. | avr. | min. | max. |
| candidates analogies | 42 122 | 1 | 544 849 | 1 374 | 0 | 14 657 |
| candidates to check | 30 629 | 0 | 381 762 | 1 001 | 0 | 0 595 |
| avr. input analogies | 4 097 | 0 | 47 176 | 179 | 0 | 1 223 |
| output equations | 512 | 0 | 47 176 | 28 | 0 | 239 |
| solutions | 278 | 0 | 1918 | 40 | 0 | 402 |
| w/o input analogy | | 37 (1.3%) | | 114 (3.9%) | | |
| w/o solution | | 69 (2.4%) | | 223 (7.7%) | | |
| locate time | 4.2s | 0.7s | 11.6s | 0.02s | 0 | 0.7s |
| check time | 3.6s | 0 | 25.5s | 0.03s | 0 | 0.3s |
| solving time | 0.7s | 0 | 7.3s | 0.01s | 0 | 0.1s |
| total time | 8.5s | 0.9s | 38.3s | 0.05s | 0 | 0.7s |

4.2 Monitoring Analogical Inference

In the following, we describe in details the FULL-TC configuration, while Table 3 reports the figures of interest for both configurations we tested. For the exhaustive configuration, most of the time is spent during the search for input analogies. Roughly 8 seconds is spent on average per input form in order to identify an average of 4 097 analogies (and a maximum of 47 176 ones). For some forms, the time for identifying input analogies can be as long as 37 seconds. This suggests a timeout strategy when too many candidate analogies are observed. Also, it is likely not necessary to consider all input analogies, which raises the issue of ranking analogies to be treated first. For 37 of the test forms, we could not identify any input analogy, leading to no response in those cases. Solving all output equations led to an average of 278 solutions per test form (minimum

0, maximum 1918). Note that many equations solved led to no solution, which explains why on average, the number of solutions is lower than the number of equations solved. The average time for solving the equations per form was 0.7 seconds, with a worst case of 7.3 seconds (because many equations needed to be solved). It is interesting to note the discrepancy between the number of input analogies identified and the number of output equations effectively solved, which is much lower. This indicates either that the input analogies were in large part fortuitous, or that the inference bias (one analogy in the input space corresponds to an analogy in the output space) does not apply well for this task. In the end, our inference procedure remained silent for 106 input forms (3.7%).

As expected, the SAMP-TC configuration runs much faster, identifying far less input analogies (179 on average) and leaving 337 input forms (11.6%) without answer. This shows that while effective at reducing computation time, the tree-count search strategy described in Section 3.2 is perfectible.

4.3 Evaluation

In this section, we analyze the different variants of the analogical devices we developed. We start by a detailed analysis of the core variants that were tested, followed by a broader evaluation conducted thanks to the evaluation scripts available on the NEWS 2009 website. In both cases, let us consider a transliteration experiment as a set of N (here, $N = 2896$) triplets, $(e_i, r_i, a_{1,\dots,n_i}^i \equiv \{c_1^i, \dots, c_{n_i}^i\})_{i \in [1,N]}$, where e_i is the i th test form, r_i its reference transliteration,⁹ and a_{1,\dots,n_i}^i the ranked list of the n_i (here, $0 \leq n_i \leq 100$) solutions proposed by analogical learning, where solutions are ranked in decreasing order of likeliness.

Detailed Evaluation Let n_k be the number of test forms for which the reference solution is seen in the k -first solutions proposed: $n_k \equiv \sum_{i=1}^N \delta(\{r_i\} \cap a_{1,\dots,k}^i \neq \emptyset)$, where $\delta(x)$ is the Kronecker function the value of which is 1 if x is true, and 0 otherwise. We define $ratio_k$ as the ratio of test forms with a sanctioned solution ranked in the k -first solutions, to the number of test forms with a sanctioned solution. We define $prec_k$ as the percentage of test forms with at least one solution that contain a sanctioned solution in the k -first solutions proposed, and rec_k the percentage of test forms with a sanctioned transliteration proposed in the k -first positions. Formally:

$$\begin{aligned} ratio_k &= n_k / \sum_{i=1}^N \delta(\{r_i\} \cap a_{1,\dots,n_i}^i \neq \emptyset) \\ prec_k &= n_k / \sum_{i=1}^N \delta(n_i > 0) \\ rec_k &= n_k / N \end{aligned}$$

Intuitively, $prec_k$ a measure of precision, and rec_k a measure of recall, while $ratio$ provides the distribution of the rank of the correct solutions identified.

⁹ For the English-into-Chinese transliteration task we consider, there is only one reference for each test form.

Those figures (expressed as percentage) are reported in Table 4 for the two configurations of the generator we tested: FULL-TC and SAMP-TC. The first line of the first column indicates for instance that 45.1% of the test forms with at least one solution have the sanctioned transliteration produced in the first position. This represents 52.9% of the test forms with a sanctioned solution, and 43.4% of the total test forms.

Table 4. Quality of the 100-top frequent solutions proposed by the FULL-TC and the SAMP-TC configurations. nb indicates the number of correct transliterations at a given rank k . Read the text for more.

| k | FULL-TC | | | SAMP-TC | | |
|----------|----------------------------|---------------------|--------------------|----------------------------|---------------------|--------------------|
| | $ratio_k$ nb (%2377) | $prec_k$ (%2790) | rec_k (%2896) | $ratio_k$ nb (%1693) | $prec_k$ (%2559) | rec_k (%2896) |
| 1 | 1258 | 52.9 | 45.1 | 43.4 | 792 | 46.8 |
| 2 | 1548 | 65.1 | 55.5 | 53.5 | 1046 | 61.8 |
| 3 | 1706 | 71.8 | 61.2 | 58.9 | 1207 | 71.3 |
| 4 | 1819 | 76.5 | 65.2 | 62.8 | 1328 | 78.4 |
| 5 | 1904 | 80.1 | 68.2 | 65.8 | 1420 | 83.9 |
| \vdots | \vdots | \vdots | \vdots | \vdots | \vdots | \vdots |
| 26 | 2304 | 96.9 | 82.6 | 79.6 | 1693 | 100.0 |
| \vdots | \vdots | \vdots | \vdots | \vdots | \vdots | \vdots |
| 86 | 2377 | 100.0 | 85.2 | 82.1 | 1693 | 100.0 |
| | | | | | | |
| | | | | | | |
| | | | | | | |

This table calls for several comments. We noted earlier that the inference procedure leaves some forms without solution (3.7% for FULL-TC and 11.6% for SAMP-TC). Furthermore, we observe that the recall is reaching a limit of 82.1% for FULL-TC and 58.5% for SAMP-TC. Therefore, there is a fair number of cases which are not handled appropriately by the inference mechanism. For the SAMP-TC configuration, this simply shows the shortcomings of a too aggressive sampling strategy. For the FULL-TC variant, and while part of this recall failure can be explained by the fact that we only consider the 100-most frequent solutions generated, it simply illustrates the silence issue discussed in Section 3.

Considering only the most frequently generated solution,¹⁰ recall drops to 43.4% and 27.4% respectively. This shows that being able to distinguish good from spurious solutions has the potential to improve the overall approach by more than 30 absolute points in both configurations.

The effect of applying the classifier described above is analyzed in Table 5 for the FULL-TC configuration (the most effective one). It is noticeable that the number

¹⁰ In the SAMP-TC configuration, 20 test forms receive several solutions with the same maximum frequency, among which the sanctioned transliteration. We do credit the system with a rank 1 solution in those cases, even if the correct transliteration is not listed first.

Table 5. Impact of the classifier applied in the aggregation step of the FULL-TC configuration.

| k | nb | FULL-TC + classifier | | |
|-----|-------------------------------|--------------------------------|--------------------------------|--------------------------------|
| | | $ratio_k$ (%1667) | $prec_k$ (%2790) | rec_k (%2896) |
| 1 | 1577 (\uparrow 319) | 94.6 (\uparrow 41.8) | 56.5 (\uparrow 11.4) | 54.5 (\uparrow 11.1) |
| 2 | 1652 (\uparrow 104) | 99.1 (\uparrow 34.0) | 59.2 (\uparrow 3.7) | 57.0 (\uparrow 3.5) |
| 3 | 1661 (\downarrow 45) | 99.6 (\uparrow 25.1) | 59.5 (\downarrow 1.7) | 57.4 (\downarrow 1.5) |
| 4 | 1662 (\downarrow 157) | 99.7 (\uparrow 23.2) | 59.6 (\downarrow 5.6) | 57.4 (\downarrow 5.4) |
| 5 | 1662 (\downarrow 242) | 99.7 (\uparrow 19.6) | 59.6 (\downarrow 8.6) | 57.4 (\downarrow 8.4) |
| : | : | : | : | : |
| 16 | 1667 (\downarrow 545) | 100.0 (\uparrow 6.9) | 59.7 (\downarrow 19.6) | 57.6 (\downarrow 18.8) |

of test forms with the sanctioned solution ranked first increases significantly, which is good: another 319 forms are ranked first, which translates into an increase of precision and recall at rank 1 of above 11 absolute points. Precision and recall at rank 2 also gain over 3 absolute points. This illustrates that classifying solutions is feasible and leads to better performance than simply picking the most frequently generated form. It must be noted however, that this gain comes at the cost of precision and recall at higher ranks: some correct solutions are actually being removed by our classifier.

NEWS 2009 Table 6 reports the results of several of the transliteration devices we devised, as measured by the official metrics of the NEWS 2009 evaluation campaign [37]. Since the MAP_{ref} metric reported always equals ACC in our experiments, we only report the three first metrics output by the evaluation script¹¹ we used: ACC which corresponds to rec_1 ,¹² F-score which gives a partial credit proportional to the longest subsequence between the reference transliteration and the first candidate one, and the Mean Reciprocal Rank (MRR), where $100/MRR$ roughly indicates the average rank of the correct solution over the session. See [37] for the details of their computation. Those figures are expressed as percentage.

Table 6 also calls for several comments. First, we tested different variants of the SAMP-TC strategy (lines 1 to 6). The dimension κ of the space into which the (input) forms are represented is of importance, and setting κ to 10 000 (line 2) improves the accuracy by almost 4 absolute points compared to setting it to 5 000 (line 1), as we did in the previous section. Increasing κ further does not help, but ultimately, this parameter should be adjusted for each task. Naturally, considering a larger number (η) of neighbors leads to better performance, as

¹¹ We downloaded the script `news_evaluation.py` at <http://translit.i2r.a-star.edu.sg/news2009/evaluation/>.

¹² To the exception of the way we broke ties in the unfrequent cases where several solutions are produced with the top frequency.

Table 6. Evaluation of different configurations with the metrics used at the NEWS 2009 evaluation campaign. For comparison, *first* and *last* indicate the first and last performing systems respectively, as reported in [37]. In the cascading configurations (lines 11 and 14), components are listed in order in which they are consulted; those marked with a * sign are using the VP_{all} classifier. See text for further explanations.

| Configurations | | | ACC | F-score | MRR | no sol. |
|----------------|----------------------------------|-------------------------------|------|---------|------|-------------|
| 1 | SAMP-TC | $\eta=1k, \kappa=5k$ | 26.7 | 56.7 | 34.7 | 337 (11.6%) |
| 2 | SAMP-TC | $\eta=1k, \kappa=10k$ | 30.6 | 60.2 | 39.1 | 268 (9.3%) |
| 3 | SAMP-TC | $\eta=2k, \kappa=10k$ | 33.7 | 63.1 | 42.5 | 202 (7.0%) |
| 4 | SAMP-TC | $\eta=5k, \kappa=10k$ | 36.7 | 66.3 | 46.1 | 148 (5.1%) |
| 5 | SAMP-TC | $\eta=10k, \kappa=10k$ | 39.4 | 68.2 | 49.0 | 120 (4.1%) |
| 6 | SAMP-TC | $\eta=15k, \kappa=10k$ | 41.0 | 69.2 | 50.5 | 110 (3.8%) |
| 7 | FULL-TC | | 43.4 | 70.5 | 52.4 | 106 (3.7%) |
| 8 | FULL-TC | $d \leq 4$ | 46.6 | 72.6 | 55.7 | 173 (6.0%) |
| 9 | FULL-TC | $d \leq 3$ | 49.7 | 74.0 | 57.6 | 265 (9.2%) |
| 10 | FULL-TC | $d \leq 2$ | 55.1 | 73.4 | 61.4 | 411 (14.2%) |
| 11 | cascading(10,9,8,7) | | 56.4 | 79.2 | 62.9 | 106 (3.7%) |
| 12 | FULL-TC | + VP_{all} | 53.3 | 77.0 | 54.5 | 106 (3.7%) |
| 13 | FULL-TC | + $\text{VP}_{\text{subset}}$ | 54.5 | 77.6 | 55.9 | 106 (3.7%) |
| 14 | cascading(10*,9*,8*,10,9,8,7*,7) | | 57.0 | 79.5 | 61.7 | 106 (3.7%) |
| 15 | last NEWS 2009 | | 19.9 | 60.6 | 22.9 | — |
| 16 | first NEWS 2009 | | 73.1 | 89.5 | 81.2 | — |

shown by lines 2 to 6. Still, considering all the input forms (line 7) yields the best performance.

Second, we tested the influence of the maximum degree accepted for an analogy (input or output), which we call *d*-limit in the sequel. Considering only degree 2 analogies (line 10) yields a much higher score than considering all analogies, and this, even though the number of unsolved test forms increases. The optimal *d*-limit should be adjusted for each task. For instance, in their experiments on unsupervised morphology acquisition, Lavallée and Langlais [14] found that a *d*-limit of 5 was optimal for the German language, while a *d*-limit of 2 was enough for capturing the English morphology. Since identifying low degree analogies can be implemented more efficiently, and seems to produce correct solutions here, this suggests a cascading strategy where degree-2 analogies are searched first. Then, degree-3 analogies are searched for the only test forms for which no solution has been provided yet, and so on, until the *d*-limit (if any) is met. This is simulated in line 11 of Table 6. By doing so, we increase the recall of analogical inference, which benefits the overall performance of the device.

Third, we tested the influence of the classifier during the aggregation step. We trained and tested different flavors of the voted-perceptron algorithm, varying

the feature representation, as well as the number of epochs. We report two of these variants in lines 12 and 13 of Table 6. The first one has been trained on the totality of the features we computed (VP_{all}) and leads to a performance (ACC) of 53.3%, while the second one obtains 54.5% by using a subset of the features only ($\text{VP}_{\text{subset}}$).¹³ Note that for these two variants, when all the solutions of a test form are being pruned by the classifier, we keep the one with the largest classification score. The classifier trained on less features outperforms the other, showing that feature selection should be considered for optimal performance. In both cases, it is important to note that the gain in accuracy is huge compared to the FULL-TC configuration in line 7 (above 10 absolute points in accuracy), which shows the importance of the aggregation step.

Fourth, we tested a last configuration by cascading 8 variants: the ones described in lines 7 to 10, that is, the FULL-TC configuration with varying d -limits, and their variants using a classifier. While we could have train a classifier specific to each variant, we used the same classifier here (the one with all the features, VP_{all}) for all the aggregation steps. The order of the cascading reflected the intuition that classified variants should be consulted first (since their precision is expected to be higher), in the order of their d -limit (2, then 3, etc.). The results are reported in line 14 and show a slight improvement over the cascading configuration reported in line 11.

Last, we observe that our best system is not among the leading ones at NEWS 2009 (see line 16); 18 systems participated to the 2009 English-into-Chinese transliteration exercise. In fact, we would have ended up at the 12th rank according to accuracy (ACC), the official metric at NEWS.

Since our major goal was to monitor analogical learning, we did not put efforts into improving those figures, although there are straightforward things that could be done, such as always providing 10 candidate solutions, even if the classifier filtered in much less (except for accuracy, the other metrics are assuming a list of 10 candidates). Also, we did not attempt anything for dealing with silent test forms. In [36], the authors show that combining in a simple way analogical learning with statistical machine translation can improve upon the performance of individual systems. Last, it is shown in [36] that representing examples as sequences of syllables instead of characters (as we did here) leads to a significant improvement of analogical learning on a English-into-Indi transliteration task.

4.4 Transliteration Session

As an example, we illustrate in Figure 4 how the name Zemansky was transliterated into 泽曼斯基.¹⁴ 1 247 candidate analogies were identified thanks to the tree-count strategy, 405 of them were filtered thanks to the property defined by equation 2. Verifying the 842 remaining candidates took 0.15 seconds, and 53 (input) analogies were finally identified. After projection, this led to 7 (output)

¹³ 17 features were considered, based on a greedy search over the feature space, minimizing the training error rate over 100 epochs.

¹⁴ We took this example just because the number of analogies involved is small enough.

equations with at least one solution. Those equations, together with the input analogies that fired them are reported in the figure. Only two equations yielded the sanctioned transliteration; both involving analogies of degree 2. All together, this session took slightly less than four seconds, mostly spent for searching the (input) candidates.

The two pairs of analogies yielding the sanctioned transliteration are illustrated in Figure 3. We observe that the first pair of analogies passively captures the fact that the two substrings *Sch* and *Z* correspond respectively to 谢 and 泽. Similarly, the second pair of analogies somehow captures that the substrings *U* and *Ze* correspond respectively to 鸟 and 泽. However, there is no attempt to learn such a mapping.

| | |
|-----------------------|---------------------|
| Sch ell → 谢 尔 | U linski → 鸟 林 斯 基 |
| Z ell → 泽 尔 | U mansky → 鸟 曼 斯 基 |
| Sch emansky → 谢 曼 斯 基 | Ze linski → 泽 林 斯 基 |
| Z emansky → 泽 曼 斯 基 | Ze mansky → 泽 曼 斯 基 |

Fig. 3. Details of the two productive pairs of analogies involved in solving *Zemansky*. Factors that commute are aligned vertically for lisibility.

5 DISCUSSION

In this study, we presented in Section 1 a number of works on formal analogy dedicated to various NLP tasks. We described in Section 2 the analogical inference procedure and discussed in Section 3 a number of issues that we feel remain to be investigated for the approach to meet higher acceptance among the NLP community. In particular, we presented a number of approaches for tackling large input spaces, and discussed their limitations. We pointed out the noise of the inference procedure that must be taken care of. We also noted that the inference procedure suffers a recall problem for which very few solutions have been proposed. In Section 4, we presented a case study, transliteration of proper names, for which we reported encouraging results. More importantly, we used this case study for illustrating some of the issues behind the scene of analogical learning.

We believe that analogical inference over sequences has not delivered all its potential yet. We already discussed a number of issues that remain to be investigated. In particular, our experiments in Section 4 show that reducing the time complexity of the search for analogies without impacting performance is still challenging. Identifying low degree analogies first might be of help here, since they are easier to spot. We also observed that there is a large room for improvements in the aggregation step. The experiments we conducted with a classifier trained to recognize correct analogies in this study have just scratched

31 solutions:

| | | | |
|-----------------|------------------|----------------|-----------------|
| 泽 曼 斯 基 (f=77) | 泽 门 斯 基 (f=59) | 齐 斯 曼 基 (f=29) | 曼 泽 斯 基 (f=29) |
| 齐 斯 基 曼 (f=22) | 兹 梅 斯 卡 尼 (f=20) | 泽 门 基 斯 (f=11) | 泽 斯 门 基 (f=9) |
| 齐 曼 斯 基 (f=9) | 泽 曼 基 斯 (f=8) | 曼 斯 泽 基 (f=8) | 泽 斯 曼 基 (f=8) |
| 兹 梅 斯 尼 卡 (f=8) | 兹 梅 尼 斯 卡 (f=6) | 泽 基 门 斯 (f=5) | 兹 尼 梅 斯 卡 (f=4) |
| 齐 基 斯 曼 (f=4) | 泽 斯 基 门 (f=4) | 斯 泽 门 基 (f=3) | 斯 齐 基 曼 (f=3) |
| 斯 基 齐 曼 (f=2) | 曼 泽 基 斯 (f=2) | 泽 基 曼 斯 (f=2) | 泽 斯 基 曼 (f=2) |
| 斯 齐 曼 基 (f=2) | 尼 兹 梅 斯 卡 (f=2) | 曼 斯 基 泽 (f=2) | 斯 基 泽 门 (f=1) |
| 基 齐 斯 曼 (f=1) | 基 泽 门 斯 (f=1) | 斯 泽 基 门 (f=1) | |

[Stephens : Stephansky :: Zemens : Zemansky] (d=2)

| | |
|--------------------------------------|---------------|
| → [斯 蒂 芬 斯 : 斯 蒂 芬 斯 基 :: 泽 门 斯 : ?] | (9 solutions) |
| 斯 泽 基 门 (d=6) | 泽 基 门 斯 (d=3) |
| 基 泽 门 斯 (d=3) | 斯 泽 门 基 (d=4) |
| 泽 门 基 斯 (d=3) | 斯 基 泽 门 (d=4) |

[Rovine : Rovensky :: Zieman : Zemansky] (d=2)

| | |
|------------------------------|---------------|
| → [罗 文 : 罗 文 斯 基 :: 齐 曼 : ?] | (6 solutions) |
| 斯 基 齐 曼 (d=3) | 斯 齐 曼 基 (d=4) |
| 齐 曼 斯 基 (d=2) | 齐 斯 曼 基 (d=4) |

[Schell : Zell :: Schemansky : Zemansky] (d=2)

| | |
|------------------------------|---------------|
| → [谢 尔 : 泽 尔 :: 谢 曼 斯 基 : ?] | (4 solutions) |
| 曼 斯 泽 基 (d=4) | 泽 曼 斯 基 (d=2) |
| 曼 斯 基 泽 (d=4) | 曼 泽 斯 基 (d=4) |

[Beale : Zmeskal :: Beaney : Zemansky] (d=8)

| | |
|--------------------------------|-----------------|
| → [比 尔 : 兹 梅 斯 卡 尔 :: 比 尼 : ?] | (5 solutions) |
| 兹 梅 斯 尼 卡 (d=4) | 兹 尼 梅 斯 (d=4) |
| 尼 兹 梅 斯 卡 (d=4) | 兹 梅 斯 卡 尼 (d=2) |

[Clise : Zisman :: Cleskey : Zemansky] (d=6)

| | |
|----------------------------------|---------------|
| → [克 莱 斯 : 齐 斯 曼 :: 克 莱 斯 基 : ?] | (8 solutions) |
| 斯 齐 基 曼 (d=5) | 斯 齐 曼 基 (d=6) |
| 基 齐 斯 曼 (d=3) | 齐 基 斯 曼 (d=3) |
| 齐 斯 曼 基 (d=4) | 齐 曼 斯 基 (d=4) |

[Heale : Zmeskal :: Heaney : Zemansky] (d=8)

| | |
|--------------------------------|-----------------|
| → [希 尔 : 兹 梅 斯 卡 尔 :: 希 尼 : ?] | (5 solutions) |
| 兹 梅 斯 尼 卡 (d=4) | 兹 尼 梅 斯 卡 (d=4) |
| 尼 兹 梅 斯 卡 (d=4) | 兹 梅 斯 卡 尼 (d=4) |

[Ulinski : Umansky :: Zelinski : Zemansky] (d=2)

| | |
|--------------------------------------|---------------|
| → [乌 林 斯 基 : 乌 曼 斯 基 :: 泽 林 斯 基 : ?] | (9 solutions) |
| 泽 斯 曼 基 (d=5) | 曼 泽 斯 基 (d=5) |
| 泽 斯 基 曼 (d=5) | 泽 基 曼 斯 (d=5) |
| 曼 泽 基 斯 (d=7) | 泽 曼 基 斯 (d=5) |

Fig. 4. Log of the FULL-TC transliteration session for the English proper name Zemansky. 31 solutions have been identified; the one underlined (actually the most frequently generated) is the sanctioned one. ($f=\bullet$) indicates the frequency of a solution, while ($d=\bullet$) indicates the degree of an analogy.

the surface. We also discussed the problem of silence that the approach meet in typical NLP applications, including transliteration we visited in this work. Here again, low degree analogies could be put at use in order to guide the split of forms in parts that analogical inference can handle.

We are currently investigating three follow up issues. First, we are engineering a much larger feature set than the one we considered in this work. In particular, we are considering features extracted from often co-occurring pairs of (English/Chinese) sequences of symbols. This raises drastically the dimensionality of the representations given to the classifier, for hopefully better discriminative power. Second, we are investigating different machine learning algorithms. Preliminary results indicate that support vector machines [40] slightly outperform the voted perceptron algorithm we used in this study. Also, we are testing a number of rescoring approaches that seem to lead to higher overall performance. Further investigations are needed to be conclusive. Last but not least, we mentioned that our equation solver produces many (spurious) solutions. Filtering them out *a posteriori*, as we tried in this study, is fruitful even though the classifier we trained is far not perfect. But, one may legitimately argue that a better solution consists in fixing the solver in the first place. Currently, our solver ranks the solutions it produces in decreasing order of frequency. Some preliminary work we conducted on solving analogical equations among English proper names show that ranking the solutions with a language model is a much better solution (the correct solution is ranked higher in the list of solutions). This observation, which likely depends on the domain investigated as well as the language considered, at the very least suggests that finding ways of guiding the solver to produce less, but promising solutions is feasible.

Learning to solve equations is therefore a promising avenue that we plan to address. Currently, our solver randomly reads out paths of the non deterministic finite-state automaton which recognizes all possible solutions to an equation, according to the definition of formal analogy given in section 3.1. While learning to weight its arcs can be tempting (see for instance the algorithm described in [41]), we must not forget that each equation leads to a particular automaton, and that building the union of all possible such automata is not realistic, which seriously questions the applicability of a learning algorithm that assumes a given topology as input. One way to tackle the problem consists in discriminatively learn a function the features of which would depend on some characteristics of each automaton (for instance features based on n-gram sequences), and using this function to guide the sampling of paths in the automaton. Another way of addressing the issue of producing less but better solutions is to introduce some knowledge directly into the solver. In a way, this is what Lepage [42] did with his solver, which produces only a subset of the solution that our solver would generate. Of course, introducing some knowledge into the solver raises the issue of its generality (to domains and to languages). This clearly deserves investigations.

While we concentrated in this study on analogical learning over sequences of symbols, it should be stressed that this learning paradigm is not limited to this

scenario only. Stroppa [32] shows it can be generalized to structures of interest in NLP such as trees and lattices. In particular, he proposed a definition of formal analogy on trees, based on the notion of factorization of trees, very much in line with the definition of formal analogies between sequences of symbols defined in [9]. Based on this definition, the authors of [10] described an exact algorithm for solving an analogical equation on trees which complexity is at least exponential in the number of nodes of the largest tree in the equation. They also proposed two approximate solvers by constraining the type of analogies captured (notably, passive/active alternations are not anymore possible). Ben Hassena [12] proposed a solution for reasoning with trees based on tree alignment. The constraints imposed over the possible alignments are much more restrictive than the ones of [10], but the author reports a solver (a dynamic programing algorithm) which has a polynomial complexity. Unfortunately, none of the aforementioned approaches scale to even medium-sized corpora of trees. For instance in [12] the author applied analogical learning on a training set of less than 300 tree structures, a very small corpus by today’s standards. See also the work of Ando and Lepage [35] for a very similar setting.

Acknowledgments

This work has been partially founded by the Natural Sciences and Engineering Research Council of Canada (NSERC). We are grateful to the anonymous reviewers of the short paper submitted to the 2012 SAMAI workshop, as well as those that reviewed this article. We found one review in particular especially inspiring.

References

1. Turney, P., Littman, M.: Corpus-based learning of analogies and semantic relations. *Machine Learning* **60**(1-3) (2005) 251–278
2. Duc, N.T., Bollegala, D., Ishizuka, M.: Cross-language latent relational search: Mapping knowledge across languages. In: AAAI’11. (2011) 1237 – 1242
3. Marshall, J.B.: A self-watching model of analogy-making and perception. *Journal of Experimental and Theoretical Artificial Intelligence* **18**(3) (2002) 267–307
4. Hofstadter, D.R.: The copycat project: An experiment in nondeterminism and creative analogies. *Massachusetts Institute of Technology* **755** (1984)
5. Mitchell, M.: *Analogy-making as Perception*. MIT Press/Bradford Books, Cambridge, MA (1993)
6. Yvon, F.: Paradigmatic cascades: a linguistically sound model of pronunciation by analogy. In: In Proceedings of 35th ACL. (1997) 429–435
7. Lepage, Y., Shin-ichi, A.: Saussurian analogy: A theoretical account and its application. In: 7th COLING. (1996) 717–722
8. Yvon, F.: Finite-state machines solving analogies on words. Technical Report D008, École Nationale Supérieure des Télécommunications (2003)
9. Yvon, F., Stroppa, N., Delhay, A., Miclet, L.: Solving analogies on words. Technical Report D005, École Nationale Supérieure des Télécommunications, Paris, France (2004)

10. Stroppa, N., Yvon, F.: Formal Models of Analogical Proportions. Available on HAL Portal (2007)
11. Miclet, L., Bayroud, S., Delhay, A.: Analogical dissimilarity: Definitions, algorithms and two experiments in machine learning. *Journal of Artificial Intelligence Research* (2008) 793–824
12. Ben Hassena, A.: Apprentissage analogique par analogie de structures d’arbres. PhD thesis, Univ. de Rennes I, France (2011)
13. Bhargava, A., Kondrak, G.: How do you pronounce your name? improving g2p with transliterations. In: 49th ACL/HLT, Portland, USA (2011) 399–408
14. Lavallée, J.F., Langlais, P.: Moranapho: un système multilingue d’analyse morphologique basé sur l’analogie formelle. *TAL* **52**(2) (2011) 17–44
15. Kurimo, M., Virpioja, S., Turunen, V., Blackwood, G., Byrne, W.: Overview and results of morpho challenge 2009. In: 10th Workshop of the Cross-Language Evaluation Forum (CLEF 2009). Lecture Notes in Computer Science (2009) 578–597
16. Creutz, M., Lagus, K.: Inducing the morphological lexicon of a natural language from unan- notated text. In: International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning (AKRR’05), Espoo, Finland (2005) 106–113
17. Spiegler, S.: Emma: A novel evaluation metric for morphological analysis - experimental results in detail. Technical Report CSTR-10-004, University of Bristol, Bristol (2010)
18. Stroppa, N., Yvon, F.: An analogical learner for morphological analysis. In: 9th Conf. on Computational Natural Language Learning (CoNLL), Ann Arbor, USA (2005) 120–127
19. van den Bosch, A., Daelemans, W.: Data-oriented methods for grapheme-to-phoneme conversion. In: EACL, Utrecht, Netherlands (1993) 45–53
20. Lepage, Y., Denoual, E.: Purest ever example-based machine translation: Detailed presentation and assesment. *Mach. Translat* **19** (2005) 25–252
21. Lepage, Y., Lardilleux, A., Gosme, J.: The greyc translation memory for the iwslt 2009 evaluation campaign: one step beyond translation memory. In: 6th IWSLT, Tokyo, Japan (2009) 45–49
22. Langlais, P., Patry, A.: Translating Unknown Words by Analogical Learning. In: EMNLP, Prague, Czech Republic (2007) 877–886
23. Denoual, E.: Analogical translation of unknown words in a statistical machine translation framework. In: MT Summit XI, Copenhagen, Denmark (2007) 135–141
24. Langlais, P., Yvon, F., Zweigenbaum, P.: Improvements in Analogical Learning: Application to Translating multi-Terms of the Medical Domain. In: 12th EACL, Athens (2009) 487–495
25. Koehn, P.: Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In: 6th AMTA, Washington DC (2004)
26. Somers, H., Sandapat, S., Naskar, S.K.: A review of ebmt using proportional analogies. In: 3rd Workshop on Example-Based Machine Translation, Dublin, Ireland (2009) 53–60
27. Gosme, J., Lepage, Y.: Structure des trigrammes inconnus et lissage par analogie. In: 18e TALN, Montpellier, France (2011)
28. Claveau, V., L’Homme, M.C.: Structuring terminology by analogy-based machine learning. In: 7th International Conference on Terminology and Knowledge Engineering, Copenhagen, Denmark (2005)

29. Moreau, F., Claveau, V., Sébillot, P.: Automatic morphological query expansion using analogy-based machine learning. In: 29th European conference on IR research (ECIR'07), Berlin, Heidelberg (2007) 222–233
30. Correa, W.F., Prade, H., Richard, G.: When intelligence is just a matter of copying. In: ECAI'12. (2012) 276–281
31. Langlais, P., Yvon, F.: Scaling up analogical learning. Technical report, Paritech, INFRES, IC2, Paris, France (Oct. 2008)
32. Stroppa, N.: Définitions et caractérisations de modèles à base d'analogies pour l'apprentissage automatique des langues naturelles. PhD thesis, Telecom Paris, ENST, Paris, France (2005)
33. Eisner, J.: Parameter estimation for probabilistic finite-state transducers. In: 40th ACL, Philadelphia, USA (2002) 1–8
34. Lepage, Y., Lardilleux, A.: The greyc translation memory for the iwslt 2007 evaluation campaign. In: 4th IWSLT, Trento, Italy (2008) 49–54
35. Ando, S., Lepage, Y.: Linguistic structure analysis by analogy: Its efficiency. In: NLPRS, Phuket, Thailand (1997) 401–406
36. Dandapat, S., Morrissey, S., Naskar, S.K., Somers, H.: Mitigating problems in analogy-based ebmt with smt and vice versa: a case study with named entity transliteration. In: PACLIC, Sendai, Japan (2010)
37. Li, H., Kumaran, A., Pervouchine, V., Zhang, M.: Report of news 2009 machine transliteration shared task. In: Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration. NEWS '09 (2009) 1–18
38. Langlais, P.: Formal analogy for natural language processing: a review of issues to be addressed. In: 1st International Workshop Similarity and Analogy-based Methods in AI (ECAI Workshop), Montpellier, France (aug. 2012) 49–55
39. Freund, Y., Schapire, R.E.: Large margin classification using the perceptron algorithm. *Mach. Learn.* **37**(3) (1999) 277–296
40. Cortes, C., Vapnik, V.: Support-Vector Networks. *Mach. Learn.* **20**(3) (1995) 273–297
41. Eisner, J.: Parameter estimation for probabilistic finite-state transducers. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia (July 2002) 1–8
42. Lepage, Y.: Solving analogies on words: an algorithm. In: COLING-ACL, Montreal, Canada (1998) 728–733

Analogy as an Organizational Principle in the Construction of Large Knowledge-Bases

Tony Veale, Guofu Li

Web Science and Technology Division, KAIST, Yuseong, Korea.
tony.veale@gmail.com

Abstract. A capacity for analogy is an excellent acid test for the quality of a knowledge-base. A good knowledge-base should be balanced and coherent, so that its high-level generalities are systematically reflected in a variety of lower-level specializations. As such, we can expect a rich, well-structured knowledge-base to support a greater diversity of analogies than one that is imbalanced, disjoint or impoverished. We argue here that the converse is also true: when choosing from a large pool of candidate propositions, in which many propositions are invalid because they are extracted automatically from corpora or volunteered by untrained web-users, we should prefer those that are most likely to enhance the analogical productivity of the knowledge-base. We present a simple and efficient means of finding potential analogies within a large knowledge-base, using a corpus-constrained notion of pragmatic comparability rather than the typically less-constrained notion of semantic similarity. This allows us to empirically demonstrate, in the context of a substantial knowledge-base of simple generalizations automatically extracted from the Google n-grams, that knowledge acquisition proceeds at a significantly faster pace when candidate additions are prioritized according to their analogical potential.

1 Introduction

As a knowledge-base grows in scale and diversity, its capacity for insightful analogy should grow also ([19]). Analogy is a knowledge-hungry cognitive mechanism for learning and generalization, one that allows us to project our knowledge of one domain onto another ([9], [12]). Computational approaches to analogy work best when different domains are represented in largely isomorphic ways that facilitate systematic mappings, but this is a challenge for very large KBs constructed by diverse teams of engineers (e.g., [14]). Fortunately, the converse also seems to be true: if we design a KB to maximize its capacity for analogy, the resulting structure should be well-balanced, coherent and rich in isomorphisms. So given a large pool of candidate propositions to choose from, we should prefer those that increase the potential for analogy in the growing KB.

A good knowledge-base is like a city park or a fashionable nightspot: each is subject to *neighborhood effects*, insofar as their appeal is in direct proportion to the value of the members they can attract ([29]). An empty nightclub has

little appeal, but this grows as new high-profile members are added. Each new member provides what economists call a *positive externality* ([29]), since each addition makes a club or park livelier and more enjoyable for others. Likewise, each addition to a KB can have marginal benefits for the propositions that are already there, by increasing their connectedness and making it likelier that they can be used in longer and more complex inferential chains. But economists also speak of *negative externalities*, and invalid propositions can undermine the workings of a knowledge-base just as surely as drunken gatecrashers can detract from the appeal of a trendy nightspot. Nightclubs use doormen to discourage negative externalities while expediting the entry of those candidates queuing outside that are most aligned with the club’s image. We similarly want to expedite the addition of candidate propositions that provide the most positive fit to the current knowledge-base, as measured in terms of their analogical potential. In effect, analogy is our doorman to the KB.

Though analogy is cognitively important ([9],[12],[7],[24]), it is just one of many capacities that we expect from our knowledge-bases, and many KB applications demand no analogical competence at all. So in this chapter we also consider a practical corollary, and ask whether incrementally constructing a knowledge-base so as to maximize its capacity for analogy results in a faster and more streamlined acquisition process overall. We have strong *a priori* reasons to presume so: balance, coherence and wide coverage are desirable qualities in any knowledge-base, while the value of high-level generalizations lies in the number and variety of lower-level specializations that they can explain. Analogy can directly pinpoint those additions that will flesh out the knowledge-base in the most coherent and well-rounded fashion.

Our argument is presented with an empirical evaluation in the following sections. We first discuss related background work on analogy and knowledge representation in section 2, before section 3 recasts analogy in terms of a corpus-based notion of comparability. Section 4 describes how a simple but relatively large knowledge-base of generic propositions is extracted from the Google n-grams ([3]). A coarse net is trawled over these n-grams, so many more invalid propositions are retrieved than are actually valid or desirable. Section 5 presents a model of analogy-guided acquisition, in which a mixed bag of candidate propositions is sorted according to their analogical *fit* to the current KB. Section 6 presents an evaluation of this approach, and quantifies the efficiency gains that arise from the use of analogy to shake out the best candidates. Final remarks are offered in section 7.

2 Related Work and Ideas

Computational treatments of analogy have traditionally focused on two principal forms. The first form, typified by the work of Gentner ([9]) and Holyoak and Thagard ([12]) and formally evaluated by Veale and Keane ([24]), is the *system analogy* that builds rich networks of cross-domain mappings via the systematic alignment of two complex structures. These analogies can be quite deep, and are

most common in scientific reasoning, legal argumentation and education. The second form, typified by the work of Hofstadter ([11]) and Turney ([22], [23]), limits itself to the fixed frame A:B::C:D. These proportional analogies have, for many years, been a fixture of IQ and SAT-style aptitude tests ([25]). Despite a simplicity of form, these often highlight the subtle nuances that separate close conceptual neighbors, as in *mercenary:soldier::hack:writer*. Both forms have attracted very different computational treatments, yet each is mutually compatible, since complex system analogies can be seen as coherent global combinations of smaller, more local, proportional analogies.

System analogies are typically modeled using a structure-mapping process applied to graph representations of domain knowledge ([7]), in which mappings are constructed by finding the largest sub-graph isomorphisms between two logical representations. Finding optimal mappings is thus an NP-hard problem ([24]), so various heuristics and pragmatic assumptions are employed to generate good mappings in polynomial time. For Hofstadter and his fluid analogies group ([11]), it is these very trade-offs and heuristics that make analogy so interesting. They model analogy as a non-deterministic process subject to competing slippage pressures that can shape very different (but valid) solutions. Their work focuses on proportional analogies within closed micro-worlds, dealing e.g. with letter sequences or table-top place settings. Yet though insightful, it is not immediately clear how a micro-world approach can gain traction on analogies between arbitrary propositions extracted from open-domain texts.

Turney ([22], [23])) works with word-based proportional analogies drawn from SAT-style tests, in which one must find the best analogical match for a given pair of terms (e.g., *jury:verdict*) among a set in which the right answer (e.g., *courier:package*) is mingled with distractors (e.g., *judge:trial*). To avoid a fragile reliance on a knowledge-base of hand-coded representations, Turney employs a distributional model of relational meaning in which a vector space of distributional features is derived from corpus or web text, and smoothed via singular value decomposition (SVD). In his approach, called *Latent Relational Analysis* (LRA), the features are words and phrases that can link the two elements of a relational pair, such as "delivers" for *jury:verdict* and *courier:package*. LRA achieves impressive performance on real SAT analogies, comparable to that of the average human test-taker. Turney shows that the distributional approach is more effective than a pure knowledge-based approach; applying WordNet ([8]) to Turney's dataset of 374 SAT analogies, Veale ([25]) reports a lower precision of 38%-44% using WordNet alone, attaining a bare pass mark. Nonetheless, WordNet-based approaches are much less computationally demanding, hinging on a lightweight measure of semantic similarity that can be efficiently applied on a large scale to tens of thousands of potential analogies.

The *AnalogySpace* model ([19]) sees analogy as a natural outgrowth of a large knowledge-base, in this case the *ConceptNet* project of Liu and Singh ([15], [20]). Comprising facts and generalizations acquired from the template-structured contributions of web volunteers, ConceptNet expresses many relationships that accurately reflect a public, common-sense view on a given topic (from *vampires*

to *dentists*), but also many that are idiosyncratic or ill-formed. As in Turney’s LRA [22], AnalogySpace builds a representation with reduced-dimensionality in which analogies emerge from the mulching together of perspectives that have deep similarities despite their superficial dissimilarities. AnalogySpace does not concern itself either with complex system analogies or even with proportional analogies between individual propositions, but with the identification of concepts that share analogical similarity (e.g., things that evoke similar feelings and are pleasant or unpleasant in similar ways).

As in AnalogySpace, this current work assumes that analogy occurs within a large knowledge-base of diverse propositions. Our approach is also corpus-based, like LRA, but we do not use representations with reduced-dimensionality, nor is our main goal here the detection of analogies within an existing KB or dataset. Rather, we use analogy as a guide to how the knowledge-base should grow, and provide a wholly symbolic implementation of proportional analogy that efficiently hinges on the simple notion of pragmatic comparability, which we describe next.

3 Learning to Compare, Pragmatically

A taxonomic model of word meaning like WordNet can be used to provide a numeric similarity score for any two terms one cares to compare, no matter how odd the pairing. Budanitsky and Hirst [5] evaluate a menu of WordNet-based similarity functions, and whether one is comparing *prawns* and *protons* or *galaxies* and *footballs*, WordNet can be used to provide a sensible measure of their semantic similarity. Experiments have shown that WordNet-based similarity measures broadly reflect human intuitions ([16], [18], [27]), though such measures are best viewed as relative, to know e.g., that protons are more similar to electrons than to crustaceans.

Yet the biggest advantage to this approach is also its greatest weakness. The space of sensible comparisons is far smaller than the space of possible comparisons, and WordNet can be used to attach a non-zero similarity score to the most ill-judged of comparisons. This may not be a concern if we can trust the client application to only seek similarity scores for terms it has good reason to compare, as when interpretation of the analogy *runner:marathon::oarsman:regatta* requires a comparison of *runner* to *oarsman* and *marathon* to *regatta*. However, speculative matches of *runner:marathon* to hundreds or even thousands of potential analogues in the KB will inevitably result in silly comparisons that no human would ever make, even if they do yield good similarity scores. The situation is exacerbated by lexical ambiguity, since the word senses that yield the best scores may not be the intended or most natural senses for the analogy in question.

WordNet-based measures are semantic, objective, and context-free, uninfluenced by subjective and pragmatic considerations. Any measurement typically involves a small number of static category structures, such as *mammal* when comparing *cats* and *dogs*, or *vehicle* when comparing *cars* and *buses*. In contrast,

distributed corpus-based approaches implicitly capture the diverse contexts in which we experience two terms/ideas ([28]). For instance, *pirates*, *astronauts* and *cowboys* are semantically similar by virtue of being *human beings*, but are pragmatically similar for a variety of tacit cultural reasons, not least because they represent dashing heroic types that make for “cool” characters in movies and “cool” costumes on Halloween. The distributed approach is successful because we cannot hope to articulate all the reasons why two terms are pragmatically comparable, much less express them as static categories in WordNet.

The coordination “*astronauts* and *cowboys*” suggests that both terms are comparable because they occupy the same context- or task- specific ad-hoc category ([2]). Set-building linguistic constructs provide evidence of subtle pragmatic categories that cannot be lexicalized in WordNet. Parsing such constructs, like lists and coordinations, is the basis for *Google Sets*, a tool that allows Google to perform on-demand set completion ([21]). To obtain our own comparability judgments for generic-level concepts, we harvest all coordinations of bare plurals (e.g., “*cats* and *dogs*” and even “*atoms* and *galaxies*”) and of proper names (such as “*Paris* and *London*” or “*Zeus* and *Hera*”) from Google’s 1T database of web n-grams ([3]). For each pair of coordinated terms, we calculate a similarity score based on the relative depth of their senses and their common hypernym in the WordNet sense hierarchy ([18]), and populate the comparability matrix accordingly. The n-grams link 35,019 unique terms, but each is coordinated with relatively few other terms, so only a tiny fraction of the possible cells in the $35,019 \times 35,019$ comparability matrix will have non-zero similarity scores.

This matrix is symmetric, since we assume that the similarity score for X and Y is the same as the similarity score for Y and X. By finding the cell at the intersection of row X and column Y (or of row Y and column X), a system can quickly look up the pre-compiled similarity score of X to Y. One important reason to encode a similarity function as a pre-compiled matrix is that we wish to re-imagine similarity as a generative phenomenon. Conventional similarity metrics implement a convergent process: given a pair of concepts X and Y, the metric converges on a single, objective score. But such convergent metrics cannot readily be used to generate a divergent set of possible comparisons $\{Y_1 \dots Y_n\}$ for a given term X ([27]). Likewise, a convergent similarity metric can converge on a single score to represent the similarity underpinning a proportional analogy A:B::X:Y as a function of the similarity of A to X and of B to Y (and of the latent relationship R_{AB} to R_{XY}). However, a convergent metric cannot be used to generate a set of plausible pairs $\{X_1:Y_1 \dots X_n:Y_n\}$ when presented with only half of an analogy, A:B. By using corpus analysis to construct a comparability matrix, and a convergent similarity metric to populate this matrix, a system can generate plausible, divergent comparisons $\{Y_1 \dots Y_n\}$ for a given term X simply by reading off the row for X in the matrix. Such a system need not worry about generating dud comparisons, as every non-zero cell in the matrix corresponds to a corpus-attested (if implicit) comparison.

This sparse matrix is compact enough to store in memory, yet contains all of the most plausible comparisons a system is ever likely to consider. The matrix

can be used to suggest as well as interpret comparisons. So to suggest sensible analogical substitutions for a term/concept X, we simply read off the row for X in the matrix. Rows can also be intersected to suggest sensible answers for missing elements in proportional analogies, as in:

- a) priest : church :: ? : mosque (*A: imam*)
- b) church : spire :: mosque : ? (*A: minaret*)
- c) chef : recipe :: scientist : ? (*A: formula*)
- d) school : bus :: hospital : ? (*A: ambulance*)

In (a), the answer must be comparable to *priest* and coordinated with *mosque*, and is found by looking for the most similar terms to *priest* among the intersection of the rows for *mosque* and *priest*. So the answer to (a) lies at the intersection of the 3-grams “priests and imams” and “imams and mosques”. The matrix yields sensible answers to (b), (c) and (d) in the same way.

This approach is elaborated in section 5, to consider analogies of propositions with a specific relation and a minimum similarity threshold.

4 A Knowledge-Base of Commonplace Generalizations

A bare plural like “dogs” typically denotes the generic use of the category, rather than a specific group of instances ([6]). Thus, the 3-gram “dogs and cats” typically denotes a generic connection between the concepts CAT and DOG at the category level, which suggests that one might be a viable substitute for the other in an analogy. If we replace the coordinator “and” with an appropriate verb, we can determine the generic relationship that links both concepts ([17]). For instance, “dogs *chase* cats” expresses the generalization that, all things being equal and if given the chance, an instance of DOG will chase an instance of CAT. By their very nature, most generalizations are easily falsified, yet they convey defeasible commonsense insights into the workings of the world as seen through stereotypes.

To acquire a large number of generalizations, we can look for generic propositions that predicate over generic uses of concepts. Looking to the Google 3-grams again, we harvest all instances of the template “Xs <verb> Ys”. Matches include “birds lay eggs”, “scientists conduct experiments” and “chefs create dishes”. In all, we find 158,911 matches for “Xs <verb> Ys” in the Google 3-grams. But so simple a template will inevitably retrieve a great deal of noise: some of the retrieved matches will be syntactically ill-formed (e.g., the verb is part of a phrasal verb, or has a complex sub-categorization frame that does not fit into a 3-gram, as in “priests tell parishioners [that]”) while many more will express generalizations that are simply false (e.g., “mammals lay eggs”), obscene (this is web content, after all), or not worth adding to a knowledge-base. One cannot be totally objective when dealing with generalizations, so there is no official count for the number of valid generalizations to be mined from the Google n-grams (or any other open-domain source). What is added to a KB is determined by domain relevance, tolerance for imprecision and personal taste. Individual consideration

of all 158,911 matches suggests that only one in eight of these raw matches (or 21,258 of 158,911 to be precise) yields a generalization that is sound enough to merit a place in a knowledge-base. A full analysis of this KB of n-gram-derived generalizations is presented in section 6.

We can use a more sophisticated extraction process, or target longer n-grams for more complex generalizations. But for our current purposes, these matches, noise and all, make an ideal test-set. We view our task as the dynamic ranking of these 158,911 matches so that the 21,258 valid generalizations that are competing with the remaining 137,653 rejects for entry to the KB (i.e., 158,911 - 21,258) are prioritized and brought to the front of the knowledge-acquisition queue. Success with this noisy dataset will indicate the potential for analogy to expedite the results of more sophisticated mining algorithms with a lower tolerance for noise.

5 Analogy-Guided Knowledge Acquisition

The knowledge-acquisition queue contains all those propositions of yet-unproven value that are extracted from corpora or volunteered by web users. In an ongoing KB effort, this queue may be constantly growing, and like celebrities at a night-club, new arrivals may jump the queue if perceived as a better fit for the KB. Of course, when the knowledge-base is initially empty there is no analogical basis for preferring one queued proposition over another, as there are no propositions in the KB with which to form an analogy.

Creating a new knowledge-base requires a knowledge engineer to walk through the queue, accepting propositions that are valid and rejecting those that are not. This task is made increasingly more efficient by analogy: as new propositions are added, the system looks for analogies between any valid proposition in the KB and any unseen proposition remaining on the queue. As the KB grows, so too will the number and diversity of these potential analogies. The system sorts the queue dynamically, after each new addition to the KB, ranking candidates by the number of potential analogies that can link them to the growing KB. In a growing queue, ill-fitting propositions will always be pushed to the back.

We use simple (and flat) structure-mapping to perform analogical matching. For propositions of the form $\text{pred}(X, Y)$, structure-mapping is simple to apply, both to flat and to recursively-nested structures:

$\text{pred}^S(X, Y)$ is a match for $\text{pred}^T(A, B)$ iff $\text{pred}^S = \text{pred}^T$ so that
X is mapped to A and Y is mapped to B.

At the heart of any structure-mapping problem then is a series of one of more proportional analogies of the form A:B::X:Y, where the mapping of A to B and of X to Y is suggested by, and in a sense guaranteed by, the identically of the relation that connects A to B and the relation that connects X to Y. Predicate identicality is a standard meaning-preserving constraint in structure mapping (see [9], [7]), one that also reduces the search space of the mapping problem. Structure-mapping is typically applied to nested structures whose isomorphism strongly suggests the comparability of literal predicate arguments. For shallower

structures, as with our simple generalizations, we can also ensure the comparability of arguments by demanding corpus evidence of their substitutability. Generalizations such as *create(artist, artwork)* and *create(chef, dish)* can be mapped in an analogy because both use the predicate *create* with the same arity, while the 3-grams “*artists and chefs*” and “*artworks and dishes*” also suggest that *artists* are comparable to *chefs* while *dishes* are comparable to *artworks*.

Comparability ensures sensibility when interpreting analogies and efficiency when generating analogies. So to find a match for the proposition *pred(A, B)*, we consider only those KB propositions *pred(X, Y)* that have the same predicate *pred* such that A is comparable to X and B is comparable to Y. Since a system can read off the set of comparable terms for A and for B from the comparability matrix, it can divergently generate all sensible analogues of *pred(A, B)* within a given similarity threshold and look for them in the KB. For instance, at a similarity threshold of 80%, we generate the analogue *pred(X, Y)* only if A and X have a similarity of 80% or more in the comparability matrix, and if the similarity of B and Y is also at or above this threshold. At a threshold of 0%, we allow all analogies between propositions with the same predicate and arity.

Analogies are therefore proposed using a *generate-and-test* rather than a *find-and-match* approach, so the system can suggest meaningful analogies that the KB does not yet (but perhaps should) support. Clearly, fewer analogies are generated at higher similarity thresholds, while many more are generated as we lower the threshold. There is an inevitable recall-versus-precision trade-off here: as the similarity threshold is lowered, we open the door to more creative analogies with greater semantic distance, but we may also reduce the average quality of the larger pool of analogies that is proposed. In the next section we consider the effect of the similarity threshold on analogue retrieval, and its knock-on effect on the workings of analogy-guided knowledge-acquisition.

6 Empirical Evaluation

This chapter has put forth two broad claims that require empirical validation. The first concerns our corpus-based model of comparability, which assumes – for the sake of naturalness and efficiency – that the space of sensible comparisons is sparse. We verify that this sparseness does not impair coverage, to show that comparability provides a robust and compact basis for aligning like with like. The second concerns the efficacy of analogy-guided acquisition, which assumes that candidate additions to the KB should be sorted according to analogical fit. We show that a KB grows fastest when this is so, and slowest in the worst-case scenario when this fitness metric is perversely ignored.

6.1 The Reliability of Comparability

An analysis of coordination patterns in the Google 3-grams fills the comparability matrix with similarity scores for 35,019 unique terms. Any pair of terms is likely to yield a non-zero score using a WordNet-based measure, yet n-gram analysis

finds just 1,363,184 pairs, or just 1.36×10^6 pairs out of a possible $1,225 \times 10^6$ pairings. With a density of approx. 0.1% ($1.36/1,225$), the matrix is sparse enough to hold in memory, but one can ask whether it is too sparse to be a general model of comparability. So to see if it provides sufficient coverage for robust category-level reasoning, we use it to replicate the category-formation experiments of [1].

The authors of [1] select 214 words from 13 different categories in WordNet. Using query patterns in the style of Hearst ([10]) to retrieve informative text fragments for each word from the web, they then harvest a large body of features for each word. These features include attributes, such as TEMPERATURE for coffee, and attribute values, such as FAST for car. In all, they harvest a set of approx. 60,000 web features for the 214-word dataset, and use CLUTO ([13]) to automatically group the words into 13 clusters on the basis of their web-harvested features. The *purity* of a cluster is a measure of its homogeneity, and of the extent to which the members of the cluster all belong to the same category. When the average purity of a set of clusters is 1.0, this indicates that the clusters faithfully replicate the category boundaries inherent in the original data (which, in this case, are the boundaries of WordNet’s categories). These 13 CLUTO-built clusters have a purity of 0.85 relative to WordNet’s own categories, which represents an 85% replication of WordNet’s structure.

We replicate the experiment using the same 214 words and 13 WordNet categories. However, rather than harvesting web features for each word, we use the corresponding rows of our comparability matrix. Thus, the features for “chair” are the comparable terms for “chair”, that is, the set of X such that either “chairs and Xs” or “Xs and chairs” is a Google 3-gram. We ignore the similarity scores in the matrix, as these were produced using WordNet, but we do treat every word as a feature of itself, so e.g., CHAIR is a feature of “chair”. The sparse matrix yields a much smaller set of features – 8,300 in total for all 214 words. CLUTO builds 13 clusters from these features, and achieves a cluster purity of 0.934 relative to WordNet. The space of sensible comparisons is indeed a compact and precise means of representing the semantic potential of words.

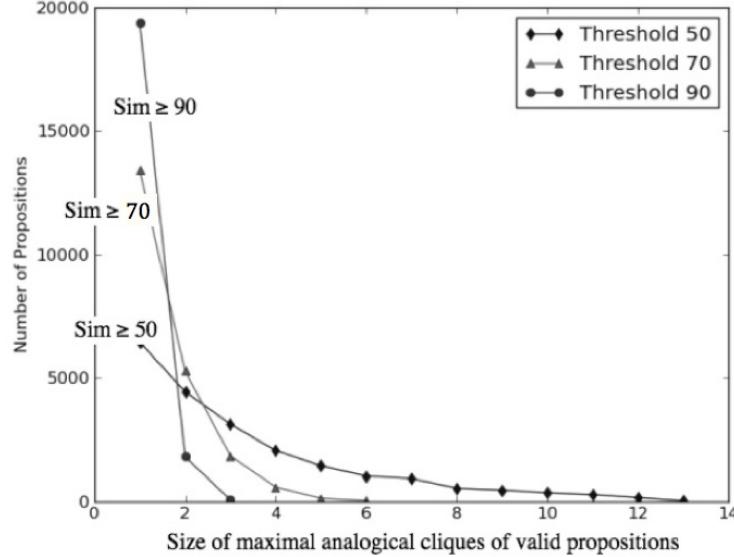
6.2 Analogical *Fit* as a Guide to Knowledge Acquisition

For evaluation purposes we annotate each of the 158,911 generic propositions from the 3-gram dataset as *valid* (if it is sensible to add it to a KB) or *invalid* (if it should be rejected from any KB). No attempt at global coherence is made: a proposition is hand-tagged as valid if it has the ring of commonsense truth, even if only as a stereotype. Different engineers may quibble about the wisdom of individual propositions, and might build slightly different KBs of their own. We consider the effect of different choices at the end of this section.

In all, we tag 21,258 propositions as valid, or about 1 in 8. Analogy makes neighbors of similar propositions, and a clique analysis (see [4]) reveals the neighborhood structure imposed by analogy on this dataset. An *analogical clique* is a set of propositions that are all mutually connected by analogy at a given similarity threshold. An analogical clique is maximal if no valid proposition can be

added to make it larger. Figure 1 shows the range of valid propositions in maximal analogical cliques of varying sizes. As the similarity threshold is lowered, the number of larger analogical cliques increases significantly.

Fig. 1. Distribution of maximal analogical cliques of propositions tagged as valid, using three similarity thresholds (50%, 70%, 90%) for analogical matching.

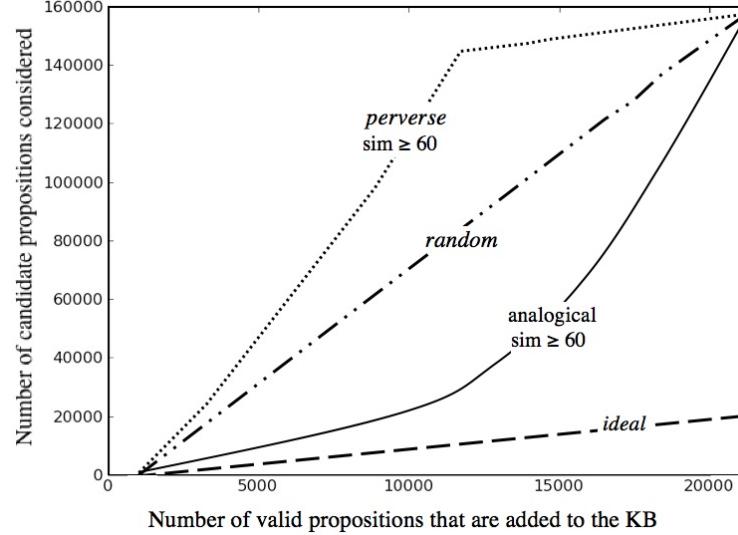


We use this hand-tagged set of valid propositions as the basis of an oracle for simulating different knowledge-acquisition scenarios. We start with a KB of 1,000 *valid* propositions, randomly chosen from those annotated as valid. Figure 2 shows the growth of this seed KB as the queue of the remaining 157,911 candidates is processed.

The *ideal* case shows the growth of the KB when every candidate served from the queue is valid: so we need process just 19,000 additional candidates with a hit-rate of 100% to grow the KB to a size of 20,000. The *random* case shows the baseline growth of the KB when the queue is unsorted. The *analogical* case shows growth when the queue is sorted in descending order of the number of possible analogies from each proposition on the queue to valid propositions in the KB. The *perverse* case aims to model the worst-case scenario, by sorting the queue in ascending order, so candidates with the least analogical potential are processed first. Note that any non-ideal curve will be anchored at (1000,0) and (21,258, 158,911) since, lacking perfect information, a system must consider all 158,911 candidate propositions to identify every valid proposition. Each curve varies in the middle part of its trajectory between these two anchor points, turning

away from the ideal line as valid propositions are encountered with diminishing frequency among the remaining candidates.

Fig. 2. A graph of the number of queued candidates that must be considered (y-axis) to grow the KB to a given number of valid propositions (x-axis). A similarity threshold of 60% is used for making analogies.

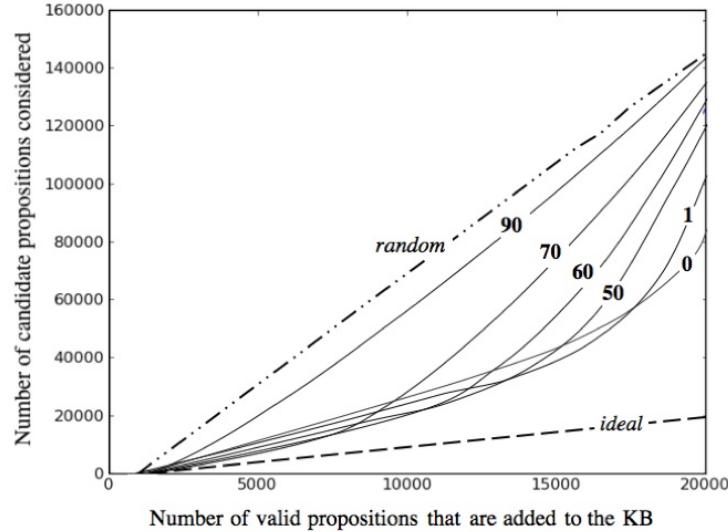


A similarity threshold of 60% on the depth-based WordNet metric of Seco, Veale and Hayes ([18]) was used for these experiments, though any WordNet-based similarity metric will suffice ([5]). The *perverse* case is clearly worse than the random baseline, while the *analogical* case is closer to ideal growth. Figure 3 shows that as the similarity threshold is lowered, and the potential for analogy is increased, we see the performance of the analogy-sorted queue move even closer to the *ideal* case. However, even at a 0% threshold, Figure 3 shows there is a considerable gap between the system’s performance and the ideal case. A 0% threshold turns the selection of valid propositions into a *predicate-popularity contest*: the propositions with predicates (and matching arities) that are most representative of the valid propositions already in the knowledge-base will be favored over those with previously unseen or rare predicates. This is a reasonable but imperfect strategy that shows the limits of analogical ordering, and no matter how low one sets the similarity threshold, a system cannot close the gap with the ideal case.

Each valid proposition in the knowledge-base can be viewed as marking an area of semantic space in which semantically similar, and equally valid, propositions might be found. The size of this space is dictated by the similarity threshold we choose: a high-threshold shrinks the space around each landmark proposi-

tion, reducing the number of analogical cliques that might be built around that proposition, and resulting in higher-precision but reduced recall for the analogical inferences we may draw. Conversely, a low threshold expands the space around each valid proposition, increasing the recall but diminishing the precision of our analogical inferences regarding the validity of novel propositions. Each threshold thus represents a compromise between recall and precision; the point on each curve where the curve takes a markedly upward turn is the point where this compromise tilts from effective to ineffective. So few analogies are permitted by a 90% similarity threshold that performance of the analogy-sorted queue is close to that of the random baseline. However, each successive lowering of the threshold, all the way down to 1%, shows obvious gains. At a threshold of 0% – which permits pure structure-mapping with no comparability constraints – this weak compromise produces mixed results. Remember, at a 0% threshold our analogies are no longer constrained to employ corpus-supported substitutions, and analogical matching becomes simple predicate and arity matching. At a 0% threshold, the system favors those propositions whose combination of predicate and arity are observed most frequently in the existing knowledge-base.

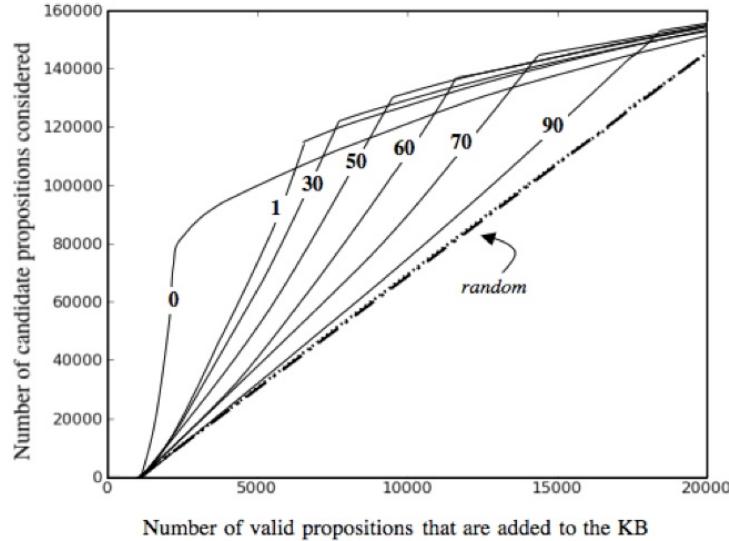
Fig. 3. Performance of analogy-guided acquisition at different similarity thresholds for making analogies.



As shown in Figure 4, lowering the threshold for analogy makes the *perverse* case even worse, especially at the 0% similarity threshold. Recall that in the perverse case –in which we attempt to model the worst-case scenario – the propositions that are deemed to have no analogical potential are considered first. However, the higher the similarity threshold, the greater the number of

propositions that the system will deem to have no analogical potential. Many propositions that have some analogical potential (at a lower similarity threshold) will be incorrectly deemed to have none, and processed first anyway. As the similarity threshold is lowered, the perverse case ensures that propositions with even a hint of analogical potential are processed last. Each graph for the *perverse* case thus shows two stages: an early stage where candidate propositions that are deemed to have no analogical potential are considered first, and a second stage where candidates with some potential are considered in reverse order of this potential. As Figure 4 shows, each lowering of the similarity threshold for analogy moves candidates from the first to the second stage.

Fig. 4. Worst-case acquisition scenarios (in the *perverse* case) at varying similarity thresholds for making analogies. The queue is sorted so that propositions with the most analogical potential are served last.



But the real test of analogy-guided acquisition is how much time it saves a knowledge-engineer. Imagine we want to double the current size of our KB by adding only valid propositions from the queue of candidate additions. Figure 5 graphs the speed-up factor achieved when doubling the KB size, relative to the random baseline of an unsorted queue, at different similarity thresholds.

Figure 5 shows that using analogy as a guide, doubling the KB size from 3000 to 6000 valid propositions is 4 times faster than using an unsorted queue, at a similarity threshold of 70%. Doubling from 5000 to 10000 is 3.5 times faster at a similarity threshold of 60%. This speed-up declines as the threshold is raised, and at 90% similarity there is practically no speed-up at all, because so few

analogies are made at this level. But the speed-up also declines as the KB grows in size, due to the *knowledge dilution effect*.

As valid propositions are removed from the pool of candidates and added to the KB, the concentration of valid propositions in the remaining pool is diluted. Initially around 1 in 8, the *dilution rate* of valid to invalid candidates can drop to 1 in 40 for the last 1000 valid propositions. Analogy-guided retrieval gives priority to those candidates with the most analogical potential, pushing those with little or no potential to the back of the queue. As the KB approaches its maximum size and valid propositions are crowded out by higher levels of noise, they are less differentiated by their analogical potential, causing overall performance to regress toward the random baseline.

Fig. 5. Efficiency savings when using analogy-guided acquisition. A system that uses analogy-guided acquisition with a given similarity threshold will double the initial size (X valid propositions) of its knowledge-base Y times faster than a system that does not use analogy-guided acquisition (the random case). Thus, e.g., a system that uses a similarity threshold of 10% will double its knowledge-base from X=2000 valid propositions to 4000 valid propositions Y=3 times faster than a system that does not use analogy-guided acquisition. These efficiency savings dwindle as the initial size (X) of the knowledge-base approaches 50% of its final size, as a system must then consider *all* incoming propositions if it is to double its size.

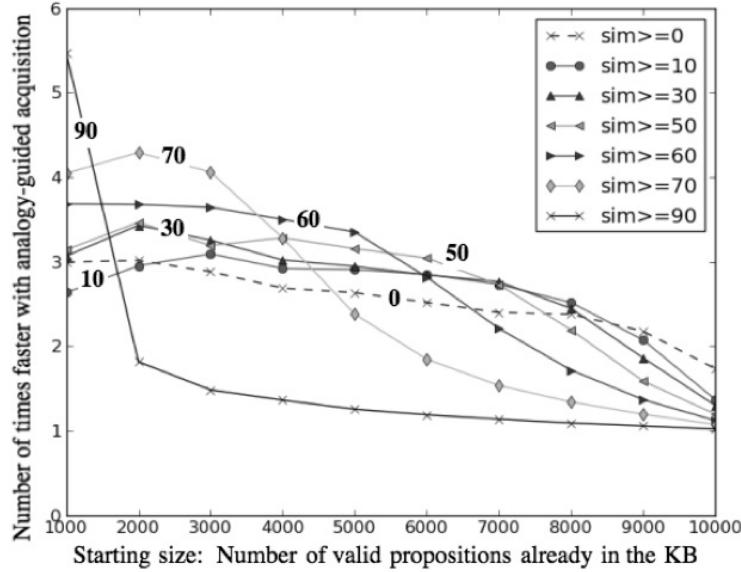
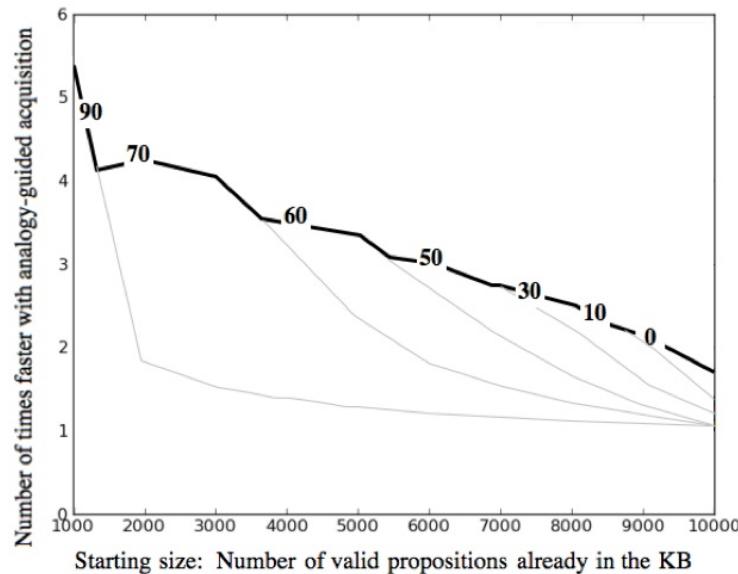


Figure 5 suggests that a mixed acquisition strategy makes the most sense: analogy-guided acquisition should use higher similarity thresholds in the earlier stages of KB construction, both to achieve higher throughput and so that system

suggestions can be explained to the user in the form of obvious, semantically-grounded analogies. As the KB grows, acquisition can shift to incrementally-lower similarity thresholds to offset the effects of knowledge-dilution. As low-hanging fruit is harvested, a system must rely on increasingly creative (and perhaps unsafe) analogies to maintain an efficient acquisition process.

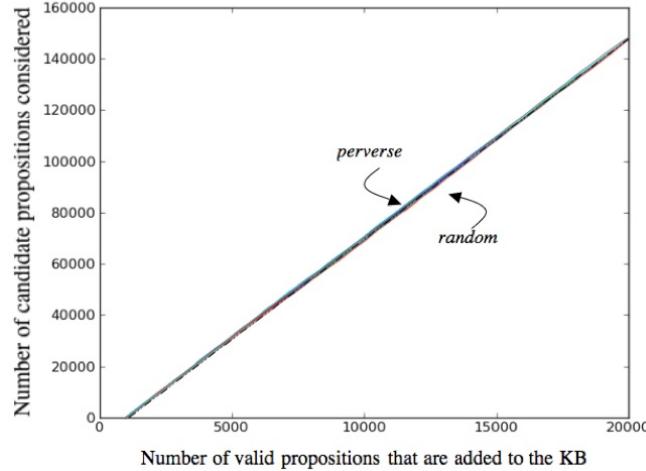
Fig. 6. Best performance is achieved by gear-changes from higher to lower similarity thresholds.



Higher-levels of similarity produce safer and more obvious analogies, but yield unsustainable efficiency gains. Figure 6 shows the gear changes that are needed to maximize the throughput of the acquisition process.

But can subjective differences in opinion about which propositions are valid or invalid alter the dynamics of analogy-guided acquisition? To find out, we imagine an extreme case: rather than use our manual annotations of propositions as valid or invalid, we randomly label 21,258 propositions from our pool of 158,911 n-gram candidates as *valid*, and label all others as *invalid*. This yields a randomized candidate pool with the same 1-in-8 distribution of *valid* propositions. Figure 7 shows the resulting collapse in performance at all similarity thresholds on the fully randomized KB. We see virtually no analogical structure at all in this KB, and the *perverse*, *random*, and *analogical* cases become indistinguishable. This illustrates that a knowledge-base is much more than a jumble of random facts and generalizations: the more coherent our knowledge, the greater the potential for analogy and the bigger the role of analogy-guided acquisition.

Fig. 7. Change in performance of analogy-guided acquisition on a randomly-constructed KB.



7 Conclusions and Future Work

Large knowledge-bases are subject to *neighborhood effects* ([29]) since propositions add more value to a KB when they interact effectively with others. This chapter has argued that analogy and comparability are an effective means of predicting precisely this kind of neighborly interaction.

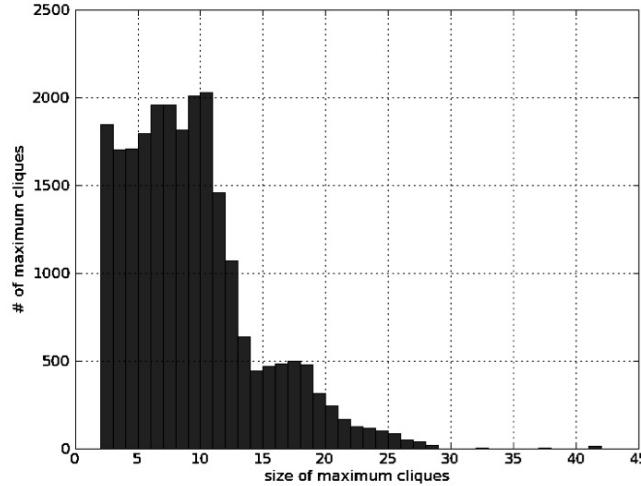
More specifically, we have presented a means of searching the pool of potential additions to a knowledge-base using analogy-guided best-first search. Queued additions that suggest the most analogies to valid propositions in the KB (at a specific similarity threshold) are moved to the front of the queue. A knowledge-base that strives to model an evolving world will always be in need of new propositions, and analogy-guided acquisition is assumed to occur within a knowledge-base that is constantly growing. That is, analogy-guided acquisition works best when used with a queue that is constantly receiving new candidate propositions from external sources. As new candidates arrive, the most promising are ushered to the front of the queue based on their likelihood of forming productive analogies and analogical cliques with the propositions that already reside in the KB.

We define analogy simply, in terms of structured comparability, which is a corpus-trainable pragmatic version of semantic similarity. Comparability is simultaneously a more generative and a more restrictive notion than semantic similarity, and it is this combination of generativity and restrictiveness that makes sensible analogical mapping efficient on a large scale. A generative measure

of semantic similarity can be used to both evaluate and suggest analogies. We have shown that analogy is a practical predictor of a candidate’s marginal value to a KB, even at very low thresholds of similarity.

In future work we must also examine the performance of analogy-guided acquisition on more complex types of propositional content than that considered here. In this vein, a productive starting point is the large set of implicit clique structures that naturally coalesce within the coordination graph that underpins our notion of corpus-driven pragmatic comparability. Recall that this graph contains an edge between two terms X and Y if corpus evidence (i.e., the Google 3-grams) suggests that X and Y are comparable. More specifically, an edge links X to Y if either of the 3-grams “ Xs and Ys ” or “ Ys and Xs ” can be found within the Google web n-grams database. These edges form complete sub-graphs of k nodes, or k -cliques, in which each node is connected by a corpus-attested edge to the $k - 1$ other nodes of the k -clique ([4]). A clique is maximal if it is not a proper-subset of another clique in the same graph. Each maximal k -clique within the coordination graph represents a tight cluster of highly-interrelated knowledge, of a kind that should be treated as a single whole, as a pragmatic category of sorts. The distribution of maximal cliques in the coordination graph for different sizes of k (clique size) is shown in Figure 8.

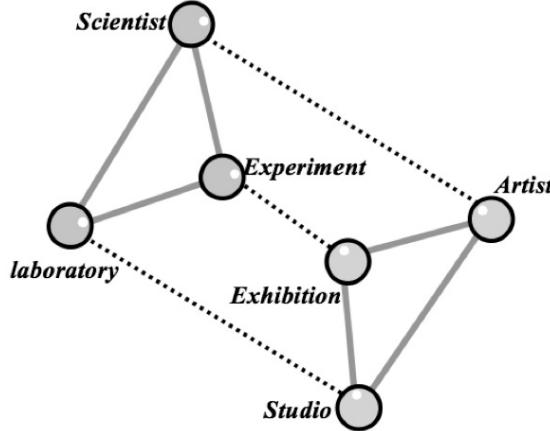
Fig. 8. Cliques of different sizes in the coordinations graph mined from the Google 3-grams.



Consider a simple example for $k = 3$. The 3-gram-derived coordination graph contains the following pair of 3-cliques (and many more besides, as shown in Figure 8): $\{\text{scientist}, \text{laboratory}, \text{experiment}\}$ and $\{\text{artist}, \text{studio}, \text{exhibition}\}$. But the coordination graph also contains edges that link *scientist* to *artist*, *laboratory* to *studio*, and *exhibition* to *experiment*. As such, if a KB contains propositions

to label each edge in the 3-clique $\{scientist, laboratory, experiment\}$ with an apt predicate, it can project these labels/predicates onto the 3-clique $\{artist, studio, exhibition\}$ and thereby form a truly systematic analogy (of the kind discussed in [9], [7], [24]). This situation is illustrated in Figure 9.

Fig. 9. A potential analogy between a pair of 3-cliques in the coordination graph.



In other words, a KB system can do more than seek to prioritize the addition of propositions that are analogous with previously acquired propositions. A system can actively seek to acquire propositions that allow it to build ever more systematic mappings between cliques of concepts, and between cliques of propositions ([26]).

We shall additionally look to exploit the generative capabilities of cliques and of the *generate-and-test* approach to analogy-making. This should allow the analogy-guided acquisition process to propose its *own* additions to the KB, over and above those in the queue of candidates extracted from corpora. For analogy has an intriguing role to play in shaping as well as recognizing valid knowledge.

This research was supported by the WCU (World Class University) program under the National Research Foundation of Korea, and funded by the Ministry of Education, Science and Technology of Korea (Project No: R31-30007).

References

1. Almuharesh A. and Massimo P.: Attribute-Based and Value-Based Clustering: An Evaluation. Proceedings of EMNLP, Empirical Methods in NLP, 158–165. (2004)
2. Barsalou, L.W.: Ad hoc categories. Memory and Cognition, 11:211–227. (1983)
3. Brants, T. and Franz, A.: Web 1T 5-gram Version 1. Linguistic Data Consortium. (2006)

4. Bron, C. and Kerbosch, J.: Algorithm 457: Finding all cliques of an undirected graph. Communications of the ACM 16(9). (1973)
5. Budanitsky, A. and Hirst, G.: Evaluating WordNet-based Measures of Lexical Semantic Relatedness. Computational Linguistics, 32(1):13–47. (2006)
6. Carlson, G. N. and Pelletier, F. (eds.): The Generic Book. University of Chicago Press. (1995)
7. Falkenhainer, B., Forbus, K. D., and Gentner, D.: Structure-Mapping Engine: Algorithm and Examples. Artificial Intelligence, 41:1–63. (1989)
8. Fellbaum, C.: (ed.). WordNet: An electronic lexical database. Cambridge, MA: MIT Press. (1998)
9. Gentner, D.: Structure-mapping: A Theoretical Framework. Cognitive Science 7:155–170. (1983)
10. Hearst, M.: Automatic acquisition of hyponyms from large text corpora. Proceedings of the 14th International Conference on Computational Linguistics, 539–545. (1992)
11. Hofstadter, D.R.: Tracking sentiment in mail: how genders differ on emotional axes. Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought. New York, NY: Basic Books. (1995)
12. Holyoak, K. J. and Thagard, P.: Mental Leaps: Analogy in Creative Thought. Cambridge, MA: Basic Books. (1995)
13. Karypis, G.: CLUTO: A clustering toolkit. Technical Report 02-017. University of Minnesota. <http://www-users.cs.umn.edu/~karypis/cluto/> (2002)
14. Lenat, D. and Guha, R.V.: Building Large Knowledge-based Systems. New York, NY: Addison Wesley. (1990)
15. Liu, H. and Singh, P.: ConceptNet: A Practical Commonsense Reasoning Toolkit. BT Technology Journal, 22(4):211–226. (2004)
16. Miller, G. A. and Charles, W.G.: Contextual correlates of semantic similarity. Language and Cognitive Processes 6(1):1–28. (1991)
17. Nakov, P. and Hearst, M.: Using verbs to characterize noun-noun relations. Artificial Intelligence: Methodology, Systems, and Applications, 233–244. (2006)
18. Seco, N., Veale, T. and Hayes, J.: An intrinsic information content metric for semantic similarity in WordNet. Proceedings of ECAI-2004, the 16th Annual Meeting of the European Association for Artificial Intelligence. (2004)
19. Speer, R., Havasi, C. and Lieberman, H.: AnalogySpace: Reducing the Dimensionality of Common Sense Knowledge. Proceedings of the 23rd AAAI Conference on Artificial Intelligence. (2008)
20. Singh, P.: The public acquisition of commonsense knowledge. Proceedings of AAAI Spring Symposium on Acquiring (and Using) Linguistic (and World) Knowledge for Information Access. Palo Alto, CA. (2002)
21. Tong, S. and Dean, J.: System and methods for automatically creating lists. US Patent 7,350,187 (granted to Google, March 25, 2008).
22. Turney, P. D.: Measuring semantic similarity by latent relational analysis. Proceedings of IJCAI-2005, the 19th International Joint Conference on Artificial Intelligence, Edinburgh, 1136–1141. (2005).
23. Turney, P.D.:A uniform approach to analogies, synonyms, antonyms, and associations, Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008), Manchester, UK, 905-912. (2008).
24. Veale, T. and Keane, M.T.: The Competence of Sub-Optimal Structure Mapping on ‘Hard Analogies’. Proceedings of ICJAI-1997, the 15th International Joint Conference on Artificial Intelligence, Nagoya, Japan. (1997)

25. Veale, T.: WordNet sits the SAT: A knowledge-based approach to lexical analogy. Proceedings of ECAI-2004, the 16th Annual Meeting of the European Association for Artificial Intelligence. (2004)
26. Veale, T. and Li, G.:Ontological cliques-analogy as an organizing principle in ontology construction. Proceedings of The International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, Madeira. (2009).
27. Veale, T. and Li, G.:Creating Similarity: Lateral Thinking for Vertical Similarity Judgments. In Proceedings of ACL 2013, the 51st Annual Meeting of the Association for Computational Linguistics, Sofia, Bulgaria (2009).
28. Weeds, J. and Weir, D.: Co-occurrence retrieval: A flexible framework for lexical distributional similarity. Computational Linguistics, 31(4):433–475. (2005)
29. Weigher, J.C. and Zerbst, R. H.: The Externalities of Neighborhood Parks: An Empirical Investigation. Land Economics 49(1):99–105. (1973)

Analogy in Automated Deduction – A Survey

Thierry Boy de la Tour and Nicolas Peltier

University of Grenoble (CNRS/LIG)

Abstract. We provide a survey of the main approaches for analogical reasoning in automated deduction. We start by proposing a general framework for reasoning by analogy based on a constrained sequent calculus in which higher-order variables denote first-order formulæ. Then we briefly review the most successful approaches, ranging from early work in Horn logic to proof reuse in interactive higher-order theorem provers. With the help of many examples, we present the main ideas and basic features of each approach and briefly discuss their common points and differences.

1 Introduction

The formalization of reasoning by analogy has been very early identified as a fundamental challenge in Automated Deduction (see for instance [3, 36]). At a very general level, reasoning by analogy consists in using information deduced when solving a problem or collection of problems (the *source* problems) in order to guide or facilitate the solving of a new formula (the *target* problem). While this process is extensively used by human beings, it is completely ignored by most existing theorem provers that try to prove theorems from basic axioms by applying uniform inference systems. Of course, these provers can use libraries of previously proven theorems as axioms in order to derive more elaborate results and to construct new proofs. However, they are not able to acquire any general knowledge about the process of proof construction itself, which could possibly be fruitfully applied in completely different contexts. In contrast, the human brain seems to be inherently programmed to extract general structures in the information it processes, and is able to re-use these derived patterns, in an amazingly short time, in seemingly completely unrelated contexts. Inducing general principles from specific facts is generally viewed as a fundamental aspect of intelligent thinking. “*An idea is always a generalization, and generalization is a property of thinking. To generalize means to think*” [21].

Analogy can be used in order to increase the performances of theorem-provers, but it is also an interesting way of reasoning in itself. Detecting similarities or recurrent patterns inside proofs can yield very interesting results. It allows one for instance to improve the presentation and readability of the proofs, by adding new lemmata and auxiliary definitions (see for instance [23]), to detect and prune cycles in the derivations [32], but can also pave the ground for the development of new theories or to the discovery of new theorems.

Although the importance of analogy has long been recognized in automated deduction, analogy-related studies are surprisingly scarce. In this paper, we provide a brief high-level survey of the main approaches in this field. We start by providing a general framework for generalizing and re-using proofs, and then we briefly review some of the techniques that have been proposed for this purpose, summarizing their main ideas and strong points and giving simple examples of applications.

2 Preliminaries

We assume the reader is familiar with the main notions in logic and automated deduction (see for instance [33] for more details). Terms and formulæ are built inductively as usual on a set of *function symbols* \mathcal{F} , a set of *predicate symbols* \mathcal{P} and a set of *variables* \mathcal{V} , using the standard logical connectives \neg , \vee , \wedge , \Rightarrow and the quantifiers \forall and \exists . Every function and predicate symbol has a unique *arity*. For simplicity we assume that all the individuals in the domain have the same type (denoted by `obj`). Formulæ are of type `bool`. The equality predicate is written in infix notation and denoted by \simeq . The notions of interpretations, models, validity etc. are defined as usual.

Although we only deal with first-order theorem-proving in the present paper, we shall also consider higher-order terms and formulae to encode generalized theorems, obtained by replacing function symbols by second-order variables of the same type. The types of such expressions are constructed from the basic types `obj` and `bool` by the operator \rightarrow . The expressions themselves are built by using the previous constructions together with the *lambda abstraction* $\lambda x t$, yielding an expression (term or formula) of type $s \rightarrow s'$, where x is a variable of type s and t is an expression of type s' , and the *function application* $t(s)$, yielding an expression of sort s' if t and s are of types $s \rightarrow s'$ and s , respectively. We assume that each variable is mapped to a unique type. Expressions are taken modulo α - and β -equivalences, i.e. modulo the renaming of bound variables and the application of the β -reduction rule: $(\lambda x t)(s) = t[s/x]$, i.e., the term obtained from t by replacing every occurrence of x by s . We assume that β -reduction is only applied on subterms where bound variables do not occur free; this is not a restriction since any term is α -equivalent to such terms and it guarantees that local free variables are preserved by β -reductions.

We shall use the *sequent calculus* as a basic logical framework. Indeed, although not suited for automated theorem-proving, this calculus is well-known and very general, and thus the presented notions will be easy to generalize and re-use in other contexts. This calculus aims at proving *sequents*, that are expressions of the form $\Gamma \vdash \Delta$, where Γ and Δ are two multisets¹ of formulæ. Semantically, a sequent $\phi_1, \dots, \phi_n \vdash \psi_1, \dots, \psi_m$ is to be interpreted as an implication: $(\wedge_{i=1}^n \phi_i) \Rightarrow (\vee_{i=1}^m \psi_i)$. We write $(\Gamma \vdash \Delta) \subseteq (\Sigma \vdash \Pi)$ if $\Gamma \subseteq \Sigma$ and $\Delta \subseteq \Pi$. Sequents are proven by mean of a finite set of *inference rules*, which

¹ A multiset is a generalization of sets in which members are allowed to appear more than once.

| | |
|---|---|
| $\frac{\phi, \Gamma \vdash \phi, \Delta}{\Gamma \vdash \phi, \Delta}$ (Axiom) | $\frac{\Gamma \vdash t \simeq t, \Delta}{\Gamma \vdash t \simeq t, \Delta}$ (Ref) |
| $\frac{\Gamma, \phi \vdash \Delta \quad \Gamma \vdash \Delta, \phi}{\Gamma \vdash \Delta}$ (Cut) | |
| $\frac{t \simeq s, \phi[t], \Gamma \vdash \Delta}{t \simeq s, \phi[s], \Gamma \vdash \Delta}$ (Param-L) | $\frac{t \simeq s, \Gamma \vdash \psi[t], \Delta}{t \simeq s, \Gamma \vdash \psi[s], \Delta}$ (Param-R) |
| $\frac{\Gamma, \phi, \psi \vdash \Delta}{\Gamma, \phi \wedge \psi \vdash \Delta}$ (\wedge -L) | $\frac{\Gamma \vdash \phi, \Delta}{\Gamma \vdash \phi \vee \psi, \Delta}$ (\vee -R) |
| $\frac{\Gamma, \phi \vdash \Delta \quad \Sigma, \psi \vdash \Pi}{\Gamma, \Sigma, \phi \vee \psi \vdash \Delta, \Pi}$ (\vee -L) | $\frac{\Gamma \vdash \phi, \Delta \quad \Sigma \vdash \psi, \Pi}{\Gamma, \Sigma \vdash \phi \wedge \psi, \Delta, \Pi}$ (\wedge -R) |
| $\frac{\Gamma \vdash \phi, \Delta \quad \Sigma, \psi \vdash \Pi}{\Gamma, \Sigma, \phi \Rightarrow \psi \vdash \Delta, \Pi}$ (\Rightarrow -L) | $\frac{\Gamma, \phi \vdash \psi, \Delta}{\Gamma \vdash \phi \Rightarrow \psi, \Delta}$ (\Rightarrow -R) |
| $\frac{\Gamma \vdash \phi, \Delta}{\Gamma, \neg \phi \vdash \Delta}$ (\neg -L) | $\frac{\Gamma, \phi \vdash \Delta}{\Gamma \vdash \neg \phi, \Delta}$ (\neg -R) |
| $\frac{\Gamma, \phi[t/x] \vdash \Delta}{\Gamma, \forall x \phi \vdash \Delta}$ (\forall -L) | $\frac{\Gamma \vdash \phi[t/x]\Delta}{\Gamma \vdash \exists x \phi, \Delta}$ (\exists -R) |
| $\frac{\Gamma, \phi[y/x] \vdash \Delta}{\Gamma, \exists x \phi \vdash \Delta}$ (\exists -L) | $\frac{\Gamma \vdash \phi[y/x], \Delta}{\Gamma \vdash \forall x \phi, \Delta}$ (\forall -R) |
| Where y denotes a variable not occurring freely in Γ, Δ, ϕ (eigenvariable condition) | |

Fig. 1. The sequent calculus LK

are recalled in Figure 1 (for conciseness we omit the structural rules, which consist in duplicating or removing formulæ, and the variants of the rules obtained by commutativity of \vee , \wedge). A rule of the form $\frac{\Gamma_1 \vdash \Delta_1 \quad \dots \quad \Gamma_n \vdash \Delta_n}{\Gamma \vdash \Delta}$ (with $n \in \{0, 1, 2\}$) indicates that a sequent $\Gamma \vdash \Delta$ (modulo a matching of the metavariables) is provable if for all $i \in [1, n]$, the sequents $\Gamma_i \vdash \Delta_i$ are provable. If $n = 0$ then the sequent $\Gamma \vdash \Delta$ is an *axiom*.

In Figure 1 the metavariables x, y denote variables, s, t denote terms, ϕ, ψ denote formulæ and $\Gamma, \Delta, \Sigma, \Pi$ denote multisets of formulæ. The expression $\phi[t]$ denotes a formula ϕ in which a number (possibly zero) of occurrences of t are singled out, so that $\phi[s]$ then denotes the formula obtained from ϕ by replacing all the singled out occurrences of t by s .

Substitutions are partial functions mapping variables to terms of the same type. For any expression (term, formula, sequent, ...) t we denote by $t\sigma$ the expression obtained from t by replacing every free occurrence of a variable $x \in \text{dom}(\sigma)$ by $\sigma(x)$. We shall denote by $\{x_1 \mapsto t_1, \dots, x_n \mapsto t_n\}$ the substitution of domain $\{x_1, \dots, x_n\}$ mapping every variable x_i to t_i ($1 \leq i \leq n$). A substitution σ is *ground* if for every $x \in \text{dom}(\sigma)$, $x\sigma$ contains no free variable.

Most of the terms considered in the present paper are of type `obj`, `bool`, `obj → bool`, or `obj → bool → bool`. However, we shall also consider terms of type `bool*` → `obj` (i.e. `bool` → ... → `bool` → `obj`), used to denote the functions mapping formulæ to Skolem terms. We impose the additional restriction that no bound variable can occur in the scope of a term of this type, i.e., we

dismiss the expressions of the form $\lambda x t$ such that t contains a term $u(v)$, where v contains the variable x and u is of type $\text{bool}^* \rightarrow \text{obj}$. A term is *admissible* if it contains no expression of the previous form. A substitution σ is *admissible* if $x\sigma$ is admissible for every variable x . Note that admissibility is preserved by β -reduction. In the following we assume (unless otherwise specified) that all the terms and substitutions are admissible.

3 A Framework for Reasoning by Analogy

According to Peirce [20], analogy can be seen as an induction and an abduction followed by a deduction. The proposed formalizations of this notion in automated deduction follow more or less implicitly this scheme. First the source theorem is generalized by abstracting away irrelevant details and extracting the essential information. Second, a matching is established between the source theorem and the target theorem. Finally, the matching is applied to the proof of the source theorem. In the best case, one obtains a direct proof of the target theorem, but in general, the analogy is only partial in the sense that some deductive steps are required to complete or adapt the proof.

3.1 Proof Generalization

The first step towards generalization consists in abstracting away the symbols occurring in the source statement. Consider for instance the following proof tree:

$$\frac{\overline{p(a) \vdash p(a), p(b)} \quad \overline{p(a), p(b) \vdash p(b)}}{p(a), p(a) \Rightarrow p(b) \vdash p(b)}$$

By definition, the inference is valid for all possible values of the predicate p and constant symbols a, b , which can be expressed by interpreting the previous end-sequent as a second-order implication, where a and b denote variables of the base type obj and p is a variable of type $\text{obj} \rightarrow \text{bool}$. Of course, if a sorted signature is used then the type constants can also be abstracted away and replaced by type variables. The obtained formula is however still not general enough: it is clear that for instance the sequent $p(a), p(a) \Rightarrow q(b) \vdash q(b)$ can be proven exactly in the same way as the previous one. More formally, we can remark that the fact that the head symbol of the premiss $p(a)$ and the conclusion $p(b)$ are identical plays no rôle in the proof: relaxing this condition by replacing these two occurrences of p by different symbols preserves the validity of all the inferences in the derivation. Thus, the next generalization step consists in replacing every *occurrence* of all symbols in the original statement by distinct second-order variables, yielding the end-sequent:

$$p_1(a_1), p_2(a_2) \Rightarrow p_3(b_1) \vdash p_4(b_2)$$

However, this time the obtained assertion is not valid anymore, since it is clear that the modus ponens rule can only be applied if $p_1(a_1) = p_2(a_2)$ and $p_3(b_1) =$

$p_4(b_2)$; these equalities hold in particular if $p_1 = p_2$, $a_1 = a_2$, $p_3 = p_4$ and $b_1 = b_2$.

We are thus faced with the problem of extracting, from given end-sequents and proofs, the conditions on the second-order variables introduced during the generalization process that guarantee that the generalized derivation remains a proof, in the sense that all the first-order instances of this generalized derivation are *LK* proofs. These conditions can be computed by collecting the equations that make applicable all the inference steps occurring in the proof tree. For instance, a sequent of the form $\phi \vdash \psi$ is an axiom if ϕ and ψ are syntactically equivalent. If ϕ and ψ are terms built on higher-order variables, then the condition $\phi \doteq \psi$ can be asserted as a constraint on these variables that ensures the soundness of the derivation. This idea can be extended to structural rules, by introducing variables for all formulæ occurring in the proof and by using equations to encode the structural relations between those formulæ. For instance the \Rightarrow -L rule is applicable on two sequents $\Gamma \vdash \phi, \Delta$ and $\Sigma, \psi \vdash \Pi$, yielding a sequent $\Gamma, \Sigma, \chi \vdash \Delta, \Pi$, if the equation $\chi \doteq (\phi \Rightarrow \psi)$ holds. Note that we use the predicate \doteq (instead of the semantic equality \simeq) in order to denote syntactic identities between terms or formulæ.

The previous idea extends easily to the first-order case, although the handling of quantifiers requires some care. Consider for instance the \forall -L rule, which is applicable only on universal quantification. A seemingly natural idea would be to encode this condition by an equation of the form $\phi \doteq \forall x \psi$, where x and ψ are new variables of the appropriate types; but then the replacement of x by t in the formula ψ would have to be handled by using an explicit encoding of substitutions and substitution applications, which would be rather tedious. A much simpler and natural solution is to consider the quantifiers as *unary* functions operating on second-order terms, more precisely as functions of type $(\text{obj} \rightarrow \text{bool}) \rightarrow \text{bool}$. Then a formula of the form $\forall \psi$ (resp. $\exists \psi$) holds iff $\psi(x)$ is true for every x (resp. for at least one x). The previous condition would then be written $\phi \doteq \forall \psi$ (thus abstracting away the name of the quantified variable x) and the replacement of x by a new term t is simply denoted by function application: $\psi(t)$. The conditions for the paramodulation rule are handled in a similar way. The replacement of the term t in a formula ϕ by a new term s is feasible if $\phi \doteq \psi(t)$, where ψ is a new variable of type $\text{obj} \rightarrow \text{bool}$, and the result is $\psi(s)$.

The constraints corresponding to the conditions allowing to apply all the inference steps can thus be collected and attached to the end-sequent, yielding a constrained formula language. This idea of separating the structure of the proof tree from the unification conditions ensuring the soundness of the inferences has been used in various contexts, for instance in [22], for analyzing sequent proofs and reducing the complexity of the derivations, in [17] for reasoning modulo theories, or in [9, 4] for building models of clause sets.

Definition 1. A unification problem is either \perp or a (possibly empty) conjunction of equations $t \doteq s$ between expressions of the same type (empty conjunctions are denoted by \top). An admissible ground substitution σ is a solution of a prob-

| | |
|---|---|
| $\frac{\Gamma, \phi \vdash \Delta, \psi \mid \phi \doteq \psi}{\Gamma, \phi \vdash \Delta} \text{ (Ax)}$ $\frac{\Gamma, \phi \vdash \Delta \quad \Gamma \vdash \Delta, \psi}{\Gamma \vdash \Delta \mid \phi \doteq \psi} \text{ (Cut)}$ $\frac{\chi, \psi, \Gamma \vdash \Delta}{\chi, \phi, \Gamma \vdash \Delta \mid \chi \doteq (s \simeq t) \wedge \phi \doteq \xi(s) \wedge \psi \doteq \xi(t)} \text{ (Param-L)}$ $\frac{\chi, \psi, \Gamma \vdash \Delta}{\chi, \Gamma \vdash \psi, \Delta} \text{ (Param-R)}$ $\frac{\Gamma, \phi, \psi \vdash \Delta}{\Gamma, \chi \vdash \Delta \mid \chi \doteq \phi \wedge \psi} \text{ (\wedge-L)}$ $\frac{\Gamma, \phi \vdash \Delta \quad \Sigma, \psi \vdash \Pi}{\Gamma, \Sigma, \chi \vdash \Delta, \Pi \mid \chi \doteq \phi \vee \psi} \text{ (\vee-L)}$ $\frac{\Gamma \vdash \phi, \Delta \quad \Sigma, \psi \vdash \Pi}{\Gamma, \Sigma, \chi \vdash \Delta, \Pi \mid \chi \doteq \phi \Rightarrow \psi} \text{ (\Rightarrow-L)}$ $\frac{\Gamma \vdash \phi, \Delta}{\Gamma, \chi \vdash \Delta \mid \chi \doteq \neg \phi} \text{ (\neg-L)}$ $\frac{\Gamma, \phi \vdash \Delta}{\Gamma, \chi \vdash \Delta \mid \chi \doteq \forall \xi \wedge \phi \doteq \xi(t)} \text{ (\forall-L)}$ $\frac{\phi, \Gamma \vdash \Delta}{T, \chi \vdash \Delta \mid \chi \doteq \exists \xi \wedge \phi \doteq \xi(f(\mathbf{x}))} \text{ (\exists-L)}$ | $\frac{\Gamma \vdash \Delta, \phi \mid \phi \doteq s \simeq t \wedge s \doteq t}{\Gamma \vdash \Delta \mid \phi \doteq s \simeq t} \text{ (Ref)}$ $\frac{\chi, \psi, \Gamma \vdash \Delta}{\chi, \Gamma \vdash \psi, \Delta} \text{ (Param-L)}$ $\frac{\chi, \psi, \Gamma \vdash \Delta}{\chi, \Gamma \vdash \psi, \Delta} \text{ (Param-R)}$ $\frac{\Gamma \vdash \phi, \Delta}{\Gamma \vdash \chi, \Delta \mid \chi \doteq \phi \vee \psi} \text{ (\vee-R)}$ $\frac{\Gamma \vdash \phi, \Delta \quad \Sigma \vdash \psi, \Pi}{\Gamma \vdash \Sigma, \chi, \Delta, \Pi \mid \chi \doteq \phi \wedge \psi} \text{ (\wedge-R)}$ $\frac{\Gamma, \phi \vdash \psi, \Delta}{\Gamma \vdash \chi, \Delta \mid \chi \doteq \phi \Rightarrow \psi} \text{ (\Rightarrow-R)}$ $\frac{\Gamma, \phi \vdash \Delta}{\Gamma \vdash \chi, \Delta \mid \chi \doteq \neg \phi} \text{ (\neg-R)}$ $\frac{\Gamma \vdash \phi, \Delta}{\Gamma \vdash \chi, \Delta \mid \chi \doteq \exists \xi \wedge \phi \doteq \xi(t)} \text{ (\exists-R)}$ $\frac{\Gamma \vdash \phi, \Delta}{\Gamma \vdash \chi, \Delta \mid \chi \doteq \forall \xi \wedge \phi \doteq \xi(f(\mathbf{x}))} \text{ (\forall-R)}$ |
|---|---|

f is a function symbol not occurring in Δ, Γ, χ (and distinct from all other Skolem symbols occurring in the proof tree) and \mathbf{x} is the vector of variables free in Δ, Γ, χ .

Fig. 2. The generalized sequent calculus LK^g

lem \mathcal{P} if \mathcal{P} is of the form $\bigwedge_{i=1}^n (t_i \doteq s_i)$ (with $n \geq 0$) and $\forall i \in [1, n] t_i \sigma = s_i \sigma$. The set of solutions of a problem \mathcal{P} is denoted by $Sol(\mathcal{P})$.

Definition 2. A generalized sequent is an expression of the form $[\Gamma \vdash \Lambda \mid \mathcal{P}]$ where Γ and Λ are two sequences of expressions of type `bool` and \mathcal{P} is a unification problem.

It is easy to adapt the rules of Figure 1 in order to collect automatically all the necessary constraints, instead of checking them on the fly during proof construction. The obtained calculus is depicted in Figure 2. There, $\Gamma, \Delta, \Sigma, \Pi$ are multisets of terms of type `bool`, χ, ϕ, ψ are terms of type `bool`, ξ is a term of type `obj → bool` and s, t are terms of type `obj`.

More precisely, a rule of the form

$$\frac{\Gamma_1 \vdash \Delta_1 \dots \Gamma_n \vdash \Delta_n}{\Gamma \vdash \Delta \mid \mathcal{P}}$$

means that a generalized sequent of the form $[\Gamma \vdash \Delta \mid \mathcal{P}_1 \wedge \dots \wedge \mathcal{P}_n \wedge \mathcal{Q}]$ (modulo an instantiation of the meta-variables) can be inferred from n generalized sequents $[\Gamma_1 \vdash \Delta_1 \mid \mathcal{P}_1], \dots, [\Gamma_n \vdash \Delta_n \mid \mathcal{P}_n]$. Note that the constraints of the premisses are always weaker than those in the conclusion.

The only rules that require some care are the \exists -L and \forall -R rules, that introduce a new variable y in the sequent by Skolemisation. Due to the eigenvariable condition, this variable cannot occur in the sequences of formulæ Γ and Δ occurring in the end-sequent; but this condition cannot be tested because these formulæ are not known at this point. Instead, it can easily be enforced by making y depend on the sequences Γ and Δ (this idea is essentially the same as that of the Skolemisation rule, which replaces an existentially quantified variable by a term depending on the variables freely occurring in the formula). This is done by introducing a new function symbol f depending on the variables occurring in the considered sequents.

Example 1. We consider the following proof tree in LK .

$$\frac{\frac{\frac{\frac{q(a) \vdash q(a)}{q(a) \vdash \exists x q(x)}}{p(b), p(b) \Rightarrow q(b) \vdash \exists x q(x)}}{p(b), \forall x p(x) \Rightarrow q(x) \vdash \exists x q(x)}}{q(a) \vee p(b), \forall x p(x) \Rightarrow q(x) \vdash \exists x q(x)}$$

It is immediately possible to transform this tree into a proof tree in LK^g by adding trivial unification problems to each sequent. For instance, the top left sequent is replaced by $[q(a) \vdash q(a) \mid q(a) \doteq q(a)]$ and the last sequent by

$$[q(a) \vee p(b), \forall x p(x) \Rightarrow q(x) \vdash \exists x q(x) \mid q(a) \vee p(b) \doteq q(a) \vee p(b)].$$

According to the definition, the last unification problem should actually be the conjunction of all the unification problems in the tree. Of course the resulting proof tree has a lot of redundant information and lacks in readability. But it is possible to be even more concise than the LK proof tree by systematically using variables to represent formulæ and terms. This yields the following proof tree (again we only depict the local unification problems).

$$\frac{\frac{\frac{\frac{[u_4 \vdash u_{11} \mid u_4 \doteq u_{11}]}{[u_4 \vdash u_3 \mid u_3 \doteq \exists u_{10} \wedge u_{11} \doteq u_{10}(v_3)]}}{[u_5 \vdash u_7 \mid u_5 \doteq u_7]}}{[u_5, u_{12} \vdash u_3 \mid u_3 \doteq \exists u_9 \wedge u_{13} \doteq u_9(v_2)]}}{[u_1, u_2 \vdash u_3 \mid u_1 \doteq u_4 \vee u_5]}$$

We consider this proof tree as the generalized version of the LK proof. It represents the minimal syntax required of the theorem by this LK proof.

By definition, the end-sequent of every proof in LK^g is of the form $[\Gamma \vdash \Delta \mid \mathcal{P}]$ where Γ and Δ are sequences of expressions of type `bool` and \mathcal{P} is a unification problem. \mathcal{P} involves not only the variables in Γ and Δ but also other variables, denoting the subformulæ and terms occurring in the proof. The calculus LK can be viewed as a particular form of LK^g , in which all the sequents are completely instantiated, with valid unification problems (and in which the eigenvariables are replaced by terms depending on sequences of formulæ). This remark yields the following two theorems.

Theorem 1. *For every proof tree π in LK of end-sequent $\Gamma \vdash \Delta$ and for every sequent $\Gamma^g \vdash \Delta^g$ such that there exists an admissible substitution σ with $(\Gamma \vdash \Delta) = (\Gamma^g \vdash \Delta^g)\sigma$, there exists a proof tree π^g in LK^g of end-sequent $[\Gamma^g \vdash \Delta^g \mid \mathcal{P}]$ and an extension σ' of σ (coinciding with σ on all the variables in $\Gamma^g \vdash \Delta^g$) that is a solution of \mathcal{P} .*

Proof. (Sketch) The proof tree π^g is constructed by induction on π . By definition, the end-sequent $\Gamma \vdash \Delta$ must be inferred from n sequents $\Gamma_1 \vdash \Delta_1, \dots, \Gamma_n \vdash \Delta_n$ by one of the rules in LK (with $n = 0, 1, 2$). By definition of the calculus LK^g , it is easy to check (by inspection of the different rules) that for every instance of a rule ρ in LK , there exists a corresponding instance of a rule ρ^g in LK^g , with constraints \mathcal{Q} , such that ρ is an instance of ρ^g , in the sense that there exists a solution θ of \mathcal{Q} such that the sequents occurring as premisses and conclusion in the rule $\rho^g\theta$ are identical to the premisses and conclusion of ρ . Since $(\Gamma^g \vdash \Delta^g)\sigma = \Gamma \vdash \Delta$, θ can be written as $\theta'\sigma$, where θ' maps the antecedent and succedent of the conclusion of ρ^g to Γ^g and Δ^g respectively. In each rule, the constraints of the premisses are simply passed to the conclusion without affecting the soundness of the inference and, similarly, the antecedent and succedent of the conclusion can be arbitrary. Thus, for all $\mathcal{P}_1, \dots, \mathcal{P}_n$, an end-sequent of the form $[\Gamma^g \vdash \Delta^g \mid \mathcal{P}_1 \wedge \dots \wedge \mathcal{P}_n \wedge \mathcal{Q}\theta']$ can always be deduced from n generalized sequents of the form $[\Gamma_i^g \vdash \Delta_i^g \mid \mathcal{P}_i]$ ($1 \leq i \leq n$) by applying the rule $\rho^g\theta'$. We have $(\Gamma^g \vdash \Delta^g)\sigma = \Gamma \vdash \Delta$, and since ρ is equivalent to $\rho^g\theta$, there exists an extension σ'' of σ such that $\sigma'' \in \text{Sol}(\mathcal{Q}\theta')$ and $\forall i \in [1, n] (\Gamma_i^g \vdash \Delta_i^g)\sigma'' = \Gamma_i \vdash \Delta_i$. By the induction hypothesis, there exist n proof trees π_1^g, \dots, π_n^g in LK^g of end-sequents $[\Gamma_1^g \vdash \Delta_1^g \mid \mathcal{P}_1], \dots, [\Gamma_n^g \vdash \Delta_n^g \mid \mathcal{P}_n]$ and n extensions $\sigma'_1, \dots, \sigma'_n$ of σ'' such that $\forall i \in [1, n] \sigma'_i \in \text{Sol}(\mathcal{P}_i)$. We build the proof tree π^g in LK^g by appending the rule $\rho^g\theta'$ to the subtrees π_1^g, \dots, π_n^g , with end-sequent $[\Gamma^g \vdash \Delta^g \mid \mathcal{P}_1 \wedge \dots \wedge \mathcal{P}_n \wedge \mathcal{Q}\theta']$.

Without loss of generality we can assume that the variables occurring at non-root level in the proof trees π_1, \dots, π_n are distinct (otherwise they can simply be renamed), and also distinct from that occurring in $\Gamma^g \vdash \Delta^g$. Thus, since the σ'_i coincide on all the variables occurring in the end-sequent, we can define the substitution $\sigma' = \bigcup_{i=1}^n \sigma'_i$, which fulfills all the desired properties: it is admissible since it is an extension of the admissible σ_i 's, these are extensions of σ'' which is an extension of σ , hence σ' is also an extension of σ , and as an extension of solutions of the \mathcal{P}_i 's and $\mathcal{Q}\theta'$ it is a solution of their conjunction.

Theorem 2. *Let π be a proof tree in LK^g with end-sequent $[\Gamma \vdash \Delta \mid \mathcal{P}]$. For every solution σ of \mathcal{P} , there is a proof in LK of $(\Gamma \vdash \Delta)\sigma$.*

Proof. (Sketch) It is sufficient to remark that every first-order instance of a rule in LK^g satisfying the constraints of the rule is an instance of a rule in LK . The detailed proof is by a straightforward induction on the proof tree. For the sake of conciseness, we only handle the case of the rule $\exists\text{-}L$ (this is the most complex case, since it involves eigenvariables). The end-sequent of π is then of the form $[\Gamma, \chi \vdash \Delta \mid \chi \doteq \exists \xi \wedge \phi \doteq \xi(f(\mathbf{x})) \wedge \mathcal{Q}]$ and the premiss is $[\phi, \Gamma \vdash \Delta \mid \mathcal{Q}]$. We denote by π' the proof tree whose end-sequent is $[\phi, \Gamma \vdash \Delta \mid \mathcal{Q}]$. We assume, w.l.o.g.,

that the LK^g -rules are applied in such a way that all the terms $\chi, \phi, \psi, \xi, s, t$ are variables (note that if this is not the case, then we can simply replace these terms by new variables and replace the substitution σ by $\sigma'.\sigma$, where σ' maps every newly introduced variable to the corresponding term). Hence the equations in \mathcal{Q} are very simple, and particularly there is no occurrence of f or λ in \mathcal{Q} (other occurrences of the rule \exists -L in π' introduce other function symbols than f).

Let σ be a solution of $\chi \doteq \exists \xi \wedge \phi \doteq \xi(f(\mathbf{x})) \wedge \mathcal{Q}$. Let y be a fresh variable, not occurring in $\Gamma\sigma, \chi\sigma \vdash \Delta\sigma$ and consider the substitution θ obtained from σ by replacing all occurrences of $f(\mathbf{x})\sigma$ by y , formally defined as follows: $\forall x \in \mathcal{V}, x\theta \stackrel{\text{def}}{=} x\sigma[y/f(\mathbf{x})\sigma]$. Since by definition σ is admissible, so is θ . Assume that θ is not a solution of \mathcal{Q} . Then \mathcal{Q} necessarily contains an equation $u \doteq v$ such that $u\theta \neq v\theta$. By hypothesis we have $u\sigma = v\sigma$, hence $u\sigma[y/f(\mathbf{x})\sigma] = v\sigma[y/f(\mathbf{x})\sigma]$, so we must have, say, $u\sigma[y/f(\mathbf{x})\sigma] \neq u\theta$. Since f does not occur in u , the term $u\theta$ must allow for a β -reduction that eventually creates an occurrence of $f(\mathbf{x})\sigma$. This means that in $u\theta$ must occur a term $\lambda x w$ in which occurs some $f(\mathbf{w})$ where x occurs in \mathbf{w} . But this term is not admissible, and since λ does not occur in u this means that θ is not admissible, a contradiction.

Consequently, θ is a solution of \mathcal{Q} and by the induction hypothesis there is a proof in LK of $\phi\theta, \Gamma\theta \vdash \Delta\theta$. Furthermore, we have $\chi\sigma = \exists\xi\sigma$ and $\phi\sigma = \xi\sigma(f(\mathbf{x})\sigma)$. Let x be any variable free in Γ, χ, Δ , by definition of \mathbf{x} the term $x\sigma$ is a strict subterm of $f(\mathbf{x})\sigma$, thus $f(\mathbf{x})\sigma$ cannot be a subterm of $x\sigma$. Hence the term $f(\mathbf{x})\sigma$ cannot occur in $(\Gamma, \chi \vdash \Delta)\sigma$ (note that f does not occur in Γ, χ, Δ) which is therefore equal to $(\Gamma, \chi \vdash \Delta)\theta$. Obviously $f(\mathbf{x})\sigma$ cannot occur either in $\xi\sigma$ which similarly entails $\xi\sigma[y/f(\mathbf{x})\sigma] = \xi\sigma$. By our convention ϕ is a variable hence we may prove as above that $\phi\theta = \phi\sigma[y/f(\mathbf{x})\sigma]$, thus $\phi\theta = \xi\sigma(f(\mathbf{x})\sigma)[y/f(\mathbf{x})\sigma] = \xi\sigma(y)$. By definition y does not occur free in $(\Gamma, \chi \vdash \Delta)\theta$, hence the rule \exists -L of LK is applicable on $\phi\theta, \Gamma\theta \vdash \Delta\theta$ yielding $\Gamma\theta, \exists\xi\sigma \vdash \Delta\theta$, which is the expected sequent $(\Gamma, \chi \vdash \Delta)\sigma$.

3.2 Matching

According to Theorems 1 and 2, a general technique for detecting that a sequent $\Gamma_t \vdash \Delta_t$ is provable by analogy with a previously proven sequent $\Gamma_s \vdash \Delta_s$ can be described as follows.

1. Compute the generalization $[\Gamma_g \vdash \Delta_g \mid \mathcal{P}]$ of $\Gamma_s \vdash \Delta_s$. This is done by reconstructing the LK^g derivation corresponding to the proof of $\Gamma_s \vdash \Delta_s$ (as explained in the proof of Theorem 1), starting with an end-sequent $\Gamma_g \vdash \Delta_g$ in which all formulae are replaced by pairwise distinct variables of type `bool`. This yields a unification problem \mathcal{P} stating constraints on these variables ensuring that the tree is an LK -proof.
2. Check that there exists an (admissible) solution σ of \mathcal{P} such that $(\Gamma_g \vdash \Delta_g)\sigma \subseteq \Gamma_t \vdash \Delta_t$, i.e. that the unification problem $\mathcal{P} \wedge (\Gamma_g \cup x) \doteq \Gamma_t \wedge (\Delta_g \cup y) \doteq \Delta_t$ has a solution (up to the AC-properties of the operator \cup and of the logical connectives \vee and \wedge).

Second-order unification is undecidable in general [19], but it is clear that the class of unification problems that we obtain by applying the rules above are of a very restricted form. We forget the cut and equality rules for the time being. Since the target formula contains no free variable², the condition $(\Gamma_g \vdash \Delta_g)\sigma \subseteq \Gamma_t \vdash \Delta_t$ is a matching problem (modulo AC). Furthermore, by inspection of the constraints of the inference rules, it is easy to check that the value of a second-order variable occurring in a given sequent only depends on the variables occurring in the *descendant* of this sequent. Consequently, once the values of the variables of the end-sequent are fixed, all second-order variables can be eliminated, and we are left with a first-order unification problem. Consequently, the problem can easily be handled by using existing algorithms for solving second-order matching problems in presence of additional equational axioms such as commutativity or associativity [13, 12], and usual first-order unification algorithms [27]. Solving this problem is essentially equivalent to a proof verification problem: it consists in testing, given an end-sequent $\Gamma \vdash \Delta$ and a proof π (represented by a generalized proof tree in LK^g), whether π is a proof of $\Gamma \vdash \Delta$. Things become more complicated when the cut or equality rules are considered, because in this case the second-order variables cannot be eliminated so easily.

Example 2. We consider the generalized proof tree of Example 2 and the following target sequent:

$$q(a) \vee p(b, b), \forall x p(x, x) \Rightarrow q(x) \vdash \exists x q(x)$$

First, the sequent is matched against the end-sequent of the source proof tree. This yields the following solution:

$$\{u_1 \mapsto q(a) \vee p(b, b), u_2 \mapsto \forall x p(x, x) \Rightarrow q(x), u_3 \mapsto \exists x q(x)\}$$

Then we have to check whether this substitution can be extended in order to satisfy all the constraints of the proof tree. First, the variables u_{11} , u_{12} and u_{13} are replaced by the terms $u_{10}(v_3)$, $u_6(v_1)$ and $u_9(v_2)$, respectively (using the equations $u_{11} \doteq u_{10}(v_3)$, $u_{12} \doteq u_6(v_1)$ and $u_{13} \doteq u_9(v_2)$ occurring in the proof tree). By decomposition, the constraints $u_1 \doteq u_4 \vee u_5$, $u_2 \doteq \forall u_6$, $u_3 \doteq \exists u_9$ and $u_3 \doteq \exists u_{10}$ yield the mapping:

$$\{u_4 \mapsto q(a), u_5 \mapsto p(b, b), u_6 \mapsto \lambda x p(x, x) \Rightarrow q(x), u_{10} \mapsto \lambda x q(x), u_9 \mapsto \lambda x q(x)\}$$

Of course, since the constraints are solved modulo commutativity, the solution $\{u_4 \mapsto p(b, b), u_5 \mapsto q(a)\}$ should also be considered, leading to an eventual failure. By replacing the variables u_6 , u_9 and u_{10} by their values and applying the β -reduction rule, the equations $u_6(v_1) \doteq (u_7 \Rightarrow u_8)$ and $u_8 \doteq u_9(v_2)$ can be reduced to $(p(v_1, v_1) \Rightarrow q(v_1)) \doteq (u_7 \Rightarrow u_8)$ and $u_8 \doteq q(v_2)$. By decomposition, the first formula yields the solution:

$$\{u_7 \mapsto p(v_1, v_1), u_8 \mapsto q(v_1)\}$$

² Remember that quantification is denoted by function application, for instance the formula $\forall x p(x)$ is represented by $\forall \lambda x p(x)$

The second formula is reduced by replacement to $q(v_1) \doteq q(v_2)$ which yields the mapping $\{v_2 \mapsto v_1\}$. Note that in contrast to the previous variables, the value of u_7, u_8 are not fixed at this point: they depend on the first-order variable v_1 .

The remaining equations are $u_4 \doteq u_{10}(v_3)$ and $u_5 \doteq u_7$. By replacement and β -normalization, they are equivalent to $q(a) \doteq q(v_3)$, $p(b, b) \doteq p(v_1, v_1)$, which produces the mapping: $\{v_3 \mapsto a, v_1 \mapsto b\}$.

3.3 Partial Analogy

In many cases, no exact solution of the matching problem will exist, due to the fact that the target problem is not a mere instance of the generalized source problem. In this case, it may be necessary to adapt the proof of the source theorem in order to make it match the target formula. This adaptation may consist in adding some proof steps or removing others, or even to transform some parts of the proof completely. This is typically the case when some hypotheses of a source theorem are not explicitly available in the target problem, but are *provable* from other axioms. Consider for instance the following sequent:

$$r(a) \vee p(b, b), \forall x p(x, x) \Rightarrow q(x), \forall x r(x) \Rightarrow q(x) \vdash \exists x q(x)$$

Any attempt to match this sequent against the generalized sequent of Example 1 would fail, since we would eventually obtain the mappings $u_4 \mapsto r(a)$ and $u_9 \mapsto \lambda x q(x)$, which entails that the constraints $u_4 \doteq u_{11}$ and $u_{11} \doteq u_{10}(v_3)$ would have no solution, meaning that the sequent $u_4 \vdash u_{11}$ (i.e. $r(a) \vdash q(a)$) is not an axiom in LK . However, it is clear that the formula $q(a)$ can be inferred from $r(a)$, due to the additional hypothesis $\forall x r(x) \Rightarrow q(x)$ occurring in the antecedent of the end-sequent. Thus the axiom $r(a) \vdash q(a)$ can be replaced by the proof tree:

$$\frac{\overline{r(a) \vdash r(a)} \quad \overline{q(a) \vdash q(a)}}{\overline{r(a), r(a) \Rightarrow q(a) \vdash q(a)}} \quad \frac{}{r(a), \forall x r(x) \Rightarrow q(x) \vdash q(a)}$$

The constraint $u_4 \doteq u_{11}$ is then irrelevant (since the corresponding subtree has been removed) hence a partial solution of the matching problem has been computed.

In general, the matching process will thus produce a partial solution σ , together with a finite set of sequents which still have to be proven in order to check that the analogy is indeed correct. This can be implemented by ignoring the equations leading to failure (due to an application of the Clash or Occur Check rule) when solving the matching problem. Since failures are ignored, a substitution will always eventually be produced, satisfying some of the constraints at hand; and the sequents whose constraints cannot be satisfied are exactly those that will have to be proven later on.

Another related case in which a partial mapping is relevant is the case in which some of the hypotheses of the target theorem are actually stronger than

those of the source theorem, making some part of the proof tree useless. Consider the following example:

$$q(a) \vee q(b) \vdash \exists x q(x)$$

Obviously, this sequent does not match the generalized proof tree of Example 1. Still, the partial solution $\{u_4 \mapsto q(a), u_5 \mapsto q(b), u_3 \mapsto \lambda x q(x), v_3 \mapsto a\}$ can be generated. The left part of the proof tree can be reconstructed, but the constraints $u_2 \doteq \forall u_6$ cannot be satisfied. However the target sequent corresponding to this equation is $q(b) \vdash \exists x q(x)$. This sequent is actually an instance of a subsequent $[u_8 \vdash u_3 \mid u_3 \doteq \exists u_9 \wedge u_{13} \doteq u_9(v_2)]$ occurring at a deeper level in the proof tree. In this case, the obtained solution corresponds to a proof tree obtained from the source tree by removing the proof steps corresponding to the handling of $\forall x p(x) \Rightarrow q(x)$.

Formally, the process of partial analogy discovery can be viewed as a second-order proof problem: one has to find, given a sequent $\Gamma \vdash \Delta$, a second-order substitution σ such that $\Gamma\sigma \vdash \Delta\sigma$ is provable. This problem can be solved by applying the LK^g rules “in reverse”. Each time an inference is applied, the necessary constraints are added to the current unification problem in order to make this rule applicable. Derivations leading to constraints with no solutions are to be dismissed. Furthermore, this process can be interleaved with standard second-order matching, i.e., one can try to match each of the subsequent generated during the search for a proof against the subsequents occurring in the source proof tree. Of course, this yields a huge search space, and such a procedure is very unlikely to be practically successful.

4 Practical Algorithms

The partial analogy relation defined in the previous section is a very general one, and any direct attempt to implement the previous definitions and informal remarks into a concrete calculus would yield a very flexible procedure, with a huge search space. There is an obvious risk of *over-generalization*: due to the generality and flexibility of the procedure, failures will be very seldom, and in general a partial solution will be computed, together with a set of sequents which will have to be proven afterwards. However, proving (or disproving) these sequents could actually turn out to be much more difficult than trying to prove the original statement. Such a general procedure would be relevant only if strong human guidance is available, in interactive theorem proving. To devise *automated* methods, it is thus necessary to drastically reduce the search space and to add constraints in order to control the way the generalization and matching operations are performed. In this section, we review the most successful approaches. We do not present the proposed procedures in detail, but we explain their relevant features w.r.t. our previous framework and we provide examples of applications.

4.1 Earlier Approaches

To the best of our knowledge, the work presented in [25] is the first published method for handling analogy in Automated Deduction. It describes a system,

named ZORBA, that aims at detecting analogies between a newly presented statement and a previously proven theorem, in order to select an appropriate set of axioms. Such axioms are used to construct the proof of the target theorem, which allows one to reduce the number of available hypotheses and thus of irrelevant inferences. The system does not try to reuse the *proof* of the source theorem, but only uses it to identify the *relevant axioms*. Theorems and axioms are represented as (Horn) clauses and the used proof procedure is the resolution method [34, 26]. The general idea consists in trying to construct a pairing between the set of symbols of the source and target theorems. This can be viewed as a very restricted way of solving the matching problem in Section 3.2, by keeping the structure of the original statement and by solving second-order equalities by decomposition, exactly as first-order ones (i.e., $f(t_1, \dots, t_n) = g(s_1, \dots, s_m)$ is decomposed into $f = g \wedge n = m \wedge \bigwedge_{i=1}^n t_i = s_i$). ZORBA uses additional semantic information about the considered input clauses in order to restrict the search for analogies: every predicate symbol is mapped to a *semantic template* which specifies the type of the arguments, and the mapping between the two statements must preserve this typing. For example, the semantic template for the predicate “group” indicates that the first argument is a set and the second argument is an operator; when trying to match another structure, e.g., “ring”, the system will not try to associate sets with operators. In our context, semantic templates can be viewed as additional type information which is shared between the source and target theorem, and thus which would not be considered for generalization (i.e., the considered type constants are not replaced by variables). Although this assumption restricts the search space and is relevant in many applications, it can dismiss some potentially interesting and unexpected similarities.

The work presented in [29] uses similar ideas, but applies them to construct the whole proof of the target statement by reusing the proof of the source theorem, while replacing the symbols and axioms by their images in the target theorem.

These techniques have been extended in [31], in which a much more flexible and powerful matching algorithm is presented. Flexible, partial mappings are allowed, taking into account properties such as commutativity, and possibly leaving some parts of the target theorem unmatched (for instance $p(x) \wedge p(y) \wedge x \neq y$ can be paired to $p(x) \wedge p(y)$). When applying the decomposition rules, the system tries to delay the application of the rules that have the greatest branching factor, in the hope that they will be dismissed when solving other constraints. Heuristics are presented to evaluate and compare the different pairings that can be computed from two statements, taking into account, for instance, the number of paired symbols, the number of inconsistencies or the number of constant symbols.

4.2 Analogy in Inductive Theorem Proving

An approach is presented in [35] for reusing proofs in inductive theorem proving based on second-order matching modulo evaluation. The considered statements are of the form $\mathcal{A} \vdash_{ind} C$, where C is a Horn equational clause and \mathcal{A} is a

set of equations, and where \vdash_{ind} denotes the inductive entailment relation, i.e., $\mathcal{A} \vdash_{ind} C$ iff any instance of C is a logical consequence of \mathcal{A} . Rewriting [1] is used for deciding the validity of equations, together with explicit induction schemes.

Given a set of equations $\mathcal{A} = \{t_i \simeq s_i \mid i \in [1, n]\}$, it is well known that it is possible to test whether $\mathcal{A} \models u \simeq v$ by normalizing the terms u and v according to the rewrite system $\{t_i \rightarrow s_i \mid i \in [1, n]\}$, if this system is convergent, i.e., confluent and terminating. This, in general, requires to exhibit a reduction ordering \succ such that $\forall i \in [1, n], t_i \succ s_i$. In our setting, rewriting can be of course simulated by a derived inference rule consisting in first applying the $\forall\text{-L}$ rule on the axioms and second applying the paramodulation rule from the generated equations into the conclusion of the sequent. The procedure presented in [35] aims at proving inductive properties of functions defined by induction on some given data-structure. It thus assumes that the considered axioms can be oriented in such a way that ground equations can be solved by rewriting.

The generalization phase is performed by replacing each symbol occurrence by second-order variables as explained in Section 3.1. This is done by indexing each symbol by different superscripts. Sort constants are also replaced by sort variables, thus allowing to discover similarities between objects of different types. The obtained second-order formula is called a *schematic formula*, and the corresponding proofs are called *proof shells*. The soundness of the generalization is ensured by collecting all the unification conditions, i.e. the set of equations of the form $f^i(\dots) \simeq f^j(\dots)$ that occurs in the constraints corresponding to the proof, as defined in Section 3.1. Such a pair is called a *collision set*. Two distinct approaches are proposed to handle such conditions. The first one is syntactic: it consists in simply identifying the two considered occurrences f^i and f^j . An equivalence relation \sim among occurrences of the same symbols is thus computed, and all the occurrences in the same equivalence class are mapped to the same function variable. Again, this technique can be viewed as a restricted form of second-order unification. During this computation, the axioms can be duplicated, for instance a statement: $f(a) \simeq a \vdash f(f(a)) \simeq a$ yields the generalized sequent: $F_1(A_1) \simeq A_2, F_2(A_2) \simeq A_3 \vdash F_2(F_1(A_1)) \simeq A_3$, where F_1, F_2, A_1, A_2, A_3 are new variables. This technique yields a stronger generalization since different occurrences of the same axioms can now be instantiated differently, but has the drawback of increasing the number of hypotheses. The second approach is semantic and consists in asserting the condition $\forall x_1, \dots, x_n f(x_1, \dots, x_n) \simeq g(x_1, \dots, x_n)$ as an additional axiom. This obviously increases the generality of the obtained assertion, but also increases the number of hypotheses, which, as mentioned in [35], makes the matching more difficult. Semantic generalization is applied in a systematic way on the functions occurring in the conclusion of the sequent, whereas syntactic generalization is performed on those only occurring in the axioms.

An important feature of the approach in [35] is that the matching is not done in a purely syntactic way. The conclusion of the target statement is first matched syntactically against the source statement; then the corresponding substitution is applied to the antecedent of the source theorem, thus yielding a set of *mixed*

formulæ, containing both function symbols and function variables. Instead of trying to make these instantiated axioms correspond to hypotheses of the target theorem, a partial evaluation technique is used to evaluate the first-order terms they contain, by applying the target axioms to reduce these terms to normal forms. The resulting formulæ are then solved syntactically. Matching modulo evaluation can be viewed as an elegant and efficient way of solving the partial analogy problem presented in Section 3.3, interleaving inferences (here rewrite steps) with higher-order matching. Of course such an approach is possible only because the considered proof procedure (based on rewriting) is purely deterministic (i.e. that a unique normal form exists for each term, otherwise the procedure would yield a huge search space).

Heuristics are proposed to rate and select the matchers. For instance, a matcher mapping all variables to function symbols is considered simpler than – and thus preferred to – those associating variables with more complex lambda expressions. A hierarchy of matchers of increasing complexity is presented, and those belonging to the lowest complexity classes are promoted.

Example 3. Consider the following theorem (adapted from [35]):

$$\text{sum}(x) + \text{sum}(y) = \text{sum}(\text{append}(x, y))$$

The term $\text{sum}(x)$ denotes the sum of the elements in the list x , and $\text{append}(x, y)$ denotes the concatenation of the lists x and y , defined by the following axioms.

$$\text{sum}(\text{nil}) = 0 \tag{1}$$

$$\text{sum}(\text{cons}(u, x)) = u + \text{sum}(x) \tag{2}$$

$$\text{append}(\text{nil}, y) = y \tag{4}$$

$$\text{append}(\text{cons}(u, x), y) = \text{cons}(u, \text{append}(x, y)) \tag{5}$$

We will focus on the proof of the inductive step, namely:

$$\begin{aligned} \text{sum}(x) + \text{sum}(y) &= \text{sum}(\text{append}(x, y)) \vdash \\ \text{sum}(\text{cons}(u, x)) + \text{sum}(y) &= \text{sum}(\text{append}(\text{cons}(u, x), y)) \end{aligned}$$

The proof is constructed by rewriting as follows.

$$\begin{aligned} \text{sum}(\text{cons}(u, x)) + \text{sum}(y) &= \text{sum}(\text{append}(\text{cons}(u, x), y)) \\ &\equiv \text{sum}(\text{cons}(u, x)) + \text{sum}(y) = \text{sum}(\text{cons}(u, \text{append}(x, y))) \tag{5} \\ &\equiv \text{sum}(\text{cons}(u, x)) + \text{sum}(y) = u + \text{sum}(\text{append}(x, y)) \tag{2} \\ &\equiv (u + \text{sum}(x)) + \text{sum}(y) = u + \text{sum}(\text{append}(x, y)) \tag{2} \\ &\equiv (u + (\text{sum}(x))) + \text{sum}(y) = u + (\text{sum}(x) + \text{sum}(y)) \tag{Hyp} \\ &\equiv u + (\text{sum}(x) + \text{sum}(y)) = u + (\text{sum}(x) + \text{sum}(y)) \tag{Assoc. of +} \\ &\equiv \text{true} \tag{Refl.} \end{aligned}$$

The statement is generalized by introducing second-order variables and collecting all the necessary unification conditions as explained before, yielding the following statement:

$$P^1(S^1(x), S^2(y)) = S^3(A^1(x, y)) \vdash P^1(S^1(C^1(u, x)), S^2(y)) = S^3(A^1(C^1(u, x), y))$$

together with the 4 following axioms (notice axiom 2 is used twice, yielding two different axioms in the generalized form).

$$\begin{aligned} A^1(C^1(u, x), y) &= C^2(u, A^1(x, y)) && \text{(gen. 5)} \\ S^3(C^2(u, x), y) &= P^2(u, S^3(x, y)) && \text{(gen. 2)} \\ S^1(C^1(u, x), y) &= P^3(u, S^1(x, y)) && \text{(gen. 2')} \\ P^1(P^3(x, y), z) &= P^2(x, P^1(y, z)) && \text{(gen. Assoc)} \end{aligned}$$

Note that some of the symbols occurring in the axioms do not occur in the initial statement. Now consider the following conjecture (where $\text{length}(x)$ denotes as usual the number of elements of a list x).

$$\begin{aligned} \text{length}(x) + \text{length}(y) &= \text{length}(\text{append}(x, y)) \vdash \\ \text{length}(\text{cons}(u, x)) + \text{length}(y) &= \text{length}(\text{append}(\text{cons}(u, x), y)) \end{aligned}$$

The syntactic second-order matching yields the solution: $\{S^1, S^2, S^3 \mapsto \text{length}, A^1 \mapsto \text{append}, P^1 \mapsto +, C^1 \mapsto \text{cons}\}$ Note that the symbols C^2, P^2, P^3 are not instantiated at this point. This mapping is applied to the above generalized axioms, yielding:

$$\begin{aligned} \text{append}(\text{cons}(u, x), y) &= C^2(u, \text{append}(x, y)) \\ \text{length}(C^2(u, x), y) &= u + \text{length}(x, y) \\ \text{length}(\text{cons}(u, x), y) &= u + \text{length}(x, y) \\ P^3(x, y) + z &= P^2(x, y + z) \end{aligned}$$

Before trying to solve these equations syntactically, the system tries to evaluate the terms occurring in them by applying the axioms of the target theorem. Here, for instance the term $\text{append}(\text{cons}(u, x), y)$ is reduced to $\text{cons}(u, \text{append}(x, y))$. Then, the matching of $\text{cons}(u, \text{append}(x, y))$ with $C^2(u, \text{append}(x, y))$ yields the solution $C^2 \mapsto \text{cons}$.

The authors describe the system Plagiator, together with several examples of theorems successfully handled by proof reuse.

4.3 Analogy-Driven Proof Plan Construction

In [28] techniques are presented for analogy-driven proof plan construction in inductive theorem proving. The presented method stands at a slightly different level than the previous approaches: it focuses on the construction of *proof plans*, instead of proofs. A proof plan is a sequence of methods (or tactics), each method can be viewed as a high-level inference rule, defined by a set of sequents (the premisses) and an end-sequent (the conclusion), together with a set of constraints encoding the conditions that make the application of the method possible. The execution of a proof plan possibly requires calling an external prover to check that some proof step is feasible. As the one described in the previous section, the procedure handles statements of the form $\mathcal{A} \vdash_{\text{ind}} C$, where C is an equation and \mathcal{A} a set of axioms, using rewriting and explicit induction schemes.

It first attempts to find a second-order mapping from the source theorem and rules to the target theorem and rules. This is done, as in previous approaches, by replacing all function occurrences by new second-order variables, and by collecting all the equality relations between these variables that make the proof applicable. Such relations are called *C-equations*. Once the pairing is computed, the system tries to replay the source plan methods node by node if this is possible. The main originality of the proposed approach lies in its flexibility and handling of failures. If the constraints of the method corresponding to a given node does not hold in the target, then the system tries to analyze these failure conditions in order to reformulate the proof plan before replay, by inserting, deleting or modifying proof nodes. Various transformations are proposed on proof plan in order to try to recover from a failure. Each of these transformations are heuristically triggered by specific conditions. For instance, if a function symbol is eliminated by mapping it to the identity, then the rewrite step which ought to be applied on this symbol will be deleted. Similarly, if a function is duplicated, i.e. if f is mapped to $\lambda x g(g(x))$, then the rewrite rule will be applied twice on the two occurrences of g (this is called the 1to2 transformation). Finally, if this reformulation does not work, then the system tries to speculate a lemma.

Example 4. (adapted from [28]) Assume that the system tries to prove the statement: $\text{half}(x + y) = \text{half}(y + x)$ (where half denotes the function $x \mapsto \frac{x}{2}$ and x, y are even numbers) by analogy with the theorem $\text{length}(\text{append}(x, y)) = \text{length}(\text{append}(y, x))$ (where x, y are lists and length and append are defined as usual).

The axioms of the source theorem are the following.

$$\begin{aligned} \text{append}(\text{cons}(u, x), y) &= \text{cons}(u, \text{append}(x, y)) & (1) \\ \text{length}(\text{cons}(u, x)) &= \text{succ}(\text{length}(x)) & (2) \\ \text{length}(\text{append}(x, \text{cons}(u, y))) &= \text{succ}(\text{length}(\text{append}(x, y))) & (3) \end{aligned}$$

The inductive case of the source theorem can be proven by rewriting, as follows:

$$\begin{aligned} \text{length}(\text{append}(\text{cons}(u, x), y)) &= \text{length}(\text{append}(y, \text{cons}(u, x))) \\ \equiv \text{length}(\text{cons}(u, \text{append}(x, y))) &= \text{length}(\text{append}(y, \text{cons}(u, x))) & (1) \\ \equiv \text{succ}(\text{length}(\text{append}(x, y))) &= \text{length}(\text{append}(y, \text{cons}(u, x))) & (2) \\ \equiv \text{succ}(\text{length}(\text{append}(x, y))) &= \text{succ}(\text{length}(\text{append}(y, x))) & (3) \\ \equiv \text{succ}(\text{length}(\text{append}(y, x))) &= \text{succ}(\text{length}(\text{append}(y, x))) & (\text{Hyp.}) \\ \equiv \text{true} & & (\text{Refl.}) \end{aligned}$$

The inductive case of the target theorem can be written as follows.

$$\text{half}(x + y) = \text{half}(y + x) \vdash \text{half}(\text{succ}(\text{succ}(x)) + y) = \text{half}(y + \text{succ}(\text{succ}(x)))$$

The axioms of the target theorem are given below.

$$\begin{aligned} \text{succ}(x) + y &= \text{succ}(x + y) & (1') \\ \text{half}(\text{succ}(\text{succ}(x))) &= \text{succ}(\text{half}(x)) & (2') \\ \text{half}(x + \text{succ}(\text{succ}(y))) &= \text{succ}(\text{half}(x + y)) & (3') \end{aligned}$$

The matching between the conclusions of the two theorems yields the following mapping (for conciseness we omit the generalization phase and simply regard the function symbols of the source statement as variables³).

$$\{length \mapsto half, append \mapsto +, cons \mapsto \lambda x, y succ(succ(y))\}$$

Then the first proof step can be carried out, by replacing the axioms of the source theorem by the corresponding ones in the target.

$$\begin{aligned} half(succ(succ(x)) + y) &= half(y + succ(succ(x))) \\ &\equiv half(succ(succ(x) + y)) = half(y + succ(succ(x))) \end{aligned}$$

The second proof step cannot be replicated because the term $half(succ(succ(x) + y))$ is not an instance of $half(succ(succ(x)))$. More precisely, this is due to the fact that the occurrence $cons^1$ of $cons$ in the conclusion of the source theorem is mapped to $\lambda x, y succ(succ(y))$, whereas the other symbol occurrence $cons^2$ in the axiom is mapped to $\lambda x, y succ(y)$ (so that the Axiom 2' becomes an instance of Axiom 2). In this solution, we have $cons^1 = cons^2.cons^2$, which heuristically triggers the 1to2 transformation. This transformation consists in applying twice the axiom corresponding to the second occurrence of $cons^2$. This yields the equation $half(succ(succ(x + y))) = half(y + succ(succ(x)))$, on which the axiom 2' can be applied. The rest of the proof can be carried out without modification.

The presented procedure is able to handle the construction of induction schemes. Heuristics are presented to evaluate the different mappings, which again, tries to delay, or if possible to avoid, decisions with high branching factors. For instance, the application of projection rules during second-order matching is discouraged whereas functions mapping variables to function symbols (i.e. of the form $x \mapsto \lambda y f(y)$) are promoted (as in the heuristics used by the Plagiator system).

The approach has been implemented in the system ABALONE, on the top of the proof planner CLAM [8] and many examples of applications are presented.

4.4 Proof Generalization in Resolution-Based Theorem Proving

In [14], an algorithm is introduced to transform a given formula into a more general one, while preserving the proof structure of the initial formula. The method is defined in the context of the resolution calculus, thus the considered formulæ are sets of clauses and the proofs are resolution refutations (i.e. derivations of the empty clause). In this specific context, the goal of the generalization process is to *weaken* as far as possible the clauses occurring in the initial set since they are either hypotheses or negated conclusions. The algorithm is defined as a set of satisfiability-preserving rules, operating on clause sets. These rules are listed and briefly described below.

³ Of course, such symbols would have to be indexed.

- **Creation of New Functional Symbols.** The goal of the first rule is to replace different occurrences of the same symbol f by new pairwise distinct function symbols, if these occurrences are not unified during resolution inferences. This idea is similar to the computation of the relation \sim based on collision sets in [35].
- **Instantiation.** This rule instantiates a variable x by a term of the form $f(x_1, \dots, x_n)$, where x_1, \dots, x_n are fresh pairwise distinct variables, if x is always instantiated by a term unifiable with $f(x_1, \dots, x_n)$ (and at least once by an instance of $f(x_1, \dots, x_n)$) during the proof. Note that all the copies of the clauses occurring in the derivation must be taken into account, as well as implicit occurrences of the variable x in all the descendants of the clause in which it occurs. For instance, consider the following derivation:

| | | |
|---|----------------------------------|-------------|
| 1 | $p(x, y) \vee r(y)$ | |
| 2 | $\neg p(a, u)$ | |
| 3 | $\neg r(f(a)) \vee \neg r(f(b))$ | |
| 4 | $r(u)$ | (res, 1, 2) |
| 5 | $\neg r(f(b))$ | (res, 4, 3) |
| 6 | \square | (res, 4, 5) |

Here the variable x can be replaced by the constant a , since it is never unified with a term distinct from a . The obtained clause is strictly weaker than the initial one (in the sense that it is a logical consequence of it) hence the corresponding theorem is more general. Similarly, u and y can be replaced by $f(y_1)$, since they are unified only with $f(a)$ and $f(b)$, which are both of the form $f(\dots)$. Note that the number of copies of each clause do not increase since each variable is instantiated only once.

- **Variable Elimination.** This rule aims at replacing two variables occurring in the same clause by the same variable, if they are never unified with non-unifiable terms. Again, implicit instantiations of these variables in the descendants of the considered clause must be taken into account. The idea is dual from that of the first rule since in this case the number of symbols decreases. The intuition is that these symbols are *universally* quantified, whereas function symbols implicitly correspond to *existential* quantifiers. Thus, in the former case, instantiation makes the statement more general (it weakens the hypotheses), whereas in the later case it makes it less general.
- **Functional Symbols Elimination.** The systematic application of the previous rules frequently generates derivations with many recurrent patterns. In particular, it may be the case that a function symbol is always applied on an argument having some specific head symbol g . In this case, it is possible to simplify the considered statement by introducing a new symbol f' denoting the application of f to the terms of this specific form; the arguments of f' being the union of those of f and g . Formally, this rule corresponds to the application of the following rewriting rule (applied if argument $i+1$ of f is always of head g):

$$f(x_1, \dots, x_i, g(y_1, \dots, y_m), x_{i+2}, \dots, x_n) \rightarrow f'(x_1, \dots, x_i, y_1, \dots, y_m, x_{i+2}, \dots, x_n)$$

- **Parameter Elimination.** A similar rule can be applied to eliminate redundant parameters. If the arguments i and j (with $i < j$) of a function f are always identical (or at least unifiable), then one of these two terms is redundant, and the following rewrite rule can be applied:

$$f(x_1, \dots, x_n) \rightarrow f'(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$$

Similar ideas can be applied to handle model construction rules [5]. In the latter case, however, since the formula is actually satisfiable (the model being a witness of satisfiability) a more general statement is obtained by *generalizing* the clauses it contains, instead of weakening or instantiating them. Thus the rules have to be applied in a dual way. For instance, different function symbols f and g can be replaced by the same symbol if their interpretations in the constructed model are identical (or at least compatible, if partial models are constructed). Similarly, distinct occurrences of first-order variables can be replaced by distinct new variables when possible. In this case, the constraints that make possible the construction of the model are not unification problems, but rather dis-unification ones [10].

[16] tackles the problem of analogy discovery, namely the comparison between the generalized statements and the target problem. It uses a standard higher-order matching algorithm (modulo AC), but with an original feature: instead of making a comparison between two statements, it uses the whole source derivation and tries to match the clauses of the target problem with some of the clauses occurring in the source refutation, in such a way that a so-called “cut-set” is obtained in the proof tree, namely a minimal set of nodes that is sufficient to derive the empty clause. Formally, a “cut-set” is a set of nodes S in the proof tree such that any path from the leaves (the hypotheses) to the root of the tree (the empty clause) contains a node in S . This matching process can thus relate a source problem to a problem in which some of the clauses that were *derived* in the original statement are *asserted* in the new one.

[15] introduces another, different matching algorithm, which allows for more flexible analogies. The main idea is to compare the two statements modulo the *subsumption* relation. Given two source and target clause sets S_S and S_T , the goal is to find a higher-order substitution σ such that every clause occurring in $S_S\sigma$ is subsumed by some clause in S_T . If such a substitution exists, then it is easy to see that a proof of S_T can be automatically reconstructed from that of S_S . The problem is clearly undecidable in general. Indeed, second-order unification can be reduced to it, simply by considering two formulae of the form $S_S = \{p(t, s)\}$ and $S_S = \{\forall x p(x, x)\}$: then the matching problem has a solution iff t and s are unifiable. The devised (semi-decision) method consists in encoding the considered problem as a unification problem with an additional prefix of nested quantifiers. The nesting of quantifiers is essential to express the dependencies between variables that appear when stating subsumption relations. Indeed, $C\sigma$ subsumes D iff σ is a solution of the equational formula $\forall \mathbf{x} \exists \mathbf{y} \exists z D \doteq (C \vee z)$, where \doteq denotes syntactic equality modulo AC and where \mathbf{x} and \mathbf{y} denote the

vectors of first-order variables occurring in D and C , respectively. Note that the domain of σ contains the function and predicate symbols occurring in C , but *not* the first-order variables of C (which are also quantified in the obtained equational problem). The procedure used to solve the obtained problems mixes standard second-order unification rules with first-order unification (modulo AC) and specific quantifier elimination rules (similar to those in [10]). The matching may be partial, in the sense that the solving process can generate formulae of the form $\phi = \text{false}$, which obviously have no solution. If ϕ is actually refutable (thus interpreting \doteq as a *semantic* equivalence), then the proof of the source statement can still be reconstructed.

4.5 Expressing and Using Analogies via the Curry-Howard Isomorphism

The previous methods are based on the idea of analogy as a yes/no relation between proofs: a new proof may be obtained from another one if and only if the two proofs are analogous. Another philosophy is to try to make distinctions between different kinds of analogies. Analogy may not be considered as a uniform relation between different situations, but rather as a particular correspondence between these situations, so that two given situations may be analogous in more than one way. In the context of theorem proving an analogy is therefore a correspondence between proofs. We may imagine many ways of expressing such correspondences, we may even consider that the previous methods for obtaining new proofs from old ones are specific analogies. But for practical purposes we need to offer a simple and convenient way of expressing proof correspondences.

The method proposed in [6, 7] relies on the Curry-Howard isomorphism which offers a way of representing proofs in a compact form as typed λ -terms. In the Curry-Howard interpretation, propositions are viewed as types whose members are terms which are themselves proofs of the non-emptiness (or truth) of their type. A sequent $x_1 : T_1, \dots, x_n : T_n \vdash t : T$ is interpreted as: under the assumption that x_i are variables of type T_i a member t of type T can be constructed. This interpretation works as a semantics in intuitionist logic, but it can be extended to other logics if we stick to a purely syntactic notion of isomorphism between proofs and terms, which is enough here. The rules for typing terms can be read as inference rules, for instance, in the NuPRL system [11] the rules

$$\frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B}{\Gamma \vdash \text{pair}(a, b) : A \wedge B} \quad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x t : A \Rightarrow B}$$

are called the *and-i* (*i* is short for *introduction*) and *imp-i* rules. As long as it is possible to construct a proof of $\vdash t : T$ by following the term t (i.e., if type-checking is decidable), we can consider t as a representation of a proof of $\vdash t : T$, or simply of the proposition T .

Correspondences between proofs can then be expressed as correspondences between λ -terms, which in [6] are formalized as single second order rewriting rules. A rule $lhs \rightarrow rhs$ is applied only once, at the top-level of a proof term t :

second order pattern-matching yields a finite set $S = \{\sigma \mid \text{lhs } \sigma \doteq t\}$ of solutions some of which are then applied to rhs . The resulting proof terms are then type-checked against the new conjecture; if one term passes the test, a proof of the conjecture has been found.

In a rewriting rule first-order variables are used to hold (proof terms of) subproofs while second-order variables may hold sub-derivations of the source proof. For example one can write

$$h(\text{pair}(x, y)) \rightarrow h(\text{pair}(\text{pair}(x, y), a))$$

where h is a second-order variable, x and y are first-order variables and a is the label of a new hypothesis, here considered as a constant. The rule replaces occurrences of a term $\text{pair}(x, y)$ by $\text{pair}(\text{pair}(x, y), a)$ and the variable h denotes in some sense the context in which these replacements occur. The variable h may be matched to a huge part of the proof t , hence this kind of rule may formalize subtle modifications in a rather concise way. However, the solution $h \doteq \lambda z t$, where z is a variable that does not occur in t , necessarily belongs to S but is clearly not relevant in this context: the term $\text{pair}(x, y)$ has to be matched against a subterm of t , thus giving values to x and y . It is therefore necessary to impose some limitations on the elements of S , and keep only those where:

1. every free variable of lhs is instantiated,
2. all these instantiations are closed (the extracted values should not depend on arbitrary variables),
3. a free higher-order variable h of arity n in lhs must be instantiated by a term $\lambda x_1 \dots x_n u$ where each x_i occurs in u .

This filter dramatically reduces the number of candidate proof terms. Another possibility is to put in rhs free variables that do not occur in lhs . The candidate proof terms may then contain free variables and type-checking should be extended to search possible values for them. This is clearly not decidable but offers much flexibility in writing transformation rules by using variables to be filled by a theorem prover.

4.6 Reusing Derivations in Proof Assistants

Techniques closely related to the previous ones have been developed in interactive proof assistants in order to *reuse* partial proofs. Developing proofs in systems like Coq or Isabelle requires a lot of input from the user since the process of proof refinement by tactics only provides a restricted and constrained form of automation. This input is of course error prone and it can be very time consuming to backtrack from an almost completed proof only for a slight mistake. This situation can also occur in proving properties of programs, when a bug is detected; the correction proof has to be endeavored from the start once the bug is corrected. Most proof assistants keep a record of the user's input and offer the possibility to "replay" a sequence of tactics (or *tactic proof*) on a different

conjecture. However, a tactic proof as input by the user is aimed at a particular conjecture and has no claim to generality; it is therefore very sensitive to changes in the conjecture which often leads to failure. Reusing a derivation in a different context thus involves some amount of generalization and modification to this derivation. This is clearly a form of analogy between derivations, though a restricted one since the conclusions of the derivations are supposed to be rather close.

In [18] the authors propose to link each inference step of a derivation to a *justification* which is basically a schema containing free variables together with the tactic that has produced the step. For instance, the tactic *and_i* (corresponding to the rule $\wedge\text{-}R$ of the sequent calculus depicted in Figure 1) on the conjecture $p, q \vdash p \wedge q$ produces two new conjectures:

$$\frac{p, q \vdash p \quad p, q \vdash q}{p, q \vdash p \wedge q}$$

and to this inference step is attached a justification containing *and_i* (which is a λ -term) and the schema $(H \vdash A \wedge B, (H \vdash A, H \vdash B))$, where H, A, B are variables. The schema is also represented by a λ -term, as are all objects in [18] since this proof reuse algorithm is implemented in λ -prolog. This allows to use the interpreter's higher order unification in a transparent way.

Of course, a tactic may not return a one-step derivation and the problem is then to find a way of computing a schema justifying a derivation. This schema has to be as general as possible since proof reuse is performed by instantiating these schemas. Assuming that all inference rules of the object logic can be represented by step schemas, this problem can be transformed into a unification problem, as we have seen in Section 3.1: each time the conclusion of an inference is used as a premiss of another, an equation relating the corresponding schemas is produced. A unifier is computed and then applied to the schemas corresponding to the conclusion and the premisses of the derivation. The result is the schema justifying the whole derivation.

For example, consider the schema $(H, A \wedge B \vdash C, (H, A \wedge B, A, B \vdash C))$ corresponding to a rule *and_e* of the object logic (corresponding to the rule $\wedge\text{-}L$ in Figure 1), and a tactic *and_e*_tac* that repeatedly applies this rule. On a given conjecture $p \wedge q \wedge r \vdash \phi$ the tactic *and_e*_tac* applies the rule *and_e* twice, which corresponds to the following derivation:

$$\frac{\frac{p \wedge q \wedge r, p, q \wedge r, q, r \vdash \phi}{p \wedge q \wedge r, p, q \wedge r \vdash \phi}}{p \wedge q \wedge r \vdash \phi}$$

We now identify the premiss schema of the bottom inference, say $H, A \wedge B, A, B \vdash C$, with the conclusion schema of the top one, say $H', A' \wedge B' \vdash C'$ (with renamed variables), which yields, e.g., the solution $\{B \doteq A' \wedge B', H' \doteq (H, A \wedge B, A, B), C \doteq C'\}$. The corresponding unifier is then applied to the conclusion schema of the bottom inference and the premiss schema of the top one,

which yields the schema $(H, A \wedge A' \wedge B' \vdash C, (H, A \wedge A' \wedge B', A, A' \wedge B', A', B' \vdash C))$ justifying the derivation

$$\frac{p \wedge q \wedge r, p, q \wedge r, q, r \vdash \phi}{p \wedge q \wedge r \vdash \phi}$$

(together with the tactic *and_e*_tac*). Only this shorter derivation is kept in memory (with its justification) and shown to the user.

Reusing a derivation is very similar to this problem: a unifier is computed in the same way by adding an equation relating the schema of the conclusion of the proof to the new conjecture, as explained in Section 3.2. If there is a unifier then it can be applied to all the schemas in the proof, which yields a proof of the new conjecture.

Following our example, we can reuse our derivation on the new conjecture $p \wedge q \wedge r \wedge s \vdash \phi$ by identifying it to the conclusion schema $H, A \wedge A' \wedge B' \vdash C$ and applying the corresponding unifier to the premiss schema, which yields the new derivation

$$\frac{p \wedge q \wedge r \wedge s, p, q \wedge r \wedge s, q, r \wedge s \vdash \phi}{p \wedge q \wedge r \vdash \phi}$$

with the same justification. The tactic has not been replayed, and if it were it would yield a premiss different than this one.

This way of generalizing proofs is of course very similar to what can be performed by using LK^g . A difference is that in LK^g all constraints are gathered as one unification problem, whereas in [18] unification is performed at each application of a tactic. This is not equivalent since most general unifiers do not exist in higher order unification; in this reuse method a unifier is found at each step by λ -prolog which is therefore implementation dependent.

Other proof reuse methods have been developed for specific proof assistants. For instance in [24] an implementation uses the proof terms automatically constructed in the system Isabelle [30]. As explained in Section 4.5 these proof terms provide a compact representation of derivations, stripped of many redundant details. Since the type system of Isabelle allows type variables, proof terms can be transformed in a straightforward way so that they contain no constant symbol and all variables are abstracted. This proof term can then be type-checked with a new conjecture (a new type) in order to obtain a new proof. For example, starting from a proof of $\forall l, l @ [] \simeq l$ in an inductive theory of lists (where $@$ is the append function), abstracting the variables in the proof term yields a proof term whose type is

$$\begin{aligned} & \forall P [P(nil) \wedge \forall l (P(l) \Rightarrow \forall x P(cons(x, l)))] \Rightarrow \forall l P(l) \\ & \wedge \forall x, append(nil, x) \simeq x \\ & \wedge \forall x, y, z, append(cons(x, y), z) \simeq cons(x, append(y, z)) \\ & \Rightarrow \forall x, append(x, nil) \simeq x \end{aligned}$$

where the constructors of the theory of lists have been replaced by new variables *nil*, *cons* and *append* (we give them convenient names for sake of readability).

This new proof term can be instantiated by applying the substitution $\{nil \doteq 0, cons \doteq \lambda x \lambda y s(y), append \doteq \lambda x \lambda y y + x\}$. This yields a proof of

$$\begin{aligned} & \forall P [P(0) \wedge \forall l (P(l) \Rightarrow P(s(l)))] \Rightarrow \forall l P(l) \\ & \wedge \forall x, x + 0 \simeq x \\ & \wedge \forall y, z, z + s(y) \simeq s(z + y) \\ & \Rightarrow \forall x, 0 + x \simeq x \end{aligned}$$

which is a proof of $\forall x, 0 + x \simeq x$ in the inductive theory of Peano integers. Obviously this method cannot be reversed; the target proof could not be generalized into the source proof. This method implements a transformation by signature morphism hence requires to start from the most general signature.

Another method has been proposed in [2] that allows for a more flexible change in data representations. This is required in situations where results from different formalizations of mathematical objects are used. For example, proving some property of an efficient program on natural numbers often needs a binary representation of integers but also theorems that may have been proved only in Peano arithmetic. Is it possible to reuse these theorems and their proofs in a context where integers have a completely different representation? Rather than transforming the proofs [2] propose to extend the type theory of the Coq system with rewriting rules on types and elements that realize isomorphisms between types. With such back and forth computation rules between isomorphic types it is possible to equate different propositions based on different (but isomorphic) types. This preserves membership of proof terms into equated propositions as well as decidability of proof-checking, hence a proof term can be reused without change. Details are out of the scope of a survey on analogy. It should however be noted that silent changes of representations, such as considering natural numbers as finite ordinals or coercing integers into rational numbers, are ubiquitous in mathematics. These can be seen as analogies with very precise justifications since they are supported by mathematical proofs.

5 Conclusion

Analogy can be looked at from a number of different perspectives, such as proof generalization, error recovery from partial proofs, morphisms, type coercions, proof transformations etc. Since analogy-based reasoning can be seen as a kind of meta-reasoning, it is rather natural to see that most existing approaches use higher-order techniques to represent and manipulate proofs and express correspondences between them. These approaches mainly differ by the kinds of correspondences that they are able to deal with and the way such correspondences are computed. The proposed methods range from automatized techniques aiming at handling minor adaptations or slight changes inside existing proofs (e.g., by propagating a necessary correction in a definition), to more powerful generalization techniques which are capable of handling much more general classes of relations between proofs, at the cost of a much larger search space, and therefore require more human guidance. The former approaches are well-suited to proof

reuse, where the same pattern is applied repeatedly to establish strongly related results (with only some slight changes in the argumentation), whereas the latter are capable of expressing and even discovering hidden correspondences between seemingly unrelated proofs.

References

1. F. Baader and T. Nipkow. *Term rewriting and all that*. Cambridge University Press, 1998.
2. G. Barthe and O. Pons. Type isomorphisms and proof reuse in dependent type theory. In *FoSSaCS*, Springer LNCS 2030, pages 57–71, 2001.
3. W. W. Bledsoe. Non-resolution theorem proving. *Artificial Intelligence*, 9:1–35, 1977.
4. C. Bourely, R. Caferra, and N. Peltier. A method for building models automatically. Experiments with an extension of OTTER. In *Proceedings of CADE-12*, Springer LNAI 814, pages 72–86, 1994.
5. C. Bourely, G. Défourneaux, and N. Peltier. Building proofs or counterexamples by analogy in a resolution framework. In *Proceedings of JELIA 96*, Springer LNAI 1126, pages 34–49, 1996.
6. T. Boy de la Tour and R. Caferra. Proof analogy in interactive theorem proving: A method to express and use it via second order matching. In *Proceedings of the Sixth National Conference on Artificial Intelligence AAAI-87*, Morgan Kaufmann, pages 95–99, 1987.
7. T. Boy de la Tour and C. Kreitz. Building proofs by analogy via the Curry-Howard isomorphism. In A. Voronkov, editor, *Proceedings of the Conference on Logic Programming and Automated Reasoning LPAR’92*, Springer Verlag LNAI 624, pages 202–213, 1992.
8. A. Bundy, F. van Harmelen, C. Horn, and A. Smaill. The Oyster-Clam system. In *Proceedings of the 10th International Conference on Automated Deduction*, Springer-Verlag, pages 647–648, 1990.
9. R. Caferra and N. Zabel. A method for simultaneous search for refutations and models by equational constraint solving. *Journal of Symbolic Computation*, 13:613–641, 1992.
10. H. Comon and P. Lescanne. Equational problems and disunification. *Journal of Symbolic Computation*, 7:371–475, 1989.
11. R. Constable. *Implementing Mathematics with the Nuprl Proof Development System*. Prentice-Hall, Englewood Cliffs, New Jersey, 1986.
12. R. Curien. Second order e-matching as a tool for automated theorem proving. In M. Filgueiras and L. Damas, editors, *Progress in Artificial Intelligence*, Springer LNCS 727, pages 242–257, 1993.
13. R. Curien, Z. Qian, and H. Shi. Efficient second-order matching. In H. Ganzinger, editor, *Rewriting Techniques and Applications*, Springer LNCS 1103, pages 317–331, 1996.
14. G. Défourneaux, C. Bourely, and N. Peltier. Semantic generalizations for proving and disproving conjectures by analogy. *Journal of Automated Reasoning*, 20(1 & 2):27–45, 1998.
15. G. Défourneaux and N. Peltier. Analogy and abduction in automated reasoning. In M. E. Pollack, editor, *Proceedings of IJCAI’97*, August 23–29 1997.

16. G. Défourneaux and N. Peltier. Partial matching for analogy discovery in proofs and counter-examples. In W. McCune, editor, *Proceedings of CADE 14*. Springer LNAI 1249, July 1997.
17. G. Dowek, T. Hardin, and C. Kirchner. Theorem proving modulo. *J. Autom. Reason.*, 31(1):33–72, Oct. 2003.
18. A. P. Felty and D. J. Howe. Generalization and reuse of tactic proofs. In F. Pfenning, editor, *LPAR*, Springer LNCS 822, pages 1–15, 1994.
19. W. D. Goldfarb. The undecidability of the second-order unification problem. *Theoretical Computer Science*, 13(2):225 – 230, 1981.
20. Hartshorne, Weiss, and Burks. *Collected Papers of C.S. Peirce (1930–1958)*. Harvard U. Press.
21. G. Hegel. *Elements of the Philosophy of Right*. Cambridge University Press, 1991.
22. S. Hetzl. A sequent calculus with implicit term representation. In *CSL*, Springer, LNCS 6247, pages 351–365, 2010.
23. S. Hetzl, A. Leitsch, and D. Weller. Towards algorithmic cut-introduction. In N. Bjørner and A. Voronkov, editors, *LPAR*, Springer LNCS 7180, pages 228–242, 2012.
24. E. B. Johnsen and C. Lüth. Theorem reuse by proof term transformation. In K. Slind, A. Bunker, and G. Gopalakrishnan, editors, *TPHOLs*, Springer LNCS 3223, pages 152–167, 2004.
25. R. E. Kling. A paradigm for reasoning by analogy. In *Proceedings of the 2nd international joint conference on Artificial intelligence*, IJCAI’71, Morgan Kaufmann, pages 568–585, 1971.
26. A. Leitsch. *The resolution calculus*. Springer. Texts in Theoretical Computer Science, 1997.
27. A. Martelli and U. Montanari. An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems*, 4(2):258–282, 1982.
28. E. Melis and J. Whittle. Analogy in inductive theorem proving. *J. Autom. Reasoning*, 22(2):117–147, 1999.
29. J. Munyer. *Analogy as a mean of discovery in problem-solving and learning*. PhD thesis, Univ. Calif. Santa Cruz, 1981.
30. T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer-Verlag, 2002.
31. S. Owen. *Analogy for Automated Reasoning*. Academic Press, 1990.
32. N. Peltier. A general method for using terms schematizations in automated deduction. In *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR’01)*, Springer LNCS 2083, pages 578–593, 2001.
33. A. Robinson and A. Voronkov, editors. *Handbook of Automated Reasoning*. North-Holland, 2001.
34. J. A. Robinson. A machine-oriented logic based on the resolution principle. *J. Assoc. Comput. Mach.*, 12:23–41, 1965.
35. C. Walther and T. Kolbe. Proving theorems by reuse. *Artif. Intell.*, 116(1-2):17–66, 2000.
36. L. Wos. The problem of reasoning by analogy. *Journal of Automated Reasoning*, 10:421–422, 1993.

Part2: Modeling analogy

Applying belief revision to case-based reasoning

Julien Cojan¹ and Jean Lieber^{2,3,4}

¹ INRIA Sophia-Antipolis, Wimmics Project, France

² Université de Lorraine, LORIA, UMR 7503—54506 Vandœuvre-lès-Nancy, France

³ CNRS—54506 Vandœuvre-lès-Nancy, France

⁴ Inria—54602 Villers-lès-Nancy, France

Julien.Cojan@inria.fr

Jean.Lieber@loria.fr

Abstract. Adaptation is a task of case-based reasoning (CBR) that aims at modifying a case to solve a new problem. Now, belief revision deals also about modifications. This chapter studies how some results about revision can be applied to formalize adaptation and, more widely, CBR. Revision operators based on distances are defined in formalisms frequently used in CBR and applied to define an adaptation operator that takes into account the domain knowledge and the adaptation knowledge. This approach to adaptation is shown to generalize some other approaches to adaptation, such as rule-based adaptation.

1 Introduction

Case-based reasoning and belief revision are two domains in which the notions of similarity and modification play an important role.

Case-based reasoning (CBR [1]) is a reasoning process using a case base, where a case is a representation of a problem-solving episode, in general, in the form of a problem-solution pair. CBR aims at solving a *target problem* and generally consists in a retrieval step (selection of one or several case(s) from the case base that is/are similar to the target problem), an adaptation step (modification of the retrieved case(s) to propose a solution to the target problem), and a possible storage of the case formed by the target problem and its solution.

Belief revision is the process of changing a belief base about a static world by incorporating new beliefs while keeping the belief base consistent. When the old beliefs are inconsistent with the new beliefs, the formers have to be modified in order to restore consistency with the latters. Usually, belief revision is based on the minimal change principle [2]: most of the old beliefs should be kept. One way to measure change (so that it is minimal) is to use a similarity metric (to be maximized) or a distance (to be minimized).

Thus, the question raised is whether the modification performed during CBR could be performed by a belief revision operator. This question has been addressed in several publications and this chapter gives a synthesis of some of them.

The chapter is organized as follows. Some preliminaries about CBR are given in section 2. Section 3 introduces belief revision. In CBR, the modifications are performed during the adaptation step, section 4 is the core of the chapter and describes

revision-based adaptation from a theoretical viewpoint and with a few examples. More globally, belief revision can be applied to CBR as a whole as section 5 shows. Several revision operators have been implemented for different formalisms, together with their revision-based adaptation functions. Some of them are gathered in REVISOR (<http://revisor.loria.fr>), a system that is briefly described in section 6. Finally section 7 concludes the chapter.

2 Preliminaries

2.1 Formalism

The approach to CBR presented in this chapter can be applied to a variety of representation languages. It is assumed that there exists a representation language \mathcal{L} : a formula is an element of \mathcal{L} . The semantics of \mathcal{L} is given by a (possibly infinite) set \mathcal{U} and by a function $\text{Mod} : \varphi \in \mathcal{L} \mapsto \text{Mod}(\varphi) \in 2^{\mathcal{U}}$, defining, in a model-theoretical manner, the semantics of \mathcal{L} : a is a model of φ if $a \in \text{Mod}(\varphi)$; φ_1 entails φ_2 ($\varphi_1 \models \varphi_2$) if $\text{Mod}(\varphi_1) \subseteq \text{Mod}(\varphi_2)$; φ_1 and φ_2 are equivalent ($\varphi_1 \equiv \varphi_2$) if $\text{Mod}(\varphi_1) = \text{Mod}(\varphi_2)$. A subset A of \mathcal{U} is *representable* in \mathcal{L} if there exists a formula φ such that $\text{Mod}(\varphi) = A$.

It is assumed that \mathcal{L} is stable under conjunction, which means that for every $\varphi_1, \varphi_2 \in \mathcal{L}$ there exists a formula denoted by $\varphi_1 \wedge \varphi_2$ such that $\text{Mod}(\varphi_1 \wedge \varphi_2) = \text{Mod}(\varphi_1) \cap \text{Mod}(\varphi_2)$.

Some formalisms are stable under negation (or complement), which means that for every $\varphi \in \mathcal{L}$, there exists a formula denoted by $\neg\varphi$ such that $\text{Mod}(\neg\varphi) = \mathcal{U} \setminus \text{Mod}(\varphi)$. For such formalisms, $\varphi_2 \vee \varphi_2$ is an abbreviation for $\neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\varphi_1 \Rightarrow \varphi_2$ is an abbreviation for $\neg\varphi_1 \vee \varphi_2$ and $\varphi_1 \Leftrightarrow \varphi_2$ is an abbreviation for $(\varphi_1 \Rightarrow \varphi_2) \wedge (\varphi_2 \Rightarrow \varphi_1)$.

Propositional logic with n variables is an example of such a formalism: \mathcal{U} denotes the set of interpretations on the variables. Every $A \subseteq \mathcal{U}$ is representable in this logic.

2.2 Case-based reasoning: principles and notations

For CBR, \mathcal{U} is called the *case universe*. A *case instance* a is, by definition, an element of \mathcal{U} : $a \in \mathcal{U}$. A *case* C is a class of case instances: $C \in 2^{\mathcal{U}}$ (in this chapter, a case represents a class of experiences, it is what is called an *ossified case* in [1] and a *generalized case* in [3]). For instance, when the formalism is propositional logic with n variables, \mathcal{U} is the set of the 2^n interpretations and a case C is represented by a formula φ : $C = \text{Mod}(\varphi)$.

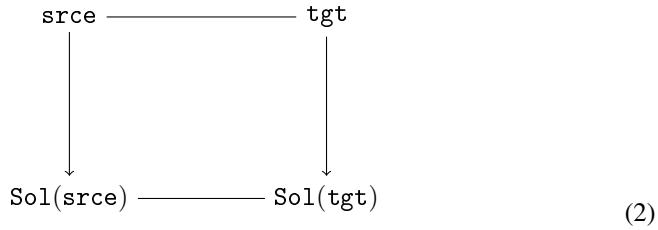
A *source case* is denoted by *Source*: it is a case of *CaseBase* (the case base). The *target case* is denoted by *Target*: it is the input of the CBR system. In many applications, the source cases *Source* are *specific*: each of them represents a single case instance a ($\text{Source} = \{a\}$). By contrast, the target case specifies only its “problem part” and needs to be completed by a “solution part”. The aim of the CBR process is to perform this completion:

$$\begin{aligned} \text{CBR} : (\text{CaseBase}, \text{Target}) &\mapsto \text{ComplTarget} \\ &\text{with } \text{ComplTarget} \subseteq \text{Target} \end{aligned} \tag{1}$$

Usually, this inference is decomposed into two steps:

$$\begin{aligned} \text{Retrieval} : (\text{CaseBase}, \text{Target}) &\mapsto \text{Source} \in \text{CaseBase} \\ \text{Adaptation} : (\text{Source}, \text{Target}) &\mapsto \text{ComplTarget} \end{aligned}$$

In many CBR applications, a case instance a can be decomposed into a problem part x and a solution part y : $a = (x, y)$. Let \mathcal{U}_{pb} and \mathcal{U}_{sol} be the universes of problem and solution instances: $x \in \mathcal{U}_{pb}$, $y \in \mathcal{U}_{sol}$, $\mathcal{U} = \mathcal{U}_{pb} \times \mathcal{U}_{sol}$. A source case Source is decomposed in a *source problem* $\text{srce} \in 2^{\mathcal{U}_{pb}}$ and its solution $\text{Sol}(\text{srce}) \in 2^{\mathcal{U}_{sol}}$, thus $\text{Source} = \text{srce} \times \text{Sol}(\text{srce})$ that is interpreted as: for all $x \in \text{srce}$ there exists $y \in \text{Sol}(\text{srce})$ such that $a = (x, y)$ is a licit case (i.e., y solves x). The solution part of the target problem is unknown, thus $\text{Target} = \text{tgt} \times \mathcal{U}_{sol}$, where tgt is called the *target problem*. When cases are decomposed in problems and solutions, CBR aims at solving the target problem tgt , thus $\text{ComplTarget} = \text{tgt} \times \text{Sol}(\text{tgt})$ where $\text{Sol}(\text{tgt}) \in 2^{\mathcal{U}_{sol}}$. In general, a target problem is specific: it is a singleton, i.e., $\text{tgt} = \{x^t\}$ with $x^t \in \mathcal{U}_{pb}$. This problem-solution decomposition is not needed to present revision-based adaptation but it is a prerequisite for other approaches to adaptation mentioned in the chapter. When cases are decomposed in problem and solution parts, it is common to consider the adaptation problem as an analogical problem represented by the following diagram:



that can be read as “ $\text{Sol}(\text{tgt})$ is to $\text{Sol}(\text{srce})$ as tgt is to srce ” (transformational analogy [4]) or “ $\text{Sol}(\text{tgt})$ is to tgt as $\text{Sol}(\text{srce})$ is to srce ” (derivational analogy [5]). In other words, adaptation aims at solving an analogical problem.

The domain knowledge DK is a knowledge base giving a necessary condition for a case instance to be licit. Thus, the domain knowledge can be represented by a subset DK of \mathcal{U} and for each $a \in \mathcal{U}$, $a \notin \text{DK}$ involves that a is not licit. When the case universe is decomposed in $\mathcal{U}_{pb} \times \mathcal{U}_{sol}$, $a = (x, y) \notin \text{DK}$ means that y is not a solution of x or that x and/or y are meaningless (i.e., they are objects represented in the language that have no correspondence in the real world, e.g., in the domain of zoology, a cat that is not a mammal). Having no domain knowledge (or not taking it into account) amounts to $\text{DK} = \mathcal{U}$.

Each source case is assumed to be consistent with the domain knowledge, i.e.,

$$\text{DK} \cap \text{Source} \neq \emptyset \quad (3)$$

Similarly, if a target case is inconsistent with the domain knowledge, it has not to be considered for the CBR inference (the system has to reject it before the retrieval). Thus, if Target is an input of the adaptation procedure, it is required that

$$\text{DK} \cap \text{Target} \neq \emptyset \quad (4)$$

The result of adaptation must also be consistent with DK, therefore:

$$\text{DK} \cap \text{ComplTarget} \neq \emptyset \quad (5)$$

(It can be noted that (4) is a consequence of (1) and (5).)

2.3 Distances and metric spaces

A *distance* on a set \mathcal{U} is defined in this chapter as a function $d : \mathcal{U} \times \mathcal{U} \rightarrow [0; +\infty]$ such that $d(\mathbf{a}, \mathbf{b}) = 0$ iff $\mathbf{a} = \mathbf{b}$ (the properties of symmetry and triangle inequality are not required in this chapter). Let $\mathbf{b} \in \mathcal{U}$ and $A, B \in 2^{\mathcal{U}}$. The usual abbreviations are used:

$$d(A, \mathbf{b}) = \inf_{\mathbf{a} \in A} d(\mathbf{a}, \mathbf{b}) \quad d(\mathbf{b}, A) = \inf_{\mathbf{a} \in A} d(\mathbf{b}, \mathbf{a}) \quad d(A, B) = \inf_{\mathbf{a} \in A, \mathbf{b} \in B} d(\mathbf{a}, \mathbf{b})$$

By convention, the infimum on the empty set is $+\infty$, e.g., $d(A, \emptyset) = +\infty$. A is said to be closed under d if

$$\{\mathbf{b} \in \mathcal{U} \mid d(A, \mathbf{b}) = 0\} = \{\mathbf{b} \in \mathcal{U} \mid d(\mathbf{b}, A) = 0\} = A \quad (\text{closeness of } A)$$

A *metric space* is an ordered pair (\mathcal{U}, d) where d is a distance on \mathcal{U} .⁵

If $\mathcal{U} = \mathbb{R}^n$ (where \mathbb{R} is the set of the real numbers), a L_1 -distance is a distance d parametrized by a base \mathcal{B} of the vector space \mathcal{U} such that

$$d(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^n |v_i - u_i| \quad (L_1 \text{ distance})$$

where (u_1, u_2, \dots, u_n) (resp., (v_1, v_2, \dots, v_n)) is the representation of \mathbf{a} (resp., of \mathbf{b}) in \mathcal{B} . When \mathcal{B} is the canonical base, $u_i = \mathbf{a}_i$ and $v_i = \mathbf{b}_i$ for each $i \in \{1, 2, \dots, n\}$. By extension, if \mathcal{U} is a subset of \mathbb{R}^n , a distance d on \mathcal{U} that is the restriction of a L_1 distance on \mathbb{R}^n is also called a L_1 distance on \mathcal{U} (for example, if $\mathcal{U} = \mathbb{Z}^n$ where \mathbb{Z} is the set of integers).

If $\mathcal{U} = \mathbb{B}^n$ where $\mathbb{B} = \{\text{true}, \text{false}\}$, the weighted Hamming distance with weights (w_1, w_2, \dots, w_n) (where $w_i > 0$) is defined, for $\mathbf{a} = (\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n)$ and $\mathbf{b} = (\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_n)$ by

$$d(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^n \begin{cases} 0 & \text{if } \mathbf{a}_i = \mathbf{b}_i \\ w_i & \text{else} \end{cases}$$

The Hamming distance is the weighted Hamming distance with $w_i = 1$ for each $i \in \{1, 2, \dots, n\}$. An interpretation in propositional logic with n variables is assimilated to an element of \mathbb{B}^n , hence the notion of weighted Hamming distance between such interpretations.

⁵ Calling it a metric space is an abuse of the usual mathematical language, since d does not necessarily verify all the postulates of a distance metric. However, the term metric space is used in this chapter with this generalized meaning, for the sake of simplicity.

3 Belief revision

3.1 Belief revision in propositional logic

In [2], postulates of belief revision are proposed in a general logical setting. These postulates are based on the idea of minimal change. They are applied to propositional logic in [6] which presents 6 postulates that a belief operation $\dot{+}$ has to verify in this formalism: given ψ and μ , two belief bases, $\psi \dot{+} \mu$ is a revision of ψ by μ . One of these postulates states that $\dot{+}$ is independent to syntax:

$$\text{if } \psi_1 \equiv \psi_2 \text{ and } \mu_1 \equiv \mu_2 \text{ then } \psi_1 \dot{+} \mu_1 \equiv \psi_2 \dot{+} \mu_2 \quad (6)$$

where ψ_1 , ψ_2 , μ_1 , and μ_2 are formulas representing beliefs. As a consequence of (6), a formula φ can be assimilated to $\text{Mod}(\varphi)$, the set of its models: in the rest of the chapter, formulas and subsets of \mathcal{U} are used indifferently. Then, the other 5 postulates can be rewritten as follows (using subsets of \mathcal{U} , the set of interpretations, instead of propositional formulas):

- (+1) $A \dot{+} B \subseteq B$.
- (+2) If $A \cap B \neq \emptyset$ then $A \dot{+} B = A \cap B$.
- (+3) If $B \neq \emptyset$ then $A \dot{+} B \neq \emptyset$.
- (+4) $(A \dot{+} B) \cap C \subseteq A \dot{+} (B \cap C)$.
- (+5) If $(A \dot{+} B) \cap C \neq \emptyset$ then $A \dot{+} (B \cap C) \subseteq (A \dot{+} B) \cap C$.

(A , B , and C are subsets of \mathcal{U} .) The interpretation of these postulates is made further (in section 4.3), for their application to revision-based adaptation.

Intuitively, to revise A by B , the idea is to modify minimally A into A' so that $A' \cap B \neq \emptyset$, and then $A \dot{+} B = A' \cap B$. Now, there are many ways to model minimal modifications. Among them, there is the modification based on a distance d on \mathcal{U} (figure 1 illustrates it): given $\lambda \in \mathbb{R}$ with $\lambda \geq 0$, $G_\lambda^d(A)$ is the generalization (a kind of modification) of $A \subseteq \mathcal{U}$ defined by

$$G_\lambda^d(A) = \{\mathbf{b} \in \mathcal{U} \mid d(A, \mathbf{b}) \leq \lambda\}$$

Then, the revision operator $\dot{+}^d$ is defined by

$$A \dot{+}^d B = G_\delta^d(A) \cap B \quad \text{where } \delta = \inf\{\lambda \mid G_\lambda^d(A) \cap B \neq \emptyset\}$$

Note that the infima on \mathcal{U} are always reached when \mathcal{U} is finite, which is the case when \mathcal{U} is the set of interpretations over n propositional variables. This kind of revision operators is a direct generalization of the Dalał revision operator [7], which is based on the Hamming distance between propositional interpretations. The following equivalent definition can be given:

$$A \dot{+}^d B = \{\mathbf{b} \in B \mid d(A, \mathbf{b}) = \delta\} \quad \text{where } \delta = d(A, B)$$

(the δ 's in the two definitions are equal).

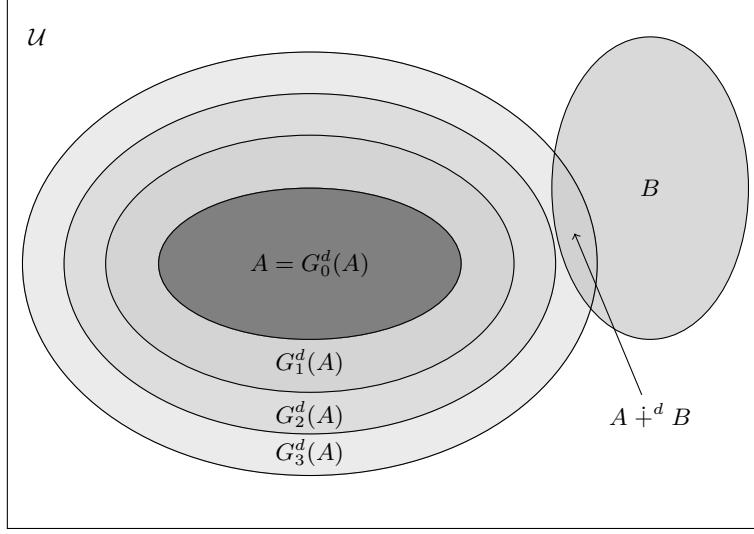


Fig. 1: Illustration of a distance-based revision operator in propositional logic. The set of interpretations is mapped to a finite subset of the plan and the distance between interpretations corresponds to a distance in the plan. A is generalized step by step ($G_1^d(A)$, $G_2^d(A)$, etc.) until the generalized step $A' = G_\lambda^d(A)$ intersects with B . The result is $A +^d B = A' \cap B$.

3.2 Belief revision in a metric space

The (+1-5) postulates can be straightforwardly generalized to other formalisms \mathcal{L} where \mathcal{U} is (a priori) any set and each formula φ of \mathcal{L} is assimilated to a subset $\text{Mod}(\varphi)$ of \mathcal{U} . In this generalization, A , B and C are sets assumed to be representable in \mathcal{L} .

Given a distance d on \mathcal{U} , $+^d$ can be defined as above but it must be noticed that $+^d$ may not satisfy the (+1-5) postulates. This issue is considered further, in section 4.4.

The representability issue must also be addressed in this generalization, thus we propose the following postulate:

(+6) For any $\psi, \mu \in \mathcal{L}$, there exists $\rho \in \mathcal{L}$ such that $\text{Mod}(\rho) = \text{Mod}(\psi) +^d \text{Mod}(\mu)$.

In propositional logic, for any operator $+$, this postulate is particularly holds, since every subset of the set \mathcal{U} of the interpretations of propositional variables is representable in this logic.

3.3 Integrity constraint belief merging

Let $\psi_1, \psi_2, \dots, \psi_k$, and μ be $k + 1$ belief bases. Merging $\psi_1, \psi_2, \dots, \psi_k$, given the integrity constraint μ consists in building a belief base φ such that $\varphi \models \mu$ and φ keeps “as much as possible” from the ψ_i ’s. A merging operator

$$\Delta : (\mu, \{\psi_i\}_{1 \leq i \leq k}) \mapsto \varphi = \Delta_\mu (\{\psi_i\}_{1 \leq i \leq k})$$

is assumed to satisfy some postulates similar to the postulates for a revision operator (in [8], such postulates are defined in propositional logic but can be easily generalized to metric spaces). Actually, the notion of integrity constraint extends the notion of belief revision in the sense that if Δ is such a merging operator, then $\dot{+}$ defined by $\psi \dot{+} \mu = \Delta_\mu(\{\psi\})$ satisfies the (+1-5) postulates.

4 Revision-based adaptation

This section defines revision-based adaptation (subsection 4.1), presents an example in propositional logic (subsection 4.2), studies its properties (subsection 4.3), describes with details revision-based adaptation in metric spaces (subsection 4.4), mentions briefly an extension to multiple case adaptation (subsection 4.5), relates this approach to adaptation with other approaches to adaptation (subsection 4.6) and gives pointers on other work related to revision-based adaptation (subsection 4.7).

4.1 Definition

Let \mathcal{U} be the case universe and $\dot{+}$ be a revision operator on \mathcal{U} . The $\dot{+}$ -adaptation is defined as follows [9]:

$$\text{ComplTarget} = (\text{DK} \cap \text{Source}) \dot{+} (\text{DK} \cap \text{Target}) \quad (7)$$

$(\text{DK} \cap \text{Source})$ (resp., $(\text{DK} \cap \text{Target})$) is the source (resp., target) case interpreted within the domain knowledge (i.e., case instances known to be not licit are removed). Thus (7) can be interpreted as a minimal modification of the source case to satisfy the target case, given the domain knowledge, knowing that the minimality of modification is the one associated with the operator $\dot{+}$.

4.2 Example in propositional logic

Let us consider the following story. Léon is about to invite Thècle and wants to prepare her an appropriate meal. His target problem can be specified by the characteristics of Thècle about food. Let us assume that Thècle is vegetarian (denoted by the propositional variable v) and that she has other characteristics (denoted by o) not detailed in this example:

$$\text{Target} = v \wedge o$$

From his experience as a host, Léon remembers that he had invited Simone some times ago and he thinks that Simone is very similar to Thècle according to food behavior, except that she is not a vegetarian ($\neg v \wedge o$). He had proposed to Simone a meal with salad (s), beef (b), and a dessert (d), and she was satisfied by the two formers but has not eaten the dessert, thus Léon has retained the source case

$$\text{Source} = (\neg v \wedge o) \wedge (s \wedge b \wedge \neg d)$$

Besides that, Léon has some general knowledge about food: he knows that beef is meat, that meat and tofu are protein foods, and that vegetarians do not eat meat. Moreover, the

only protein food that he is willing to cook, apart from meat, is tofu. Thus, his domain knowledge is

$$\text{DK} = b \Rightarrow m \wedge m \vee t \Leftrightarrow p \wedge v \Rightarrow \neg m$$

where b , m , t , and p are the propositional variables for “some beef/meat/tofu/protein food is appreciated by the current guest”. According to $\dot{+}$ -adaptation, what meal should be proposed to Thècle? If $\dot{+}$ is the Dalal revision operator, the $\dot{+}$ -adaptation of the meal for Simone to a meal for Thècle is

$$\text{ComplTarget} \equiv \text{DK} \wedge \text{Target} \wedge (s \wedge t \wedge \neg d)$$

In [9], this adaptation is qualified as conservative: the salad and the absence of dessert is reused for the target case and, though the beef is not kept (to ensure a consistent result), the consequence of b that is consistent with DK , i.e., p , is kept, and thus, t is proposed instead of beef (since $v \wedge p \models_{\text{DK}} t$; in other words, some protein food is required, the only vegetarian protein that Léon is willing to cook is tofu, thus there will be tofu in the meal).

4.3 Properties

The (+1-6) postulates entail some properties of revision-based adaptation.

(+1) applied to $\dot{+}$ -adaptation gives $\text{ComplTarget} \subseteq \text{DK} \cap \text{Target}$, which entails the property (1) required for the adaptation process (cf. section 2.2) and the fact that $\text{ComplTarget} \subseteq \text{DK}$ (no instance case a known to be illicit— $a \in \mathcal{U} \setminus \text{DK}$ —is in the result).

Let us assume that $\text{DK} \cap \text{Source} \cap \text{Target} \neq \emptyset$. Then, (+2) entails that $\text{ComplTarget} = \text{DK} \cap \text{Source} \cap \text{Target}$. This means that if the target case is consistent with the source case, given the domain knowledge, then it can be inferred by $\dot{+}$ -adaptation that Source solves Target. This is consistent with the principle of this kind of adaptation: ComplTarget is obtained by keeping from Source as much as possible, and if no modification is needed then no modification is applied.

(+3) gives: if $\text{DK} \cap \text{Target} \neq \emptyset$ then $\text{ComplTarget} \neq \emptyset$. Since $\text{ComplTarget} \subseteq \text{DK}$ (cf. (+1)), $\text{DK} \cap \text{ComplTarget} \neq \emptyset$, which is a property required by an adaptation operator (cf. equation (5), section 2.2).

According to [6], (+4) and (+5) capture the minimality of modifications. Thus they express the minimality of modification made by a $\dot{+}$ -adaptation. This can be interpreted as follows. The conjunction of (+4) and (+5) can be reformulated as:

$$\begin{cases} \text{Either } (A \dot{+} B) \cap C = \emptyset, \\ \text{Or } (A \dot{+} B) \cap C = A \dot{+} (B \cap C). \end{cases} \quad (8)$$

Let F represent some additional features about the target problem: the new target case is $\text{Target}_2 = \text{Target} \cap F$. If ComplTarget is consistent with F , then (+4) and (+5) entail that the adaptation of Source to Target_2 gives $\text{ComplTarget}_2 = \text{ComplTarget} \cap F$. In other words, if F does not involve needs on modifications (corresponding to an inconsistency) then the result of the $\dot{+}$ -adaptation can be reused straightforwardly.

(+6) involves that ComplTarget is representable in \mathcal{L} .

4.4 Revision-based adaptation in metric spaces

In this section, $\dot{+}^d$ -adaptation is considered on a metric space (\mathcal{U}, d) . This study is motivated by applications of CBR which are frequently based on attribute-value formalisms, where the attributes range frequently in simple domains (numeric domains, Boolean, etc.). Thus, the first idea was to study revision for such formalisms, but it appeared that the more general framework of metric space was a better level of study (particularities of attribute-value formalisms did not add much to the study). Therefore, this has involved the necessity to study revision in this framework. This section goes from general to specific. First, revision in (\mathcal{U}, d) is considered according to the revision postulates. Then, it is applied to attribute-constraint formalisms (that include attribute-value formalisms), which is further applied to formalisms of linear constraints, leading to a practical algorithm for revision in this case. Finally, a practical application to the cooking domain is presented.

The (+1-6) postulates in metric spaces

This section studies the revision postulates for $\dot{+}^d$. Some of these postulates are not satisfied by $\dot{+}^d$ and some additional assumptions on the representation language \mathcal{L} and on d are proposed that are sufficient conditions for their satisfaction (recall that the subsets A , B , and C of \mathcal{U} involved in postulates (+1-5) are representable in \mathcal{L}).

(+1) is always satisfied by $\dot{+}^d$ (direct consequence from the definition of $\dot{+}^d$).

(+2) is not always satisfied by $\dot{+}^d$ as the following counterexample shows. Let $\mathcal{U} = \mathbb{IR}$ and let \mathcal{L} be the language of intervals of \mathbb{IR} , e.g., $[0; 1] = \{a \in \mathcal{U} \mid 0 \leq a < 1\}$. Let $d : (a, b) \mapsto |b - a|$. It can be shown that $[0; 1] \dot{+}^d [0; 1] = [0; 1] \not\subseteq [0; 1]$. Now, let us consider the following additional assumption about the formalism \mathcal{L} :

($\mathcal{L}1$) Every subset A of \mathcal{U} that can be represented in \mathcal{L} is closed under d .

Under this assumption, (+2) is satisfied as proven hereafter. Let A and B be two subsets of \mathcal{U} such that A is closed and $A \cap B \neq \emptyset$. Thus, $d(A, B) = 0$ and

$$\begin{aligned} A \dot{+}^d B &= \{b \in B \mid d(A, b) = 0\} \\ &= \{b \in \mathcal{U} \mid d(A, b) = 0\} \cap B \\ &= A \cap B \end{aligned} \quad \text{since } A \text{ is closed}$$

Therefore $A \dot{+}^d B = A \cap B$ and (+2) is satisfied.

(+3) is not always satisfied by $\dot{+}^d$ as the following counterexample shows. Let \mathcal{U} and d be the same as in the counterexample of (+2). Let $A = [0; 1]$ and $B =]2; 3]$. $B \neq \emptyset$ but $A \dot{+}^d B = \emptyset$. This suggests that A and B should be closed but even if the ($\mathcal{L}1$) assumption was made, (+3) may be not satisfied. Figure 2 presents a counterexample of (+3) with A and B , two closed sets. Now, let us consider the following assumption:

($\mathcal{L}2$) For every A and B non empty subsets of \mathcal{U} representable in \mathcal{L} , the distance between A and B is always reached: there exist $a \in A$ and $b \in B$ such that $d(A, B) = d(a, b)$.

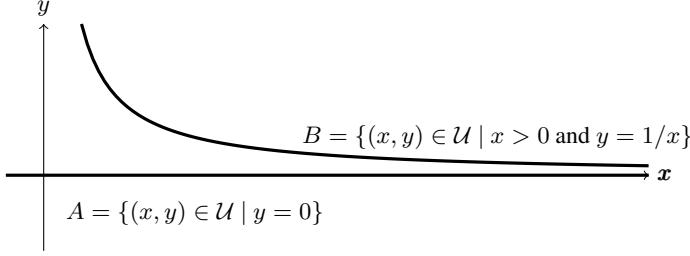


Fig. 2: A counterexample of $(\dot{+}3)$. $\mathcal{U} = \mathbb{R}^2$, d is a L_1 distance on \mathcal{U} , \mathcal{L} is chosen so that A, B and \emptyset are representable in this formalism, which is consistent with assumption $(\mathcal{L}1)$, since A and B are closed under d . Finally, $B \neq \emptyset$. Despite these facts, $A \dot{+} B = \emptyset$ (since $d(A, B) = 0$ and no element of B is at distance 0 from an element of A).

Under this assumption, $(\dot{+}3)$ is satisfied as proven hereafter. Let $A, B \in 2^\mathcal{U}$ such that $B \neq \emptyset$. If $A = \emptyset$, $d(A, B) = d(A, b) = +\infty$ for every $b \in B$, thus $A \dot{+}^d B = B \neq \emptyset$. If $A \neq \emptyset$ and if A and B are representable in \mathcal{L} then $(\mathcal{L}2)$ entails that $d(A, B) = d(a, b)$ for some $(a, b) \in A \times B$. Since $d(a, b) \geq d(A, b) \geq d(A, B) = d(a, b)$, $d(A, b) = d(a, b)$. Therefore, b is such that $d(A, b) = d(A, B)$ thus $b \in A \dot{+}^d B$ and so, $A \dot{+}^d B \neq \emptyset$. So $(\dot{+}3)$ is satisfied.

If $\mathcal{U} = \mathbb{R}^n$, if d is a L_1 distance on \mathcal{U} , and if each A representable in \mathcal{L} is closed and bounded, then $(\mathcal{L}2)$ is satisfied. More generally, if d is a distance in the classical mathematical sense (it verifies separation, symmetry, triangle inequality, and $d(a, b) < +\infty$ for every $a, b \in \mathcal{U}$), and if every A representable in \mathcal{L} is a compact space, then $(\mathcal{L}2)$ is satisfied.

$(\dot{+}4)$ and $(\dot{+}5)$ are always satisfied by $\dot{+}^d$ as proven hereafter. The conjunction of these postulates is equivalent to (8). Let $A, B, C \in 2^\mathcal{U}$. If $(A \dot{+}^d B) \cap C = \emptyset$ then (8) is verified. Now, assume that $(A \dot{+}^d B) \cap C \neq \emptyset$ and let $b \in (A \dot{+}^d B) \cap C$. Then $b \in (B \cap C)$ and $d(A, b) = d(A, B)$. Thus, the following chain of relations can be established:

$$d(A, B) \leq d(A, B \cap C) \leq d(A, b) = d(A, B)$$

(cf. the infimum appearing in the definition of $d(A, \cdot)$ and the fact that $b \in B \cap C$). Therefore, these numbers are all equal and $d(A, B) = d(A, B \cap C)$. Hence

$$\begin{aligned} A \dot{+}^d (B \cap C) &= \{b \in B \cap C \mid d(A, b) = d(A, B \cap C)\} \\ &= \{b \in B \cap C \mid d(A, b) = d(A, B)\} \\ &= \{b \in B \mid d(A, b) = d(A, B)\} \cap C \\ &= (A \dot{+}^d B) \cap C \end{aligned}$$

$(\dot{+}6)$ is not always satisfied by $\dot{+}^d$ as the following counterexample shows. Let $\mathcal{U} = \mathbb{R}^2$ and let \mathcal{L} be the language of *horizontal rectangles*, i.e., for every $\varphi \in \mathcal{L}$, there exists $[x_1; x_2]$ and $[y_1; y_2]$, two intervals of \mathbb{R} , such that $\text{Mod}(\varphi) = [x_1; x_2] \times [y_1; y_2]$. Now, let d be the distance on \mathcal{U} defined by $d(a, b)$ is the Euclidian distance rounded up

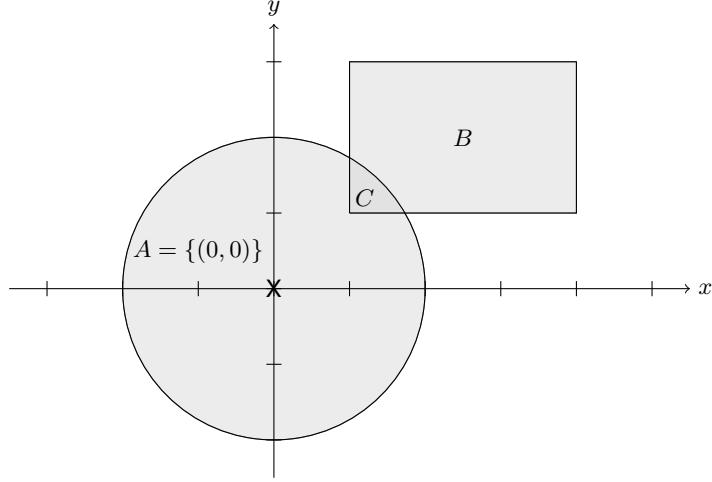


Fig. 3: A counterexample of $(\dot{+}6)$. $A = [0; 0] \times [0; 0] = \{(0, 0)\}$ and $B = [1; 4] \times [1; 3]$ are two subsets of \mathcal{U} representable in the language \mathcal{L} of horizontal rectangles, though $A \dot{+}^d B = C$ (where d is the Euclidian distance rounded up to integers) is not representable in \mathcal{L} . Indeed, C is the intersection of $G_2^d(A)$, the closed disk of radius 2 and centered on $(0, 0)$, with B .

to integers $(d(\mathbf{a}, \mathbf{b}) = \lceil d_2(\mathbf{a}, \mathbf{b}) \rceil)$ where d_2 is the canonical Euclidian distance on \mathbb{R}^2 and $\lceil x \rceil$ is the smallest integer n such that $x \leq n$). As illustrated by figure 3, $A \dot{+}^d B$ is a subset of \mathcal{U} that is not representable in \mathcal{L} .

Now, let us consider the following assumption:

- $(\mathcal{L}3)$ For each A , subset of \mathcal{U} representable in \mathcal{L} , for each $\lambda \in \mathbb{R}$ with $\lambda \geq 0$, $G_\lambda^d(A)$ is representable in \mathcal{L} .

It is a sufficient condition for $(\dot{+}6)$. Indeed, if A and B is representable in \mathcal{L} then $(\mathcal{L}3)$ entails that $G_\delta^d(A)$ is representable in \mathcal{L} , with $\delta = d(A, B)$. Therefore, $G_\delta^d(A) \cap B = A \dot{+}^d B$ is representable in \mathcal{L} , since this formalism is stable under conjunction. So $(\dot{+}6)$ holds. Note that $(\mathcal{L}3)$ is not a necessary condition for $(\dot{+}6)$.

Table 1 summarizes these results.

Attribute-constraint formalisms

Definitions. In this section, it is assumed that $\mathcal{U} = V_1 \times V_2 \times \dots \times V_n$ where each V_i is a “simple value” space, i.e. either \mathbb{R} (the set of real numbers), \mathbb{Z} (the set of integers), any interval of \mathbb{R} or \mathbb{Z} , $\mathbb{B} = \{\text{true}, \text{false}\}$, or another set defined in extension. For $i \in \{1, 2, \dots, n\}$, the attribute a_i is the i^{th} projection:

$$a_i : (\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n) \in \mathcal{U} \mapsto \mathbf{a}_i \in V_i$$

| | |
|-------------------------------------|---------------------------|
| | (+1) is always satisfied. |
| Under assumption $(\mathcal{L}1)$, | (+2) is satisfied. |
| Under assumption $(\mathcal{L}2)$, | (+3) is satisfied. |
| | (+4) is always satisfied. |
| | (+5) is always satisfied. |
| Under assumption $(\mathcal{L}3)$, | (+6) is satisfied. |

Table 1: Conditions for having the revision postulates satisfied by a distance-based revision operator.

A formula φ of the representation language is a constraint, i.e., a Boolean expression based on the attributes a_i : $\varphi = P(a_1, a_2, \dots, a_n)$. The semantics of φ is

$$\text{Mod}(\varphi) = \{\mathbf{a} \in \mathcal{U} \mid P(a_1(\mathbf{a}), a_2(\mathbf{a}), \dots, a_n(\mathbf{a}))\}$$

These formalisms contain propositional logic with n variables: $V_i = \mathbb{IB}$ (for each $i \in \{1, 2, \dots, n\}$), knowing that the Boolean expressions are based on the Boolean operations **and**, **or**, and **not**. For example, if $n = 3$, and $a_1 = o$, $a_2 = t$ and $a_3 = v$:

$$\text{Mod}(\neg v \vee o) = \{(\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3) \in \mathcal{U} \mid \text{or}(\text{not}(\mathbf{a}_3), \mathbf{a}_1)) = \text{true}\}$$

These formalisms also contain the attribute-value formalisms often used for representing cases in CBR [10]: a specific case C is defined by $C = (a_1 = v_1) \wedge (a_2 = v_2) \wedge \dots \wedge (a_n = v_n)$ and thus $C = \{(v_1, v_2, \dots, v_n)\}$. When problem-solution decomposition is made, in general, the attributes are split in problem attributes (a_1, \dots, a_p) and solution attributes (a_{p+1}, \dots, a_n). Classically, the distance used on \mathcal{U} for the retrieval is the weighted sum of distances on each problem attribute.

Application to the numerical case with linear constraints. Now, it is assumed that each V_i is either \mathbb{IR} or \mathbb{Z} and each formula is a conjunction of linear constraints on the attributes. A linear constraint is an expression of the form $\sum_{i=1}^n \alpha_i \cdot a_i \leq \beta$ where $\alpha_1, \dots, \alpha_n, \beta \in \mathbb{IR}$.

Let d be the L_1 distance on \mathcal{U} parametrized by a base \mathcal{B} . It can be shown that $\dot{+}^d$ satisfies all the (+1-6) postulates (where A , B , and C are defined thanks to conjunctions of linear constraints). $\dot{+}^d$ -adaptation amounts to solve the following optimization problem:

$$\mathbf{a} \in \text{DK} \cap \text{Source} \tag{9}$$

$$\mathbf{b} \in \text{DK} \cap \text{Target} \tag{10}$$

$$\text{minimize } \sum_{i=1}^n |v_i - u_i| \tag{11}$$

where (u_1, \dots, u_n) and (v_1, \dots, v_n) are the respective representations of \mathbf{a} and \mathbf{b} in the base \mathcal{B} . ComplTarget is the set of the \mathbf{b} that solve this optimization problem.

In this optimization problem, (9) and (10) are linear constraints, but the function to be minimized in (11) is not linear. However, this optimization problem can be solved thanks to the solving of the following linear problem (introducing the new variables z_1, \dots, z_n):

$$\begin{aligned} \mathbf{a} &\in \text{DK} \cap \text{Source} \\ \mathbf{b} &\in \text{DK} \cap \text{Target} \\ v_i - u_i &\leq z_i \quad (1 \leq i \leq n) \\ u_i - v_i &\leq z_i \quad (1 \leq i \leq n) \\ \text{minimize} \quad &\sum_{i=1}^n z_i \end{aligned}$$

It can be shown that the optimal values of \mathbf{a} and \mathbf{b} in the two optimization problems are the same. Therefore, in this formalism, $\dot{+}^d$ -adaptation amounts to a linear programming problem, which is NP-complete if some $V_i = \mathbb{Z}$ but is polynomial when all $V_i = \text{IR}$ [11].

More details about this process can be found in [12].

A cooking application. This principle has been applied to a CBR system called Taaable (<http://taaable.fr>) that has been a contestant of the CCC (*Computer Cooking Contest*, organized during the ICCBR conferences). The CCC provides a recipe base. A contestant of the CCC is a system that has to solve cooking problems using these recipes (a case of this application is a recipe). These problems are specified by a set of desired ingredients or dish types, and undesired ones (e.g., “I’d like a pear pie but I don’t like cinnamon.”). Taaable has won the main challenge and the adaptation challenge of this contest in 2010 [13]. The adaptation of ingredient quantities was made possible thanks to a reduction to linear programming as mentioned before. Details can be found in [13] but the idea, explained on a simplified example, is as follows. Suppose that the user wants a recipe of a pear pie and that Taaable retrieves an apple pie. The domain knowledge is expressed by linear constraints on these properties, such as:

$$\begin{aligned} \text{mass}_{\text{fruit}} &= 120 \cdot \text{nb}_{\text{apple}} + 100 \cdot \text{nb}_{\text{pear}} \\ \text{mass}_{\text{sweet}} &= \text{mass}_{\text{sugar}} + 10 \cdot \text{nb}_{\text{apple}} + 15 \cdot \text{nb}_{\text{pear}} \end{aligned}$$

(these knowledge can be found in a free nutritional database). Each mass_\bullet is an attribute on IR and each nb_\bullet is an attribute on IN (non negative integers). The source case is a singleton $\{\mathbf{a}\}$ such that $\text{nb}_{\text{apple}}(\mathbf{a}) = 4$ and $\text{mass}_{\text{sugar}}(\mathbf{a}) = 40$. The target case corresponds to the constraint $\text{nb}_{\text{apple}} = 0$ (the substitution of apples by pears is inferred by a previous step similar to a $\dot{+}$ -adaptation in propositional logic). The $\dot{+}^d$ -adaptation leads to a maximal preservation of the attributes $\text{mass}_{\text{fruit}}$ and $\text{mass}_{\text{sugar}}$ and since the pears contain more sweet than the apples, the mass of added sugar is lowered (there is a sort of “compensation effect”). More precisely, the $\dot{+}^d$ -adaptation (at least for some base \mathcal{B}) gives $\text{ComplTarget} = \{\mathbf{b}\}$ with $\text{nb}_{\text{pear}}(\mathbf{b}) = 5$ (the total fruit mass from Source to ComplTarget is modified from 480 to 500) and $\text{mass}_{\text{sugar}}(\mathbf{b}) = 5$ (the total sweet mass is unchanged).

4.5 Multiple case adaptation

Some CBR systems retrieve several cases and then adapt them in order to solve the target case:

$$\begin{aligned} \text{Retrieval} : (\text{CaseBase}, \text{Target}) &\mapsto \{\text{Source}_i\}_{1 \leq i \leq k} \subseteq \text{CaseBase} \\ \text{Adaptation} : \left(\{\text{Source}_i\}_{1 \leq i \leq k}, \text{Target}\right) &\mapsto \text{ComplTarget} \end{aligned}$$

This adaptation is called multiple case adaptation and is also known as case combination. Multiple case adaptation extends single case adaptation (which is a case combination with $k = 1$) in the same way as integrity constraint belief merging extends belief revision (cf. section 3.3), hence the idea⁶ to use a merging operator Δ on \mathcal{U} to define a multiple case adaptation process:

$$\text{ComplTarget} = \Delta_{\text{DK} \cap \text{Target}} \left(\{\text{DK} \cap \text{Source}_i\}_{1 \leq i \leq k} \right)$$

which generalizes (7).

This approach to multiple case adaptation is studied in [12].

4.6 Revision-based adaptation and other approaches to adaptation

Other approaches to adaptation have been defined in the CBR literature. This section compares revision-based adaptation to some of them.

Adaptation by generalization and specialization

The principle of this adaptation is to generalize the source case and then to specialize it to the target case. Since

- the source and target cases must be considered w.r.t. the domain knowledge and
- this generalization of the source case must be minimal so that it meets the constraints of the target case,

this kind of adaptation amounts to generalize minimally $A = \text{DK} \cap \text{Source}$ into A' such that $A' \cap B \neq \emptyset$ with $B = \text{DK} \cap \text{Target}$. Therefore, it is a $\dot{+}$ -adaptation for which the modification operation $A \mapsto A'$ is a generalization ($A \subseteq A'$).

Conversely, if $\dot{+}$ is a revision operator and $A, B \in 2^{\mathcal{U}}$, $A \dot{+} B = A' \cap B$ such that A has been modified minimally in A' . This modification is not necessarily a generalization but if $\widehat{A} = A \cup A'$, then $A \subseteq \widehat{A}$ and $A \dot{+} B = \widehat{A} \cap B$. Indeed, either $A \cap B \neq \emptyset$ so $A' = A$ and then $\widehat{A} = A$, or $A \cap B = \emptyset$ so $\widehat{A} \cap B = (A \cup A') \cap B = (A \cap B) \cup (A' \cap B) = \emptyset \cup (A' \cap B) = A' \dot{+} B$. So, without loss of generality, it can be considered that the modification of A done when revising A by any revision operator on \mathcal{U} is a generalization. Hence, $\dot{+}$ -adaptation can be considered as a formalization of the general model of adaptation by generalization and specialization.

When the revision operator is based on a distance d , $\dot{+}^d$ -adaptation can be read as the composition of

⁶ Once suggested by Pierre Marquis. Thanks Pierre!

a generalization $\text{DK} \cap \text{Source} \mapsto G_\delta^d(\text{DK} \cap \text{Source})$ and
a specialization $G_\delta^d(\text{DK} \cap \text{Source}) \mapsto G_\delta^d(\text{DK} \cap \text{Source}) \cap \text{DK} \cap \text{Target}$.

To illustrate this idea, it can be noticed that the example of section 4.2 (when Léon invites Thècle) of revision-based adaptation in propositional logic, can be redescribed as an adaptation by generalization and specialization. First, the meal is generalized by substituting beef by protein food (generalization motivated by the fact that Thècle is vegetarian). Then, the generalized meal is specialized by substituting protein food by tofu.

For the other approaches to adaptation considered below, the assumptions of problem-solving decomposition ($\mathcal{U} = \mathcal{U}_{\text{pb}} \times \mathcal{U}_{\text{sol}}$, $\text{Source} = \text{srce} \times \text{Sol}(\text{srce})$, $\text{Target} = \text{tgt} \times \mathcal{U}_{\text{sol}}$, $\text{ComplTarget} = \text{tgt} \times \text{Sol}(\text{tgt})$) and of specificities of the source case and the target problem ($\text{Source} = \{\text{a}\} = \{(x^s, y^s)\}$, $\text{tgt} = \{x^t\}$) are made.

Null adaptation, orthogonal revision operators and conservative adaptation

In [1], null adaptation is justified by the assertion “People often do little adaptation.” This adaptation is defined by $\text{Sol}(\text{tgt}) = \text{Sol}(\text{srce})$. It is often used when the solution space representation is very simple (e.g., a set of predefined categories). It is also used when CBR is assimilated to a simple approximate reasoning method: if $\text{Sol}(\text{srce})$ solves srce and srce is similar to tgt then it is likely that $\text{Sol}(\text{tgt}) = \text{Sol}(\text{srce})$ approximately solves tgt . Under the following assumptions, $\dot{+}$ -adaptation coincides with null adaptation:

- (A1) No domain knowledge is considered (i.e., $\text{DK} = \mathcal{U}$);
- (A2) $\dot{+}$ is *orthogonal*,⁷ i.e. there exists two revision operators, $\dot{+}_{\text{pb}}$ on \mathcal{U}_{pb} and $\dot{+}_{\text{sol}}$ on \mathcal{U}_{sol} , such that

$$(A_{\text{pb}} \times A_{\text{sol}}) \dot{+} (B_{\text{pb}} \times B_{\text{sol}}) = (A_{\text{pb}} \dot{+}_{\text{pb}} B_{\text{pb}}) \times (A_{\text{sol}} \dot{+}_{\text{sol}} B_{\text{sol}}) \quad (12)$$

for any $A_{\text{pb}}, B_{\text{pb}} \in 2^{\mathcal{U}_{\text{pb}}}$ and $A_{\text{sol}}, B_{\text{sol}} \in 2^{\mathcal{U}_{\text{sol}}}$.

Indeed, under these assumptions:

$$\begin{aligned} \text{ComplTarget} &= (\text{srce} \times \text{Sol}(\text{srce})) \dot{+} (\text{tgt} \times \mathcal{U}_{\text{sol}}) \\ &= (\text{srce} \dot{+}_{\text{pb}} \text{tgt}) \times (\text{Sol}(\text{srce}) \dot{+}_{\text{sol}} \mathcal{U}_{\text{sol}}) \end{aligned}$$

Therefore:

$$\begin{aligned} \text{Sol}(\text{tgt}) &= \text{Sol}(\text{srce}) \dot{+}_{\text{sol}} \mathcal{U}_{\text{sol}} \\ &= \text{Sol}(\text{srce}) \cap \mathcal{U}_{\text{sol}} \quad (\text{according to postulate } (+2)) \\ &= \text{Sol}(\text{srce}) \end{aligned}$$

⁷ This adjective is justified further in the chapter.

The intuition behind this notion of orthogonality of $\dot{+}$ is that the modifications are made independently in the problem space and in the solution space. For example, if d is a distance on \mathcal{U} such that there exists a distance d_{pb} on \mathcal{U}_{pb} and a distance d_{sol} on \mathcal{U}_{sol} with

$$d((x^1, y^1), (x^2, y^2)) = d_{\text{pb}}(x^1, x^2) + d_{\text{sol}}(y^1, y^2) \quad (13)$$

for each $(x^1, y^1), (x^2, y^2) \in \mathcal{U}$, then $\dot{+}^d$ is orthogonal. Note that the reverse is not true. For example, if $d((x^1, y^1), (x^2, y^2)) = \sqrt{d_{\text{pb}}(x^1, x^2)^2 + d_{\text{sol}}(y^1, y^2)^2}$ then $\dot{+}^d$ is orthogonal, though d cannot be written as in (13) in general.

Now, let us consider the situation when the assumption (A2) is made but the assumption (A1) is not. Source is a singleton $\{(x^s, x^t)\}$ consistent with DK, thus $\text{DK} \cap \text{Source} = \{(x^s, x^t)\}$. Target = $\{x^t\} \times \mathcal{U}_{\text{sol}}$ so:

$$\begin{aligned} \text{DK} \cap \text{Target} &= \{(x, y) \in \text{DK} \mid x = x^t\} = \{x^t\} \times \text{DK}_{\text{sol}}(x^t) \\ \text{where } \text{DK}_{\text{sol}}(x^t) &= \{y \in \mathcal{U}_{\text{sol}} \mid (x^t, y) \in \text{DK}\} \end{aligned}$$

Therefore:

$$\begin{aligned} \text{ComplTarget} &= \{(x^s, y^s)\} \dot{+} (\{x^t\} \times \text{DK}_{\text{sol}}(x^t)) \\ &= (\{x^s\} \dot{+}_{\text{pb}} \{x^t\}) \times (\{y^s\} \dot{+}_{\text{sol}} \text{DK}_{\text{sol}}(x^t)) \quad (\text{cf. (12)}) \\ &= \{x^t\} \times (\{y^s\} \dot{+}_{\text{sol}} \text{DK}_{\text{sol}}(x^t)) \end{aligned}$$

$(\{x^s\} \dot{+}_{\text{pb}} \{x^t\}) = \{x^t\}$ is a consequence of (+1) and (+3)). So, the $\dot{+}$ -adaptation consists in “repairing” the solution $\text{Sol}(\text{srce}) = \{y^s\}$ according to the constraint $\text{DK}_{\text{sol}}(x^t)$, the repair being done by $\dot{+}_{\text{sol}}$. Note that, in this process, neither $\dot{+}_{\text{pb}}$ nor x^s are used.

In the following, a $\dot{+}$ -adaptation with an orthogonal revision operator $\dot{+}$ is called a *conservative adaptation*.⁸

Rule-based adaptation

Rule-based adaptation is the adaptation based on a set of *adaptation rules*. Following the formalization of [14], an adaptation rule is an ordered pair (r, \mathcal{A}_r) where r is a binary relation on \mathcal{U}_{pb} and \mathcal{A}_r is such that, for $x^s, x^t \in \mathcal{U}_{\text{pb}}$ and $y^s \in \mathcal{U}_{\text{sol}}$ ($\text{Source} = \{(x^s, y^s)\}$, $\text{Target} = \{x^t\} \times \mathcal{U}_{\text{sol}}$):

$$\text{if } x^s \ r \ x^t \text{ then } \mathcal{A}_r(x^s, y^s, x^t) = y^t \text{ probably solves } x^t$$

The rule is not certain (hence the “probably”).

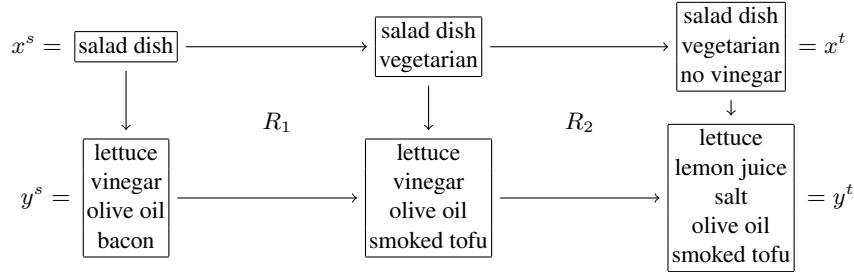
The adaptation rules can be composed as explained hereafter. Let AK be the finite set of adaptation rules that are available to the CBR system. Let $\text{AK}_{\text{pb}} = \{r \mid (r, \mathcal{A}_r) \in \text{AK}\}$.

⁸ In the first publications about $\dot{+}$ -adaptation, the term “conservative adaptation” was used for any $\dot{+}$ -adaptation. Then, it has appeared that some $\dot{+}$ -adaptation could hardly be qualified as conservative (see below), thus, from now on, the term of conservative adaptation is only used for revision-based adaptation using an orthogonal revision operator.

R_1 = Bacon can be substituted by smoked tofu

R_2 = In a salad dish, vinegar can be substituted by lemon juice and salt

(a) Examples of rules.



(b) An example of rule-based adaptation.

$$\text{DK} = \text{lettuce} \Rightarrow \text{green salad} \quad \wedge \quad \text{escarole} \Rightarrow \text{green salad}$$

$$\quad \wedge \quad \text{vegetarian} \Rightarrow \neg \text{meat} \quad \wedge \quad \text{bacon} \Rightarrow \text{meat}$$

$$\text{Source} = \text{salad dish} \wedge \text{lettuce} \wedge \text{vinegar} \wedge \text{olive oil} \wedge \text{bacon}$$

$$\text{Target} = \text{salad dish} \wedge \neg \text{vinegar} \wedge \text{vegetarian}$$

$$\text{AK} = \{R_1, R_2\}$$

with $R_1 = \text{bacon} \rightsquigarrow \text{smoked tofu}$

and $R_2 = \text{salad dish} \wedge \text{vinegar} \rightsquigarrow \text{salad dish} \wedge \text{lemon juice} \wedge \text{salt}$

$$\text{ComplTarget} \equiv \text{DK} \wedge \text{Target} \wedge \text{lettuce} \wedge \text{lemon juice} \wedge \text{salt} \wedge \text{olive oil} \wedge \text{smoked tofu}$$

(c) Formalization of this example as a $\dot{+}^{d_{\text{AK}}}$ -adaptation.

$$\text{Target}_2 = \text{Target} \wedge \neg \text{lettuce}$$

(d) A new target case, slightly different from Target but which cannot be solved by rule-based adaptation of Source given DK and AK (i.e., by $\dot{+}^{d_{\text{AK}}}$ -adaptation).

$$\text{ComplTarget}_2 = (\text{DK} \wedge \text{Source}) \dot{+}^d (\text{DK} \wedge \text{Target}_2)$$

$$\equiv \text{DK} \wedge \text{Target}_2 \wedge \text{green salad} \wedge \text{lemon juice} \wedge \text{salt} \wedge \text{olive oil} \wedge \text{smoked tofu}$$

(e) Solving Target_2 by $\dot{+}^d$ -adaptation, where d is defined by equation (14), with d_0 the Hamming distance. This adaptation consists in applying R_1 , R_2 and in removing lettuce.
 $\text{ComplTarget}_2 \models \text{green salad} \wedge \neg \text{lettuce}$: the lettuce can be replaced by any other kind of green salad, e.g., escarole.

Fig. 4: Rule-based adaptation and revision-based adaptation on an example.

AK_{pb} provides a structure on \mathcal{U}_{pb} . A similarity path from $x^s \in \mathcal{U}_{\text{pb}}$ to $x^t \in \mathcal{U}_{\text{pb}}$ is a path in $(\mathcal{U}_{\text{pb}}, \text{AK}_{\text{pb}})$: it is a sequence of relations $r^i \in \text{AK}_{\text{pb}}$ such that there exist $x^0, x^1, \dots, x^q \in \mathcal{U}_{\text{pb}}$ with $x^0 = x^s$, $x^q = x^t$, and $x^{i-1} \xrightarrow{r^i} x^i$ ($1 \leq i \leq q$). Given such a similarity path, $y^t \in \mathcal{U}$ that probably solves x^t can be computed by applying successively the rules $(r^1, \mathcal{A}_{r^1}), \dots, (r^q, \mathcal{A}_{r^q})$: $y^i = \mathcal{A}_{r^i}(x^{i-1}, y^{i-1}, x^i)$ for i taking the successive values 1, 2, ..., q . Finally, $y^t = y^q$ probably solves x^t . This can be graphically represented by the following diagram, composed of q diagrams like the one of (2), section 2.2:

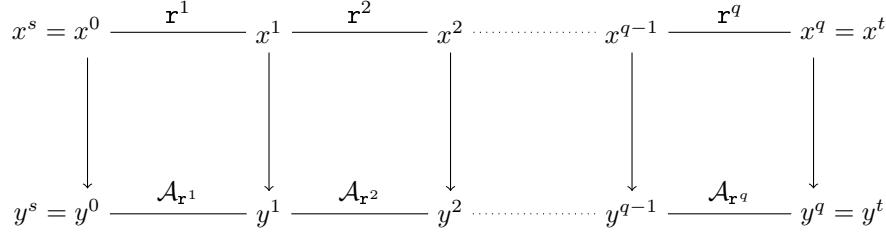


Figure 4(b) illustrates such a rule-based adaptation with a similarity path in two steps, using rules of figure 4(a).

There may be several similarity paths from x^s to x^t . The choice between them is usually based on a cost function such that if SP_1 and SP_2 are two similarity paths from x^s to x^t and $\text{cost}(SP_1) < \text{cost}(SP_2)$ then SP_1 is preferred to SP_2 , which is interpreted as “ SP_1 is more likely to lead to an appropriate solution to x^t than SP_2 .” The function cost is usually assumed to be additive, that is $\text{cost}(SP)$ is the sum of $\text{cost}(r)$ for r a relation of SP . To each $(r, \mathcal{A}_r) \in \text{AK}$, $\text{cost}(r) > 0$ is an information associated with this adaptation rule.⁹

Let d_{AK} be the distance on \mathcal{U} defined by

$$d_{\text{AK}}((x^s, y^s), (x^t, y^t)) = \min \left\{ \text{cost}(SP) \middle| \begin{array}{l} SP: \text{similarity path from } x^s \text{ to } x^t \\ \text{such that the application of } SP \\ \text{on } \{(x^s, y^s)\} \text{ gives } y^t \end{array} \right\}$$

with the convention $\min \emptyset = +\infty$. Let $\text{ComplTarget} = \text{tgt} \times \text{Sol}(\text{tgt})$ be the result of $\dot{+}^{d_{\text{AK}}}$ -adaptation without domain knowledge ($\mathcal{U} = \text{DK}$). If there is no similarity path from x^s to x^t , then $\text{ComplTarget} = \text{Target}$ (the adaptation fails: it does not add any information to the target case). Else, $b = (x^t, y^t) \in \text{ComplTarget}$ iff y^t is obtained by application of a similarity path of minimal cost. Therefore, revision-based adaptation includes rule-based adaptation. Moreover, DK can be taken into account in $\dot{+}^{d_{\text{AK}}}$ -adaptation, thus this enables to specify a rule-based adaptation taking into account the domain knowledge. Conversely, if some adaptation knowledge AK in the form of rules has been acquired (e.g., by means of knowledge discovery and data-mining techniques [15, 16]), this can be useful to specify a relevant revision operator. Indeed, there

⁹ A coarse modeling of this cost is $\text{cost}(r) = -\log P$ where P is the probability that y^t is a licit solution of x^s . Thus, the additivity of the cost corresponds to an independence assumption of the q adaptation steps.

are many possible revision operators and the adaptation knowledge enables to make some choices among them.

Figure 4(c) illustrates how the rule-based adaptation of figure 4(b) can be formalized by a revision-based adaptation using $\dot{+}^{d_{\text{AK}}}$.

A limitation of rule-based reasoning is that it can fail, meaning that there is no similarity path from x^s to x^t ($d(x^s, x^t) = +\infty$), which involves that $\text{ComplTarget} = \text{Target}$. This is particularly true when there are few adaptation rules. Indeed, if $\text{AK} \subseteq \text{AK}'$ then $d_{\text{AK}}(\mathbf{a}, \mathbf{b}) \geq d_{\text{AK}'}(\mathbf{a}, \mathbf{b})$ so if $\dot{+}^{d_{\text{AK}'}}$ -adaptation fails then $\dot{+}^{d_{\text{AK}}}$ -adaptation fails. One way to overcome this limitation is to combine this kind of adaptation with another approach to adaptation and the principle of revision-based adaptation can be used to formalize this combination. This idea is formalized as follows. Let us assume that the other approach to adaptation that has to be combined with rule-based adaptation can be formalized as a revision-based adaptation and let d_0 be a distance on \mathcal{U} such that this adaptation coincides with the $\dot{+}^{d_0}$ -adaptation (intuitively, this adaptation is a “novice” adaptation, hence the 0 in d_0). The $\dot{+}^d$ -adaptation with d defined below combines rule-based adaptation with $\dot{+}^{d_0}$ -adaptation (for $\mathbf{a}, \mathbf{b} \in \mathcal{U}$):

$$d(\mathbf{a}, \mathbf{b}) = \inf_{\mathbf{c} \in \mathcal{U}} (W_{\text{AK}} \cdot d_{\text{AK}}(\mathbf{a}, \mathbf{c}) + W_0 \cdot d_0(\mathbf{c}, \mathbf{b})) \quad (14)$$

where W_{AK} and W_0 are two positive constants. When $\text{AK} = \emptyset$, $d = d_0$ (intuitively: with no adaptation knowledge, the adaptation process is a novice). If the infimum above is reached on $\mathbf{c} = (x^c, y^c)$, the $\dot{+}^d$ -adaptation consists in a rule-based adaptation of $\text{Source} = \{\mathbf{a}\}$ to solve $\{x^c\}$ and then a $\dot{+}^{d_0}$ -adaptation of \mathbf{c} to solve the target case.

Figure 4(d) presents an example of adaptation that cannot be solved by rule-based adaptation using the two rules of figure 4(a) but can be solved by $\dot{+}^d$ -adaptation according to equation (14).

Differential calculus as a $\dot{+}$ -adaptation

First-order differential calculus rely on the equation

$$dy_j = \sum_i \frac{\partial y_j}{\partial x_i} dx_i$$

where $x \mapsto y$ is a differentiable function from \mathbb{R}^p to \mathbb{R}^q . dx_i and dy_j can be interpreted as small differences and substituted respectively by $x_i^t - x_i^s$ and $y_j^t - y_j^s$ when x^s and x^t are similar and thus can be seen as a way to adapt $\text{Source} = \{(x^s, y^s)\}$ in order to solve $\text{Target} = \{x^t\} \times \mathcal{U}_{\text{sol}}$. In this adaptation process, $\left\{ \frac{\partial y_j}{\partial x_i} \right\}_{ij}$ constitutes the adaptation knowledge: it represents how a solution descriptor is modified when a problem descriptor is modified.

In the following, we restrict ourselves to a differentiable function $f : \mathbb{R} \rightarrow \mathbb{R}$, though the principle presented below can be generalized to $f : \mathbb{R}^p \rightarrow \mathbb{R}^q$. Thus $\mathcal{U}_{\text{pb}} = \mathcal{U}_{\text{sol}} = \mathbb{R}$.

A first-order approximation approximation of f in x^t is

$$y^t = x^s + \varrho^s(x^t - x^s) \quad (15)$$

The best choice for ϱ^s is $\varrho^s = f'(x^s)$ (where f' is the differential of f): this choice is the best in the sense that $f'(x^s)$ is the only ϱ^s such that

$$y^t - (y^s + \varrho^s(x^t - x^s)) \underset{x^t \rightarrow x^s}{=} o(x^t - x^s)$$

(this is a direct consequence of the definition of differentials). The adaptation knowledge of this first-order approximation is ϱ^s (either defined for each source case or more globally).

The adaptation defined in (15) coincides with the $\dot{+}^d$ -adaptation with $\text{DK} = \mathcal{U}$ and d defined by

$$d((x^s, y^s), (x^t, y^t)) = |x^t - x^s| + |y^t - y^s - \varrho^s(x^t - x^s)| \quad (16)$$

(the value of ϱ^s when $\{(x^s, y^s)\}$ is not a source case does not matter here so it can be chosen arbitrarily).

From this, conclusions similar to the conclusions about rule-based adaptation can be drawn. First, $\dot{+}$ -adaptation makes it possible to incorporate domain knowledge in this first-order approximation. Then, the knowledge of the adaptation knowledge (ϱ^s) can be used to parametrize the revision operator.

Let $(\vec{\varepsilon}_1, \vec{\varepsilon}_2)$ be the canonical base of \mathbb{R}^2 : $\vec{\varepsilon}_1 = (1, 0)$, $\vec{\varepsilon}_2 = (0, 1)$. Let $\{(x^s, y^s)\}$ be a fixed source case. The distance defined by (16) is the L_1 distance parametrized by the base (\vec{e}_1, \vec{e}_2) with $\vec{e}_1 = \vec{\varepsilon}_1 + \varrho^s \vec{\varepsilon}_2$ and $\vec{e}_2 = \vec{\varepsilon}_2$ (ϱ^s is a constant here, since $\{(x^s, y^s)\}$ is fixed). The base (\vec{e}_1, \vec{e}_2) is orthogonal w.r.t. $(\vec{\varepsilon}_1, \vec{\varepsilon}_2)$ iff $\varrho^s = 0$. Now, when $\varrho^s = 0$, d can be written as in (13) thus $\dot{+}^d$ is an orthogonal revision operator (which justifies a posteriori the adjective “orthogonal”: cf. note 7). So, if the base \mathcal{B} is orthogonal, $\dot{+}^d$ -adaptation is a conservative adaptation.

Other approaches to adaptation

There are other adaptation approaches described in the literature. Most of them are domain-specific and defined by their algorithms. However, there is at least one other general approach to adaptation: the case-based adaptation [17, 18] (also known as *recursive CBR* [19]). The idea is that the adaptation of a CBR system can be a CBR system itself where cases are *adaptation cases*. This approach to adaptation still remains to be formalized in order to be compared to revision-based adaptation.

4.7 Other studies related to revision-based adaptation

Some other studies related to revision-based adaptation have been carried out. Two of them are explained below.

Adaptation in the description logic \mathcal{ALC}

Description logics (DLs) form a family of formalisms that are equivalent to fragments of first-order logic [20].¹⁰ The most used semantics of DLs is based on the theory of models, as this is the case for, e.g., propositional logic. However, given the collection \mathcal{U} of all the interpretations of a DL, it is uneasy (if not impossible) to define a distance on \mathcal{U} . Therefore, the authors have not considered any more the idea of defining a $\dot{+}^d$ revision operator for a DL and have searched in another direction.

In [21], the existence of a belief contraction operator satisfying the postulates of contraction is studied for various DLs (belief contraction is an operator on belief bases that has been related to belief revision through the so-called Harper and Levi identities). There are also some research and implementations on the related issue of repairing inconsistent DL knowledge bases [22].

In [23], an approach to adaptation in the DL \mathcal{ALC} (the simplest propositionally closed DL) has been proposed that is inspired by the principle of revision-based adaptation (though it has not defined a revision operator dealing with any knowledge bases of this DL). The principle of this adaptation in \mathcal{ALC} is based on the semantic tableau method and on inconsistency repairing. The semantic tableau method is used to check the consistency of a DL knowledge base. Roughly said, it consists in applying a set of deductive rules whenever it is possible and then, to detect the clashes (a clash is a contradiction of an easy to detect predefined form). If there is a clash in each reasoning branch, then the knowledge base is inconsistent, otherwise it is consistent (provided that the set of rules have some completeness property). The adaptation in \mathcal{ALC} consists in (1) “pretending” that the source case solves the target problem, (2) applying the semantic tableau method which leads to clashes, and (3) applying a clash repairing strategy that minimizes a repair cost.

The following example illustrates this approach to adaptation in \mathcal{ALC} , using the first-order logic syntax. This example is in the cooking domain and cases represent recipes. The domain knowledge is expressed by the following formula:

$$\begin{aligned} \text{DK} = & \forall x \text{ gratedRawCarrot}(x) \Leftrightarrow \text{carrot}(x) \wedge \text{raw}(x) \wedge \text{grated}(x) \\ & \wedge \forall x \text{ root}(x) \Leftrightarrow \text{carrot}(x) \vee \text{parsnip}(x) \vee \text{celery}(x) \\ & \wedge \forall x \text{ parsnip}(x) \Rightarrow \neg \text{raw}(x) \end{aligned}$$

(the first line expresses what grated raw carrots are, the second line means that the only roots that are considered in this example are carrots, parsnips and celeries, the third line means that a parsnip in a recipe must not be raw). The source case is represented by a formula about a constant σ representing a carrot salad, which is a starter with grated raw carrot and vinaigrette as ingredients:

$$\begin{aligned} \text{Source} = & \text{starter}(\sigma) \wedge \exists x \text{ ingredient}(\sigma, x) \wedge \text{gratedRawCarrot}(x) \\ & \wedge \exists x \text{ ingredient}(\sigma, x) \wedge \text{vinaigrette}(x) \end{aligned}$$

The target case expresses the fact that the user wants the recipe of a starter without carrot in it, by the following formula about the constant θ :

$$\text{Target} = \text{starter}(\theta) \wedge \neg(\exists x \text{ ingredient}(\theta, x) \wedge \text{carrot}(x))$$

¹⁰ Some DLs use constructs that go beyond standard first-order logic, such as concrete domains, but this is not the case for \mathcal{ALC} .

The three steps of the adaptation are as follows. (1) To pretend that the source case solves the target case amounts to identify σ and θ ($\sigma = \theta$). (2) $\text{DK} \wedge \text{Source} \wedge \text{Target} \wedge (\sigma = \theta)$ is inconsistent; the tableaux method leads to the clash

$$\text{between } \exists x, \text{ingredient}(\sigma, x) \wedge \text{carrot}(x) \quad (17)$$

$$\text{and } \neg \exists x, \text{ingredient}(\theta, x) \wedge \text{carrot}(x) \quad (18)$$

(3) In order to repair this clash, the piece of knowledge of equation (17) is removed but its consequence (given DK)

$$\exists x, \text{ingredient}(\theta, x) \wedge \text{root}(x) \wedge \text{raw}(x) \wedge \text{grated}(x) \quad (19)$$

is kept. Thus, from (18), (19) and the domain knowledge, it can be deduced that

$$\exists x, \text{ingredient}(\theta, x) \wedge \text{parsnip}(x) \wedge \text{raw}(x) \wedge \text{grated}(x) \quad (20)$$

$$\text{or } \exists x, \text{ingredient}(\theta, x) \wedge \text{celery}(x) \wedge \text{raw}(x) \wedge \text{grated}(x) \quad (21)$$

Now (20) leads to another clash, since the parsnip must not be raw. By contrast, the celery can be eaten raw and does not lead to another clash. Therefore, the piece of knowledge (20) is disregarded whereas (21) is kept and, finally:

$$\begin{aligned} \text{ComplTarget} \equiv & \text{DK} \wedge \text{starter}(\theta) \\ & \wedge \exists x \text{ ingredient}(\sigma, x) \wedge \text{celery}(x) \wedge \text{raw}(x) \wedge \text{grated}(x) \\ & \wedge \exists x \text{ ingredient}(\sigma, x) \wedge \text{vinaigrette}(x) \end{aligned}$$

Adaptation of qualitative constraint networks

Qualitative algebras (QAs) constitute a family of knowledge representation formalisms. For example, Allen algebra is a well-known QA dedicated to temporal representation [24] and RCC8 is a well-known QA dedicated to spatial representation [25]. A qualitative constraint network (QCN) is a knowledge base of a QA; it is given by a set of constraints between variables. Belief revision has been studied for QAs in [26]: a revision operator $\dot{+}$ associating to a QCN ψ and a QCN μ a QCN $\psi \dot{+} \mu$ is defined, following principles of distance-based revision operators.¹¹

Therefore, revision-based adaptation can be applied to a CBR system in which cases are represented by QCNs. This has been studied in [27]. In a first application, revision-based adaptation has been applied to adapt the preparation parts of cooking recipes, for the system Taaable. The QA used was INDU [28], an extension of the Allen algebra. In a second application, revision-based adaptation has been applied to adapt a crop allocation in a farmland. The QA used was RCC8. This work has raised some implementation issues that are addressed in the paper.

¹¹ \mathcal{U} corresponds to the set of scenarios, i.e., fully constrained QCNs. $\text{Mod}(\varphi)$ is the set of the scenarios that are more specific than the QCN φ . A distance d between scenarios is defined. One problem is that if $A = \text{Mod}(\psi)$ and $B = \text{Mod}(\mu)$, $A \dot{+}^d B$ is not necessarily representable by a QCN (i.e., postulate $(\dot{+}6)$ does not hold). In this case, to write it in a simplified manner, a most specific QCN χ is chosen such that $\text{Mod}(\chi) \supseteq A \dot{+}^d B$, and $\psi \dot{+} \mu = \chi$.

The figure 5 illustrates revision-based adaptation in the Allen algebra (the figure represents the different pieces of knowledge in this formalism—using the qualitative constraint network notations for the most part—and the captions describe them in natural language, for reader non familiar with the Allen algebra). The story underlying this example is an attribution of schedule to teachers. The source case corresponds to the previous year. The target case corresponds to the fact that, for some reason, the French teacher and the Math teacher want to avoid each other. The distance used in this example is based on the distance between the Allen basic relations which are the distances in the relation neighbourhood graph of this formalism (see [29]).

5 Revision-based CBR

Let SOURCE be the union of all the cases from the case base:

$$\text{SOURCE} = \bigcup_i \text{Source}_i \text{ where } \text{CaseBase} = \{\text{Source}_i\}_i$$

The following question can be raised: according to what conditions can the CBR process with SOURCE as only source case be equivalent to the CBR process with CaseBase ? This question is addressed below with a $\dot{+}^d$ -adaptation.

Let A_1, A_2, \dots, A_n , and B be $n + 1$ subsets of \mathcal{U} . Let $\delta_i = d(A_i, B)$ and $\Delta = \min_i \delta_i$. The following equation holds:

$$\left(\bigcup_i A_i \right) \dot{+}^d B = \bigcup_{i, \delta_i = \Delta} (A_i \dot{+}^d B)$$

Indeed $d(\bigcup_i A_i, b) = \min_i d(A_i, b)$ for any $b \in B$, and so $(\bigcup_i A_i) \dot{+} B = \{b \in \mathcal{U} \mid \min_i d(A_i, b) = \Delta\} = \bigcup_{i, \delta_i = \Delta} (A_i \dot{+} B)$.

From this equation applied to $A_i = \text{DK} \cap \text{Source}_i$ and $B = \text{DK} \cap \text{Target}$, it comes that the $\dot{+}^d$ -adaptation of SOURCE to solve Target gives COMPL_TARGET such that

$$\text{COMPL_TARGET} = \bigcup_{i, \delta_i = \Delta} \text{ComplTarget}_i$$

where ComplTarget_i is the result of the $\dot{+}^d$ -adaptation of Source_i to solve Target .

First, let us consider that there is only one i such that $\delta_i = \Delta$. Then $\text{COMPL_TARGET} = \text{ComplTarget}_i$. Therefore if the retrieval process aims at selecting the source case $\text{Source}_i \in \text{CaseBase}$ that minimizes $d(\text{DK} \cap \text{Source}_i, \text{DK} \cap \text{Target})$ then

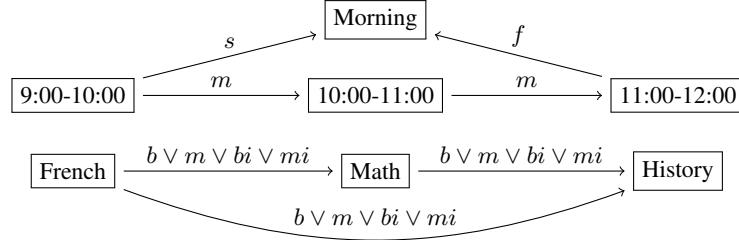
$$\text{CBR}(\{\text{SOURCE}\}, \text{Target}) = \text{CBR}(\text{CaseBase}, \text{Target}) \quad (22)$$

Now, let us consider that there are ex aequo source cases for such a retrieval process: there are several source cases Source such that $d(\text{DK} \cap \text{Source}, \text{DK} \cap \text{Target}) = \Delta$. Then the equation (22) still holds if the two following modifications are made:

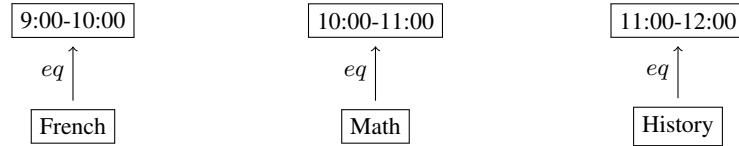
- Retrieval returns the set S of source cases Source minimizing $d(\text{DK} \cap \text{Source}, \text{DK} \cap \text{Target})$;

for each $x \in \{\text{French}, \text{History}, \text{Math}\}$ and each $y \in \{9:00-10:00, 10:00-11:00, 11:00-12:00\}$

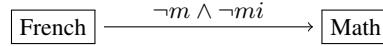
$$x (eq \vee m \vee mi \vee b \vee bi) y \quad x (s \vee d \vee f) \text{ Morning}$$



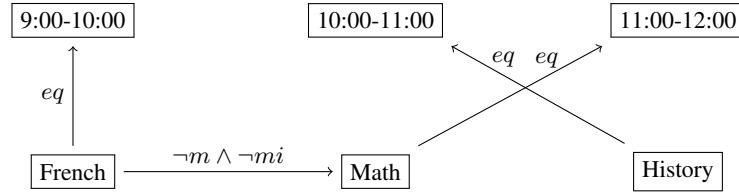
- (a) DK. For each of the three courses and each of the three slots, either the course coincides with the slot or it has an intersection of length 0 with it, moreover, the course takes place during the morning. The slot 10:00-11:00 (resp., 11:00-12:00) immediately follows the slot 9:00-10:00 (resp., 10:00-11:00). There is no intersection of length non null between two different courses.



- (b) Source. The previous year, the slot 9:00-10:00 (resp., 10:00-11:00, 11:00-12:00) was attributed to French (resp., Math, resp., History).



- (c) Target. The French course and the Math course must not meet any more. (Technically, $\neg m \wedge \neg mi$ is an abbreviation for $eq \vee b \vee bi \vee d \vee di \vee f \vee fi \vee o \vee oi \vee s \vee si$.)



- (d) Comp1Target (without the pieces of knowledge given by DK). The adaptation has consisted in the exchange of slots between the Math teacher and the History teacher.

Fig. 5: Example of revision-based adaptation in the Allen algebra.

- Adaptation first performs a $\dot{+}^d$ -adaptation of each $\text{Source} \in S$ and then takes the union of the results.

Therefore, if the distance d used for $\dot{+}^d$ -adaptation is also used for the retrieval process as it is described above, the whole CBR inference can be specified by $\dot{+}^d$ and DK; DK is the “static” knowledge (stating that some case instances are not appropriate) and d is the “dynamic” knowledge about the modification from a case instance to another one. This can be linked to the general principle of “adaptation-guided retrieval” [30] stating that the adaptation knowledge should be used during retrieval: a source case must be preferred to another one if the former requires less “adaptation effort” (this adaptation effort being measured thanks to d for $\dot{+}^d$ -based CBR).

6 REVISOR: a set of revision-based adaptation engines

REVISOR gathers several adaptation engines under GPL license and is available at <http://revisor.loria.fr>. In the current version, several engines have been developed.

REVISOR/PL implements $\dot{+}^d$ and the corresponding revision-based adaptation, where the formalism is propositional logic with a finite number of variables and where d is the weighted Hamming distance between interpretations.

REVISOR/PLAK implements $\dot{+}^d$ and the corresponding revision-based adaptation, where the formalism is propositional logic with a finite number of variables and where d is defined by equation (14) with d_0 a weighted Hamming distance and AK a set of adaptation rules defined by the user, together with their costs.

REVISOR/CLC implements $\dot{+}^d$ and the corresponding revision-based adaptation, where formulas are conjunctions of linear constraints with variables on \mathbb{Z} and on \mathbb{R} and where d is a L_1 distance.

REVISOR/QA implements $\dot{+}^d$ and the corresponding revision-based adaptation, in one of the following qualitative algebras: the Allen algebra, INDU and RCC8.

REVISOR/CLC and REVISOR/QA have been used for the system Taaable.

7 Conclusion

Case-based reasoning systems use similarity (usually in the form of a similarity measure or a distance). This is obvious for the retrieval of a case similar to the target case but this chapter shows how it can be used for adaptation: an important class of revision operators is based on distances. Indeed, $\dot{+}^d$ -based adaptation can be reformulated as the process of selecting the case instances that are the closest ones to the source case, in the metric space (\mathcal{U}, d) , with constraints given by DK. Since adaptation aims at solving a certain type of analogical problem (in which two of the four elements in the analogy are problems, the other ones—including the unknown—are solutions), this approach concretely relates analogical reasoning with belief revision.

This approach to adaptation has a good level of generality since it captures some other approaches to adaptation. This is useful for at least two reasons. First, it proposes

a general framework for specifying adaptation, covering many approaches. Second, it enables to modify an existing adaptation process by taking into account the domain knowledge in this approach. For instance, rule-based adaptation does not consider the domain knowledge (unless it is considered in the adaptation rules): it works on the case universe \mathcal{U} . Revision-based adaptation enables to re-specify it and then to have it working in the case universe $\mathcal{U} \cap \text{DK}$ (i.e., the case universe without the case instances known to be illicit).

Finally, from our own experience, this way to consider adaptation, has appeared as a useful guide to specify and to realize adaptation procedures.

However, even if revision-based adaptation would capture all the approaches to adaptation (and we do not claim that it is the case), it would not close the investigations about adaptation in CBR. Indeed, a revision operator is parametrized by a topology (usually a distance) and the issue of the choice of an appropriate topology, is far from being completely addressed. In fact, the choice of this topology is an adaptation knowledge acquisition issue and an important aspect of the research on revision-based adaptation is to have related this topology of the case universe with the adaptation knowledge.

References

1. Riesbeck, C.K., Schank, R.C.: Inside Case-Based Reasoning. Lawrence Erlbaum Associates, Inc., Hillsdale, New Jersey (1989)
2. Alchourrón, C.E., Gärdenfors, P., Makinson, D.: On the Logic of Theory Change: partial meet functions for contraction and revision. *Journal of Symbolic Logic* **50** (1985) 510–530
3. Maximini, K., Maximini, R., Bergmann, R.: An investigation of generalized cases. In Ashley, K.D., Bridge, D., eds.: Proceedings of the 5th International Conference on Case Base Reasoning (ICCBR'03). Volume 2689 of LNAI., Trondheim, Norway, Springer (June 2003) 261–275
4. Carbonell, J.G.: Learning by analogy: Formulating and generalizing plans from past experience. In R. S. Michalski and J. G. Carbonell and T. M. Mitchell, ed.: Machine Learning, An Artificial Intelligence Approach. Morgan Kaufmann, Inc. (1983) 137–161
5. Carbonell, J.G.: Derivational analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition. In: Machine Learning. Volume 2. Morgan Kaufmann, Inc. (1986) 371–392
6. Katsuno, H., Mendelzon, A.: Propositional knowledge base revision and minimal change. *Artificial Intelligence* **52**(3) (1991) 263–294
7. Dalal, M.: Investigations into a theory of knowledge base revision: Preliminary report. In: AAAI. (1988) 475–479
8. Konieczny, S., Lang, J., Marquis, P.: DA² merging operators. *Artificial Intelligence* **157**(1-2) (2004) 49–79
9. Lieber, J.: Application of the Revision Theory to Adaptation in Case-Based Reasoning: the Conservative Adaptation. In: Proceedings of the 7th International Conference on Case-Based Reasoning (ICCBR-07). Lecture Notes in Artificial Intelligence 4626. Springer, Belfast (2007) 239–253
10. Kolodner, J.: Case-Based Reasoning. Morgan Kaufmann, Inc. (1993)
11. Karmarkar, N.: A new polynomial-time algorithm for linear programming. *Combinatorica* **4**(4) (1984) 373–396
12. Cojan, J., Lieber, J.: Belief Merging-based Case Combination. In: Case-Based Reasoning Research and Development (ICCBR 2009). (2009) 105–119

13. Blansché, A., Cojan, J., Dufour Lussier, V., Lieber, J., Molli, P., Nauer, E., Skaf Molli, H., Toussaint, Y.: TAAABLE 3: Adaptation of ingredient quantities and of textual preparations. In: 18th International Conference on Case-Based Reasoning - ICCBR 2010, "Computer Cooking Contest" Workshop Proceedings. (2010)
14. Lieber, J., Napoli, A.: Correct and Complete Retrieval for Case-Based Problem-Solving. In Prade, H., ed.: Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98), Brighton, United Kingdom. (1998) 68–72
15. Craw, S., Wiratunga, N., Rowe, R.C.: Learning adaptation knowledge to improve case-based reasoning. *Artificial Intelligence* **170**(16-17) (2006) 1175–1192
16. d'Aquin, M., Badra, F., Lafrogne, S., Lieber, J., Napoli, A., Szathmary, L.: Case Base Mining for Adaptation Knowledge Acquisition. In Veloso, M.M., ed.: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07), Morgan Kaufmann, Inc. (2007) 750–755
17. Jarmulak, J., Craw, S., Rowe, R.: Using Case-Base Data to Learn Adaptation Knowledge for Design. In: Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI'01), Morgan Kaufmann, Inc. (2001) 1011–1016
18. Leake, D.B., Kinley, A., Wilson, D.C.: Acquiring Case Adaptation Knowledge: A Hybrid Approach. In: AAAI/IAAI. Volume 1. (1996) 684–689
19. Stahl, A., Bergmann, R.: Applying Recursive CBR for the Customization of Structure Products in an Electronic Shop. In Blanzieri, E., Portinale, L., eds.: Advances in Case-Based Reasoning — Proceedings of the fifth European Workshop on Case-Based Reasoning (EWCBR-2k). Lecture Notes in Artificial Intelligence 1898. Springer (2000) 297–308
20. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., eds.: The Description Logic Handbook. Cambridge University Press, cambridge, UK (2003)
21. Flouris, G., Plexousakis, D., Antoniou, G.: On Applying the AGM theory to DLs and OWL. In Gil, Y., Motta, E., eds.: Proceedings of the 4th International Semantic Web Conference (ISWC 2005). LNCS 3729, Springer (November 2005) 216–231
22. Kalyanpur, A., Parsia, B., Sirin, E., Hendler, J.: Debugging unsatisfiable classes in OWL ontologies. *Web Semantics: Science, Services and Agents on the World Wide Web* **3**(4) (2005) 268–293
23. Cojan, J., Lieber, J.: An Algorithm for Adapting Cases Represented in \mathcal{ALC} . In: 22th International Joint Conference on Artificial Intelligence, Barcelone Espagne (07 2011)
24. Allen, J.F.: An interval-based representation of temporal knowledge. In: Proceedings 7th International Joint Conference on Artificial Intelligence (IJCAI 1981). (1981) 221–226
25. Randell, D.A., Cui, Z., Cohn, A.G.: A spatial logic based on regions and connection. In: Knowledge Representation. (1992) 165–176
26. Condotta, J.F., Kaci, S., Marquis, P., Schwind, N.: A Syntactical Approach to Qualitative Constraint Networks Merging. In: Proc. of the 17th LPAR (Logic for Programming, Artificial Intelligence and Reasoning). (2010) 233–247
27. Dufour-Lussier, V., Le Ber, F., Lieber, J., Martin, L.: Adapting Spatial and Temporal Cases. In Ian Watson, B.D.A., ed.: International Conference for Case-Based Reasoning. Volume 7466 of Lecture Notes in Artificial Intelligence., Lyon, France, Amélie Cordier, Marie Lefevre, Springer (September 2012) 77–91
28. Pujari, A.K., Kumari, G.V., Sattar, A.: INDU: An Interval & Duration Network. Advanced Topics in Artificial Intelligence (1999) 291–303
29. Ligozat, G.: On generalized interval calculi. In: Proceedings of the 9th National Conference of the American Association for Artificial Intelligence (AAAI), Anaheim, CA, AAAI Press/MIT Press (1991) 234–240
30. Smyth, B., Keane, M.T.: Using adaptation knowledge to retrieve and adapt design cases. *Knowledge-Based Systems* **9**(2) (1996) 127–135

Heuristic-Driven Theory Projection: An Overview

Martin Schmidt, Ulf Krumnack, Helmar Gust, and Kai-Uwe Kühnberger

Institute of Cognitive Science, University of Osnabrück

Abstract. This paper provides a concise overview of Heuristic-Driven Theory Projection (HDTP), a powerful framework for computing analogies. The paper attempts to illuminate HDTP from several different perspectives. On the one hand, the syntactic basis of HDTP is formally specified, in particular, restricted higher-order anti-unification together with a complexity measure is described as the core process to compute a generalization given two input domains (source and target). On the other hand, the substitution-governed alignment and mapping process is analyzed together with the transfer of knowledge from source to target in order to induce hypotheses on the target domain. Additionally, this paper presents some core ideas concerning the semantics of HDTP as well as the algorithm that computes analogies given two input domains. Finally, some further remarks describe the different (but important) roles heuristics play in this framework.

1 Introduction

Analogy making is a high-level cognitive process [1, 3, 18] that occurs in every day life. It is considered a core component of cognition [10]. Analogies use already known information from a *source domain* to acquire knowledge in new situations in the *target domain*. They require an abstract mapping to be established between these domains. Empirical research supports the belief that structural commonalities between the two domains are the main guidance for the construction of this mapping. Many computational models, most prominently Gentner’s structure mapping engine (SME) [2], have been proposed. Altogether, many different mechanisms to analyze and extract structural commonalities have been developed [6, 17, 19]. A recent survey of different analogy models is presented in [7]. For more information on analogies and analogical reasoning the interested reader is invited to have a look at the extensive catalogue of works compiled by [8], which will convey a good impression of the ubiquitousness of the topic.

Heuristic-driven Theory Projection (HDTP) [11] is a logic based analogy framework and thereby differs from many of above mentioned models, that are mostly psychologically or neurally inspired. The domains given as input to HDTP are a sets of first order formulas that constitute the domain’s axiomatization, and an analogy is computed based on syntactic generalization using anti-unification. In this respect HDTP resembles earlier works by Hirowatari [16], Hirata [14], and Furtado [4]. However, it should be noted that the overall setup as well as

the role of generalization in these approaches is fundamentally different. HDTP has a strict separation of input domains that can be described over different signatures. Furthermore, in HDTP the idea of structure mapping is crucial and therefore a restricted form of higher-order anti-unification is used to capture even deep structural commonalities between formulas.

This work provides a basic overview with references to extensions of HDTP. An introduction in regards to the framework with its different phases, is outlined in section 2. Section 3 is dedicated to explaining restricted higher-order anti-unification, which is a central mechanism of the framework. Section 4 introduces general concepts used in the core phase named mapping. Section 6 complements the previous syntactic description by a brief outline of semantic aspects. The following section 7 explains how mapping can be viewed as a search problem. It gives a general outline of how HDTP’s mapping algorithm is structured using heuristics. Goals and characteristics of heuristics used by strategies for mapping, are discussed in section 8. Section 5 provides a short look on how the generated mappings can be used in the transfer phase. The paper closes with a brief summary.

2 Heuristic-Driven Theory Projection

2.1 Principles

Heuristic-driven Theory Projection is a formal framework for solving analogies, based on well-known standard symbolic methods of artificial intelligence [11]. HDTP’s input consists of a finite many-sorted first-order logic axiomatization of a source and a target domain. The use of first-order logic makes it easy to incorporate classical reasoning mechanisms of artificial intelligence and also distinguishes it from other often neurally inspired theories of analogical reasoning. However, HDTP still retains the goal of only drawing inferences that are cognitively inspired.

HDTP analyzes source and target domains to determine common structural patterns. These patterns are represented by a mapping between source and target domain via a generalized axiom system that describes the generalized theory of both input domains. The process of finding common structural patterns is described as aligning or mapping of source and target domain and the result is called an *alignment* or *mapping*. The source domain usually contains a set of formulas that axiomatizes an informationally richer domain than the target domain. Hence, an analogical mapping can be used to transfer knowledge from the better understood source, to the target domain, thereby fulfilling the goal of analogy making to gain more information about the target domain. However, transferred knowledge is not factual information but rather hypotheses that have to be tested for consistency and further evaluated. Thus, it is possible to infer new knowledge in a creative way. Depending on the domains compared and the representations chosen, analogical reasoning can result in a number of different analogies and inferences.

The following sections 2.2 and 2.3 focus on a brief overview about the general process flow and application of formal mechanisms that HDTP employs.

2.2 Language

The well-known *Rutherford Analogy*, which compares the solar system to the Rutherford atom model, is used as a running example to show the different aspects of HDTP's language for description of domains. Figure 1 shows a formalization of the Rutherford Analogy. The source domain on the left side is the solar system. The target domain on the right side is the Rutherford atom model. Both theories are simplified and not considered to be complete and accurate physical theories of the described objects. Analogies where source and target domain are from different fields of knowledge, or the domains only present a partial view of the underlying theory, seem to be especially creative. As can be seen by the amount of formulas the source domain has a much richer formalization. The source domain describes that a planet revolves around the sun due to gravitational forces. At the same time, the sun and the planets do not collide with each other. This is considered in relation to the Rutherford atom model where lightweight electrons are attracted by the nucleus due to the Coulomb force. However, the electrons and the nucleus keep a distance of greater than zero, which is an abstraction of the results of Rutherford's gold foil experiment [26].

Domain descriptions can contain not only facts, but also general laws as seen in the formalization of the Rutherford atom model. To capture all the information present in the Rutherford analogy, HDTP needs an expressive formal language to specify axioms of a domain. We now provide a brief overview of the language that encodes knowledge of a domain in HDTP.

Symbols like *sun* and *nucleus* are constants in their respective domain because they have an arity of zero and are of sort object. *Real*, *object* and *time* are sorts used to tag the type of a function, which can then be used as an explicit guide in the structural information mapping employed in HDTP. They can be seen as high-level concepts in an ontology. Functions with positive arity such as *mass*, which takes one argument, are used to map constants to constants. We will denote Variables always by strings with an upper case letter.

Atomic formulas are build from predicate symbols and can be used to express relations. Predicate symbols in our example formalization include $>$, $=$ and $<$. An example for an atomic formula is $\text{mass}(\text{sun}) > \text{mass}(\text{planet})$.

Formulas are built from *atomic formulas* with the help of *logical symbols* in the form of *propositional connectives* and *quantifiers*. The *propositional connectives* \vee , \wedge , \rightarrow , \leftrightarrow and \neg are used to express complex facts. Using the aforementioned components complex expressions can be formed by chaining. Furthermore, complex rules can be formed by using the classical logical *quantifiers* \forall and \exists . Such a rule can be a law like "*every two bodies with positive mass will attract each other*".

A consistent, finite set of formulas over a that vocabulary is referred to as an axiomatization of a domain. Formulas that can be logically inferred from these

| Solar System | Rutherford Atom |
|---|---|
| <p>sorts</p> <p><i>real, object, time</i></p> <p>constants</p> <p><i>sun : object, planet : object</i></p> <p>functions</p> <p><i>mass : object → real</i> <i>dist : object × object × time → real</i> <i>force : object × object × time → real</i> <i>gravity : object × object × time → real</i> <i>centrifugal : object × object × time → real</i></p> <p>predicates</p> <p><i>revolves_around : object × object</i></p> <p>facts</p> <p>$\alpha_1 : \text{mass}(\text{sun}) > \text{mass}(\text{planet})$ $\alpha_2 : \text{mass}(\text{planet}) > 0$ $\alpha_3 : \forall(T : \text{time}) : \text{gravity}(\text{planet}, \text{sun}, T) > 0$ $\alpha_4 : \forall(T : \text{time}) : \text{dist}(\text{planet}, \text{sun}, T) > 0$</p> <p>laws</p> <p>$\alpha_5 : \forall(T : \text{time}, O_1 : \text{object}, O_2 : \text{object}) :$ $\quad \text{dist}(O_1, O_2, T) > 0$ $\quad \wedge \text{gravity}(O_1, O_2, T) > 0$ $\quad \rightarrow \text{centrifugal}(O_1, O_2, T) =$ $\quad -\text{gravity}(O_1, O_2, T)$</p> <p>$\alpha_6 : \forall(T : \text{time}, O_1 : \text{object}, O_2 : \text{object}) :$ $\quad 0 < \text{mass}(O_1) \wedge 0 < \text{mass}(O_2)$ $\quad \wedge \text{dist}(O_1, O_2, T) > 0$ $\quad \wedge \text{centrifugal}(O_1, O_2, T) < 0$ $\quad \rightarrow \text{revolves_around}(O_1, O_2)$</p> | <p>sorts</p> <p><i>real, object, time</i></p> <p>constants</p> <p><i>nucleus : object, electron : object</i></p> <p>functions</p> <p><i>mass : object → real</i> <i>dist : object × object × time → real</i> <i>coulomb : object × object × time → real</i></p> <p>facts</p> <p>$\beta_1 : \text{mass}(\text{nucleus}) > \text{mass}(\text{electron})$ $\beta_2 : \text{mass}(\text{electron}) > 0$ $\beta_3 : \forall(T : \text{time}) :$ $\quad \text{coulomb}(\text{electron}, \text{nucleus}, T) > 0$ $\beta_4 : \forall(T : \text{time}) :$ $\quad \text{dist}(\text{electron}, \text{nucleus}, T) > 0$</p> |

Fig. 1. Logic formalization of the solar system and the Rutherford atom.

axioms constitute the *domains theory*. Whereas the expressive power of first-order logic allows the possibility of expressing complex systems, at the same time it confronts us with the problem of having more than one equivalent axiomatization for a given domain. However, this expressive power enables not only situation descriptions, but also general laws that are not possible in other well-known analogy models [21]. Furthermore, the representation of knowledge and reasoning over formulas is possible with this logic-based formalization.

2.3 Phases

The formal framework of HDTp can employ the three common phases of *retrieval*, *mapping*, and *transfer* of analogy-making. However, we will mainly focus on the mapping phase in the coming chapters. HDTp is very modular and can be extended with more subprocesses that complement the main phases as seen in figure 2.

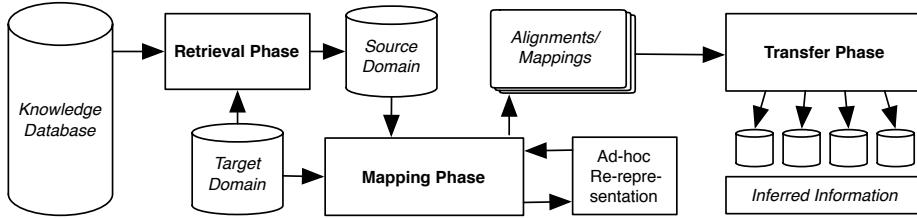


Fig. 2. Overview of HDT's phases.

Retrieval is the process of selecting a suitable set of formulas and therefore theory to represent the source domain. This source domain should be related to the given target domain in that they share a common structure. This can be identified in detail during the mapping phase. Different source domains lead to different analogical inferences in the target domain subsequently. The goal, therefore, is not necessarily to only find the structurally most similar domain to the target domain, but a multitude of source domains leading to different newly discovered knowledge about the target domain. Without a retrieval phase, the source and target domain must be given explicitly to the mapping phase. The modular and sequential nature of the HDT framework makes it possible to add an optional *retrieval* process that generates this input before the mapping phase.

During *mapping*, the analogical relation between the given domains is established by aligning formulas of the source domain with formulas of the target domain. This alignment does not necessarily use all available formulas. Restructuring of the axiomatization while describing the same domain can also lead to different results and may be preferred for building a more plausible outcome depending on the relative goal of the analogy. This difference in mapping possibilities is regarded as the *representation problem*.

Ad-hoc re-representation during mapping can be done by deduction with the help of laws of the respective domain and background-knowledge. An example of background-knowledge is that *distance* is a symmetric function, which can be expressed by the $\forall X \forall Y \forall T : \text{distance}(X, Y, T) = \text{distance}(Y, X, T)$. Therefore, $\text{distance}(\text{nucleus}, \text{electron}, T)$ can be rewritten as $\text{distance}(\text{electron}, \text{nucleus}, T)$ in the axiomatization of the Rutherford atom model. Another more general example of *background-knowledge* is $\forall X \forall Y \forall Z : (X > Y \wedge Y > Z) \rightarrow (X > Z)$ which states the transitivity of the *greater than* relation.

The structural matching within the mapping phase is created stepwise by aligning a formula from the source domain with a formula from the target. The selection criteria for both are guided by heuristics and may use information regarding already aligned parts and still to-be-aligned parts. Foremost, the alignment quality depends on a metric of how well two formulas are equal in structure and semantics. A plausible pair of terms to make an alignment is $\text{mass}(\text{sun}) > \text{mass}(\text{planet})$ and $\text{mass}(\text{nucleus}) > \text{mass}(\text{electron})$ because they have the matching top-level predicate symbol $>$. This is complemented by the

```

types
real, object, time
constants
X : object, Y : object
functions
mass : object → real
dist : object × object × time → real
F : object × object × time → real
centrifugal : object × object × time → real
predicates
revolves_around : object × object
facts
γ1 : mass(X) > mass(Y)
γ2 : mass(Y) > 0
γ3 : ∀(T : time) : F(X, Y, T) > 0
γ4 : ∀(T : time) : dist(Y, X, T) > 0
laws
γ5* : ∀(t : time, O1 : object, O2 : object) :
    dist(O1, O2, T) > 0 ∧ F(O1, O2, T) > 0
    → centrifugal(O1, O2, T) = -F(O1, O2, T)
γ6* : ∀(T : time, O1 : object, O2 : object) :
    0 < mass(O1) < mass(O2) ∧
    dist(O1, O2, T) > 0 ∧
    centrifugal(O1, O2, T) < 0
    → revolves_around(O1, O2)

```

Fig. 3. Generalization of the solar system and the Rutherford atom (including the generalizations from the transfer marked with *).

similarity that *sun*, *planet*, *nucleus* and *electron* are all of the sort *object*. The plausibility of the structural alignment of two formulas can be expressed by a metric assigning costs to mappings. Summation over the costs associated with the metric of all mappings, provides a relative measure of how good an overall domain wide mapping is regarding another alternative mapping. The operation of structural mapping a pair of formulas is carried out by using *anti-unification*. This core operation is very important because it constrains the possible mapping of terms and therefore shapes the generalized theory that HDTP generates. For an explanation of the specific form of anti-unification, which is used by HDTP, see section 3.2.

The generated formalization for the generalized theory of the solar system and Rutherford atom model as computed by HDTP is shown in figure 3. We can observe that our already mentioned alignment candidates of formulas $\text{mass}(\text{sun}) > \text{mass}(\text{planet})$ and $\text{mass}(\text{nucleus}) > \text{mass}(\text{electron})$ were generalized to $\text{mass}(X) > \text{mass}(Y)$. Here uppercase letters are variables that must be appropriately instantiated to gain back the original terms of source and target domain. Notably *electron* in the target domain is the corresponding entity of *planet* in the source domain. The same relation holds between *nucleus* and *sun*.

In the final step of HDTP, called the *transfer* phase, the generated mapping is used to transfer information from the source domain to the target do-

main. Inferred knowledge in the target domain can only be used as a guide or hypothesis that must be proven using other mechanisms of logic or empirical tests outside the theoretic realm. From a cognitive perspective, this can be viewed as the inspirational mechanism that leads the way for creative ideas that have to be put to test by trying them out in a real world setting. The transferred information is usually not already covered by the generalized theory. HDTDP will detect inconsistent inferences and reject them by applying a theorem prover. In contrast, the general usefulness of such transferred information can not be inferred by formal processes. In our running example we can transfer *revolves_around(planet, sun)* from the solar system domain to the Rutherford atom model and infer *revolves_around(electron, nucleus)* by means of the mapping of *planet* to *electron* as well as *sun* to *nucleus*.

3 Anti-unification

3.1 General concepts

The concept of anti-unification was first outlined by Reynolds [24] and Plotkin [23]. It is the formal counterpart to the widely known (weak) *unification*, but in contrast, has less research specifically dedicated to it. Whereas unification is one of the core concepts used in logic programming, best known through the programming language Prolog, anti-unification is mostly used for inductive learning and proof generalization. The following chapter will define a vocabulary that is used to describe anti-unification and especially a form of higher-order anti-unification called *restricted higher-order anti-unification*.

Definition 1 (Substitution). A substitution on terms is a partial function, mapping variables to terms, formally represented by $\tau = \{X_1/t_1, \dots, X_n/t_n\}$ (provided $X_i \neq X_j$ for $i, j \in \{1, \dots, n\}$, $i \neq j$). We say τ acts on variables X_1, \dots, X_n . An application of a substitution τ on a term is defined by simultaneously replacing all occurrences of the variables X_1, \dots, X_n and thus:

$$\begin{aligned} - \text{apply}(X, \tau) &= \begin{cases} t' & \text{if } X/t' \in \tau \\ X & \text{otherwise} \end{cases} \\ - \text{apply}(f(t_1, \dots, t_m), \tau) &= f(\text{apply}(t_1, \tau), \dots, \text{apply}(t_m, \tau)) \end{aligned}$$

Anti-unification operates on terms. These operations are specifically called substitutions and when applied to terms produce altered terms. A simple substitution such as $\tau = \{X/a\}$ applied to a term $t = f(X, b)$ yields a new term $t' = f(a, b)$. If the variable X to be replaced occurs more than once within t , all occurrences of X within t are replaced by the constant a . Substitutions are not restricted to constants but may replace variables with complex terms such as $g(Y, e)$. The application of substitution $\tau = \{X/g(Y, e)\}$ to $f(X, b)$ results in the term $f(g(Y, e), b)$. The extended substitution $\tau = \{X/g(Y, e), Y/b\}$ applied to $f(X, Y)$ results in $f(g(Y, e), b)$ and not $f(g(b, e), b)$.

Different sets of valid substitutions result in different forms of anti-unification. In first-order anti-unification variables $V \in \mathcal{V}$ are allowed to be replaced by terms $t \in \text{Term}(\Sigma, \mathcal{V})$. Gentner et al. [5] revealed empirically that analogies are characterized by deep structural commonalities. In [27] first-order anti-unification was analyzed, which was introduced by Plotkin [23], and found that it is not powerful enough to capture complex and deep structural commonalities between two terms with its limited form of substitutions. It was shown that anti-unification can be altered to recognize complex commonalities by employing higher-order substitutions on terms and thereby produce higher-order terms. An example for a commonality that is not detected by frameworks based on first-order anti-unification is the anti-unification of $f(a, b)$ and $g(b, h(a))$. In first-order this is generalized to a single variable and high-order approaches can retain structure and produce $F(a, b)$ and $F(b, a)$ as generalized term. Examples of logic based analogy frameworks using forms of first-order (anti)-instances are [15] and [4].

Definition 2 (Higher-order Terms). Let Σ be a signature. The set of variables \mathcal{V}^* is the union of infinite sets of variables \mathcal{V}_n for every $n \in \mathbb{N}_0$. We say that a variable has arity n if it is an element of the set \mathcal{V}_n . Every variable $V \in \mathcal{V}$, has the form $V : \sigma_1 \times \sigma_2 \times \dots \times \sigma_n \rightarrow \sigma$ where $\sigma_1, \dots, \sigma_n, \sigma \in \text{Sort}_\Sigma$. The set $\text{Term}^*(\Sigma, \mathcal{V}^*)$ relative to \mathcal{V}^* is defined as the smallest set such that the following conditions hold:

1. If $X : \sigma \in \mathcal{V}_0$ then $X : \sigma \in \text{Term}^*(\Sigma, \mathcal{V}^*)$.
2. If $f : \sigma_1 \times \sigma_2 \times \dots \times \sigma_n \rightarrow \sigma \in \Sigma$ and $t_i : \sigma_i \in \text{Term}^*(\Sigma, \mathcal{V}^*)$, then $f(t_1, \dots, t_n) : \sigma \in \text{Term}^*(\Sigma, \mathcal{V}^*)$.
3. If $F : \sigma_1 \times \sigma_2 \times \dots \times \sigma_n \rightarrow \sigma \in \mathcal{V}^*$ and $t_i : \sigma_i \in \text{Term}^*(\Sigma, \mathcal{V}^*)$, then $F(t_1, \dots, t_n) : \sigma \in \text{Term}^*(\Sigma, \mathcal{V}^*)$.

We call an element of the set $\text{Term}^*(\Sigma, \mathcal{V}^*)$ a term of the term algebra and a term that contains any variable with arity higher than zero a higher-order term.

The set \mathcal{V}_0 contains all first-order variables. One can see that definition 2 is an extended version of the classical first order term algebra and therefore the set of terms $\text{Term}(\Sigma, \mathcal{V})$ is a subset of $\text{Term}^*(\Sigma, \mathcal{V}^*)$. Examples for higher-order terms are $F(a, b)$, $F(X, c)$, $f(G(d))$ and $F(X, G(Y))$.

Working in a higher-order logic and defining arbitrary complex substitutions on predicate and function structures results in many problems [12] and might lead to an overgeneralization. Therefore, substitutions in higher-order anti-unification for analogies had to be restricted to still capture complex structural commonalities and to cognitively adequately express shared structure of domains. This new formalism has been referred to as *restricted higher-order anti-unification* in [20, 29]. Restricted higher-order anti-unification is a weak type of higher-order anti-unification and can be reduced to first-order anti-unification modulo, a suitably restricted equational theory [11, 13].

3.2 Restricted higher-order anti-unification

In this section, the formal framework for restricted higher-order anti-unification is presented. We will only discuss anti-unification of terms. The extension to

atomic formulas is straight forward. For the final extension from atomic formulas to formulas assume that the connectives between atomic formulas and the quantifiers of variables have to literally match, to be anti-unifiable. In general this can be achieved by re-representing formulas into a normal form. For the sake of presentation, we will usually notate terms without their sorts in the following definitions and examples.

Using a higher-order term-algebra as defined in definition 2, we can define the notion of substitutions for restricted higher-order anti-unification. The modifier *restricted* stems directly from the fact that we do not allow arbitrary substitutions on higher-order terms, but only a few restricted types. These types were chosen to gain a useful form of anti-unification. These have proven to be sufficient for the generation of generalized domains in analogical reasoning with HDTP [29].

Definition 3 (Basic Substitutions). *Given the term algebra $\text{Term}^*(\Sigma, \mathcal{V}^*)$, we define the following set of basic substitutions for restricted higher-order anti-unification:*

1. A renaming $\rho_{F'}^F$ replaces a variable $F : \sigma_1 \times \dots \times \sigma_n \rightarrow \sigma \in \mathcal{V}_n$ by a variable $F' : \sigma_1 \times \dots \times \sigma_n \rightarrow \sigma \in \mathcal{V}_n$:

$$F(t_1 : \sigma_1, \dots, t_n : \sigma_n) : \sigma \xrightarrow{\rho_{F'}^F} F'(t_1 : \sigma_1, \dots, t_n : \sigma_n) : \sigma$$

2. A fixation ϕ_f^F replaces a variable $F : \sigma_1 \times \dots \times \sigma_n \rightarrow \sigma \in \mathcal{V}_n$ by a function symbol $f : \sigma_1 \times \dots \times \sigma_n \rightarrow \sigma \in \text{Func}_\Sigma$:

$$F(t_1 : \sigma_1, \dots, t_n : \sigma_n) : \sigma \xrightarrow{\phi_f^F} f(t_1 : \sigma_1, \dots, t_n : \sigma_n) : \sigma$$

3. An argument insertion $\iota_{G,i}^{F,F'}$ with $0 \leq i \leq n$, $F : \sigma_1 \times \dots \times \sigma_n \rightarrow \sigma \in \mathcal{V}_n$, $G : \sigma_i \times \dots \times \sigma_{i+k-1} \rightarrow \sigma_g \in \mathcal{V}_k$ with $k \leq n - i$ and $F' : \sigma_1 \times \dots \times \sigma_{i-1} \times \sigma_g \times \sigma_{i+k} \times \dots \times \sigma_n \rightarrow \sigma \in \mathcal{V}_{n-k+1}$ is defined as:

$$F(t_1 : \sigma_1, \dots, t_n : \sigma_n) : \sigma \xrightarrow{\iota_{G,i}^{F,F'}}$$

$$F'(t_1 : \sigma_1, \dots, t_{i-1} : \sigma_{i-1}, G(t_i : \sigma_i, \dots, t_{i+k-1} : \sigma_{i+k-1}) : \sigma_g, t_{i+k} : \sigma_{i+k}, \dots, t_n : \sigma_n) : \sigma$$

4. A permutation $\pi_\alpha^{F,F'}$ with $F : \sigma_1 \times \dots \times \sigma_n \rightarrow \sigma \in \mathcal{V}_n$ and $F' : \sigma_1 \times \dots \times \sigma_n \rightarrow \sigma \in \mathcal{V}_n$ together with a bijective function $\alpha : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ which is not the identity function, rearranges the arguments of a term:

$$F(t_1 : \sigma_1, \dots, t_n : \sigma_n) : \sigma \xrightarrow{\pi_\alpha^{F,F'}} F'(t_{\alpha(1)} : \sigma_{\alpha(1)}, \dots, t_{\alpha(n)} : \sigma_{\alpha(n)}) : \sigma$$

We write the chaining of basic substitutions $\tau_1, \tau_2, \dots, \tau_n$ as $[\tau_1, \tau_2, \dots, \tau_n]$ which has the application structure $\text{apply}(\text{apply}(\text{apply}(s, \tau_1), \dots), \tau_n)$ on a term s . The application order therefore is from left to right. Every chain of basic substitutions $[\tau_1, \tau_2, \dots, \tau_n]$ is a substitution.

Renaming is a relatively simple form of substitution. For example the substitution $[\rho_Y^X]$ applied to $f(X)$ results in the term $f(Y)$. This is not restricted to variables with arity zero. Applying $[\rho_G^F]$ to $F(a)$ would for example result in $G(a)$.

Fixation is used to substitute a variable by a symbol of the same arity. Again this is possible for first or higher-order variables.

Argument Insertion or just *insertion* is a more challenging substitution because it can change the argument structure and arity of a term. An example for this is when a variable X is inserted into $F(a, b, c)$ after position 2 to yield $F'(a, b, X, c)$. Note that insertion does only allow for insertion of a variable into an argument slot of a higher-order variable and not a function symbol.

Inserting a variable I of arity higher than zero basically embeds the already present subterms into a new subterm with function name I . An example for this would be $F(a, b, c) \xrightarrow{\iota_{G,1}^{F,F''}} F''(a, G(b, c))$. The insertion of a variable with arity higher than one does decrease the arity of the embedding term and can therefore be better described as embedding than insertion.

Permutation is used for rearranging arguments of a term and does not change the arity of the term. As with insertion, this operation can only operate on a higher-order variable.

Definition 4 (Instance & Anti-Instance). A term t' is an instance of t and t is an anti-instance of t' , if there is a substitution τ such that application of τ on t' results in t . In this case we write $t' \xrightarrow{\tau} t$ or simply $t \rightarrow t'$.

Hence $t' = f(a, b)$ can be called an *instance* of $t = f(X, b)$ and t is an *anti-instance* of t' because $t \xrightarrow{\tau} t'$ where τ is the fixation substitution $[\phi_a^X]$. Note that $f(a, b)$ cannot be an *instance* of $f(X, X)$ because a and b are different terms and the variable X cannot be substituted dependent on position by a fixation substitution into one or the other.

Definition 5 (Term Equivalence). Two Terms t and t' are considered equivalent if $t \rightarrow t'$ and $t' \rightarrow t$. In this case we write $t \equiv t'$. If t and t' are not equivalent we write $t \not\equiv t'$. We say the t and t' are strongly equivalent if $t \xrightarrow{\tau} t'$ and $t' \xrightarrow{\nu} t$ where τ and ν are substitutions of the form $X_1/X'_1, \dots, X_n/X'_n$ and thereby only replace variables by variables.

$F(a, b, c)$ is therefore equivalent to $F'(c, b, a)$ because they can be derived from each other in either direction by a permutation substitution. The equivalence relation has the usual properties reflexivity, symmetry and transitivity. An equivalence class of a term t is the set of all terms that are equivalent to t . Because the equivalence relation is reflexive, such a set contains t itself. If we restrict the substitutions used in the term equivalence definition to renaming substitutions, we write unique up to variable renaming to denote this special equivalence class.

Definition 6 (Anti-Unifier). An anti-unifier for a given set of terms \mathfrak{T} is a term that is an anti-instance of every term of \mathfrak{T} . We call the set \mathfrak{T} the anti-unified terms in context of the anti-unifier.

By this definition the term $f(X, Y)$ is a possible *anti-unifier* for the set of terms $f(a, b)$, $f(c, d)$ and $f(X, c)$.

Definition 7 (Most Specific Anti-Unifier). A *most specific anti-unifier (msa)* also referred to as the least general anti-unifier (*lga*) of a set of terms \mathfrak{T} is an anti-unifier a for \mathfrak{T} such that there exists no anti-unifier a' for \mathfrak{T} with $a \rightarrow a'$ and $a \not\equiv a'$.

The process of finding the *most specific anti-unifier* is called anti-unification. The search for the most specific anti-unifiers means that the number of basic substitution components is kept minimal and only those substitutions necessary to fulfill the conditions for a valid *anti-unifier* are used. Anti-instances that are *most specific anti-unifier* therefore carry maximal information about the common structure of the set of terms anti-unified. If we would change the definition of possible substitutions, the ordering governed by the anti-instance relation could be changed. Therefore the notion of what common structure is will change. The most general anti-unifier possible in restricted higher-order anti-unification is, as in first-order anti-unification, a single zero arity variable. This is a trivial anti-unifier for every set of terms, and guarantees that an anti-unifier always exists for a set of terms if we do not take sorts into account. However, this anti-unifier does not need to be and usually will not be the *most specific anti-unifier*. We will call any anti-unifier together with the corresponding substitutions for a set of two terms a *generalization*.

Definition 8 (Generalization). A *generalization* for a pair of terms $\langle s, t \rangle$ is a triple $\langle g, \tau, \nu \rangle$ with a term g and substitutions τ, ν such that $g \xrightarrow{\tau} s$ and $g \xrightarrow{\nu} t$. The term g is therefore the anti-unifier in a generalization for s and t .

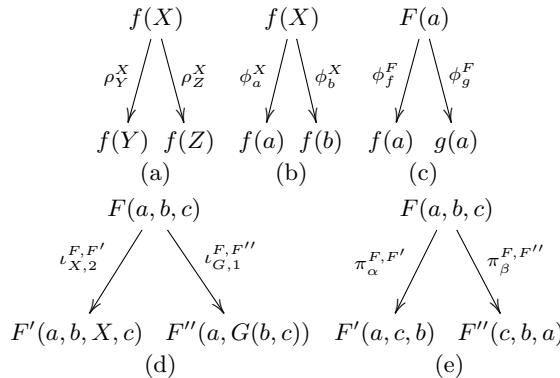


Fig. 4. Examples for all basic substitutions from definition 3.

Generalizations with the earlier discussed examples of basic substitutions can be found in figure 4. The pair of anti-unified terms at the bottom of each generalization depicted, is stated additionally to the anti-unifier placed at the top

and the two corresponding substitutions. Labeled arrows represent the substitutions. The anti-unifier in a generalization is not required to be a most specific anti-unifier for the pair of terms given. Hence, we introduce the notion of least general generalization where the anti-unifier is a most specific anti-unifier for the pair of terms given.

Definition 9 (Least General Generalization). A generalization $\langle g, \tau, \nu \rangle$ for a pair of terms $\langle s, t \rangle$ is called least general generalization if g is a most specific anti-unifier for the terms s and t .

In figure 5 we have three possible generalizations for the terms $f(a, b)$ and $f(c, b)$. Figure 5(a) includes the most general anti-unifier. (c) is the least general generalization with a most specific anti-unifier. (b) is less general than (a), but not a least general generalization because it does not include a most specific anti-unifier.

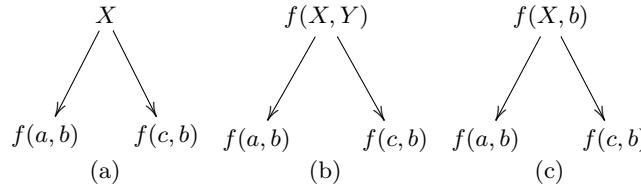


Fig. 5. Examples for generalizations in anti-unification.

Anti-unifiers of restricted higher-order anti-unification always exist for a pair of terms, and the number of anti-unifiers is finite up to the renaming of variables. This can be easily seen, as basic substitutions, and hence chains of basic substitutions, never reduce the number of symbols in a term. Hence, an anti-unifier is always less complex than the anti-unified terms as show in [27]. However, one difference in comparison to first-order anti-unification is that most specific anti-unifiers are no longer unique. As a result, least general generalizations are also not unique anymore in the context of restricted higher-order anti-unification. An example of a pair of terms with more than one most specific anti-unifier is shown in figure 6. Here $f(X, Y)$, $F(d, G(a))$ and $F'(G(a), d)$ are most specific anti-unifiers for the pair of terms $f(g(a, b, c), d)$ and $f(d, h(a))$. All three generalizations shown are therefore least general generalizations.

3.3 Complexity of substitutions

Multiple least general generalizations are not necessary disadvantageous. In analogy making several different mappings with different degrees of plausibility may coexist. We introduce a measure for ranking alternative generalizations, preferring analogies that identify more common structure. We start by assigning a

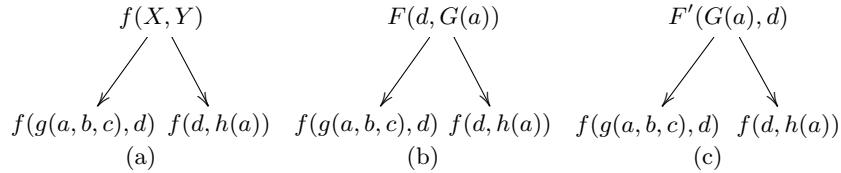


Fig. 6. Example with multiple least general generalizations.

complexity¹ value to substitutions and then extend it to generalizations in a way that promotes their reuse, as this indicates systematic mapping in the resulting analogy.

Definition 10 (Complexity Of Substitutions). *The complexity of a basic substitution τ is defined as:*

$$\mathcal{C}(\tau) = \begin{cases} 0 & \text{if } \tau = \rho & (\text{renaming}) \\ 1 & \text{if } \tau = \phi_f & (\text{fixation}) \\ k+1 & \text{if } \tau = \iota_{V,i} \text{ and } V \in \mathcal{V}_k & (\text{argument insertion}) \\ 1 & \text{if } \tau = \pi_\alpha & (\text{permutation}) \end{cases}$$

For a composition of basic substitutions we define $\mathcal{C}([\tau_1, \dots, \tau_m]) = \sum_{i=1}^m \mathcal{C}(\tau_i)$.

The complexity of a substitution is meant to reflect its computational processing effort. Therefore, permutations have a non-zero complexity even though they do not change the information load of a term. Argument insertion restructures the term, and the higher the arity of the inserted variable, the more arguments are moved. Thus, more complexity is assigned to the argument insertion operation. The renaming substitution does not change the equivalence class and structure of a term and therefore has a complexity measure of zero. A substitution that is composed of basic higher-order substitutions has the complexity of the sum of those basic substitutions. The complexity values for basic substitutions have proven to generate plausible analogies when used by HDTP. The complexity of a generalization can be defined in a straightforward manner by basing it on the complexity of its substitutions.

Definition 11 (Complexity Of Generalizations). *Let $\langle g, \tau, \nu \rangle$ be a generalization for a pair of terms $\langle s, t \rangle$. Define the complexity of the generalization by $\mathcal{C}(\langle g, \tau, \nu \rangle) = \mathcal{C}(\tau) + \mathcal{C}(\nu)$.*

Note that the only situation in which renaming is needed to produce an anti-unifier is when the variables of the two anti-unified terms have variables in the same argument positions and are unifiable. To enforce that renaming substitutions are only used when necessary, we require that chains of basic substitutions

¹ Complexity was chosen here to reflect that this is not concerned with the computation cost involved but with how structurally complex the substitutions and thereby mappings are.

should only have all basic renaming substitutions before all other basic substitutions and are restricted to the minimum amount needed as discussed in [27].

Given the restrictions on the use of renaming we can now define the notion of preferred generalizations.

Definition 12 (Preferred Generalization). *A preferred generalization $\langle g, \tau, \nu \rangle$ is a least general generalization for the pair of terms s and t such that there is no other least general generalization that has less complexity and τ and ν contain only the minimal amount of basic renaming substitutions required at their beginning.*

In contrast to least general generalizations there are only a finite number of preferred generalizations (up to renaming of the anti-unifier and the basic substitution chains) for a pair of terms. These preferred generalizations will be a core building block for mappings between domains constructed in the mapping process of HDTP. The generalization should capture the commonalities of source and target domain, while allowing differences to be replaced in a systematic fashion. Looking for the least general generalization will keep as much of the common structure as possible. More general generalizations may still result in analogical mappings but such mappings will have less structural support. As structural correspondence is considered a key feature of analogies the authors see least generality as a core heuristics for establishing analogies. See Section 4 for a detailed discussion of how they will be used.

3.4 Reuse of substitutions

Anti-unification within HDTP is used to produce a set of anti-unifiers for a set of terms of source and target domains by building generalizations. Subterms within the source and target domains can appear more than once. The constants *sun* and *planet*, for example, are used within the terms $mass(sun) > mass(planet)$ and $dist(planet, sun, T)$ on the source side within the Rutherford analogy formalized in figure 1. Furthermore, in the target domain the terms $mass(nucleus) > mass(electron)$ and $dist(electron, nucleus, T)$ have the same subterms *electron* and *nucleus*. A generalization for $mass(sun) > mass(planet)$ and $mass(nucleus) > mass(electron)$ is shown in figure 7(a). In figure 7(b) the depicted generalization is between $dist(planet, sun, T)$ and $dist(electron, nucleus, T)$.

Both generalizations use two fixations for each substitution and therefore have a complexity of 4. Both generalizations combined have a complexity of 8. However, the fixations used in these two generalizations are the same. It is cognitively plausible if substitutions already used do not add complexity on reuse. Because HDTP builds generalizations sequentially one by one, the process should be that where the generalization in (a) is already made and a new generalization as in (b) is constructed, the complexity assigned to (b) should be 0.

Definition 13 (Complexity Of Reused Substitutions). *Let g_1, g_2, \dots, g_n be generalizations constructed after each other in this order. Any basic substitu-*

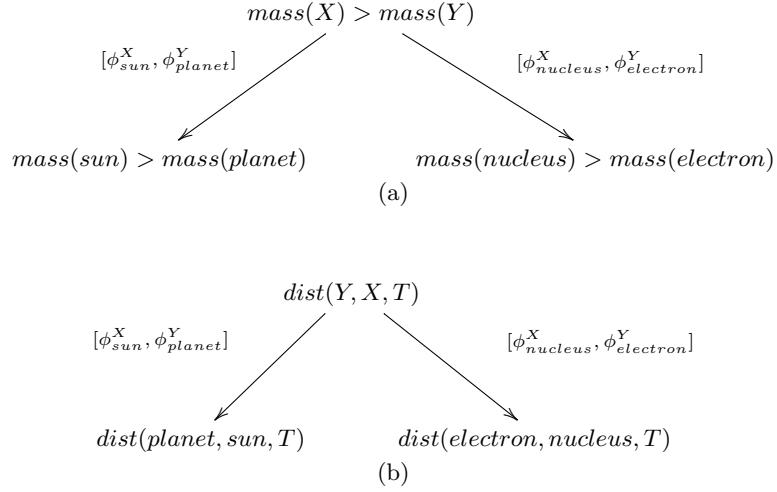


Fig. 7. Examples for generalizations in the Rutherford analogy.

tion τ that is used in g_j with the same parameters as used in g_i has a complexity of 0 if $1 \leq i < j \leq n$.

With this definition the combined complexity of the generalizations in figure 7 sums up to 4 instead of 8 when reuse of substitutions is not allowed. We will present the effect of preferred generalization for analogy making by another example, that may also demonstrate how the detection of structural commonalities is obtained by restricted higher-order anti-unification. The *heat-flow analogy* is traditionally presented in the following form: In the source domain, there are two vessels, a *beaker* and a *vial*, that are connected via a *pipe*. The vessels are filled with water and initially the height of the water in the beaker is higher than the height of the water in the vial. In the target domain, a cup of hot coffee is connected to an ice cube via a silver bar. Without going into too much detail, the following terms are relevant in understanding the analogy:

$$\text{height}(\text{in(water, beaker)}, t) \quad \text{and} \quad \text{temp}(\text{in(coffee, cup)}, t)$$

which could be generalized to $T(\text{in}(X, Y), t)$. However, this generalization would associate water with coffee and hence would predict a flow of coffee instead of the flow of heat. Furthermore, this generalization would not allow for reuse of substitutions for the following pairs of terms:

$$\text{height}(\text{in(water, vial)}, t) \quad \text{and} \quad \text{temp}(\text{ice_cube}, t)$$

A better generalization would employ higher-order substitutions, that can be reused for both pairs of terms and would result in the preferred generalizations depicted in figure 8:

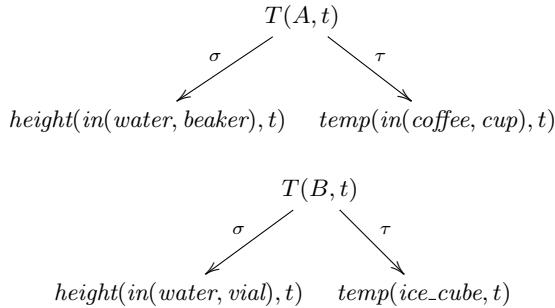


Fig. 8. Examples for generalizations in the heatflow analogy.

In that generalization, the following substitutions are applied:

$$\sigma = \{ T / \text{height}(\text{in}(water, A), t), \quad \tau = \{ T \mapsto \text{temp}(A, t), \\ A / \text{beaker}, \quad A / \text{in}(coffee, cup), \\ B / \text{vial} \} \quad B \mapsto \text{ice_cube} \}$$

We have given the substitutions in a compact notation here, but of course they can be expressed as compositions of the basic substitutions introduced above (see [29] for details).

Notice, that the definition of preferred generalization will yield a form of reuse that is not desired if we do not assume additional restrictions for substitutions in generalizations. The root of the problem is that the variable F in the basic substitutions ϕ_f^F , $\iota_I^{F,F'}$ and $\phi_\alpha^{F,F'}$ is not restricted in relation to the other parameters such as F' , I or α . Therefore, these basic substitutions can be used as if they did an additional basic renaming substitution. The additional restrictions on parameters of basic substitutions to solve this problem were discussed in [28].

The paper [28] also discusses two further aspects of anti-unification. The first is an extension to allow for anti-unification of terms with differing sorts. The purpose of sorts in HDTP is to help restrict the possible mappings during the analogical mapping process by using background knowledge about relations between classes of symbols in the involved domains. For this a fifth basic substitution named sort constriction is added that allows to change the sort of a term according to a predefined ontology of sorts.

The second topic discussed is how symbol mapping restrictions can be enforced by further restricting the use of renaming substitutions. In analogy making restrictions on mapping symbols between domains are often imposed so as to disallow arbitrary mapping from symbols to symbols. The form of anti-unification described so far allows for multiple mappings of one symbol to multiple symbols within the other domain. However, this may not be desired as discussed in [31]. Gentner's Structure-Mapping Theory is based on the finding that people prefer alignments that are structurally consistent [5]. This means that there should be a one-to-one correspondence between elements in source and target domain. The

opposite of this is a many-to-many mapping of symbols that is possible under anti-unification described here.

For an outline of an algorithm to compute preferred generalizations using restricted higher-order anti-unification see [27]. Furthermore, an analysis of the computational complexity has been done in [25]. There it was proven that e.g. due to permutation substitutions the task of restricted higher-order anti-unification is in general NP-hard.

4 Alignments and mappings

Analogous to how anti-unification is responsible for building a generalization of two formulas; the mapping phase is responsible for building generalization of two domains. The concept of domain here means a set of formulas that are either facts or rules and constitute the axiomatization for the domains theory. For exemplification we will always denote domain \mathcal{D}_α with a set of n formulas named $\alpha_1, \dots, \alpha_n$ and domain \mathcal{D}_β with a set of m formulas named β_1, \dots, β_m . Traditionally the domain with more formulas is regarded the source domain therefore we will assume $n \geq m$ and call \mathcal{D}_α the source domain and \mathcal{D}_β the target domain without loss of generality. Note the use of very compact examples that seem trivial, to clarify some properties or problems. This in no way reflects how complex actual cases of generalizations during mapping will be for fully formalized analogies. While here mapping between two domains is presented, the generalization to iteratively map multiple domains has also been shown in [9].

Definition 14 (Domain Representation). *A domain representation \mathcal{D} is a finite set of formulas $\{\alpha_1, \dots, \alpha_n\}$ with $\alpha_i \in \text{Term}^*(\Sigma, \mathcal{V}^*)$ for $i \in \{1, \dots, n\}$.*

Restricted higher-order anti-unification in the mapping phase tries to minimize the complexity of substitutions and thereby generate preferred generalizations out of a pair of given formulas $\langle \alpha, \beta \rangle$ one from each domain. Mapping is the process of selecting such pairs $\langle \alpha, \beta \rangle$ to be handed to anti-unification in a manner that minimizes the complexity of alignment between two domain representations. The complexity of alignment here means the sum of complexities of the preferred generalizations for the pairs $\langle \alpha, \beta \rangle$ that constitute the alignment. An alignment does not need to include all formulas of the two corresponding domains, but could be partial or even empty.

Definition 15 (Alignment). *An alignment for two domains \mathcal{D}_α and \mathcal{D}_β is a set of pairs $[(\alpha_1, \beta_1), \dots, (\alpha_n, \beta_n)]$ with $n \in \mathbb{N}_0$, provided $\alpha_i \neq \alpha_j$ and $\beta_i \neq \beta_j$ for $i, j \in \{1, \dots, n\}$, $i \neq j$. Furthermore, $\alpha_i \in \mathcal{D}_\alpha$ and $\beta_i \in \mathcal{D}_\beta$ for $i \in \{1, \dots, n\}$.*

From such an alignment of pairs we can build a generalized domain \mathcal{D}_γ which is the axiomatization of the generalized theory of the two aligned domains. This is a set $\gamma_1, \dots, \gamma_n$ where each γ is a most specific anti-unifier in a preferred generalization for each of the pairs in an alignment. Given that for each pair of

formulas, multiple preferred generalizations that differ not only up to renaming can exist, an alignment does not specify a specific generalized domain but multiple possible generalizations of two domains. To specify a specific generalized domain we will use the notion of a domain generalization. This notion is the extension of the generalization of a pair of formulas to the generalization of a set of formulas according to an alignment.

Definition 16 (Domain Generalization). *Given an alignment $[\langle \alpha_1, \beta_1 \rangle, \dots, \langle \alpha_n, \beta_n \rangle]$ for domains \mathcal{D}_α and \mathcal{D}_β , the domain generalization is a list of preferred generalizations $[g_1, \dots, g_n]$. Provided that g_i is a triple $\langle \gamma_i, \tau_i, \nu_i \rangle$ with $\alpha_i \xleftarrow{\tau_i} \gamma_i \xrightarrow{\nu_i} \beta_i$ for $i \in \{1, \dots, n\}$.*

Whereas each domain generalization constitutes a specific alignment, an alignment can correspond to many domain generalizations. A generalization for domains is therefore a more specific notion than an alignment. From the definition of domain generalizations we can derive the definition of generalized domain in the way we discussed earlier.

For an alignment consisting only of one pair of terms all corresponding domain generalizations must have the same complexity. This is not true for the general case with many pairs. For only one pair this follows directly from the definition 12 of preferred generalizations which states that these all must have minimal complexity. The difference for an alignment with more than one pair of terms follows again from a specific observation. Depending on the substitutions that are complexity free, because they were made already earlier, the complexity of preferred generalizations can differ. Different preferred generalizations for a pair of terms can employ different chains of substitutions as seen in figure 9.

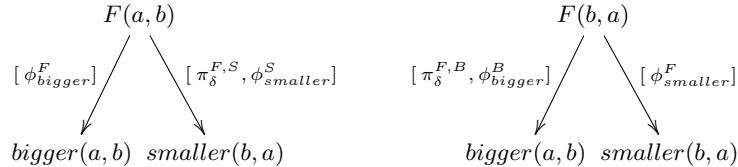


Fig. 9. Example for multiple preferred generalizations.

It follows that depending on the preferred generalization selected, for example the first pair of formulas, the complexities and structures of all possible preferred generalizations for the rest of the pairs in an alignment are affected. Thereby, this also has an effect on the sum of complexities of the specific mapping selected for that alignment. We therefore have to search for generalizations of domains with minimal complexities for an alignment of these and not just the alignment itself. This makes the task of the mapping process even more difficult.

On the other hand, given a generalization, we have specified a specific generalized domain. If we would for example just state substitutions and the source

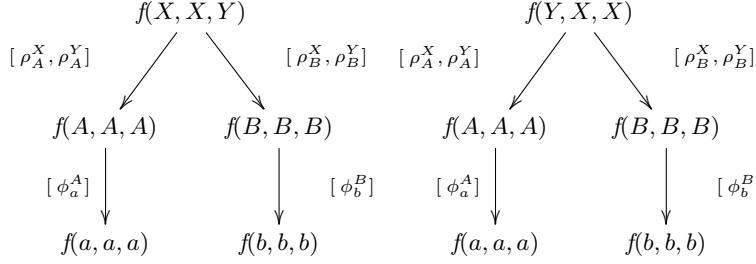


Fig. 10. Differing anti-unifier with same substitution chains and instances.

and target domain, it would be unclear which anti-unifiers for what subset of the formulas of the domains would be included and therefore build up the generalized domain. Also given two terms and substitution chains from an anti-unifier, the anti-unifier is still not unambiguously defined. See figure 10 for two generalizations with the same substitution chains to the same pair of instances but differing anti-unifiers. However, the other direction, given a term and a substitution chain the generated instance is always uniquely defined. Therefore, a domain generalization that covers all formulas of the original domains could be used to reconstruct them. However, as discussed earlier total coverage is usually not the case. Whereas an alignment gives us the source for why substitutions were made, the generalization for domains states which substitutions were made and in what anti-unifier, which is in the generalized domain, they resulted.

For simplicity we will not state the additional needed renaming substitutions in the following example. Then $\langle \langle mass(X) > mass(Y), [\phi_{sun}^X, \phi_{planet}^Y], [\phi_{nucleus}^X, \phi_{electron}^Y] \rangle, \langle mass(Y) > 0, [\phi_{planet}^Y], [\phi_{electron}^Y] \rangle$ is a domain generalization corresponding to the alignment $\langle \langle mass(sun) > mass(planet), mass(nucleus) > mass(electron) \rangle, \langle mass(planet) > 0, mass(electron) > 0 \rangle \rangle$. The formulas are part of the domains, formalized in the *Rutherford Analogy* as seen in figure 1. We will call the pair of chains of basic substitutions, employed in the generalization of domains, a mapping. For the example above this could be $[\phi_{sun}^X, \phi_{planet}^Y]$ and $[\phi_{nucleus}^X, \phi_{electron}^Y]$. We will call this pair of substitutions that lead from generalized domain to formulas in a specific domain the mapping that corresponds to that domain. Here $[\phi_{sun}^X, \phi_{planet}^Y]$ would be the mapping corresponding to the source domain which is the solar system domain.

Definition 17 (Mapping). Given a domain generalization $\langle \langle \gamma_1, \tau_1, \nu_1 \rangle, \dots, \langle \gamma_n, \tau_n, \nu_n \rangle \rangle$ for domains \mathcal{D}_α and \mathcal{D}_β , the corresponding mapping between \mathcal{D}_α and \mathcal{D}_β is the pair of chains of basic substitutions $[\tau_1, \dots, \tau_n]$ and $[\nu_1, \dots, \nu_n]$. We define the chain $[\tau_1, \dots, \tau_n]$ as the mapping from the generalized domain to \mathcal{D}_α and the chain $[\nu_1, \dots, \nu_n]$ as the mapping from the generalized domain to \mathcal{D}_β .

The mapping establishes the analogical relation between formulas of the domains. We will use the notion of a mapping also between subterms and also

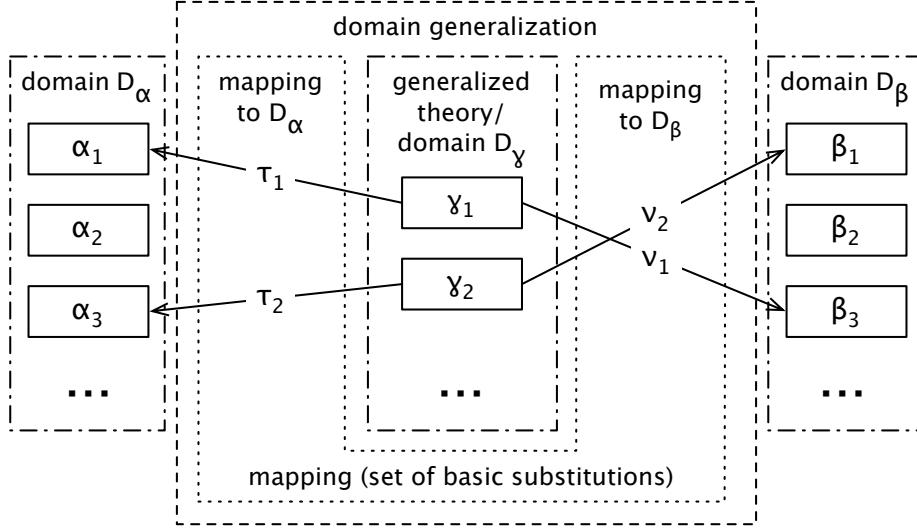


Fig. 11. Depiction of mapping, domains and domain generalization.

mere symbols, which do not need to be formulas of the domain. We will say *sun* maps to *nucleus* or vice versa if a direct relation by a mapping between those terms can be established as is the case in the earlier example. This is the case if the substitutions in a subset of a mapping can be used to build a valid generalization. Here $\langle X, \phi_{\text{sun}}^X, \phi_{\text{nucleus}}^X \rangle$ exists with *sun* being an instance of X by the substitution chain $[\phi_{\text{sun}}^X]$ and *nucleus* being an instance of X by the substitution chain $[\phi_{\text{nucleus}}^X]$.

We have used the measure of complexity for mappings, alignments and domain generalizations, but have not defined it formally. Since we now have defined the aforementioned concepts we can define the complexity of these as following.

Definition 18 (Complexity of Alignment, Mapping & Domain Generalization). *Given a domain generalization $[\langle \gamma_1, \tau_1, \nu_1 \rangle, \dots, \langle \gamma_n, \tau_n, \nu_n \rangle]$ the complexity for the domain generalization and corresponding mapping as well as alignment is $\sum_{i=1}^n \mathcal{C}(\langle \gamma_i, \tau_i, \nu_i \rangle)$.*

5 Transfer

Mappings created by the mapping phase are used afterwards in the transfer phase to infer hypothesis in the target domain. Transfer is possible by using the substitutions made from the generalized domain to the source domain and applying them in a reversed manner to transport the formula into the generalized theory from where then the substitutions to the target domain can be applied. This transfer of a formula from the source domain to the target can lead to a set of new formulas. For example, the one-to-many mapping created between

$f(a)$ and $g(b)$ as well as $f(a)$ and $h(b)$ will map a to b and f to g as well as h . So the transfer of $f(c)$ can result in $g(c)$ and $h(c)$. This shows that different alignment pairs of formulas can contribute to the transfer of a formula. In case a symbol has no mapping and does not exist in the target domain its is used as it is. Which was the case for c in the above example. If a symbol exists in the target domain but has no mapping all occurrences are replaced by a new symbol not yet used and this mapping is added. Since variables have at most formula scope they can not exist in a mapping in which that formula was not mapped. Therefore they will always be transferred as they are as can be seen for the rules transferred in the Rutherford analogy. In general it seems plausible to only try to transfer formulas that have not been mapped and thereby covered by the generalized theory. Another example of transfer in the Rutherford analogy would be to introduce the fact *yellow(sun)* in the source domain. This would lead to *yellow(nucleus)* in the target domain. By this its illustrated that transferred knowledge is just a hypothetical knowledge that does not need to follow from the axioms of the target domain. Since contradicting knowledge can be created, transferred formulas should be checked with a theorem prover to try to keep the target domain consistent. For physical domains new knowledge would serve as hypothesis that needs to be tested in experiments. Further mechanisms could be employed to decide for an inclusion in the target domain, however there is no generally applicable method to judge the usefulness of an analogical inference, as this varies with the given task. For example, in analogical problem solving the contribution to a solution of the problem can be used as a measure of usefulness and therefore decide inclusion in the target domain. Not only new statements can be derived by transfer in the target domain but also new concepts. A classical example for this kind of concept creation by analogical transfer is provided by the heat-flow analogy. There a concept, normally referred to as heat, whose properties are induced by analogical transfer is created in the target domain. For a formal treatment and detailed dicussion of this transfer see [29].

6 Semantics

As a formally defined framework, HDTP allows to accompany the syntactic processes of analogical mapping and transfer with a semantic interpretation. This distinguishes HDTP from other well-known analogy frameworks (such as SME [5] or CopyCat [17]), that use idiosyncratic formalisms for knowledge representation and quite sophisticated mapping procedures, which are hard to grasp from a formal perspective. As a result, there are to the best of our knowledge no spelled out semantics for these frameworks. In this section, we will describe the semantic implications of the aforementioned syntactic concepts of HDTP, based on model theoretic semantics. We will briefly revise the semantic core notions for first-order logic and then explain how a semantic interpretation can be given to the ideas of generalization and analogical mapping.

6.1 Basic Semantic Notions

A set theoretic model is based on a set of entities, called universe, that form the objects of the domain. An interpretation of a logical language maps terms of that language to objects of the universe, and logical formulas to statements about the universe that can be evaluated to truth values. All notions introduced in this section could be extended to the many-sorted case in the usual way, but for the sake of simplicity of presentation we will stick to the untyped version here.

Definition 19 (Σ -Interpretation). Let Σ be a signature. A Σ -interpretation is given by a pair $\mathfrak{M} = \langle U, I \rangle$ consisting of

- a non-empty set U , called universe
- a mapping I that maps
 - every n -ary function symbol f from Σ to a function $f_I : U^n \rightarrow U$
 - every n -ary predicate symbol p from Σ to a relation $p_I \subseteq U^n$.

Furthermore, a state is a mapping $s : \mathcal{V} \rightarrow U$. We denote by $s\{X/a\}$ the modified state that maps X to $a \in U$ and all other variables like s .

An interpretation allows to map terms to elements of the universe:

Definition 20 (Evaluation of Terms). Given a Σ -interpretation $\mathfrak{M} = \langle U, I \rangle$ and a state s over that interpretation, we define the evaluation of a term by induction over the term structure:

$$I_s(t) := \begin{cases} s(X) & \text{if } t = X \text{ is a variable} \\ f_I(I_s(t_1), \dots, I_s(t_n)) & \text{if } t = f(t_1, \dots, t_n) \end{cases}$$

Now truth of formulas can be established by induction over their structure:

Definition 21 (Evaluation of Formulas). Given a Σ -interpretation $\mathfrak{M} = \langle U, I \rangle$ and a state s , we define the evaluation of a formula ϕ as follows:

$$\begin{array}{ll} I \models_s p(t_1, \dots, t_n) & \text{if } \langle I_s(t_1), \dots, I_s(t_n) \rangle \in p_I \\ I \models_s \neg\varphi & \text{if } I \not\models_s \varphi \\ I \models_s \varphi_1 \wedge \varphi_2 & \text{if } I \models_s \varphi_1 \text{ and } I \models_s \varphi_2 \\ I \models_s \forall X \varphi & \text{if } I \models_{s\{X/a\}} \varphi \text{ for all } a \in U \end{array}$$

If $I \models_s \varphi$ we say that the formula φ is true in state s or simply that φ holds for the given interpretation and state. For a domain \mathcal{D} an interpretation I that makes all formulas of \mathcal{D} true for every state s is called a model of \mathcal{D} , denoted by $I \models \mathcal{D}$.

We will now discuss the model theoretic aspects of HDTP. A central part of the framework is the transition between different domains by substitution and analogical relation. In HDTP, substitutions serve to overcome particularities of specific domains while maintaining structural commonalities. A substitution

introduces a new general symbol to denote corresponding but different entities of the original domains. On the syntactic level, substitutions induce mappings to translate terms and formulas containing these general symbols into expressions of the original domain languages. On the semantic level they have an opposite role: they allow to import meaning from the original domains to interpret the new abstract symbols. We will spell out this idea for the basic substitutions of definition 3:

Definition 22 (Induced Evaluation of Terms). *Let $\mathfrak{M} = \langle U, I \rangle$ a Σ -interpretation and τ a basic substitution. Then, the induced evaluation I_s^τ in state s for terms $t \in \text{Term}(\Sigma, \mathcal{V} \cup \text{dom}(\tau))$ is defined by:*

$$I_s^\tau(t) = \begin{cases} s(X) & t = X \in \mathcal{V} \text{ and } X \notin \text{dom}(\tau) \\ f_I(I_s^\tau(t_1), \dots, I_s^\tau(t_n)) & t = f(t_1, \dots, t_n) \\ f_I(I_s^\tau(t_1), \dots, I_s^\tau(t_n)) & t = F(t_1, \dots, t_n), \phi_f^F = \tau \\ I_s^\tau(F'(t_1, \dots, t_n)) & t = F(t_1, \dots, t_n), \rho^{F,F'} = \tau \\ I_s^\tau(F'(t_1, \dots, G(t_i, \dots, t_{i+k}), \dots, t_n)) & t = F(t_1, \dots, t_n), \iota_{G,i}^{F,F'} = \tau \\ I_s^\tau(F'(t_{\alpha(1)}, \dots, t_{\alpha(n)})) & t = F(t_1, \dots, t_n), \pi_\alpha^{F,F'} = \tau \end{cases}$$

Given a chain of basic substitutions $[\tau_1, \dots, \tau_n]$, the induced evalution is defined by the composition $I^{[\tau_1, \dots, \tau_n]} = ((I^{\tau_n}) \dots)^{\tau_1}$. The resulting interpretation is denoted by $\mathfrak{M}^{[\tau_1, \dots, \tau_n]} = \langle U, I^{[\tau_1, \dots, \tau_n]} \rangle$.

This definition allows for interpreting variables that are introduced during the anti-unification process. We will make use of this idea to describe the semantics of the generalized domain and the analogical relation.

6.2 Interpreting the generalized domain

Formaly, the generalized domain \mathcal{D}_γ is a higher-order domain over the signature $\Sigma_\alpha \cup \Sigma_\beta$, i.e. a finite set of formulas from $\text{Form}^*(\Sigma_\alpha \cup \Sigma_\beta, \mathcal{V}^*)$. In fact, a careful inspection reveals, that only symbols from $\Sigma_\alpha \cap \Sigma_\beta$ can occur in formulas of the generalized domain. Furthermore, variables introduced by anti-unification will never be bound by quantifiers. Hence we can construct a generalized signature Σ_γ in the following way:

- Every $f : s^n \rightarrow s$ from $\Sigma_\alpha \cap \Sigma_\beta$ belongs to Σ_γ
- For every variable $X \in \mathcal{V}_n$ introduced by anti-unification, the string $X : s^n \rightarrow s$ belongs to Σ_γ .

Hence the generalized domain can be considered as a set of first-order formulas over Σ_γ , i.e. it is a subset of $\text{Form}(\Sigma_\gamma, \mathcal{V})$. In this sense, the semantics can be defined by providing a first-order model that interprets the new symbols. Such a model is provided in a natural way by the substitutions:

Fact 23 *Let $[(\gamma_1, \tau_1, \nu_1), \dots, (\gamma_n, \tau_n, \nu_n)]$ a domain generalization for the domains \mathcal{D}_α and \mathcal{D}_β and denote the generalized domain by \mathcal{D}_γ . Assume further*

that models $\mathfrak{M}_\alpha \models \mathcal{D}_\alpha$ $\mathfrak{M}_\beta \models \mathcal{D}_\beta$ are given. Then $\mathfrak{M}_\alpha^{[\tau_1, \dots, \tau_n]} \models \mathcal{D}_\gamma$ and $\mathfrak{M}_\beta^{[\nu_1, \dots, \nu_n]} \models \mathcal{D}_\gamma$.

This justifies the notion generalization: every model of the source or the target domain gives rise to a model for the generalized domain. Of course there can be models of the generalized domain, that neither are models of the source nor of the target domain.

7 Algorithmic treatment

A brute force approach to finding a generalization for domains with minimal complexity would simply mean to compute the complexity of all possible generalizations of two domains. Because we can not handle such a big problem space we must explore techniques to reduce the number of evaluated mappings between domains. To discuss these we first structure the problem into a more suitable form. The problem of selecting pairs of terms at each step in the mapping process for alignment can be understood as a search tree. Different alignments are different branches that arise in the search tree for each choice of formula pair for an alignment. Each alignment fences out in different branches. These are determined by the preferred generalizations deterministically computed by anti-unification. However, from a different preferred generalization for a pair of formulas a different analogy might arise. The start node in this search tree would be an empty alignment or mapping thereof. The possible end nodes are alignments with corresponding domain generalizations, which are analogies where no knowledge has been transferred yet. We will call all nodes in the search tree representing alignments with a corresponding generalization preanalyses, meaning there still might need to be some evaluation and mapping done before they are regarded as "finished" analogies. What we regard as a finished analogy and therefore, goal node in the process of search during the mapping phase, will be discussed in 8.

With this understanding of the mapping process as search through a tree structure we can employ techniques to prune this search tree to reduce the complexity of the search task at hand. Multiple points, where heuristics could be incorporated into the search strategy, to make it more informed and thereby faster, exist. We will discuss them with help of pseudo code for the operation of the mapping phase given in figure 12. Much like restricted higher-order anti-unification the general search strategy employed here is a breath first search. More specifically an A* search through the search tree of preanalyses.

As with every breadth first search we have to start the search by putting a start node in a queue, which is done at line 8. This queue is ordered by complexity of the queued preanalyses because we want to be able to do a variant of A* search. Note that the complexity here is not the usual complexity of path traversed plus expected complexity, rather just the complexity of the currently constructed mapping. We will discuss later why finding admissible heuristics, that are easily computed and not always predict 0 costs, is a complicated if not

```

01|function mapping(source, target) returns analogies
02|  // Queue and Result are ordered lists of preanalogy by complexity
03|  // preanalogy is a tuple  $\langle C, (Sopen, Sclosed), (Topen, Tclosed), Gs \rangle$  where
04|  //    $C$  is the complexity of the analogy
05|  //    $Sopen, Topen$  are lists of not aligned formulas for each domain
06|  //    $Sclosed, Tclosed$  are lists of already aligned formulas for each domain
07|  //    $Gs$  is a list of generalizations which are tuples  $(g, \tau, \nu)$ 
08|  initialize:
09|    Queue =  $\{(0, (\text{source}, []), (\text{target}, []), [])\}$ 
10|    Result = {}
11|  while Result  $\neq \{\}$  or complexity(first(Queue))  $\leq$  complexity(Result) do
12|    Preanalogy = first(Queue); Queue = rest(Queue)
13|     $\langle C, (Sopen, Sclosed), (Topen, Tclosed), Gs \rangle = \text{Preanalogy}$ 
14|    foreach Sformula in select_source_formula(Preanalogy)
15|      Sopen = Sopen \ Sformula
16|      Sclosed = Sclosed  $\cup$  Sformula
17|      C_nomatch = complexity_nomatch(Sformula)
18|      Preanalogies =  $\{C + C_{\text{nomatch}}, (Sopen, Sclosed), (Topen, Tclosed), Gs\}$ 
19|      foreach Tformula in select_target_formula(Preanalogy, Sformula)
20|        Topen = Topen \ Tformula
21|        Tclosed = Tclosed  $\cup$  Tformula
22|        Sformula, Tformula = reRepresent(Sformula, Tformula, Preanalogy)
23|        foreach C_au, G in anti-unify(Sformula, Tformula, Gs)
24|          Pre_new =  $\langle C + C_{\text{au}}, (Sopen, Sclosed), (Topen, Tclosed), Gs \cup G \rangle$ 
25|          Preanalogies = Preanalogies  $\cup$  Pre_new
26|        end foreach
27|        foreach Preanalogy in Preanalogies
28|          if is_finished(Preanalogy) then
29|            if complexity(Preanalogy) < complexity(Result) then
30|              Results = Preanalogy
31|            else if complexity(Preanalogy) = complexity(Result) then
32|              Results = merge(Result, Preanalogy)
33|            else Queue = merge(Queue, Preanalogy)
34|          end foreach
35|        end foreach
36|      end while
37|      return Results
38|end function

```

Fig. 12. Algorithm for computation of domain generalizations.

impossible task in the general case. The preanalogy with which the search queue is initialized is a special one with no alignment or mapping done. Therefore, the generalized domain (Gs) in the tuple is empty. Each preanalogy is defined in the algorithm by a complexity together with a list of aligned (closed) and not aligned (open) formulas, as well as, the domain generalization. The domain generalization is just a list of triples consisting of an anti-unifier together with

two substitution chains to the involved domains. We do not need to state the aligned formulas for each generalization of formulas explicitly as they can be constructed from the generalization itself as discussed earlier. Additionally, for the initialization of mapping we will start with an empty result list as stated in line 9. This is a list or even set that will contain the found preanalyses with the smallest complexities that are considered finished. All preanalyses in the result list therefore always have the same complexity and because therefore order does not matter, this could be as well just be considered as a set.

The mapping algorithm will start to search for finished preanalyses until none can be found. Alternatively the more usual case is that a set of finished preanalyses is found and its certain that all the preanalyses, that still could be mapped until they are finished, already have a higher complexity than the found ones. Therefore, line 10 shows that the global metric and generic goal of the mapping phase is to minimize the complexity of the mapping. The faster a finished preanalysis with minimal complexity is found the faster the results can be returned because the search tree is pruned earlier.

The search tree is traversed by selecting the preanalysis with the least complexity from the search queue and expanding the alignment between the involved domains in the analogy. Expansion of alignment is done by first finding suitable source and target formula pairs and then computing the added complexities for the possible mappings. This is a multistep process. First a set of source formulas is determined by the function *select_source_formulas* in line 13. Each will make up the first half of an alignment pair. The second half is determined relative to the source formula in line 18 by *select_target_formulas*. A special case is constructed in line 17 for each selected source formula namely that no alignment should be formed with that formula in the analogy. This is important because there might actually exist no valid anti-unifier under some selectable restrictions for the mapping phase. Another reason is that we do not want to force unreasonable alignments and mappings. In this special case the source formula is added to the closed list of aligned pairs so that it is not reconsidered for alignment later. A complexity penalty for not aligning that formula is determined by *complexity_nomatch* and added to the overall complexity of the newly constructed preanalysis. This special preanalysis will be processed in the same manner all other preanalysis are in the following steps.

For each alignment pair selected as described above a generalization is constructed by application of restricted higher-order anti-unification, as shown in line 22. The preanalysis here is an argument to the anti-unification because reuse of substitutions must be checked to calculate the complexity of the generated generalizations correctly. To minimize the complexity added, an additional representation step for each alignment pair is performed before anti-unification in line 21 by the *re_represent* function.

For each newly constructed preanalysis we check whether its mapping is regarded finished in line 27 by use of the function *is_finished*. If this is the case the complexity is compared to the currently known finished preanalyses and added to them if the complexity is equal. If complexity of the new preanalysis is lower

the result list is cleared and set to this newer preanalogy with lesser complexity. If the complexity of the new preanalogy is higher it is simply discarded to save memory. We could save it for latter processing and make the result list ordered to produce higher complexity alternative mappings. If a preanalogies mapping is not yet considered finished its sorted into to the search queue according to its complexity.

This presents the general outline of HDTP mapping phases top-level operation. Several functions were mentioned that were used in this algorithm but their exact operation was left unclear. These are *select_source_formulas*, *select_target_formulas*, *complexity_nomatch* and *is_finished*. Depending on their characteristics they influence the construction of new preanalogies and restrict the search in the search tree of preanalogies. They are used as heuristics to guide the search to find suitable analogies for drawing analogical inferences in the transfer phase. Details about these will be discussed in the next section.

8 Heuristics

We may distinguish two aspects of the mentioned heuristics in mapping. The first viewpoint we can take on heuristics is that they could incorporate knowledge that is not solely based on the formal structure of domains. They for example could use information that is based on previous knowledge of anti-unifying domains from similar fields e.g. physics. Or they can guide the search in such a way that complexity is not solely minimized, rather other goals are incorporated by pruning the search in an appropriate form. Whereas HDTP is solely a symbol manipulating system this represents a point where "intelligence" can be incorporated. As Simons and Newell have pointed out in their Turing Award lecture [22] the amount of search a system does is no measure of intelligence. The magnitude of a problem is described by how much search would be required if some form of intelligence would not be applied. Mapping for analogical reasoning as here described has an exponential explosion in the search tree and can not be solved by merely brute force search. The task of intelligence in form of heuristics here is then to advert the threat of exponential search explosion by guiding search in only "plausible" directions.

Such intelligent heuristics may be searched for by conducting experiments for analogical reasoning in humans and trying to extract guiding principles from the gathered data. This was done for example in the Cougar (COmputational Understanding of Geometric Analogy Reasoning) Project [30] were the influence of *Gestalt Laws* in proportional geometric analogies was tested. Such findings can then be integrated into HDTP as heuristics in the algorithm. Found heuristics however might make HDTP search incomplete, meaning no useable analogy, even if one exists, may be discovered during search. This is offset by usually finding analogies in a timely manner. This is not different from humans who may not be able to solve a proportional geometric analogy even if a plausible solution exists, or they might construct a more complex solution than needed. Whether an analogy or mapping seems plausible for a specific field and therefore is what

is_finished would partly decide should be uncovered by empirical studies in that field.

For further discussions we will take a purely computational optimization view on heuristics to explain their different uses and properties within the algorithm. The goal here is to find good heuristics to approximate the complexity a restricted higher-order anti-unification will have given a preexisting mapping. Therefore, with this viewpoint *select_source_formulas*, *select_target_formulas*, *complexity_nomatch* and *is_finished* guide the mapping phase in a way that it quickly and efficiently finds the mappings with a minimal overall complexity. This would be the default and general case for HDTP if no "intelligent" heuristics can be employed. We assume here that substitution complexity itself is an approximation for good mappings in general analogical reasoning. This itself could be regarded as a fundamental heuristic.

Ideally to achieve a good mapping the search queue would be ordered by the usual $f + h$ metric for A* search. Here f would be the current complexity of the preanalogy and h would be a heuristic that approximates the additional complexity, that will be added until the preanalogy is regarded finished. The problem here is that there is always the possibility of the complexity of anti-unification being 0 due to reuse of substitutions. Checking this would actually mean to bottom up construct anti-unifiers and verify that the substitutions all are complexity free. This in turn means doing what the anti-unification algorithm does anyway and thereby not gaining any computational efficiency over actually having no heuristic at all. Getting an admissible heuristic that is not computational complex, when complexity free reuse of substitutions is possible, seems therefore not achievable. A full A* search is therefore not used within the top-level loop of the mapping algorithm. Just f is used to order the search queue and h is simply set to the null heuristic yielding 0. If for a specific version of restricted higher-order anti-unification an admissible heuristic is found it can be easily incorporated into the algorithm none the less.

8.1 Non aligned formula penalty and finished mappings

Anti-unification of $f(A, g(h))$ with a variable X has a complexity of 6. Generally the complexity can be computed by (*number of symbols**2+*number of variables*–1). Taking this measure for *complexity_nomatch* seems to be a good idea. If a formula matches better against a variable, the most general term possible, then it might not share much in common with another term with higher anti-unification complexity. Hence, it may just better be not aligned. Anti-unification of two formulas with equal argument structure but totally differing symbols such as $f(A, g(h))$ and $f2(A2, g2(h2))$ is two times the number of symbols of that term. Therefore, usually two structurally equal terms will be preferred over not including a term in alignment. However, there is one corner case namely when the number of variables in the formula is zero or one. For *complexity_nomatch* to still yield a higher complexity in those cases, than actually aligning such two structurally equal terms, we will take the anti-unification heuristic and add a constant complexity of 2. Any heuristic which yields higher values will start

to favor aligning terms even if they share no structure. *complexity_nomatch* heuristics giving lower values will favor aligning terms when they share structure or reuse substitutions a lot. Thereby, they favor mappings where the same substitutions are supported by multiple alignment pairs.

The next heuristic to be discussed is *is_finished*. One understanding of it can be that if all source domain formulas were considered for alignment, a pre-analogy is regarded as complete. Whether these are actually aligned or not is left to the complexity penalty given out by *complexity_nomatch*. Another possibility for *is_finished* is to check whether relative to the number of symbols in a domain enough mappings have been made. A more goal driven variant of this is to check whether certain symbols, which are focused by the analogy making process, are mapped. If the goal of the analogy is to transfer a specific formula to the target domain, then having mappings for all symbols in the source formula is sufficient. Mapping further can help to establish more support for specific symbol mappings, because substitutions are reused or can establish different transformations into the target domain. However, it can also take considerable more time trying to align all formulas in the domains. The search tree does not have to be explored that deep and therefore *is_finished* provides a pruning for the general depth of the search tree. Depending on *is_finished* properties this can be relative (specific symbols mapped) or absolute (aligning all formulas) or just a fixed amount. In the later case we would have a depth limit of the search tree, which could be used for iterative depending if the main loop of HDTP would be changed to a depth first search.

8.2 Alignment selection

The up to now still not discussed heuristics but, none the less very important ones, are *select_source_formulas* and *select_target_formulas*. Together they select pairs of formulas for the alignment mapped next in the search. We will therefore call this the *selection heuristic*. Whether they generate one pair or many is up to the specific implementation. However, the more pairs of formulas they return the more branches are traversed during the search. If the *selection heuristic* generates one pair of terms that tries to minimize the added complexity for just the next pair formulas, the mapping phase becomes a hill-climbing search. This is also called greedy local search. Here a selection heuristic should select a pair of formulas that ideally have the minimal complexity according anti-unification of all possible pairs of formulas.

8.3 A strategy for fast structural alignment

We hereby describe a generic domain independent strategy for finding minimal complexity generalization for two domains within the HDTP's framework. This strategy is named FAST (Fast Aligns Selecting Terms).

FAST instead of full breadth first search employs a greedy general alignment strategy. This has turned out well for the analogies formalized for HDTP so far. An empirical analysis how good the mapping algorithm with the *FAST* strategy

has not been done due to the lack of a large library of analogy formalizations with analogical inferences.

As mentioned, the concrete strategy *FAST* is a locally greedy search. For this it is especially important that the *selection heuristic* selects pairs of formulas that do not constrain mapping possibilities too fast, but collect support for symbol mappings incrementally in small steps. If no mapping is available the pair of formulas is selected that is best according to anti-unification complexity or an heuristic approximation. If multiple such pairs exist, the ones that have most matching predicates on the top-level will be selected. For example aligning $f(a)$ and $f(b)$ would be preferred over the pair $f(a)$ and $g(a)$. We then take the source formula from that pair as heuristic *select_source_formula*. If the mapping already contains substitutions we simply select the formula as source formula that contains the least amount of symbols that are unmapped or do not appear in the other domain. If multiple such formulas exist we narrow down the choice again by the anti-unification complexity and choose the source formula that is contained in the pair of formulas that has the lowest approximated complexity.

Because if an anti-unification heuristic is used it is not admissible we will only heuristically select the source formula by *select_source_formula*. *select_target_formula* will be chosen to simply return all unaligned formulas and therefore *Topen* in the mapping phase pseudo algorithm. However, we will optimize this by returning them sorted by the anti-unification heuristic. To make this computationally more efficient, we exploit the fact that the restricted higher-order anti-unification algorithm is able to search for generalizations up to a specific complexity. We can pass the currently computed minimal complexity for generalizations with a specific selected source term on to the following anti-unifications with different target formulas. If they do not have generalizations that are less or equally complex they can terminate earlier than without a given cutoff value.

The algorithm is still locally optimal in that given a source formula will select the best target source formula that still needs to be aligned. However, because we employ a greedy search we will not compute anti-unifications for an arbitrary order of source formulas in the alignment, which makes the algorithm globally not optimal. Note that the search still branches, even if we do a locally greedy strategy, because given a source formula, multiple target formulas can exists, that produce preferred generalizations with the same complexity.

For the *complexity_nomatch* heuristic the balance between not trying to force mappings to hard and not leaving a lot of formulas unaligned, is aimed for in *FAST*. We therefore use the metric *number of symbols**2 + 1 to make it more complex to not align a formula, than to anti-unifying it with a formula that is equal in argument structure. At the same time it does not dismiss alignments when a term with many matching or already mapped to symbols is available.

Re-representation is done locally by *re_represent* between formulas only considering basic transformations given in background knowledge of the domains involved. The mapping process according to *FAST* is finished if the *Sopen* in the mapping phase pseudo algorithm is empty. The strategy will therefore al-

ways terminate, because in each iteration of the main mapping loop a formula will be taken from S_{open} which is finite. Whether the formula will be aligned or not is a different matter. In the end no source formulas are left for inclusion in the alignment. Thereby, trying to make the mapping cover as many formulas as possible in both domains. We now have instantiated and explained all heuristics for the concrete case of *FAST* strategy in this section.

9 Summary and Conclusion

In this overview and referenced papers, we described the syntactic, semantic, and algorithmic aspects of Heuristic-Driven Theory Projection. The presented framework for analogy-making allows the representation of complex rules (including quantifiers and variables) as they appear, for example, quite often in mathematical theories. This expressive formalism makes HDTP different to alternative systems that focus quite often on rather simple and fully instantiated facts and not on axiomatic scientific theories. In the mapping phase, the application of higher-order mechanisms allows to find correspondences between terms of different complexity if that is justified by the context. HDTP is one of the rare examples of analogy engines for which a model theoretic semantics can be formulated. This is another significant difference from systems like SME, systems that are hybrid in nature (AMBR/DUAL or Copycat) or neurally inspired (LISA). For all these alternative systems no semantics is currently available. Similar to many other approaches, HDTP uses heuristics in order to reduce the computational complexity and uses a cost function to compute a ranking of candidates for the analogical relation.

We think that HDTP is a promising approach for analogy making in a large variety of different domains. Due to the fact that analogy making is an extremely important cognitive mechanism for many different cognitive abilities such as creativity, concept invention, problem solving etc., we believe that HDTP is a crucial step to a cognitively inspired approach for modeling intelligence on a human scale, i.e. the ultimate goal of artificial intelligence. Perhaps it is the historically long standing denial of the modeling of cognitive mechanisms (such as analogy making) that led AI into a discipline that has been focusing strongly on specific problem domains and ignoring the generality aspect of intelligence. We think that it is time to change the directions of this line of research.

References

1. Chalmers, D.J., French, R.M., Hofstadter, D.R.: High-level perception, representation, and analogy: A critique of artificial intelligence methodology. *Journal of Experimental and Theoretical Artificial Intelligence* 4(3), 185–211 (1992)
2. Falkenhainer, B., Forbus, K.D., Gentner, D.: The structure-mapping engine: Algorithm and examples. *Artificial Intelligence* 41(1), 1–63 (1989)
3. Forbus, K.D., Gentner, D., Markman, A.B., Ferguson, R.W.: Analogy just looks like high-level perception: why a domain-general approach to analogical mapping

- is right. *Journal of Experimental & Theoretical Artificial Intelligence* 10, 231–257 (1998)
4. Furtado, A.L.: Analogy by generalization—and the quest of the grail. *SIGPLAN Not.* 27(1), 105–113 (Jan 1992)
 5. Gentner, D.: Structure-mapping: A theoretical framework for analogy. *Cognitive Science* 7(2), 155–170 (1983)
 6. Gentner, D.: The mechanism of analogical learning. In: Vosniadou, S., Ortony, A. (eds.) *Similarity and analogical reasoning*, pp. 199–241. Cambridge University Press, New York, USA (1989)
 7. Gentner, D., Forbus, K.D.: Computational models of analogy. *Wiley Interdisciplinary Reviews: Cognitive Science* 2(3), 266–276 (May/June 2010)
 8. Guarini, M., Butchart, A., Smith, P.S., Moldovan, A.: Resources for research on analogy: A multi-disciplinary guide. *Informal Logic* 29(2), 84–197 (2009)
 9. Guhe, M., Pease, A., Smaill, A., Schmidt, M., Gust, H., Kühnberger, K.U., Krumnack, U.: Mathematical reasoning with higher-order anti-unification. In: Proceedings of the 32nd Annual Conference of the Cognitive Science Society. pp. 1992–1997. Cognitive Science Society, Austin, TX (2010)
 10. Gust, H., Krumnack, U., Kühnberger, K.U., Schwering, A.: Analogical reasoning: A core of cognition. *Zeitschrift für Künstliche Intelligenz (KI), Themenheft KI und Kognition* 1, 8–12 (2008)
 11. Gust, H., Kühnberger, K.U., Schmid, U.: Metaphors and heuristic-driven theory projection (HDTP). *Theoretical Computer Science* 354(1), 98–117 (2006)
 12. Hasker, R.W.: The replay of program derivations. Phd thesis, University of Illinois at Urbana-Champaign (1995)
 13. Heinz, B.: Anti-Unifikation modulo Gleichungstheorie und deren Anwendung zur Lemmagenerierung. R. Oldenbourg Verlag, Wien (1996)
 14. Hirata, K.: A note on rule-finding abduction. Tech. rep., Research Institute of Fundamental Information Science, Kyushu University (1994)
 15. Hirowatari, E., Arikawa, S.: Partially isomorphic generalization and analogical reasoning. In: Proceedings of European Conference on Machine Learning (1994), Lecture Notes in Artificial Intelligence 784. pp. 363–366 (1994)
 16. Hirowatari, E., Arikawa, S.: Partially isomorphic generalization and analogical reasoning. In: Machine Learning: ECML-94, Lecture Notes in Computer Science, vol. 784, pp. 363–366. Springer (1994)
 17. Hofstadter, D.R., Mitchell, J.: The copycat project: A model of mental fluidity and analogy-making. In: Hofstadter, D.R., Fluid Analogies Research Group (eds.) *Fluid Concepts and Creative Analogies*, pp. 205–267. Basic Books (1995)
 18. Holyoak, K., Morrison, R. (eds.): *The cambridge handbook on thinking and reasoning*. Cambridge University Press (2005)
 19. Indurkhya, B.: *Metaphor and cognition*. Kluver, Dodrecht (1992)
 20. Krumnack, U., Schwering, A., Gust, H., Kühnberger, K.U.: Restricted higher-order anti-unification for analogy making. In: AI 2007: Advances in Artificial Intelligence. Lecture Notes of Artificial Intelligence, vol. 4830, pp. 273–282. Springer (2007)
 21. Kurzen, L.M.: *Analogy and Bisimulation: A Comparison of Indurkhya's Cognitive Models and Heuristic-Driven Theory Projection*, PICS (Publications of the Institute of Cognitive Science), vol. 2005-10. University of Osnabrück, Osnabrück (2005)
 22. Newell, A., Simon, H.A.: Computer science as empirical inquiry: symbols and search. *Commun. ACM* 19, 113–126 (March 1976)
 23. Plotkin, G.D.: A note on inductive generalization. *Machine Intelligence* 5, 153–163 (1970)

24. Reynolds, J.: Transformational systems and the algebraic structure of atomic formulas. *Machine Intelligence* 5, 135–153 (1970)
25. Robere, R., Besold, T.R.: Complex analogies: Remarks on the complexity of hdtp. In: Thielacher, M., Zhang, D. (eds.) *Australasian Conference on Artificial Intelligence*. Lecture Notes in Computer Science, vol. 7691, pp. 530–542. Springer (2012)
26. Rutherford, E.: The scattering of α and β particles by matter and the structure of the atom. *Philosophical Magazine Series 6* 21, 669–688 (1911)
27. Schmidt, M.: Restricted Higher-Order Anti-Unification for Heuristic-Driven Theory Projection, PICS (Publications of the Institute of Cognitive Science), vol. 2010-31. University of Osnabrück, Osnabrück (2010)
28. Schmidt, M., Krumnack, U., Kühnberger, K.U., Gust, H.: Refinements of restricted higher-order anti-unification for heuristic-driven theory projection. In: Bach, J., Edelkamp, S. (eds.) *KI 2011: Advances in Artificial Intelligence 34th Annual German Conference on AI*, Berlin, Germany, October 4-7, 2011. Proceedings. Lecture Notes in Computer Science, vol. 7006/2011, pp. 289–300. Springer, Berlin, Heidelberg (2011)
29. Schwering, A., Krumnack, U., Kühnberger, K.U., Gust, H.: Syntactic principles of Heuristic-Driven Theory Projection. Special Issue on Analogies - Integrating Cognitive Abilities. In: *Journal of Cognitive Systems Research* 10(3), 251–269 (2009)
30. Schwering, A., Krumnack, U., Kühnberger, K.U., Gust, H. (eds.): *Investigating Experimentally Problem Solving Strategies in Geometric Proportional Analogies*, Publications of the Institute of Cognitive Science, vol. 30-2010. Institute of Cognitive Science, Osnabrück (2010)
31. Vosniadou, S., Ortony, A.: Similarity and analogical reasoning. Cambridge University Press, Cambridge (1989)

Completing symbolic rule bases using betweenness and analogical proportion

Steven Schockaert¹ and Henri Prade²

¹ School of Computer Science & Informatics, Cardiff University,
5 The Parade, CF24 3AA, Cardiff, United Kingdom
s.schockaert@cs.cardiff.ac.uk

² Institut de Recherche en Informatique de Toulouse (IRIT), University of Toulouse,
118 route de Narbonne, 31062 Toulouse Cedex 09, France
prade@irit.fr

Abstract. Classical deduction is limited as a tool for reasoning about logical domain theories, in its ability to make sense of situations that are not explicitly covered. Humans on the other hand are remarkably adept at speculation about new situations, by drawing analogies or by relying on knowledge of similar situations. In this paper, we are interested in formalising this process to develop a form of commonsense reasoning about incomplete rule bases. More precisely, we discuss two methods which can be used to derive plausible rules from a given set of propositional rules and a set of analogical proportions. The first method is based on the view that whenever the antecedents of four rules are in an analogical proportion, their consequences are likely to be in an analogical proportion as well. It often produces useful results, although it may be too adventurous for some applications. The second method is more cautious and makes explicit the assumptions under which it produces sound conclusions. Finally, we show how the second method may be further refined, in such a way that we recover the first method as a special case.

1 Introduction

We consider a propositional language based on the usual propositional connectives and a set of atomic propositions (or atoms) A . We assume that a partition $A = A_1 \cup \dots \cup A_n$ is available such that intuitively in any possible world exactly one atom from each partition class A_i is true. The partition classes A_i will be referred to as *attribute domains*, and they group properties which are of the same kind. The notion of attribute domain is used under a number of different names in the literature; for example, in [9], attribute domains are called multivalued propositional variables. When talking about animals, for instance, we could consider the following attribute domains:

$$\begin{aligned} \text{Activity} &= \{\text{nocturnal}, \text{diurnal}, \text{crepuscular}\} \\ \text{Interaction} &= \{\text{domestic}, \text{predator}, \text{prey}, \text{scavenger}\} \\ \text{Size} &= \{\text{very-small}, \text{small}, \text{medium}, \text{large}, \text{very-large}\} \end{aligned}$$

$$Species = \{ dog, cat, lynx, wolf, coyote, leopard \}$$

Obviously the set *Species* is not an exhaustive enumeration of all species, but we may consider that in a given context, these are the only species that we want to consider. Moreover, to obtain pairwise disjoint properties, we should interpret properties such as *predator*, *prey* and *scavenger* in an exclusive way, e.g. *prey* only denotes animals which are not also predators and *scavenger* denotes all animals which primarily use scavenging (even if they occasionally hunt) to feed. Similarly, we will assume that properties such as *small* have a precise meaning. For example, we may consider that *small* refers to animals with a length between 25cm and 75cm. Such boundaries are obviously artificial, but modelling the vagueness of linguistic properties falls outside the scope of this chapter; we refer the interested reader to [2, 4, 10, 11, 20].

An interpretation ω is a set $\{a_1, \dots, a_n\}$ such that $a_i \in A_i$ for $i = 1, \dots, n$, i.e. an interpretation encodes for each attribute domain, which of its atoms we consider to be true. An interpretation ω is a *model* of a propositional formula if it satisfies the formula in the usual sense (interpreting the atoms in ω as true and those in $A \setminus \omega$ as false). We say that a formula α (resp. a set of formulas R) entails a formula β , written $\alpha \models \beta$ (resp. $R \models \beta$) if every model of α (resp. R) is a model of β . In other words, we consider the standard notion of propositional entailment, modulo the assumption that the atoms in each attribute domain are jointly exhaustive and pairwise disjoint (JEPD).

Consider for instance the following rules, encoding some knowledge we may have about the characteristics of animals:

$$dog \rightarrow domestic \wedge small \wedge diurnal \tag{1}$$

$$cat \rightarrow domestic \wedge small \tag{2}$$

$$lynx \rightarrow predator \tag{3}$$

$$wolf \rightarrow predator \tag{4}$$

$$coyote \rightarrow nocturnal \tag{5}$$

$$leopard \rightarrow large \tag{6}$$

From these rules, we can entail $lynx \vee wolf \rightarrow predator$, using classical deduction, or that $leopard \rightarrow \neg small$ using the assumption that the atoms in *Size* are pairwise disjoint. However, without further information, we cannot derive anything about whether leopards are nocturnal (i.e. active at night), diurnal (i.e. active during the day) or crepuscular (i.e. active at twilight).

One important source of additional information is similarity. For many attribute domains, some form of data is available which we could use to estimate the degree to which two of its atoms are similar. For example, similarity between music genres could be derived from the tagging behaviour of users of music recommendation systems such as last.fm [19], while similarity between wines could be derived by looking at guidance on wine-food pairings [22]. Open-domain semantic networks such as ConceptNet can be used to derive similarity degrees for a wide variety of properties [23]. Wikipedia has also been used as an

open-domain source for assessing semantic similarity [6]. Measuring similarity between biological species is an active topic of research in the field of taxonomy. Similarity in this context may among others refer to morphological, ecological and phylogenetic similarity, although such different forms of similarity are often highly correlated (see e.g. [12] for a discussion).

Similarity-based reasoning is a form of commonsense reasoning based on the premise that from similar conditions we should be able to draw similar conclusions. It is paramount in how humans deal with missing knowledge [24], and is closely related to prototype theory [18] and more generally to the role of similarity in models of categorisation. For example, we could use the observation that wolfs and coyotes are similar to derive from (4) and (5) that the following rules are plausible:

$$wolf \rightarrow nocturnal \quad (7)$$

$$coyote \rightarrow predator \quad (8)$$

An important disadvantage of similarity-based reasoning is that there is no clear guidance on how similar two propositions should be for such an inference to be plausible. For instance, wolfs and dogs are also quite similar, but we do not want to derive that dogs are likely to be nocturnal predators.

To avoid such problems, which are caused by the gradual and multi-dimensional nature of similarity, the approach we propose is based on two qualitative notions instead: betweenness and analogical proportion. The intuition behind both notions relies on the fact that the properties of interest can often be represented in terms of more primitive features. If we identify a property with a set of binary features, then property b is intuitively between properties a and c if all features that are shared by a and c are present in b . Similarly, the properties a, b, c and d are in an analogical proportion, written $a : b :: c : d$, if a and b differ in the same features as the ones in which c differs from d [13]. Note that while betweenness can be seen as a qualitative form of similarity, analogical proportion is defined in terms of dissimilarity, and it will allow us to derive conclusions that could not be obtained using similarity based reasoning.

Another possibility is to rely on a geometric representation of properties. It is often the case that the most primitive features with which an object can be described are measurable quantities (e.g. the features hue, saturation and intensity when modelling colours). As a result, we can think of objects as points in a vector space, and as properties as regions in such a space. This is the intuition underlying Carnap's attribute spaces [1] and, more recently, Gärdenfors' conceptual spaces [7]. In such a case, betweenness and analogical proportion can be given a geometric interpretation, in terms of geometric betweenness and parallelograms respectively. From a practical point of view, it is important to note that we do not require such representations of properties (as feature sets or geometric regions) to be available; these representations are merely used to make explicit the semantics and properties of betweenness and analogical proportion.

The notions of betweenness and analogical proportion give rise to an approach for completing rule bases which is purely qualitative, and which we discuss in

Section 2. In Section 3 we look more closely at the justification for the principles underlying this approach, based on a representation of properties as regions in a conceptual space. We will in particular look at justifications for the idea that whenever the conditions of four rules form an analogical proportion that we can then also expect their conclusions to form an analogical proportion. This gives rise to an alternative approach which is more cautious. We then show how the latter approach can be refined, and in particular, how we can retrieve the first approach by making a number of additional assumptions. Finally, note that this chapter extends and further develops [17] (in Section 2) and [21] (in Section 3).

2 Completing rule bases using analogical proportions

In this section, we outline the idea of using analogical proportions for adding rules to a given rule base. After presenting the basic idea, we briefly discuss how analogical proportions interact with the standard logical connectives in Section 2.2. Then in Section 2.3, we discuss the issue that perfect analogical proportions are rare, and suggest a way of softening the proposed mechanism.

2.1 Basic idea

The following inference rule formalises a natural strategy for completing rule bases, using the view that whenever the conditions of four rules are in an analogical proportion, it is reasonable to assume that their conclusions are in an analogical proportion as well:

$$\begin{array}{c} \alpha_1 \rightarrow \beta_1 \\ \alpha_2 \rightarrow \beta_2 \\ \alpha_3 \rightarrow \beta_3 \\ \alpha_1 : \alpha_2 :: \alpha_3 : \alpha^* \\ \beta_1 : \beta_2 :: \beta_3 : \beta^* \\ \hline \alpha^* \rightarrow \beta^* \end{array} \tag{9}$$

For example, if we consider that *cat* : *lynx* :: *dog* : *coyote*, then we can use the inference rule (9) to conclude from (2), (3) and (1) that

$$coyote \rightarrow predator$$

To make this inference, we have also used the fact that *domestic* : *domestic* :: *predator* : *predator*. The analogical proportion $\alpha : \alpha :: \beta : \beta$ is assumed to be characteristic of the analogical proportion, and is assumed to hold in general. Similarly, the analogical proportion $\alpha : \beta :: \alpha : \beta$ is assumed to always hold. Two other notable properties that are characteristic of analogical proportion are

$$(\alpha : \beta :: \gamma : \delta) \Leftrightarrow (\gamma : \delta :: \alpha : \beta) \tag{10}$$

$$(\alpha : \beta :: \gamma : \delta) \Leftrightarrow (\alpha : \gamma :: \beta : \delta) \tag{11}$$

Formally, the analogical proportion can be defined as follows [15]. Let A , B , C and D be sets of features which characterise α , β , γ and δ respectively. Then

$$(\alpha : \beta :: \gamma : \delta) \Leftrightarrow (A \setminus B = C \setminus D) \wedge (B \setminus A = D \setminus C) \quad (12)$$

It is not hard to see that this definition indeed satisfies the aforementioned properties. Equivalently, we can use Boolean feature vectors (a_1, \dots, a_n) , (b_1, \dots, b_n) , (c_1, \dots, c_n) , and (d_1, \dots, d_n) , in which case we have

$$(\alpha : \beta :: \gamma : \delta) \Leftrightarrow \forall i : ((a_i \rightarrow b_i) \equiv (c_i \rightarrow d_i)) \wedge ((b_i \rightarrow a_i) \equiv (d_i \rightarrow c_i)) \quad (13)$$

This definition also suggests how we might discover analogical proportions from data. For example, to verify whether $dog : coyote :: cat : leopard$ we first need to represent dog , $coyote$, cat , $leopard$ as Boolean feature vectors. Then we can verify whether (13) is satisfied for all considered features.

2.2 Interaction with logical connectives

Analogical proportions are defined between atoms from the same attribute domain. However, if $(a_1 : b_1 :: c_1 : d_1), \dots, (a_k : b_k :: c_k : d_k)$ are all valid, then we may also consider the following analogical proportion to be valid:

$$(a_1 \wedge \dots \wedge a_k) : (b_1 \wedge \dots \wedge b_k) :: (c_1 \wedge \dots \wedge c_k) : (d_1 \wedge \dots \wedge d_k) \quad (14)$$

The validity of (14), however, does not follow from (12) as the following example illustrates.

Example 1. Consider properties $a_1, b_1, c_1, d_1, a_2, b_2, c_2, d_2$ which are respectively characterised by the following sets of features:

$$\begin{array}{llll} A_1 = \{f_1, f_2\} & B_1 = \{f_2, f_3\} & C_1 = \{f_1, f_4\} & D_1 = \{f_4, f_3\} \\ A_2 = \{f_2, f_4\} & B_2 = \{f_3\} & C_2 = \{f_2, f_4\} & D_2 = \{f_3\} \end{array}$$

Then we have $a_1 : b_1 :: c_1 : d_1$ and $a_2 : b_2 :: c_2 : d_2$:

$$\begin{aligned} A_1 \setminus B_1 &= C_1 \setminus D_1 = \{f_1\} \\ B_1 \setminus A_1 &= D_1 \setminus C_1 = \{f_3\} \\ A_2 \setminus B_2 &= C_2 \setminus D_2 = \{f_2, f_4\} \\ B_2 \setminus A_2 &= D_2 \setminus C_2 = \{f_3\} \end{aligned}$$

To obtain feature representations of conjunctions of properties, the union of the features of the conjuncts is used. For example, as an object satisfying $a_1 \wedge a_2$ has all the features of a_1 and a_2 , a suitable feature representation for $a_1 \wedge a_2$ is the set $A_1 \cup A_2$. It follows that for $(a_1 \wedge a_2) : (b_1 \wedge b_2) :: (c_1 \wedge c_2) : (d_1 \wedge d_2)$ to hold, we would need to have that

$$(A_1 \cup A_2) \setminus (B_1 \cup B_2) = (C_1 \cup C_2) \setminus (D_1 \cup D_2)$$

$$(B_1 \cup B_2) \setminus (A_1 \cup A_2) = (D_1 \cup D_2) \setminus (C_1 \cup C_2)$$

This is not the case:

$$\begin{aligned}(A_1 \cup A_2) \setminus (B_1 \cup B_2) &= \{f_1, f_2, f_4\} \setminus \{f_2, f_3\} = \{f_1, f_4\} \\(C_1 \cup C_2) \setminus (D_1 \cup D_2) &= \{f_1, f_2, f_4\} \setminus \{f_3, f_4\} = \{f_1, f_2\}\end{aligned}$$

However, a natural sufficient condition for (14) to hold is that each of the analogical proportions $a_i : b_i :: c_i : d_i$ refer to different features. Specifically, if $A_i \cup B_i \cup C_i \cup D_i \subseteq \mathcal{F}_i$ such that $\mathcal{F}_i \cap \mathcal{F}_j = \emptyset$ for $i \neq j$ then we have:

$$\begin{aligned}(A_1 \cup \dots \cup A_k) \setminus (B_1 \cup \dots \cup B_k) &= (A_1 \setminus B_1) \cup \dots \cup (A_k \setminus B_k) \\(B_1 \cup \dots \cup B_k) \setminus (A_1 \cup \dots \cup A_k) &= (B_1 \setminus A_1) \cup \dots \cup (B_k \setminus A_k) \\(C_1 \cup \dots \cup C_k) \setminus (D_1 \cup \dots \cup D_k) &= (C_1 \setminus D_1) \cup \dots \cup (C_k \setminus D_k) \\(D_1 \cup \dots \cup D_k) \setminus (C_1 \cup \dots \cup C_k) &= (D_1 \setminus C_1) \cup \dots \cup (D_k \setminus C_k)\end{aligned}$$

From this observation, it follows that (14) can be derived from $(a_1 : b_1 :: c_1 : d_1), \dots, (a_k : b_k :: c_k : d_k)$. For example from *dog : coyote :: cat : lynx* and *juvenile : adult :: juvenile : adult* it follows that

$$(\text{dog} \wedge \text{juvenile}) : (\text{coyote} \wedge \text{adult}) :: (\text{cat} \wedge \text{juvenile}) : (\text{lynx} \wedge \text{adult})$$

We can avoid any explicit references to negation, because of our assumption that atoms from the same attribute domain are pairwise disjoint. For instance $\neg \text{nocturnal}$ could thus be seen as an abbreviation of *diurnal* \vee *crepuscular*. Disjunction in the antecedent can be avoided as well, taking into account that $\alpha_1 \vee \dots \vee \alpha_k \rightarrow \beta$ is equivalent to the set of rules $\{\alpha_1 \rightarrow \beta, \dots, \alpha_k \rightarrow \beta\}$.

It remains unclear, however, how analogical proportion should interact with disjunctions in the consequent of rules. In fact, it might be more natural to avoid analogical proportions between disjunctions, by first approximating disjunctions by single atoms for coarser domains. For example, we may consider two attribute domains referring to the size of animals:

$$\begin{aligned}\text{Size}_1 &= \{\text{very-small}, \text{small}, \text{medium}, \text{large}, \text{very-large}\} \\\text{Size}_2 &= \{\text{smaller-than-human}, \text{comparable-to-human}, \text{larger-than-human}\}\end{aligned}$$

With the following intended relations

$$\begin{aligned}\text{very-small} \vee \text{small} \vee \text{medium} &\rightarrow \text{smaller-than-human} \\\text{large} &\rightarrow \text{comparable-to-human} \\\text{very-large} &\rightarrow \text{larger-than-human}\end{aligned}$$

A rule with *very-small* \vee *medium* in the consequent may be replaced by a rule with *smaller-than-human*, and instead of considering analogical proportions involving *very-small* \vee *medium*, we may use analogical proportions in the domain *Size*₂.

2.3 Approximate analogical proportions

Perfect analogical proportions are rare. For example, a leopard's fur is marked with rosettes, which is not the case for dogs, cats or wolves, which would prevent us from concluding that $\text{dog} : \text{wolve} :: \text{cat} : \text{leopard}$ holds, if the feature of having rosettes is considered. In practice, we would therefore be more likely to look for four-tuples that satisfy (13) for the most important features, rather than for all features. One way in which this could be accomplished is by using dimensionality reduction techniques such as singular value decomposition (SVD) [5]. Such methods map Boolean vector representations to lower-dimensional, but real-valued vectors. To use SVD, we therefore need a generalisation of (13) in which a_i, b_i, c_i and d_i can be graded, which is what we discuss in this section.

The main idea is to encode weighted features as atoms in a multi-valued logic. It has been argued in [14, 16] that an appropriate way of generalising (13) is by interpreting \wedge , \rightarrow and \equiv as follows:

$$\begin{aligned} x \wedge y &= \min(x, y) \\ x \rightarrow y &= \min(1, 1 - x + y) \\ x \equiv y &= (x \rightarrow y) \wedge (y \rightarrow x) = 1 - |x - y| \end{aligned}$$

Note that \wedge , \rightarrow and \equiv are respectively interpreted as the weak conjunction, implication and equivalence operators from Łukasiewicz logic. We can then talk about the degree to which 4 propositions α, β, γ and δ form an analogical proportion. In particular, let $(a_1, \dots, a_n), (b_1, \dots, b_n), (c_1, \dots, c_n)$, and (d_1, \dots, d_n) now be real-valued feature vectors encoding α, β, γ and δ respectively, and assume that $a_i, b_i, c_i, d_i \in [0, 1]$. Then the degree to which $\alpha : \beta :: \gamma : \delta$ holds is defined as

$$(\alpha : \beta :: \gamma : \delta) = \min_{i=1}^n ((a_i \rightarrow b_i) \equiv (c_i \rightarrow d_i)) \wedge ((b_i \rightarrow a_i) \equiv (d_i \rightarrow c_i))$$

in which \wedge , \rightarrow and \equiv are interpreted as the Łukasiewicz logic connectives. It should be noted, however, that this definition does not in general satisfy $(\alpha : \beta :: \gamma : \delta) = (\alpha : \gamma :: \beta : \delta)$, the latter being a natural generalization of the central permutation property (11), e.g. $(0 : 0.2 :: 1 : 0.8) = 0.8 \neq (0 : 1 :: 0.2 : 0.8) = 0.4$. An alternative multi-valued semantics for analogical proportions, addressing this issue, has recently been proposed in [16].

Consider again the example from the introduction, and assume that we are only interested in the size of animals and whether they belong to the canidae or felidae family. In that case we only need two features. The first is a weighted feature corresponding to the length (after an appropriate rescaling to the unit interval), while the second can be encoded as a binary feature which is 0 for canidae and 1 for felidae, e.g.:

$$\begin{array}{lll} \text{dog} : (0.2, 0) & \text{coyote} : (0.3, 0) & \text{wolf} : (0.5, 0) \\ \text{cat} : (0.2, 1) & \text{lynx} : (0.3, 1) & \text{leopard} : (0.6, 1) \end{array}$$

The degree to which four atoms (from the same attribute domain) form an analogical proportion can then easily be verified, e.g:

$$\begin{aligned} \text{dog : wolf :: cat : leopard} &= \min(0.2 \rightarrow 0.5 \equiv 0.2 \rightarrow 0.6, 0.5 \rightarrow 0.2 \equiv 0.6 \rightarrow 0.2, \\ &\quad 0 \rightarrow 0 \equiv 1 \rightarrow 1, 0 \rightarrow 0 \equiv 1 \rightarrow 1) \\ &= \min(1 \equiv 1, 0.3 \equiv 0.4, 1 \equiv 1, 1 \equiv 1) \\ &= 0.9 \end{aligned}$$

Similarly, we can derive:

$$\begin{aligned} \text{dog : coyote :: cat : lynx} &= 1 \\ \text{dog : coyote :: coyote : wolf} &= 0.9 \\ \text{cat : lynx :: lynx : leopard} &= 0.8 \\ \text{dog : wolf :: dog : lynx} &= 0 \end{aligned}$$

The atoms from the attribute domain *Size* can be encoded in one dimension:

$$\text{very-small : } 0 \quad \text{small : } 0.25 \quad \text{medium : } 0.5 \quad \text{large : } 0.75 \quad \text{very-large : } 1$$

Note how this encode makes explicit a number of relations between atoms from *Size*, e.g. the fact that *small* is intermediate between *very-small* and *medium*, or the fact that *small* is more similar to *very-small* than to *very-large*. Furthermore note that the first feature in the encoding of *dog*, ..., *leopard* does not necessarily have to use the same scale as the encoding of *very-small*, ..., *very-large*.

$$\begin{aligned} \text{very-small : small :: small : medium} &= 1 \\ \text{very-small : small :: medium : large} &= 1 \\ \text{very-small : small :: small : large} &= 0.75 \\ \text{very-small : small :: medium : medium} &= 0.75 \end{aligned}$$

Note in particular how this multi-valued setting allows us to express that one proposition is intermediate between two others, e.g. *dog : coyote :: coyote : wolf*. In the Boolean setting, this could not be expressed as the only Boolean solutions to $a : b :: b : c$ are $a = b = c = 0$ and $a = b = c = 1$, i.e. $a : b :: b : c$ then simply means that *a*, *b* and *c* are equivalent w.r.t. the considered features. When restricted to analogical proportions expressing betweenness, (9) amounts to a form of interpolative reasoning: from intermediate conditions, intermediate conclusions are derived.

The inference principle (9) can now be softened, by taking the view that whenever the conditions of four rules are close to being in an analogical proportion, the same should hold for their conclusions. For example, we may use the observation that $\text{dog : wolf :: cat : leopard} = 0.9$ to derive knowledge about the size of wolves, given the rules (1), (2) and (6). This requires us to find an atom *x* from *Size* such that $\text{small : } x :: \text{small : large}$ is an analogical proportion to a sufficiently large degree. The choice $x = \text{large}$ makes a perfect analogical proportion,

hence we can consider that wolves are most likely to be large. However, given that the analogical proportion $dog : wolf :: cat : leopard$ is not perfect, we may want to be more careful about what to conclude. In particular, we have that the choices $x = medium$ and $x = very-large$ make analogical proportions to degree 0.75, which we may consider to be sufficiently high. In that case, we would rather conclude that the size of wolves is medium, large, or very large. As a second example, consider the analogical proportion $cat : lynx :: lynx : leopard = 0.8$ which we may use to derive plausible conclusions about the size of a lynx. However, since the degree of 0.8 is lower, we may be even more careful about what we want to derive.

In general, we could consider the following generalisation of the inference principle (9), which depends on the parameters $\theta \in [0, 1]$ and $\lambda \in [0, 1]$:

$$\begin{array}{c} \alpha_1 \rightarrow \beta_1 \\ \alpha_2 \rightarrow \beta_2 \\ \alpha_3 \rightarrow \beta_3 \\ \hline \alpha_1 : \alpha_2 :: \alpha_3 : \alpha^* \geq \theta \\ \hline \alpha^* \rightarrow \bigvee \{\beta^* \mid (\beta_1 : \beta_2 :: \beta_3 : \beta^*) \geq \lambda\} \end{array} \quad (15)$$

This generalisation expresses that when $\alpha_1, \alpha_3, \alpha_3$ and α^* approximately make an analogical proportion, all we can derive about α^* is that some conclusion β^* holds which approximately makes an analogical proportion together with $\beta_1, \beta_2, \beta_3$. Since there may be several choices for β^* , we end up with a disjunction. We can retrieve (9) as a special case when $\theta = \lambda = 1$, provided that there is only one β^* for which $\beta_1 : \beta_2 :: \beta_3 : \beta^* = 1$. This latter assumption is quite natural, given that atoms from the same attribute domain are mutually exclusive.

As in the case of similarity-based reasoning, we are again faced with the task of selecting thresholds on degrees in a more or less arbitrary fashion. One possibility would be to fix θ to a value which is sufficiently close to 1, and then let λ be the highest value in $[0, 1]$ which does not introduce inconsistencies in the rule base. Another idea would be to encode the result in possibilistic logic [3], such that conclusions that can be derived using a higher value of θ and a lower value of λ receive a higher degree of certainty. In the previous example on the size of wolves, we could for instance encode the conclusion as

$$\{(wolve \rightarrow large, 0.5), (wolve \rightarrow medium \vee large \vee very-large, 0.75)\}$$

A fixed set of certainty degrees could be considered for this purpose (e.g. 0.25, 0.5, 0.75, 1) which correspond to increasingly more restrictive choices of the parameters θ and λ .

3 Interpolation and extrapolation of rule bases

The notion of analogical proportion is quite strict, as $a : b :: c : d$ requires the “direction of change” from a to b to be the same as the direction of change

from c to d , but it also requires the magnitude of the change to be the same. In most situations, the magnitude of the change is difficult to measure. Moreover, in most practical scenarios the amount of change between a and b is unlikely to be exactly the same as the amount of change between c and d . Approximate analogical proportions may alleviate this problem, but they come at the cost of introducing more or less arbitrary thresholds on degrees.

In this section, we explore an alternative which completes rule bases based on a weaker counterpart of the analogical proportion, which only looks at the direction of the change, ignoring the amount of change. For reasons which will become obvious below, we need to consider interpolative reasoning (i.e. deriving intermediate conclusions from intermediate conditions) separate from extrapolative reasoning. In contrast, analogical proportions can be used straightforwardly for interpolative reasoning as well as extrapolation, considering that in the multi-valued case, $a : b :: b : c$ means that b is (half-way) between a and c .

First, in Section 3.1 we discuss an approach to interpolative reasoning which is more cautious than the analogical proportion based method. In Section 3.2 we then present a similar idea for extrapolative reasoning. Finally, in Section 3.3 we give some details on the underlying semantics of these methods, trying to make explicit under which assumptions the proposed forms of plausible inference are sound.

3.1 Interpolation

At the syntactic level, the idea of interpolating rules corresponds to the following inference rule:

$$\frac{\alpha_1 \rightarrow \beta_1 \\ \alpha_2 \rightarrow \beta_2}{\alpha_1 \bowtie \alpha_2 \rightarrow \beta_1 \bowtie \beta_2} \quad (16)$$

where we write $\alpha_1 \bowtie \alpha_2$ for the disjunction of all formulas that are conceptually between α_1 and α_2 . For example, *chianti* \bowtie *merlot* would be the disjunction of all wines whose taste is between that of *chianti* and *merlot*. In practice, it may be difficult or even impossible to characterise $\alpha_1 \bowtie \alpha_2$ and $\beta_1 \bowtie \beta_2$ using the available labels and the usual propositional connectives. For example, while we may know that *barbera* is between *chianti* and *merlot*, we may not necessarily be able to enumerate all such wines. Even worse, sometimes the available labels make it impossible to exactly characterise $\alpha_1 \bowtie \alpha_2$. For instance, let $\alpha_1 = 3\text{-bedroom-apartment}$ and $\alpha_2 = \text{penthouse}$, then we may wonder whether a *loft* should be included in the disjunction $\alpha_1 \bowtie \alpha_2$, i.e. whether $\text{loft} \rightarrow 3\text{-bedroom-apartment} \bowtie \text{penthouse}$ holds. A loft with three bedrooms can be considered intermediate between a 3 bedroom apartment and a 3 bedroom penthouse, but for a loft with fewer bedrooms this is harder to justify. In other words, while some lofts are conceptually between 3 bedroom apartments and penthouses, this does not hold for all lofts. If the language does not contain a

label for 3 bedroom lofts, we may therefore not be able to precisely characterise *3-bedroom-apartment* \bowtie *penthouse*.

Because of this observation, in practice we are left with approximating $\alpha_1 \bowtie \alpha_2$. In particular, we assume that we have access to rules of the form

$$\alpha^* \rightarrow \alpha_1 \bowtie \alpha_2 \quad \beta_1 \bowtie \beta_2 \rightarrow \beta^*$$

which indicate, respectively, that at least all situations covered by α^* are conceptually between α_1 and α_2 , and that all situations which are conceptually between β_1 and β_2 satisfy β^* . We may consider, for instance

$$\begin{aligned} wolf &\rightarrow dog \bowtie coyote \\ bistro &\rightarrow pub \bowtie restaurant \\ motorbike &\rightarrow bicycle \bowtie car \\ hard-rock \bowtie heavy-metal &\rightarrow guitar-based-music \\ tundra \bowtie temperate-forest &\equiv tundra \vee taiga \vee temperate-forest \\ very-small \bowtie medium &\equiv very-small \vee small \vee medium \end{aligned}$$

Example 2. Consider the following rules:

$$chianti \rightarrow low-tannins \wedge medium-body \tag{17}$$

$$merlot \rightarrow (low-tannins \vee mid-tannins) \wedge medium-body \tag{18}$$

From (17) and (18) we derive using interpolation:

$$chianti \bowtie merlot \rightarrow (lt \wedge mb) \bowtie ((lt \vee mt) \wedge mb)$$

where we have abbreviated the labels for the ease of presentation (e.g. *lt* stands for *low-tannins*). Considering

$$\begin{aligned} (lt \wedge mb) \bowtie ((lt \vee mt) \wedge mb) &\equiv (lt \vee mt) \wedge mb \\ barbera &\rightarrow chianti \bowtie merlot \end{aligned}$$

we find using classical deduction that

$$barbera \rightarrow (lt \vee mt) \wedge mb$$

Notice how the symbol \bowtie is essentially treated as a binary modality. We assume this modality to be reflexive and symmetric in the sense that

$$\alpha \bowtie \alpha \equiv \alpha \quad \alpha \bowtie \beta \equiv \beta \bowtie \alpha$$

for any propositional formulas α and β . We moreover assume that α and β themselves are “between α and β ”, i.e.

$$\alpha \vee \beta \rightarrow \alpha \bowtie \beta$$

In practice, we may only have information about the betweenness of atoms (i.e. individual labels) and not about the betweenness of more complex propositional formulas. We may consider, for instance, the following inference rules to lift betweenness for atoms to betweenness for formulas:

$$\frac{\alpha \rightarrow \alpha_1 \bowtie \alpha_2 \\ \beta \rightarrow \beta_1 \bowtie \beta_2}{(\alpha \vee \beta) \rightarrow (\alpha_1 \vee \beta_1) \bowtie (\alpha_2 \vee \beta_2)} \quad (19)$$

$$\frac{\alpha_1 \bowtie \alpha_2 \rightarrow \alpha}{(\alpha_1 \vee \beta) \bowtie (\alpha_2 \vee \beta) \rightarrow (\alpha \vee \beta)} \quad (20)$$

$$\frac{\alpha \rightarrow \alpha_1 \bowtie \alpha_2 \\ \alpha, \alpha_1 \text{ and } \alpha_2 \text{ are "logically independent" from } \beta}{(\alpha \wedge \beta) \rightarrow (\alpha_1 \wedge \beta) \bowtie (\alpha_2 \wedge \beta)} \quad (21)$$

$$\frac{\alpha_1 \bowtie \alpha_2 \rightarrow \alpha \\ \beta_1 \bowtie \beta_2 \rightarrow \beta}{(\alpha_1 \wedge \beta_1) \bowtie (\alpha_2 \wedge \beta_2) \rightarrow (\alpha \wedge \beta)} \quad (22)$$

Note that because \bowtie is symmetric, from the premises of (19) we can also derive

$$\begin{aligned} & (\alpha \vee \beta) \rightarrow (\alpha_1 \vee \beta_2) \bowtie (\alpha_2 \vee \beta_1) \\ & (\alpha \vee \beta) \rightarrow (\alpha_2 \vee \beta_1) \bowtie (\alpha_1 \vee \beta_2) \\ & (\alpha \vee \beta) \rightarrow (\alpha_2 \vee \beta_2) \bowtie (\alpha_1 \vee \beta_1) \end{aligned}$$

and similar for (20)–(22).

A semantic justification for these inference rules has been given in [22]; we will briefly come back to it in Section 3.3. We also refer to Section 3.3 and [22] for a precise definition of the notion of logical independence which is used in (21).

3.2 Extrapolation

Geometrically, analogical proportions correspond to the idea of a parallelogram, indicating that the direction of change to go from a to b is parallel to the direction of change from c to d , and that the amount of change is identical. As this latter amount is difficult to quantify, we will restrict our attention to the direction of change. In particular, we write $\gamma \triangleright \langle \alpha, \beta \rangle$ for the disjunction of all propositional formulas δ which differ from γ in a way which is qualitatively similar to the

way in which β differs from α . Like in the case of analogical proportions, we are essentially looking for pairs of properties that differ in the same way. In contrast to the case of analogical proportions, however, we only require this difference to be qualitatively similar.

Example 3. We may consider that

$$\text{medium} \triangleright \langle \text{very-small}, \text{large} \rangle \equiv \text{medium} \vee \text{large} \vee \text{very-large}$$

Indeed, the change from *very-small* to *large* denotes an increase in size. Therefore the sizes compatible with $\text{medium} \triangleright \langle \text{very-small}, \text{large} \rangle$ are those that are at least as large as *medium*.

As in the case of betweenness, when we move from uni-dimensional to multi-dimensional domains, it is often not possible to precisely characterise formulas of the form $\gamma \triangleright \langle \alpha, \beta \rangle$. For example, we may assume

$$\begin{aligned} \text{prog-metal} &\rightarrow \text{heavy-metal} \triangleright \langle \text{hard-rock}, \text{prog-rock} \rangle \\ \text{leopard} \vee \text{lynx} &\rightarrow \text{cat} \triangleright \langle \text{dog}, \text{coyote} \rangle \\ \text{leopard} \vee \text{lynx} &\rightarrow \text{cat} \triangleright \langle \text{dog}, \text{wolf} \rangle \end{aligned}$$

Notice in the last two rules, how the amount of change is effectively ignored.

We assume that the modality $\cdot \triangleright \langle \cdot, \cdot \rangle$ satisfies the following properties:

$$\beta \rightarrow \alpha \triangleright \langle \alpha, \beta \rangle \quad \gamma \rightarrow \gamma \triangleright \langle \alpha, \beta \rangle$$

provided that α and β are consistent.

We obtain a form of extrapolative reasoning by assuming that analogical changes in the antecedent of rules should lead to analogical changes in the consequent:

$$\begin{array}{c} \alpha_1 \rightarrow \beta_1 \\ \alpha_2 \rightarrow \beta_2 \\ \alpha_3 \rightarrow \beta_3 \\ \hline \alpha_3 \triangleright \langle \alpha_1, \alpha_2 \rangle \rightarrow \beta_3 \triangleright \langle \beta_1, \beta_2 \rangle \end{array} \tag{23}$$

To lift information about analogical changes between atomic labels to analogical changes of propositional formulas, we can make use of the following inference rules:

$$\begin{array}{c} \alpha \rightarrow \alpha_3 \triangleright \langle \alpha_1, \alpha_2 \rangle \\ \beta \rightarrow \beta_3 \triangleright \langle \beta_1, \beta_2 \rangle \\ \hline (\alpha \vee \beta) \rightarrow (\alpha_3 \vee \beta_3) \triangleright \langle (\alpha_1 \vee \beta_1), (\alpha_2 \vee \beta_2) \rangle \end{array} \tag{24}$$

$$\begin{array}{c} \alpha_3 \triangleright \langle \alpha_1, \alpha_2 \rangle \rightarrow \alpha \\ \beta_3 \triangleright \langle \beta_1, \beta_2 \rangle \rightarrow \beta \\ \hline (\alpha_3 \vee \beta_3) \triangleright \langle (\alpha_1 \vee \beta_1), (\alpha_2 \vee \beta_2) \rangle \rightarrow (\alpha \vee \beta) \end{array} \tag{25}$$

$$\frac{\alpha \rightarrow \alpha_3 \triangleright \langle \alpha_1, \alpha_2 \rangle}{\alpha, \alpha_1 \text{ and } \alpha_2 \text{ are "logically independent" from } \beta \text{ and } \gamma} \quad (26)$$

$(\alpha \wedge \gamma) \rightarrow (\alpha_3 \wedge \gamma) \triangleright \langle (\alpha_1 \wedge \beta), (\alpha_2 \wedge \beta) \rangle$

$$\frac{\alpha_3 \triangleright \langle \alpha_1, \alpha_2 \rangle \rightarrow \alpha}{\beta_3 \triangleright \langle \beta_1, \beta_2 \rangle \rightarrow \beta} \quad (27)$$

$(\alpha_3 \wedge \beta_3) \triangleright \langle (\alpha_1 \wedge \beta_1), (\alpha_2 \wedge \beta_2) \rangle \rightarrow (\alpha \wedge \beta)$

We again refer to [22] for a formal justification of these rules and the extrapolation principle (23) itself.

Example 4. Consider the following rule base about houses:

$$large \wedge detached \rightarrow comfortable \vee luxurious \quad (28)$$

$$large \wedge row\text{-}house \rightarrow comfortable \quad (29)$$

$$small \wedge detached \rightarrow basic \vee comfortable \quad (30)$$

which defines the comfort level (basic, comfortable, luxurious) of a house, based on its size (small, medium, large) and type (detached, row-house, semi-detached). From the extrapolation principle (23) we find

$$\begin{aligned} & ((small \wedge det) \triangleright \langle (large \wedge det), (large \wedge rh) \rangle) \\ & \rightarrow ((bas \vee comf) \triangleright \langle (comf \vee lux), (comf \vee comf) \rangle) \end{aligned} \quad (31)$$

where we have abbreviated some labels for the ease of presentation.

Since $\beta \rightarrow \alpha \triangleright \langle \alpha, \beta \rangle$ for any α and β , we find

$$rh \rightarrow det \triangleright \langle det, rh \rangle$$

where we have again abbreviated the labels. Using (26) this leads to

$$(small \wedge rh) \rightarrow (small \wedge det) \triangleright \langle (large \wedge det), (large \wedge rh) \rangle \quad (32)$$

From

$$\begin{aligned} bas \triangleright \langle comf, comf \rangle & \rightarrow bas \\ comf \triangleright \langle lux, comf \rangle & \rightarrow (bas \vee comf) \end{aligned}$$

we find using (25):

$$(bas \vee comf) \triangleright \langle (comf \vee lux), (comf \vee comf) \rangle \rightarrow (bas \vee comf) \quad (33)$$

Combining (31)–(33), we find

$$(small \wedge row\text{-}house) \rightarrow (bas \vee comf) \quad (34)$$

Intuitively, from the rule base (28)–(30) we derive that detached houses are more comfortable than row houses, hence a small row house can not be more comfortable than a small detached house.

If we now consider that

$$\text{semi-detached} \rightarrow \text{row-house} \bowtie \text{detached}$$

using the interpolation principle (16) we can derive from (30) and (34) that

$$(\text{small} \wedge \text{semi-detached}) \rightarrow (\text{bas} \vee \text{comf})$$

i.e. since semi-detached houses are intermediate between detached houses and row houses, their comfort level should be intermediate as well.

3.3 Semantics

To characterise interpolation and extrapolation at the semantic level, we need to make explicit what is the meaning of $\alpha \bowtie \beta$ (i.e. the notion of betweenness) and $\alpha \triangleright \langle \beta, \gamma \rangle$ (i.e. the notion of parallelism). Given the spatial nature of betweenness and parallelism, it seems natural to consider a geometric representation of properties, using the idea of conceptual spaces [7]. Below we give a sketch of this characterisation; for more details, we refer to [22].

We assume that the meaning of every property can be represented as a convex region in some geometric space, whose dimensions correspond to elementary cognitive features; they are usually called “quality dimensions” in this context. In the case of labels referring to wines, for instance, there would be quality dimensions corresponding to colour (e.g. three dimensions, encoding hue, saturation and intensity), dimensions corresponding to the texture of the wine, its taste, smell, etc. The points of the conceptual space would then correspond to specific instances, while regions correspond to categories. Essentially, the region representing a category (e.g. *chianti*) corresponds to the points which are closest to the prototypes of that category [8], which is why such regions are naturally convex. For simplicity, we assume that conceptual spaces are Euclidean spaces.

Given this geometric setting, betweenness can naturally be characterised: we say that a category b is between a and c , if some point of the region corresponding with b is between some point of the region corresponding with a and some point of the region corresponding with c . To make this more precise, let us write $\text{reg}(\alpha)$ for the conceptual space representation of a propositional formula α , where $\text{reg}(\alpha \wedge \beta) = \text{reg}(\alpha) \cap \text{reg}(\beta)$ and $\text{reg}(\alpha \vee \beta) = \text{reg}(\alpha) \cup \text{reg}(\beta)$. Note that while atomic propositions will correspond to convex regions, the region corresponding to arbitrary propositional formulas does not need to be convex. As before, we do not explicitly consider negation, and rely on the assumption that propositions are grouped in pairwise disjoint attribute domains instead. In the wine example, we may for instance consider the domain $A = \{\text{low-tannins}, \text{mid-tannins}, \text{high-tannins}\}$.

We have $\alpha \rightarrow \beta \bowtie \gamma$ iff

$$\forall q \in \text{reg}(\alpha) . \exists p \in \text{reg}(\beta), r \in \text{reg}(\gamma), \lambda \in [0, 1] . \overrightarrow{pq} = \lambda \cdot \overrightarrow{pr}$$

and $\beta \bowtie \gamma \rightarrow \alpha$ iff

$$\forall p \in \text{reg}(\beta), r \in \text{reg}(\gamma), \lambda \in [0, 1] . p + \lambda \cdot \vec{pr} \in \text{reg}(\alpha)$$

where the points $p + \lambda \cdot \vec{pr}$ for $\lambda \in [0, 1]$ are exactly the points which are between p and r . From these characterizations it is easy to verify that inference rules (19), (20) and (22) are indeed valid. The argument for (21) is a bit more subtle. The intuition is that because of the assumption of logical independence, we can see the underlying conceptual space as a Cartesian product $\mathcal{C}_1 \times \mathcal{C}_2$ such that $\text{reg}(\alpha)$, $\text{reg}(\alpha_1)$ and $\text{reg}(\alpha_2)$ are all of the form $X \times \mathcal{C}_2$ (i.e. the dimensions in \mathcal{C}_2 are irrelevant for describing the categories α , α_1 and α_2), whereas $\text{reg}(\beta)$ is of the form $\mathcal{C}_1 \times Y$ (i.e. the dimensions in \mathcal{C}_1 are irrelevant for describing the category β).

Turning our attention now to extrapolation, at the semantic level we have $\delta \rightarrow \alpha \triangleright \langle \beta, \gamma \rangle$ iff

$$\forall s \in \text{reg}(\delta) . \exists p \in \text{reg}(\beta), q \in \text{reg}(\gamma), r \in \text{reg}(\alpha), \lambda \geq 0 . \vec{rs} = \lambda \cdot \vec{pq}$$

and $\alpha \triangleright \langle \beta, \gamma \rangle \rightarrow \delta$ iff

$$\forall p \in \text{reg}(\beta), q \in \text{reg}(\gamma), r \in \text{reg}(\alpha), \lambda \geq 0 . r + \lambda \cdot \vec{pq} \in \text{reg}(\delta)$$

Again a geometric argument can be constructed from these definitions to justify the relevant inference rules, viz. (24)–(27).

To describe the interpolation and extrapolation process itself, i.e. inference rules (16) and (23), we consider a propositional rule base R containing negation-free rules. The antecedent α of a rule corresponds to a region $\text{reg}_1(\alpha)$ in some conceptual space \mathcal{C}_1 (typically corresponding to the Cartesian product of more elementary conceptual spaces). Similarly, the consequent β of a rule corresponds to a region $\text{reg}_2(\beta)$ in a conceptual space \mathcal{C}_2 . We can thus view the rule base R as a mapping f from regions of \mathcal{C}_1 to regions of \mathcal{C}_2 in such a way that a rule $\alpha^* \rightarrow \beta^*$ can be derived from the rule base R using classical deduction iff $f(\text{reg}_1(\alpha^*)) \subseteq \text{reg}_2(\beta^*)$.

By making meta-assumptions about the relationship between the conceptual spaces \mathcal{C}_1 and \mathcal{C}_2 , we can refine the mapping f . In particular, in many cases \mathcal{C}_2 will be a subspace of \mathcal{C}_1 (i.e. the quality dimensions that are needed to describe the labels in the consequents of rules are a subset of those needed to describe the antecedents). In such a case, f is the approximation of a linear mapping from points of \mathcal{C}_1 to points of \mathcal{C}_2 . We can then refine f to a mapping \hat{f} such that $f(X) \setminus \hat{f}(X)$ are all points from \mathcal{C}_2 that could never be obtained by a linear mapping from \mathcal{C}_1 which is consistent with f . It can then be shown [22] that $\hat{f}(\text{reg}_1(\alpha^*)) \subseteq \text{reg}_2(\beta^*)$ iff $\alpha^* \rightarrow \beta^*$ can be derived from R using inference rule (16), and (19)–(22) together with classical deduction. *In other words, at the semantic level, the interpolative inference rule (16) and extrapolative inference rule (23) correspond to the assumption that rule bases are approximations of a linear mapping.* This essentially corresponds to the assumption that the rule base is in some sense regular. We refer to [22] for more details.

4 Discussion

The interpolative and extrapolative inference rules from Section 3 are more cautious than the inference rule (9). However, we can refine the rules (16) and (23), by adding an explicit reference to the amount of change. To illustrate this, we will consider the following generalisation of (23):

$$\begin{array}{c} \alpha_1 \rightarrow \beta_1 \\ \alpha_2 \rightarrow \beta_2 \\ \alpha_3 \rightarrow \beta_3 \\ \hline \alpha_3 \triangleright_{[\lambda,\mu]} \langle \alpha_1, \alpha_2 \rangle \rightarrow \beta_3 \triangleright_{[\lambda,\mu]} \langle \beta_1, \beta_2 \rangle \end{array} \quad (35)$$

Intuitively, an expression such as $\beta_1 \triangleright_{[\lambda,\mu]} \langle \beta_2, \beta_3 \rangle$, where $0 \leq \lambda \leq \mu < +\infty$, covers all situations that can be obtained by changing β_1 in the same direction as the change from β_2 to β_3 , such that the amount of change is between λ and μ times as large as the amount of change between β_2 and β_3 .

Formally, we have $\delta \rightarrow \alpha \triangleright_{[\lambda,\mu]} \langle \beta, \gamma \rangle$ iff

$$\forall s \in \text{reg}(\delta). \exists p \in \text{reg}(\beta), q \in \text{reg}(\gamma), r \in \text{reg}(\alpha), \rho \in [\lambda, \mu]. \vec{rs} = \rho \cdot \vec{pq}$$

and $\alpha \triangleright_{[\lambda,\mu]} \langle \beta, \gamma \rangle \rightarrow \delta$ iff

$$\forall p \in \text{reg}(\beta), q \in \text{reg}(\gamma), r \in \text{reg}(\alpha), \rho \in [\lambda, \mu]. r + \rho \cdot \vec{pq} \in \text{reg}(\delta)$$

The most straightforward use of this generalization is to express whether the amount of change from β_1 should be smaller, equal, or larger than the amount of change from β_2 to β_3 . In particular, $\beta_1 \triangleright_{[1,1]} \langle \beta_2, \beta_3 \rangle$ corresponds to the solution X that makes $\beta_1 : X :: \beta_2 : \beta_3$ a perfect analogical proportion, while $\beta_1 \triangleright_{[0,1]} \langle \beta_2, \beta_3 \rangle$ and $\beta_1 \triangleright_{]1,+\infty[} \langle \beta_2, \beta_3 \rangle$ express amounts of change that are smaller and larger, respectively, than the amount of change from β_1 to β_2 . Note that $\beta_1 \triangleright_{[0,+\infty[} \langle \beta_2, \beta_3 \rangle$ corresponds to $\beta_1 \triangleright \langle \beta_2, \beta_3 \rangle$. Also note that in the aforementioned cases, the approach remains entirely qualitative. Other choices for the intervals $[\lambda, \mu]$ may give the approach a more numerical flavour, and would mainly be useful in scenarios where the conceptual relationships are obtained using data-driven techniques.

In the particular case where $\lambda = \mu = 1$, (35) thus coincides with (9). However, note that an analogical proportion of conjunctions corresponds to a conjunction of analogical proportions, as expressed in (14). The properties of the extrapolative operator \triangleright expressed in (26)–(27) reveal a weaker relationship with conjunction. As it turns out, in the specific case where $\lambda = \mu$ (and a fortiori when $\lambda = \mu = 1$), the following property can be shown:

$$\begin{array}{c} \alpha \rightarrow \alpha_3 \triangleright_{[\lambda,\lambda]} \langle \alpha_1, \alpha_2 \rangle \\ \beta \rightarrow \beta_3 \triangleright_{[\lambda,\lambda]} \langle \beta_1, \beta_2 \rangle \\ \alpha, \alpha_1 \text{ and } \alpha_2 \text{ are "logically independent" from } \beta, \beta_1 \text{ and } \beta_2 \\ \hline (\alpha \wedge \beta) \rightarrow (\alpha_3 \wedge \beta_3) \triangleright_{[\lambda,\lambda]} \langle (\alpha_1 \wedge \beta_1), (\alpha_2 \wedge \beta_2) \rangle \end{array}$$

The validity of using analogical proportions to complete rule bases, i.e. inference rule (9), thus depends on two assumptions

1. The rule base is the symbolic representation of a linear mapping between conceptual spaces.
2. For $a : b :: c : d$ to hold, the amount of change between a and b should be exactly the same as the amount of change between c and d .

The second assumption can be weakened, allowing the amount of change to be approximately similar, but in such a case we should be more careful with rules with conjunctions in the antecedent, and in particular refrain from using the property expressed in (14) about the interaction between analogical proportion and conjunction. The approach outlined in Section 3 allows us to drop the second assumption altogether, but at the cost of a much weaker inference relation.

The plausibility of the first assumption depends on the kind of rules that are considered. Consider for example the following rules:

$$\begin{aligned} \textit{studio} &\rightarrow \textit{small} \\ \textit{high-tannins} \wedge \textit{full-body} &\rightarrow \textit{opaque} \\ \textit{museum} \wedge \textit{has-live-animals} &\rightarrow \textit{zoo} \\ \textit{large} \wedge \textit{orchestra} &\rightarrow \textit{symphony} \end{aligned}$$

In all of these cases, the validity of the rules only depends on the meaning of the propositions in the antecedent. In terms of conceptual spaces, this indicates that the quality dimensions needed to represent the consequent are a subset of the quality dimensions that are needed to represent the antecedent. The rule base then corresponds to the projection of a higher-dimensional space to a lower-dimensional sub-space, which would indeed correspond to a linear mapping. However, for other types of rules such an assumption would be less plausible, e.g.:

$$\begin{aligned} \textit{morning} &\rightarrow \textit{heavy-traffic} \\ \textit{autumn} \wedge \textit{UK} &\rightarrow \textit{rainy} \end{aligned}$$

The validity of these rules depends on observations we make about the world, rather than on their intrinsic meaning. In such a case there is no reason why there should be a linear mapping underlying the rules. In such a case, however, we may still expect that similar conditions lead to similar results. In particular, in contrast to analogical proportion based reasoning, the plausibility of similarity based reasoning (in the sense of the example in (7)–(8)) does not rely on an assumption of linearity. Conversely, while linearity is a sufficient condition for analogical proportion based inference to be sound, it is neither sufficient nor necessary for similarity based reasoning to be sound.

Acknowledgment

The authors are indebted to the two reviewers for their useful comments about the presentation of the ideas.

References

1. R. Carnap. A basic system of inductive logic, part 2. In *Studies in Inductive Logic and Probability*, volume 2, pages 7–155. University of California Press, 1980.
2. D. Dubois, F. Esteva, L. Godo, and H. Prade. An information-based discussion of vagueness. In *Proceedings of the 10th IEEE International Conference on Fuzzy Systems*, volume 2, pages 781–784, 2001.
3. D. Dubois, J. Lang, and H. Prade. Possibilistic logic. In D. N. D. Gabbay, C. Hogger J. Robinson, editor, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 3, pages 439–513. Oxford University Press, 1994.
4. K. Fine. Vagueness, truth and logic. *Synthese*, 30(3):265–300, 1975.
5. G. W. Furnas, S. Deerwester, S. T. Dumais, T. K. Landauer, R. A. Harshman, L. A. Streeter, and K. E. Lochbaum. Information retrieval using a singular value decomposition model of latent semantic structure. In *Proceedings of the 11th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 465–480, 1988.
6. E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, volume 6, pages 1606–1611, 2007.
7. P. Gärdenfors. *Conceptual Spaces: The Geometry of Thought*. MIT Press, 2000.
8. P. Gärdenfors and M. Williams. Reasoning about categories in conceptual spaces. In *International Joint Conference on Artificial Intelligence*, pages 385–392, 2001.
9. G. Kern-Isberner and J. Fisseler. Knowledge discovery by reversing inductive knowledge representation. In *Proceedings of the 9th International Conference on Principles of Knowledge Representation and Reasoning*, pages 34–44, 2004.
10. G. Lakoff. Hedges: A study in meaning criteria and the logic of fuzzy concepts. *Journal of philosophical logic*, 2(4):458–508, 1973.
11. J. Lawry and Y. Tang. Uncertainty modelling for vague concepts: A prototype theory approach. *Artificial Intelligence*, 173(18):1539 – 1558, 2009.
12. J. B. Losos. Phylogenetic niche conservatism, phylogenetic signal and the relationship between phylogenetic relatedness and ecological similarity among species. *Ecology letters*, 11(10):995–1003, 2008.
13. L. Miclet and H. Prade. Handling analogical proportions in classical logic and fuzzy logics settings. In C. Sossai and G. Chemello, editors, *European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 638–650, 2009.
14. H. Prade and G. Richard. Multiple-valued logic interpretations of analogical, reverse analogical, and paralogical proportions. In *Proceedings of the 40th IEEE International Symposium on Multiple-Valued Logic*, pages 258–263, 2010.
15. H. Prade and G. Richard. Reasoning with logical proportions. In *Proceedings of the 12th International Conference on Principles of Knowledge Representation and Reasoning*, pages 545–555, 2010.
16. H. Prade and G. Richard. Analogical proportions and multiple-valued logics. In L. C. van der Gaag, editor, *Proceedings of the 12th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, volume 7958, pages 497–509, 2013.
17. H. Prade and S. Schockaert. Completing rule bases in symbolic domains by analogy making. In *Proceedings of the EUSFLAT-LFA Conference*, pages 928–934, 2011.
18. E. H. Rosch. Natural categories. *Cognitive Psychology*, 4(3):328–350, 1973.

19. S. Schockaert and H. Prade. Interpolation and extrapolation in conceptual spaces: A case study in the music domain. In *Proceedings of the 5th International Conference on Web Reasoning and Rule Systems*, pages 217–231, 2011.
20. S. Schockaert and H. Prade. Solving conflicts in information merging by a flexible interpretation of atomic propositions. *Artificial Intelligence*, 175(11):1815 – 1855, 2011.
21. S. Schockaert and H. Prade. Cautious analogical-proportion based reasoning using qualitative conceptual relations. In *Proceedings of the ECAI 2012 Workshop on Similarity and Analogy-based Methods in AI*, pages 41–48, 2012.
22. S. Schockaert and H. Prade. Interpolative and extrapolative reasoning in propositional theories using qualitative knowledge about conceptual spaces. *Artificial Intelligence*, to appear.
23. R. Speer, C. Havasi, and H. Lieberman. Analogyspace: reducing the dimensionality of common sense knowledge. In *Proceedings of the 23rd AAAI Conference on Artificial intelligence*, pages 548–553, 2008.
24. R. Sun. Robust reasoning: integrating rule-based and similarity-based reasoning. *Artificial Intelligence*, 75(2):241–295, 1995.

From analogical proportion to logical proportions

A survey

Henri Prade and Gilles Richard

University of Toulouse, IRIT-CNRS, France, email: {prade, richard}@irit.fr

Abstract. Analogies play an important role in many reasoning tasks. This chapter surveys a series of recent works developing a logical view of the notion of analogical proportion, and its applications. Analogical proportions are statements of the form “ A is to B as C is to D ”. The logical representation used for encoding such proportions takes both into account what the four situations A, B, C, D have in common and how they differ. Thanks to the use of a Boolean modeling extended with suitable fuzzy logic connectives, the approach can deal with situations described by features that may be binary or multiple-valued. It is shown that an analogical proportion is a particular case of a more general concept, namely the one of logical proportion. Among the 120 existing logical proportions, we single out two groups of 4 proportions for their remarkable properties: the homogeneous proportions (including the analogical proportion) which are symmetrical, and the heterogeneous proportions which are not. These 8 proportions are the only logical proportions to satisfy a remarkable code-independency property. We emphasize the interest of these two groups of proportions for dealing with a variety of reasoning tasks, ranging from the solving of IQ tests, to transductive reasoning for classification, to interpolative and extrapolative reasoning, and also to the handling of quizzes of the “find the odd one out” type. The approach does not just rely on the exploitation of similarities between pairs of cases (as in case-based reasoning), but rather takes advantage of the parallel made between a situation to be evaluated or to be completed, with triples of other situations.

1 Introduction

Reasoning is at the core of human intelligence, allowing us to infer new knowledge from available knowledge. While logical inference generates valid conclusions, analogical inference does not offer such a guarantee and only leads to plausible conclusions. An *analogy* parallels two particular situations on the basis of some *similarities* (the notions of situation and similarity should be understood here in a broad sense). Based on such an analogy, *analogical inference* tacitly assumes that these two situations might be similar in other respects and then draws conclusions on this basis. Due to the brittleness of its conclusions, analogical reasoning is not easily amenable to a formal logic framework. Nevertheless, different formal modelings have been proposed, using first order logic as in [12], second order logic as in [20], algebraic oriented frameworks as in [11, 63], or even a complexity-based approach as in [8, 5] that establishes a close link with computational learning theory.

From a more cognitive, but also computational viewpoint, analogy may be a matter of structure and proportion. When dealing with structures, an analogy between *two*

situations is determined through mappings that apply to relations on the one hand, and to objects on the other hand; this is the core of structure-mapping theory [18]; see also [71]. Proportional analogy relates *four* items, in statements of the form “*a is to b as c is to d*”, called *analogical proportions*, and usually denoted $a : b :: c : d$. Often, proportional analogies may be encountered inside a structural analogy. Thus, one can say that “a water flow is to a reservoir as a current is to a battery” in the setting of the structural analogy between hydraulics and electricity. Inference then amounts to find a value for an unknown item x such that the proportion $a : b :: c : x$ holds. The pioneering work of Thomas Evans [14], and the well-known Copycat system [24] are famous implementations of proportional analogy inferential devices. In the field of artificial intelligence, analogical reasoning has mainly been viewed as a powerful heuristic tool [22, 19, 66, 42], being useful at the meta level for improving deductive reasoning provers [35], as well as in problem solving and in learning, e.g. [25, 33, 29, 23]. This involves not only symbolic representations, but also numerical calculations [16] for evaluating similarities and/or handling quantitative estimation [43].

In the last decade, some authors starting with the pioneering investigation made in [31] with computational linguistic motivations, have started to develop a large panel of algebraic models for analogical proportions, from semi-groups to lattices, through words over finite alphabets and finite trees [68, 67, 38, 36]. Moreover, the use of analogical proportions for machine learning purposes in [6, 37] has been also proposed and studied. Since analogical proportions have various instances in natural language, it is not surprising that the field of natural language processing has also been investigated [3, 32, 30, 70, 7] providing encouraging results for automatic translation or text comprehension [69], or even recommendation systems [62].

However, more recently, a propositional logic representation of analogical proportion has been proposed [39, 40]. Still, this view has its roots in the largely ignored work of the anthropologist, linguist, and computer scientist Sheldon Klein [26, 27], who was the first to propose a truth table-like way for finding x such that the analogical proportion $a : b :: c : x$ holds. Besides, in the annex of a 1952 French book, the psychologist Jean Piaget [44] informally investigates a similar idea, still without explicitly mentioning analogy (see also [45] pp. 35–37), where a definition of a so-called *proportion logique* is given.

The logical view of analogical proportion has been further developed in a series of works [47, 48, 52, 49] leading to the general concept of logical proportions, to the introduction of two remarkable subsets, each one containing four proportions, namely the homogeneous proportions and the heterogeneous proportions. Then they are extended to multiple-valued settings [50], and various applications to reasoning and classification are considered. Our aim in this paper is to provide a survey introduction to the Boolean approach to analogy and a short overview of its potential developments and applications. The paper is organized in three main parts: section 2 and its subsections present the main results on the modeling of the homogeneous proportions (including analogical proportion). Section 3 and its subsections focus on the heterogeneous proportions, their link with homogeneous proportions and their properties. The remaining sections discuss the interest of the approach in the solving of IQ tests and of “find the odd one out” quizzes, as well as in classification or in interpolation and extrapolation reasoning.

2 Analogical proportion and related proportions

In this section, we present the basic Boolean definitions for analogical proportion, introducing the two groups of four related proportions and highlighting their strong link. Finally, we briefly investigate their multiple-valued extensions, before briefly mentioning the existence of other logical proportions.

2.1 Basic definitions

Similarity and dissimilarity indicators. Generally speaking, the comparison of two items A and B relies on their representation. We adopt here a logical setting. Let φ be a property, which can be seen as a predicate: $\varphi(A)$ may be true (in that case $\neg\varphi(A)$ is false), or false. When comparing two items A and B w.r.t. φ , it makes sense to consider A and B *similar* (w.r.t. φ):

- when $\varphi(A) \wedge \varphi(B)$ is true or
- when $\neg\varphi(A) \wedge \neg\varphi(B)$ is true.

In the remaining cases:

- when $\neg\varphi(A) \wedge \varphi(B)$ is true or
- when $\varphi(A) \wedge \neg\varphi(B)$ is true,

we can consider A and B as *dissimilar* w.r.t. property φ . $\varphi(A)$ and $\varphi(B)$ being ground formulas, they can be considered as Boolean variables (thus taking values in $\{0, 1\}$). They are denoted a and b by abstracting w.r.t. φ . If the conjunction $a \wedge b$ is true, the property is satisfied by both items A and B , while the property is satisfied by neither A nor B if $\bar{a} \wedge \bar{b}$ ¹ is true. Moving to the Boolean setting, $a \wedge b$ and $\bar{a} \wedge \bar{b}$ are *indicators of similarity* while $a \wedge \bar{b}$ and $\bar{a} \wedge b$ are *indicators of dissimilarity*. These indicators are the basis [47] for providing a formal definition of analogical proportion and more generally of a logical proportion.

Logical proportions. They are defined as follows:

Definition. A logical proportion $T(a, b, c, d)$ is the conjunction of 2 distinct equivalences between indicators of the form

$$I_{(a,b)} \equiv I_{(c,d)} \wedge I'_{(a,b)} \equiv I'_{(c,d)}$$

where $I_{(x,y)}$ and $I'_{(x,y)}$ denote indicators applied to the pair (x, y) . They are not necessarily the same, and $I_{(x',y')}$ and $I_{(x,y)}$ may be different indicators applied to different pairs. Thus logical proportions are quaternary Boolean operators. Then it can be shown that [56]

- there exist 120 distinct logical proportions
- they are true for 6 lines in their truth table and false for the 10 remaining lines.
- it means that logical proportions are quite rare. Indeed we know that we have $[^{16}_6] = 8008$ truth tables with exactly 6 valuations leading to true.

A typology of different classes of logical proportions has been laid bare in [49]. There are 5 classes made of:

- 4 homogeneous proportions, where the 4 terms related by equivalence symbols are all distinct and for which the same type of indicator (dissimilarity or similarity) is used

¹ The overline denotes Boolean negation.

in the two equivalences. These are the three analogy-related proportions, together with the inverse paralogy defined by $((a \wedge b) \equiv (\bar{c} \wedge \bar{d})) \wedge ((\bar{a} \wedge \bar{b}) \equiv (c \wedge d))$. Inverse paralogy expresses that “what a and b have in common, c and d miss it, and conversely”. The 6 lines of its truth table that makes it true are the 6 patterns that makes true two of the three analogy-related proportions: it thus excludes $(1, 1, 1, 1)$ and $(0, 0, 0, 0)$.

- 16 conditional proportions, where the 4 terms related by equivalence symbols are still all distinct. Their expression is made of the conjunction of an equivalence between similarity indicators and of an equivalence between dissimilarity indicators. They correspond to semantical equivalences between the 2 default rules (viewed as conditional objects) “if x then y ” and “if x' then y' ” by stating that they have the same examples, i.e. $(x \wedge y) \equiv (x' \wedge y')$ and the same counter-examples $(x \wedge \bar{y}) \equiv (x' \wedge \bar{y}')$. This play an important role at the core of nonmonotonic reasoning [13].

- 20 hybrid proportions (the 4 terms related by equivalence symbols are still all distinct). They are characterized by equivalences between similarity and dissimilarity indicators in their definitions. Noticing that negating anyone of the two terms of a dissimilarity indicator turns it into a similarity indicator, and conversely, we understand that changing a into \bar{a} (and \bar{a} into a), or applying a similar transformation with respect to b , c , or d , turns an hybrid proportion into an homogeneous or a conditional proportion, and an homogeneous or a conditional proportion into an hybrid proportion.

- 32 semi-hybrid proportions (the 4 terms related by equivalence symbols are still all distinct). One half of their expressions involve indicators of the same kind, while the other half requires equivalence between indicators of opposite kinds. Applying a change from a to \bar{a} (and \bar{a} to a), or applying a similar transformation with respect to b , c , or d , turns a semi-hybrid proportion into a semi-hybrid proportion.

- 48 degenerated proportions. In all the previous categories, the 4 terms related by equivalence symbols should be all distinct. In degenerated proportions, there are only 3 different terms. It can be checked that degenerated proportions correspond to a mutual exclusiveness condition between component(s) or negation of component(s) of one of the pairs (a, b) or (c, d) , together with either an identity condition pertaining to the other pair, or a tautology condition on one of the literals of the other pair without any constraint on the other literal.

Logical proportions that satisfy code independency. Situations can be described positively or negatively in terms of properties, i.e. one may say that a property is satisfied or that its negation is not satisfied. This leads to identify the logical proportions that satisfy the so-called *code independency* property: $T(a, b, c, d) \implies T(\bar{a}, \bar{b}, \bar{c}, \bar{d})$ insuring that the proportion T holds when 0 and 1 are exchanged. Only 8 among the 120 proportions satisfy *code independency* [56] and, in that perspective, these 8 proportions stand out from the crowd. They can be divided in two groups: the 4 so-called “homogeneous” proportions denoted A (for *analogy*), R (for *reverse analogy*) P (for *paralogy*), I (for *inverse paralogy*), and the 4 so-called “heterogeneous” proportions denoted H_a, H_b, H_c, H_d (for a reason that will be made clear in the following): they are shown respectively in Tables 1 and 2.

| Homogeneous proportions | |
|--|--|
| A | R |
| $(a \wedge \bar{b} \equiv c \wedge \bar{d}) \wedge (\bar{a} \wedge b \equiv \bar{c} \wedge d)$ | $(a \wedge \bar{b} \equiv \bar{c} \wedge d) \wedge (\bar{a} \wedge b \equiv c \wedge \bar{d})$ |
| P | I |
| $(a \wedge b \equiv c \wedge d) \wedge (\bar{a} \wedge \bar{b} \equiv \bar{c} \wedge \bar{d})$ | $(a \wedge b \equiv \bar{c} \wedge \bar{d}) \wedge (\bar{a} \wedge \bar{b} \equiv c \wedge d)$ |

Table 1. 4 homogeneous proportions

| Heterogeneous proportions | |
|--|--|
| H _a | H _b |
| $(\bar{a} \wedge b \equiv \bar{c} \wedge \bar{d}) \wedge (a \wedge \bar{b} \equiv c \wedge d)$ | $(a \wedge \bar{b} \equiv \bar{c} \wedge \bar{d}) \wedge (\bar{a} \wedge b \equiv c \wedge d)$ |
| H _c | H _d |
| $(\bar{a} \wedge \bar{b} \equiv \bar{c} \wedge d) \wedge (a \wedge b \equiv c \wedge \bar{d})$ | $(\bar{a} \wedge \bar{b} \equiv c \wedge \bar{d}) \wedge (a \wedge b \equiv \bar{c} \wedge d)$ |

Table 2. 4 heterogeneous proportions

2.2 Truth tables of A, R, P, I

Let us first focus on the subset of homogeneous proportions, which contains the analogical proportion. When needed, the analogical proportion $a : b :: c : d$ will be denoted $A(a, b, c, d)$. As can be seen from its formal definition, analogical proportion focuses on *differences* and should hold when the differences between a and b and between c and d are the same.

Considered as Boolean formulas, logical proportions can be seen via their truth tables. Table 3 exhibits the truth tables of A, R, P, I , where only the 6 lines leading to the truth value 1 are shown. It is interesting to take a closer look at the truth tables of

| A | R | P | I |
|---------|---------|---------|---------|
| 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 1 1 0 0 |
| 1 1 1 1 | 1 1 1 1 | 1 1 1 1 | 0 0 1 1 |
| 0 0 1 1 | 0 0 1 1 | 1 0 0 1 | 1 0 0 1 |
| 1 1 0 0 | 1 1 0 0 | 0 1 1 0 | 0 1 1 0 |
| 0 1 0 1 | 0 1 1 0 | 0 1 0 1 | 0 1 0 1 |
| 1 0 1 0 | 1 0 0 1 | 1 0 1 0 | 1 0 1 0 |

Table 3. Analogy, Reverse analogy, Paralogy, Inverse Paralogy truth tables

the four homogeneous proportions. First, one can observe in Table 3, that 8 possible valuations for (a, b, c, d) never appear among the patterns that make A, R, P , or I true: these 8 valuations are of the form $xxxy, xxyx, xyxx$, or $yxxx$ with $x \neq y$ and $(x, y) \in \{0, 1\}^2$. As can be seen, it corresponds to situations where $a = b$ and $c \neq d$, or $a \neq b$ and $c = d$, i.e., similarity holds between the components of one of the pairs, and dissimilarity holds in the other pair. Such patterns are closely related to heterogeneous proportions. Moreover, the truth table of each of the four homogeneous proportions, is built in the same manner:

1. 2 lines of the table correspond to the characteristic pattern of the proportion; namely the two lines where one of the two equivalences in its definition holds true under the form $1 \equiv 1$ (rather than $0 \equiv 0$). Thus,
 - A is characterized by the pattern $xyxy$ (corresponding to valuations 1010 and 0101), i.e. we have the same difference between a and b as between c and d ;
 - R is characterized by the pattern $xyyx$ (corresponding to valuations 1001 and 0110), i.e., the differences between a and b and between c and d are in opposite directions;
 - P is characterized by the pattern $xxxx$ (corresponding to valuations 1111 and 0000), i.e., what a and b have in common, c and d have it also;
 - I is characterized by the pattern $xxyy$ (corresponding to valuations 1100 and 0011), i.e. what a and b have in common, c and d do not have it, and conversely.
2. the 4 other lines of the truth table of an homogeneous proportion T are generated by the characteristic patterns of the two other proportions that are not opposed to T (in the sense that A and R are opposed, as well as P and I). For these four lines, the proportion holds true since its expression reduces to $(0 \equiv 0) \wedge (0 \equiv 0)$.

Thus, the six lines of the truth table of A that make it true are induced by the characteristic patterns of A , P , and I^2 , the six valuations that makes P true are induced by the characteristic patterns of P , A , and R , and so on for R and I .

Set theoretic view of A . The description of situations generally requires several properties, thus encoded by a set of Boolean variables. Let us consider 4 situations described by means of a finite subset of binary features. Let \underline{a} , \underline{b} , \underline{c} , \underline{d} denote the respective subsets of properties satisfied by these 4 situations. We say that an analogical proportion holds between \underline{a} , \underline{b} , \underline{c} , \underline{d} if and only if this proportion holds on each feature (i.e., for each Boolean variable associated to a property). Then it can be shown [40] that there exist 6 subsets (possibly empty), \underline{x} , \underline{y} , \underline{z} , \underline{t} , \underline{u} , \underline{v} of features that are disjoint and cover the set of features, and such that the following decomposition holds

$$\begin{aligned}\underline{a} &= \underline{x} \cup \underline{z} \cup \underline{u} \\ \underline{b} &= \underline{y} \cup \underline{z} \cup \underline{u} \\ \underline{c} &= \underline{x} \cup \underline{t} \cup \underline{u} \\ \underline{d} &= \underline{y} \cup \underline{t} \cup \underline{u}.\end{aligned}$$

As can be seen \underline{x} (resp. \underline{y}) is the set of properties that situation \underline{a} (resp. \underline{b}) has and situation \underline{b} (resp. \underline{a}) has not; \underline{z} (resp. \underline{t}) is the set of properties that situations \underline{a} and \underline{b} (resp. \underline{c} and \underline{d}) have in common and that \underline{c} and \underline{d} (resp. \underline{a} and \underline{b}) have not; \underline{u} (resp. \underline{v}) is the set of properties that the 4 situations have all in common (resp. have not). This set decomposition-based view of analogy was first proposed in [68], before the introduction of the logical modeling. The 6 subsets \underline{x} , \underline{y} , \underline{z} , \underline{t} , \underline{u} , \underline{v} clearly correspond to the 6 valuation patterns that make A true (see Table 3).

² The measure of analogical dissimilarity introduced in [37] is 0 for the valuations corresponding to the characteristic patterns of A , P , and I , maximal for the valuations corresponding to the characteristic patterns of R , and takes the same intermediary value for the 8 valuations characterized by one of the patterns $xxxy$, $xxyx$, $xyxx$, or $yxxx$.

It is interesting to notice that the pairs (a, b) and (c, d) are then *similar* in the following sense:

- the situations in the two pairs differ in the same way (as represented by x and y);
- the two elements of each of the pairs (a, b) and (c, d) are similar positively or negatively on the same subset of features (as represented by z and t);
- the pairs (a, b) and (c, d) share the same common context (represented by u and v).

This similarity relation over the pairs of items is in fact an equivalence relation. This agrees with the properties of reflexivity, symmetry and transitivity of Table 4 in the following.

Properties of interest for logical proportions. The following interrelations between A, R, P, I are easy to check on the truth tables and establish a strong link between them:

- $R(a, b, c, d)$ iff $A(a, b, d, c)$
- $P(a, b, c, d)$ iff $A(a, d, c, b)$
- $I(a, b, c, d)$ iff $A(a, \bar{d}, \bar{c}, b)$

Besides, starting from the definition of A , it can be easily shown:

- $A(a, b, a, b)$ and $A(a, a, b, b)$, but *not* $A(a, b, b, a)$;
- $A(a, b, c, d) \Rightarrow A(c, d, a, b)$ (symmetry);
- $A(a, b, c, d) \Rightarrow A(a, c, b, d)$ (central permutation).

which corresponds to the usual postulates of analogical proportion. The corresponding postulates for R, P and I can be easily deduced through the interrelations. Table 4 gathers these properties with some others. Moreover this Table emphasizes the parallel between logical proportions and numerical (geometric) proportions of the form $\frac{a}{b} = \frac{c}{d}$ with respect to these properties. It is worth noticing that the analogical proportion A then appears as being the exact symbolic counterpart to the numerical proportions. A even enjoys other properties mixing permutations and negation, namely $A(a, b, c, d) = A(\bar{c}, b, \bar{a}, d) = A(\bar{b}, \bar{a}, c, d)$, which have counterparts with geometric proportions, taking the multiplicative inverse in place of the negation. The same will hold with arithmetic proportions of the form $a - b = c - d$ (using opposite in place of the negation).

2.3 Equation solving

Given a proportion T and 3 items a, b, c , the problem of finding a fourth item x such that $T(a, b, c, x)$ holds is known as the equation-solving problem. The equations $A(a, b, c, x)$, $R(a, b, c, x)$, $P(a, b, c, x)$ and $I(a, b, c, x)$ have not always a solution $x \in \{0, 1\}$. For analogical proportions, the existence condition for a solution is $(a \equiv b) \vee (a \equiv c) = 1$, which just states that $A(1, 0, 0, x)$ and $A(0, 1, 1, x)$ have no solution. When a solution exists, it is unique and given by $x = a \equiv (b \equiv c)$ for the three proportions A, R, P [40, 47], as first guessed from anthropological observations by Klein [27] (without distinguishing the three proportions). Obviously, this equation-solving problem is at the

| name | Boolean property | which ones | numerical counterpart |
|---------------------|--|------------|--|
| full identity | $T(a, a, a, a)$ | A, R, P | $\frac{a}{a} = \frac{a}{a}$ |
| reflexivity | $T(a, b, a, b)$ | A, P | $\frac{a}{b} = \frac{a}{b}$ |
| sameness | $T(a, a, b, b)$ | A, R | $\frac{a}{a} = \frac{b}{b}$ |
| symmetry | $T(a, b, c, d) \rightarrow T(c, d, a, b)$ | A, R, P, I | $\frac{a}{b} = \frac{c}{d} \rightarrow \frac{c}{d} = \frac{a}{b}$ |
| central permutation | $T(a, b, c, d) \rightarrow T(a, c, b, d)$ | A, I | $\frac{a}{b} = \frac{c}{d} \rightarrow \frac{a}{c} = \frac{b}{d}$ |
| extreme permutation | $T(a, b, c, d) \rightarrow T(d, b, c, a)$ | A, I | $\frac{a}{b} = \frac{c}{d} \rightarrow \frac{d}{b} = \frac{c}{a}$ |
| transitivity | $T(a, b, c, d) \wedge T(c, d, e, f) \rightarrow T(a, b, e, f)$ | A, P | $\frac{a}{b} = \frac{c}{d} \wedge \frac{c}{d} = \frac{e}{f} \rightarrow \frac{a}{b} = \frac{e}{f}$ |

Table 4. Properties of A, R, P, I

core of the inference process associated to analogical proportion. When we know that a proportion $A(a, b, c, x)$ holds (under the above conditions), we can infer the value of x from the known values of a, b, c .

As a direct illustration of the equation solving process, let us consider an analogical puzzle, as the ones considered early by Evans [14] where a series of 3 first items a, b, c is given and the 4th item d has to be chosen among several plausible options. In [15] or [60], an optimization mechanism looks for the candidate solution that maximizes the similarity of the set of rules describing the change from a to b , with the set of rules describing the change from c to each candidate solution. Here, the application of the equation solving process for each feature contrasts with such an approach, since the solution is computed directly. This is illustrated on Figure 1, and the details of the equations for the different features are given below. When the items are pictures, our method may also apply with an encoding of the image at the pixel level; see [53] for a discussion.



Fig. 1. IQ test: Graphical analogy

square: $(1, 1, 0, x_1)$ holds $\Rightarrow x_1 = 0$, i.e. no square;

triangle: $(0, 0, 1, x_2)$ holds $\Rightarrow x_2 = 1$, i.e. triangle;

star: $(1, 0, 1, x_3)$ holds $\Rightarrow x_3 = 0$, i.e. no star;

circle: $(0, 1, 0, x_4)$ holds $\Rightarrow x_4 = 1$, i.e. circle;

black point: $(1, 1, 1, x_5)$ holds $\Rightarrow x_5 = 1$, i.e. black point;

hexagon: $(0, 0, 0, x_6)$ holds $\Rightarrow x_6 = 0$, i.e. no hexagon.

There are many examples of analogical proportions that may look slightly different at first glance. Take for instance France: Paris :: Italy: Roma. Note that

the proportion cannot just be captured with unary features such that “is a capital”, and “is a country” (since then France : Roma :: Italy : Paris would hold as well). In such a case, there is the (obvious) *functional* relation “is the capital of” between a and b , which also holds between c and d , namely $\text{Paris} = \text{capital(France)}$ and $\text{Roma} = \text{capital(Italy)}$. This is an instance of the more general analogical proportion $x : f(x) :: y : f(y)$. Indeed, it has a purely (meta) Boolean interpretation, shown in Figure 2, which exhibits the fact that we get $f(y)$ as a solution of the analogical equation just by solving the equation column per column: it yields ???=011, which encodes $f(y)$. This shows the full agreement of the logical view with this general analogical proportion, leading to a faithful encoding. However, the checking that the new

| | x | y | f |
|------|---|---|---|
| x | 1 | 0 | 0 |
| f(x) | 1 | 0 | 1 |
| y | 0 | 1 | 0 |
| ? | ? | ? | ? |

Fig. 2. A Boolean representation of $x : f(x) :: y : ?$

proportion obtained by central permutation France : Italy :: Paris : Roma is still valid, just amounts to acknowledge that $x : y :: f(x) : f(y)$ holds as well. Lastly, observe that the unicity of the solving of the equation $x : f(x) :: y : ?$ relies on the functional nature of f .

2.4 Multiple-valued logic extension for A, R, P, I

If we consider the Boolean expression of the analogical proportion, one may think of many possible multiple-valued extensions, depending of the operations chosen for modeling \wedge , \equiv (involving \rightarrow), and \neg . Moreover, a formula such as the one defining analogy in Table 1 can be written in many equivalent forms in Boolean logic. These forms are no longer necessarily equivalent in a non-Boolean setting where $[0, 1]$ is now the truth space. The choice for $\neg a$ interpretation is quite standard as $1 - a$, but it is important to make proper choices for the remaining connectors that are in agreement with the intended meaning of the considered proportion. Some properties seem very natural to preserve, such as

- i) the independence with respect to the positive or negative encoding of properties (one may describe a price as the extent to which it is cheap, as well as it is not cheap), which leads to require that $A(\neg a, \neg b, \neg c, \neg d)$ holds if $A(a, b, c, d)$ holds;
- ii) the knowledge of a and of the differences between a and b and between b and a , should enable us to recover b . Indeed in the Boolean case, we have $b = (a \wedge (a \rightarrow b)) \vee \neg(b \rightarrow a)$. A careful analysis [50] of the requirements leads to choose
 - i) the minimum operator for \wedge ; ii) $s \equiv t = 1 - |s - t|$;
 - iii) Lukasiewicz implication $s \rightarrow t = \min(1, 1 - s + t)$.

Note also that with these choices $s \equiv t = (s \rightarrow t) \wedge (t \rightarrow s)$.

This leads to the following expressions which both generalize the Boolean case to multiple-valued entries and introduce a graded view of the analogy-related proportions.

For analogy, we have $A(a, b, c, d) =$
 $1 - |(a - b) - (c - d)|$ if $a \geq b$ and $c \geq d$, or $a \leq b$ and $c \leq d$
 $1 - \max(|a - b|, |c - d|)$ if $a \leq b$ and $c \geq d$ or $a \geq b$ and $c \leq d$

Thus, $A(a, b, c, d)$ is all the closer to 1 as the differences $(a - b)$ and $(c - d)$ have the same sign and have similar absolute values. Note that $A(1, 0, c, d) = 0$ as soon as $c \leq d$.

For reverse analogy, we have $R(a, b, c, d) =$
 $1 - |(a - b) - (d - c)|$ if $a \leq b$ and $c \geq d$ or $a \geq b$ and $c \leq d$
 $1 - \max(|a - b|, |c - d|)$ if $a \geq b$ and $c \geq d$, or $a \leq b$ and $c \leq d$

The definition of paralogy is a little bit simpler:

$P(a, b, c, d) = \min(1 - |(a \wedge b) - (c \wedge d)|, 1 - |(a \vee b) - (c \vee d)|)$, with $a \vee b = 1 - (1 - a) \wedge (1 - b)$. Again we take $a \wedge b = \min(a, b)$; see [50] for justifications. The definition for I is deducible from the definition of P and the link $I(a, b, c, d) \equiv P(a, b, \bar{c}, \bar{d})$.

With respect to equation solving, it can be shown that it exists x such that $A(a, b, c, x) = 1$ if and only if $x = c + b - a \in [0, 1]$, and when it exists, the solution is unique. Similar equations may be solved as well for the three other proportions.

There exists another slightly different definition [55] for graded analogies which is smoother in the sense that more patterns receive intermediary truth-values; for instance, the pattern $0 : 0.5 :: 0.5 : 1$ is true at degree 0.5 with this alternative view. But this proportion is true at degree 1 with the above definition (since the amount of change from a to b is the same as from c to d , although the changes do not take place at the same position in the scale). Moreover, extensions of analogical proportions for handling non-applicable properties, or feature having unknown values have been proposed. See [55] on all these issues.

3 Heterogeneous proportions

As mentioned in the introduction, apart from the homogeneous proportions, there are 4 other outstanding proportions, namely the heterogeneous proportions, which are defined in section 2.1 and whose truth tables are given in Table 5.

| H_a | H_b | H_c | H_d |
|---------|---------|---------|---------|
| 1 1 1 0 | 1 1 1 0 | 1 1 1 0 | 1 1 0 1 |
| 0 0 0 1 | 0 0 0 1 | 0 0 0 1 | 0 0 1 0 |
| 1 1 0 1 | 1 1 0 1 | 1 0 1 1 | 1 0 1 1 |
| 0 0 1 0 | 0 0 1 0 | 0 1 0 0 | 0 1 0 0 |
| 1 0 1 1 | 0 1 1 1 | 0 1 1 1 | 0 1 1 1 |
| 0 1 0 0 | 1 0 0 0 | 1 0 0 0 | 1 0 0 0 |

Table 5. H_a, H_b, H_c, H_d Boolean truth tables

Let us note that these truth tables exactly involve the 8 missing tuples of the homogeneous tables, i.e. those ones having an odd number of 0 and 1. It is remarkable that they satisfy the same association properties as the homogeneous ones: Indeed we observe on their truth tables that any combination of 2 heterogeneous proportions is satisfiable by 4 4-tuples, any combination of 3 heterogeneous proportions is satisfiable by only 2 4-tuples, and that the conjunction $H_a(a, b, c, d) \wedge H_b(a, b, c, d) \wedge H_c(a, b, c, d) \wedge H_d(a, b, c, d)$ is not satisfiable. This fact contributes to make the heterogeneous proportions the perfect dual of the homogeneous ones.

3.1 Properties

The formal definitions given in Tables 1 and 2 lead to immediate equivalences between heterogeneous and homogeneous proportions that we summarize in Table 6. Obviously,

| H_a | H_b |
|--|--|
| $H_a(a, b, c, d) \equiv I(\bar{a}, b, c, d)$ | $H_b(a, b, c, d) \equiv I(a, \bar{b}, c, d)$ |
| $H_a(a, b, c, d) \equiv P(\bar{a}, b, \bar{c}, \bar{d})$ | $H_b(a, b, c, d) \equiv P(a, \bar{b}, \bar{c}, \bar{d})$ |
| $H_a(a, b, c, d) \equiv P(a, \bar{b}, c, d)$ | $H_b(a, b, c, d) \equiv P(\bar{a}, b, c, d)$ |
| H_c | H_d |
| $H_c(a, b, c, d) \equiv I(a, b, \bar{c}, d)$ | $H_d(a, b, c, d) \equiv I(a, b, c, \bar{d})$ |
| $H_c(a, b, c, d) \equiv P(\bar{a}, \bar{b}, \bar{c}, d)$ | $H_d(a, b, c, d) \equiv P(\bar{a}, \bar{b}, c, \bar{d})$ |
| $H_c(a, b, c, d) \equiv P(a, b, c, \bar{d})$ | $H_d(a, b, c, d) \equiv P(a, b, \bar{c}, d)$ |

Table 6. Equivalences between heterogeneous and homogeneous proportions

the heterogeneous proportions are strongly linked together: for instance, using the symmetry of I ,

$$H_a(a, b, c, d) \equiv I(\bar{a}, b, c, d) \equiv I(c, d, \bar{a}, b) \equiv H_c(c, d, a, b).$$

In the following we discuss two different ways of viewing these proportions:

- *a semantic viewpoint*: the *full identity* postulate $T(a, a, a, a)$ asserts that proportion T holds between identical values. Negating only one variable position generates an *intruder*, as in $T(\bar{a}, a, a, a)$, $T(a, \bar{a}, a, a)$, $T(a, a, \bar{a}, a)$ and $T(a, a, a, \bar{a})$, and leads to new postulates respectively denoted Int_a , Int_b , Int_c and Int_d . For a proportion, satisfying the property Int_a means that *the first variable may be an intruder*. Since each postulate Int_a , Int_b , Int_c and Int_d is validated by 2 distinct 4-tuples, it is clear that 3 of them are enough to define a logical proportion having exactly 6 valid 4-tuples. There is no proportion satisfying all these postulates since it would lead to 8 valid 4-tuples, which excludes any logical proportion. It can be easily checked that H_a satisfies Int_b , Int_c , Int_d and does not satisfy Int_a : We can interpret $H_a(a, b, c, d)$ as the following assertion *the first position is not an intruder and there is an intruder among the remaining positions*. In other words, having H_a true means “there is an intruder which is not a ”. As a consequence, $H_a(a, b, c, d)$ does not hold when there is no intruder (i.e. an even number of

0) or when a is the intruder. The same reasoning applies to H_b, H_c, H_d .

- a *syntactic viewpoint*: here we start from the inverse paralogy I definition: $(ab \equiv \bar{c}\bar{d}) \wedge (\bar{a}\bar{b} \equiv cd)$. To get the definition of an heterogeneous proportion satisfying postulates where the intruder is in position 1, 2 or 4 for instance, we add a negation on the 3rd variable in both equivalences defining I . Here we get H_c as:

$$(ab \equiv \bar{c}\bar{d}) \wedge (\bar{a}\bar{b} \equiv \bar{c}d)$$

This process, allowing us to generate the 4 heterogeneous proportions, shows that, in some sense, they are “atomic perturbations” of I : for this reason and since they are heterogeneous proportions, they have been respectively denoted H_a, H_b, H_c and H_d where the subscript corresponds to:

- the postulate which is not satisfied by the corresponding proportion or, equivalently,
- the negated variable in the equivalence with I .

For instance, $H_a(a, b, c, d) \equiv I(\bar{a}, b, c, d)$, H_a satisfies $\text{Int}_b, \text{Int}_c, \text{Int}_d$ and does not satisfy Int_a . So, when $H_a(a, b, c, d) = 1$, there is an intruder, and \bar{a} is the value of the intruder, since a is not the intruder. The different possible cases where $H_a(a, b, c, d) = 1$ are as follows:

- $(\bar{a}, b, c, d) = (1, 1, 0, 0)$ or $(0, 0, 1, 1)$ and the intruder is b .
- or $(\bar{a}, b, c, d) = (0, 1, 0, 1), (1, 0, 1, 0), (0, 1, 1, 0)$, or $(1, 0, 0, 1)$ and the intruder is c in the first two cases, and d in the two others.

In other words, there is an intruder in (a, b, c, d) , which is not a , iff the properties common to \bar{a} and b (positively or negatively) are not those common to c and d , and conversely.

From a practical viewpoint, these proportions are closely related with the idea of spotting the odd one out (the intruder), or if we prefer of picking the one that doesn’t fit among 4 items. This will be further discussed in Section 7, but we first consider the extension of heterogeneous proportions to the case of graded properties with intermediate truth values.

3.2 Multiple-valued semantics

When we have to handle properties whose satisfaction is a matter of levels, an extension of the Boolean interpretation to multiple-valued models, where the truth values belong to $[0, 1]$, is necessary. This has been done for homogeneous proportions in [50] and we do it here for heterogeneous proportions. Roughly speaking, in the case of H_a , the graded truth value of $H_a(a, b, c, d)$ estimates how far we are from having a as an intruder.

Two questions arise:

- 1) what are the 4-tuples that correspond to a “perfect” proportion of a given type (i.e. having 1 as truth degree)? For instance, we want the value of $H_a(0, u, 0, u)$ to be equal to 1 (as well as the truth value of $H_c(0, u, 0, u)$) because in that context, it is true that $a = 0$ (resp. c) cannot be the intruder, whatever the value of u .

2) are there 4-tuples that could be regarded as “approximate” proportions of a given type (with an intermediate truth degree) and in that case, what is their truth value? For instance, in $(0.7, 1.0, 1.0, 0.9)$, it is likely that a is the intruder just because the other candidate, d , has a value very close to 1, and the closer d is to 1, the more likely a is the intruder: then the truth value of $H_a(0.7, 1.0, 1.0, 0.9)$ should be small and related to $1 - d = 0.1$ (since H_a excludes a as intruder).

A rigorous way to proceed is to start from the definition of [50] for multiple-valued paralogy P , leading to 15 4-tuples fully true, and 18 fully false. The 48 remaining patterns get intermediate truth value given by the general formula (1)

$$P(a, b, c, d) = \min(1 - |\max(a, b) - \max(c, d)|, \\ 1 - |\min(a, b) - \min(c, d)|) \quad (1)$$

which, thanks to the symmetry of P and stability w.r.t. the permutation of its two first variables, has the following behavior

$$\left| \begin{array}{ll} \text{general case} & \text{case } u = v \\ P(1, 1, u, v) = \min(u, v) & P(1, 1, u, u) = u \\ P(1, 0, u, v) = \min(\max(u, v), 1 - \min(u, v)) & P(1, 0, u, u) = \min(u, 1 - u) \\ P(0, 0, u, v) = 1 - \max(u, v) & P(0, 0, u, u) = 1 - u \end{array} \right.$$

As a consequence of the equivalences given in Table 6, we get the multiple-valued definition for H_a (we get similar definitions for H_b, H_c, H_d), still leading to 15 true patterns, 18 false patterns and 48 with intermediate values:

$$H_a(a, b, c, d) = \min(1 - |\max(a, 1 - b) - \max(c, d)|, \\ 1 - |\min(a, 1 - b) - \min(c, d)|)$$

Let us note that $H_a(0, 0, u, v) = H_a(1, 1, u, v)$ due to the equivalences $H_a(0, 0, u, v) = P(0, 1, u, v) = P(1, 0, u, v)$. This leads to the following cases:

$$\left| \begin{array}{ll} \text{general case} & \text{case } u = v \\ H_a(1, 1, u, v) = \min(\max(u, v), 1 - \min(u, v)) & H_a(1, 1, u, u) = \min(u, 1 - u) \\ H_a(1, 0, u, v) = \min(u, v) & H_a(1, 0, u, u) = u \\ H_a(0, 1, u, v) = 1 - \max(u, v) & H_a(1, 0, u, u) = 1 - u \end{array} \right.$$

Let us analyze 2 examples to highlight the fact that our definition really fits with the intuition.

- Considering the general pattern $(1, 0, 0, u)$, its truth value is:
 - u for P : if u is close to 1, we are close to the fully true paralogical pattern $(1, 0, 0, 1)$ and the truth value is high. In the opposite case, u is close to 0 and we are close to a fully false paralogical pattern $(1, 0, 0, 0)$.
 - $1-u$ for H_b, H_c, H_d : if u is close to 1, we are close to the pattern $(1, 0, 0, 1)$ which is definitely not a valid pattern for H_b, H_c, H_d : so $1-u$ is a low truth value. But if u

is close to 0, we are close to the pattern $(1, 0, 0, 0)$ which is valid for H_b, H_c, H_d and 1-u is a high truth value.

- finally 0 for H_a : whatever the value of u, $(1, 0, 0, u)$ means “an intruder is in first position” where the semantics of H_a is just the opposite.
- Back to the graded pattern $(0.7, 1, 1, 0.9)$ introduced above
 - regarding P , the truth value as given by the formula is 0.8, i.e. the pattern is close to be a true paralogy.
 - regarding the heterogeneous proportions, we understand that we have 2 candidate intruders namely $a = 0.7$ and $d = 0.9$. But they are not equivalent in terms of intrusion and it is more likely to be a than d . This is consistent with the fact that the truth value of $H_a(0.7, 1, 1, 0.9)$ is 0.1 (very low), but the truth value of $H_d(0.7, 1, 1, 0.9)$ is 0.3 (a bit higher).
 - in fact, $(0.7, 1, 1, 0.9)$ does not give a strong belief that there is an intruder, and it is not a surprise that $H_b(0.7, 1, 1, 0.9) = H_c(0.7, 1, 1, 0.9) = 0.4$.

4 Reasoning with proportions

Due to their dissimilarity / similarity-based semantics, logical proportions, and specially analogy-related ones, seem to have a great potential in reasoning about particular situations. Inferring the value of d starting from the values of a, b, c and the fact that some proportion holds in the 4-tuple (a, b, c, d) is an equation solving problem: find d such that the considered proportion holds knowing the truth values of a, b, c . Such an equation may have no solution, or may have one or two solutions. Regarding the unicity of the solution when it exists, the solution will be always unique for proportions such that each of the 6 lines of their truth table starts with a different triple of values for a, b, c . There are 64 proportions that have this property, and there are 56 proportions for which the 4-tuple (a, b, c, x) may have 2 solutions for some entries a, b, c . Besides, since any logical proportion relating (a, b, c, d) is true for only 6 patterns of values, and (a, b, c) may take $2^3 = 8$ different triples of values, there are at least 2 entries a, b, c leading to no solution. Thus, as already said for analogy, the two equations $A(1, 0, 0, x)$ and $A(0, 1, 1, x)$ have no solution.

Since logical proportions are Boolean formulas, it is natural to describe what can be inferred from a given situation with a set of valid inferences, involving these proportions in the premises of the inference schemes. Let us start from a simple example to understand our point. Suppose we observe $\neg a, b$ and $\neg c$ and we get a new d knowing only that d is in *analogical* proportion with the 3 previous values. We are faced to the problem of inferring the value of d . One may use the clausal form of this proportion [48], namely

$$\{\neg a \vee b \vee c, \neg a \vee b \vee \neg d, a \vee \neg c \vee d, \neg b \vee \neg c \vee d, \\ a \vee \neg b \vee \neg c, a \vee \neg b \vee d, \neg a \vee c \vee \neg d, b \vee c \vee \neg d\}^3$$

³ Similarly, the clausal form for paralogy is: $\{\neg a \vee c \vee d, \neg a \vee \neg b \vee d, a \vee b \vee \neg c, b \vee \neg c \vee \neg d, a \vee \neg c \vee \neg d, a \vee b \vee \neg d, \neg a \vee \neg b \vee c, \neg b \vee c \vee d\}$. More generally, analogy, paralogy, reverse analogy, inverse paralogy are each described by a set of 8 clauses which cannot be further reduced by resolution, and these 4 sets do not share any clause.

Each clause is falsified by a pattern of 3 literals for which there does not exist a 4th literal with which they form a proportion. Thus, the first clause $\neg a \vee b \vee c$ expresses syntactically that $a \ \neg b \ \neg c$ (i.e., 1 0 0 in semantical terms) cannot be analogically completed, while $a \vee \neg b \vee \neg c$ expresses the same w. r. t. $\neg a \ b \ c$ and 0 1 1. Since the unknown x may be any of the 4 literals of the proportion, this makes 2×4 clauses. Going back to our inference example, we have:

$$\frac{\neg a \ b \ \neg c \ a : b :: c : d}{d}$$

by resolution from the clausal form of $a : b :: c : d$ (6th clause). As expected, there are 6 valid inferences given in Table 7

| | | |
|---|---|---|
| $\frac{a \ b \ c \ a:b:c:d}{d}$ | $\frac{\neg a \ \neg b \ \neg c \ a:b:c:d}{\neg d}$ | $\frac{\neg a \ \neg b \ c \ a:b:c:d}{d}$ |
| $\frac{a \ \neg b \ c \ a:b:c:d}{\neg d}$ | $\frac{\neg a \ b \ \neg c \ a:b:c:d}{d}$ | $\frac{a \ b \ \neg c \ a:b:c:d}{\neg d}$ |

Table 7. Valid inferences with an analogical proportion

5 Analogical-proportion based reasoning

In the above section, we have assumed the knowledge that some particular logical proportion holds in the inference patterns we considered. Where may such knowledge come from? A natural answer is to consider that the proportion has been observed between four items for some features, and we assume on this basis that the same proportion holds for other features as well. This makes sense at least for proportions such as the 3 analogy-related proportions that express regularities in change as well as they leave room for the expression of similarity for other features.

More formally, let us consider items or situations described by vectors (a_1, \dots, a_n) of Boolean values that encode the different binary features that describe a situation. More generally, one may have multiple-valued features. Starting from a triple of items completely informed with respect to all features, we consider a new item $d = (d_1, \dots, d_n)$, which is only partially informed, i.e. where only some features $k(d) = (d_1, \dots, d_p)$, for $p < n$, are known, the values of the missing features $u(d) = (d_{p+1}, \dots, d_n)$ having to be predicted. For that purpose, we adopt the following transfer pattern (where $T \in \{A, R, P\}$ denotes any analogy-related proportion):

$$\frac{\forall i \in [1, p], \ T(a_i, b_i, c_i, d_i)}{\forall j \in [p + 1, n], \ T(a_j, b_j, c_j, d_j)}$$

It simply means that *if the known part $k(d)$ of d is componentwise in formal proportion T with $k(a), k(b)$ and $k(c)$ then it should be also true for the unknown part $u(d)$ of d for the same proportion T .* This form of reasoning is clearly not sound, but may be useful for trying to guess unknown values. Then, for all $j \in [p + 1, n]$, one may infer the truth value of d_j assuming that $T(a_j, b_j, c_j, d_j)$ holds, given a_j, b_j , and c_j .

Let us consider an *example* where $n = 5$ and $a = (1, 1, 0, 0, 1)$, $b = (1, 0, 1, 1, 0)$, $c = (0, 1, 0, 0, 1)$, with an incompletely known item $d = (0, 0, 1, d_4, d_5)$ (here $p = 3$, $k(d) = (0, 0, 1)$, $u(d) = (d_4, d_5)$). For the first three features, we can check that analogical proportion $k(a) : k(b) :: k(c) : k(d)$ holds. Then using the above inference scheme, we should have $0 : 1 :: 0 : d_4$ and $1 : 0 :: 1 : d_5$, which leads to $d_4 = 1$ and $d_5 = 0$. Although this type of reasoning basically amounts to copy existing similarity / dissimilarity relationships, it is very powerful since it may produce new compound patterns where the vector representing d is not similar in all respects to any of the vectors representing a , b , or c , (as in the example above).

The transfer pattern also encompasses the following basic analogical reasoning schema. We have 2 situations or cases at hand, x and y , that both share a property P , i.e. $P(x)$ and $P(y)$ hold, and x also satisfies another property Q , the schema amounts to conclude that $Q(y)$ also holds, by an “analogical jump”. Indeed the proportion $P(x) : Q(x) :: P(y) : Q(y)$ can be justified in our approach, viewing the presence of instances, properties or functions as features (see [51] for details). Similarly, one can also justify the extended analogical pattern $x : f(x) :: y : f(y)$. This pattern, combined with the general ones, can be successfully used for solving IQ tests, starting from the completion of a sequence of 3 geometric figures as in [53] to more complex tests like the Raven’s tests [10] that we summarize in the following section.

6 Solving IQ tests

IQ tests play a special role in the AI litterature as they can be, in some sense, considered as a kind of scale on which to measure the effectiveness of an AI theory or system (see [28] for instance where the authors target Bennett Mechanical Comprehension Test Problems). Among the most well-known IQ tests, we have the Raven Progressive Matrices (RPM) [61]. They are visual tests where a sequence of 8 pictures has to be completed in a logical way and the solution has to be chosen among a set of 8 candidate pictures. The resulting performance is considered as a measure of the reasoning ability of the participant. An example⁴ is given with its solution (a simple big square) in Figure 3. Solving an RPM heavily relies on the representation of the space and objects at

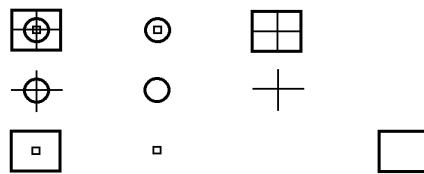


Fig. 3. Modified Raven test 12 and its solution

⁴ For copyright reasons and to protect the security of the tests, the original Raven test is replaced by specifically designed examples (still isomorphic in terms of logical encoding to the original ones).

hand. A Raven matrix pic is a 3×3 matrice where $pic[i, j]$ ($i, j \in \{1, 2, 3\} \times \{1, 2, 3\}$) denotes the picture at row i and column j and where $pic[3, 3]$ is unknown. Assuming that the Raven matrices can be understood in the following way, with respect to rows and columns:

$$\forall i \in [1, 2], \exists f \text{ such that } pic[i, 3] = f(pic[i, 1], pic[i, 2])$$

$$\forall j \in [1, 2], \exists g \text{ such that } pic[3, j] = g(pic[1, j], pic[2, j])$$

the two complete rows (resp. columns) are examples supposed to help to discover f (resp. g), and then to predict the missing picture $pic[3, 3]$ as $f(pic[3, 1], pic[3, 2])$ (resp. $g(pic[1, 3], pic[2, 3])$). This representation is summarized in Figure 4. In fact, the

| | | |
|-------------------------|-------------------------|-------------------------|
| $pic[1,1]$ | $pic[1,2]$ | $f(pic[1,1], pic[1,2])$ |
| $pic[2,1]$ | $pic[2,2]$ | $f(pic[2,1], pic[2,2])$ |
| $g(pic[1,1], pic[2,1])$ | $g(pic[1,2], pic[2,2])$ | |

Fig. 4. Raven matrix representation

problem can be restated as an analogical equation-solving problem. Using an extended scheme where proportion $(a, b) : f(a, b) :: (c, d) : f(c, d)$ holds for lines and proportion $(a, b) : g(a, b) :: (c, d) : g(c, d)$ for columns, which translates into:

$$(pic[1, 1], pic[1, 2]) : pic[1, 3] :: (pic[2, 1], pic[2, 2]) : pic[2, 3]$$

$$(pic[1, 1], pic[2, 1]) : pic[3, 1] :: (pic[1, 2], pic[2, 2]) : pic[3, 2]$$

Similar analogical proportions are supposed to relate line 1 to line 3 and line 2 to line 3, and similarly for the columns. We have two options to solve the problem:

- Each picture $pic[i, j]$ is represented as a Boolean vector of dimension n . The Boolean coding is done manually so far. In the example of Figure 4, we consider 4 binary features in the following order: big square, small square, circle, cross. For instance, the contents of $pic[1, 2]$ is encoded by the vector (0110) in Figure 5.

| | column1 | column2 | column3 |
|------|---------|---------|---------|
| row1 | 1111 | 0110 | 1001 |
| row2 | 0011 | 0010 | 0001 |
| row3 | 1100 | 0100 | ???? |

Fig. 5. Raven test 12 encoding

Considering the first missing feature, the horizontal pattern (1 0) to be completed in line 3 appears in line 1 and leads to the solution 1. For the second missing feature, there is no corresponding horizontal pattern for (1 1), so we have to move to a vertical analysis and to look for a vertical pattern starting with (0 0) for this second feature. We fail again. We have then to consider that a Raven matrix expresses a set of analogical proportions without any consideration of a particular feature. This is why, even if we fail to find a suitable pattern in row or column for this feature, we look for a similar valid pattern but related to another feature. In other terms, in order to find the solution for feature i, we allow us to take lesson from other features, then looking for a solution in a row or a column coming from another feature j. In the example, we thus find 0 for the second feature, using the context of feature 3 in column 1 or in column 2 ; in fact when proceeding this way, it is especially important to make sure that there is no contradictory patterns (here it would be (1 1 1)) observable in row or in column. Using the same approach for feature 3 (circle) we get 0 from observing feature 2 in row 2, or feature 4 in column 2. Feature 4 is easier, we get 0 from column 1 for the same feature. Altogether, we get (1 0 0 0) as the answer, which is indeed the encoding of the solution. This method is sufficient for solving 32 Raven tests over 36 [10].

- If we consider an image as a matrix of pixels of dimension $n \times m$ (considering only non compressed format as BMP for instance), this is simply a Boolean-like coding automatically performed by the picture processing device (e.g. the camera or the scanner). In that case, instead of dealing with 8 hand-coded Boolean vectors, we deal with BMP files for instance. Apart from the fact that we have to take care of the headers of the files (which do not obey any proportion pattern), there is no reason to change our method and our basic algorithm still applies allowing to solve 16 tests (over 36).

7 Finding the odd one out

With homogeneous proportions, we can solve sophisticated IQ tests where a missing item has to be found. Namely, a sequence of items being given and assumed to be incomplete, the problem is to find the item that completes the sequence in a “natural” way among a given set of candidate solutions, or better, to build it. As it is the case for homogeneous proportions, heterogeneous proportions can be also related to the solving of another type of quiz problem [57]. In the “Finding the odd one out” problem where a complete sequence of 4 or more items being given, we have to find the item that does not fit with the other ones and which is, in some sense, an intruder or an anomaly. On this basis, a complete battery of IQ tests has been recently developed in [21]. The problem of spotting anomalies can be found in other areas as well, e.g., [1]. Solving Odd One Out tests (which are visual tests) has been recently tackled in [34] by using analogical pairing between fractal representation of the pictures. It is worth noticing that the approach of these authors takes its root in the idea of analogical proportion. However, this method relies on the use of similarity/dissimilarity measures rather than referring to a formal logical view of analogical proportion. In the following, we show that heterogeneous proportions provide a convenient way to code and tackle this problem.

Let us first consider the case of 4 items. The idea of intruder in a set of Boolean values amounts to have a unique value distinct from all the others (which are thus identical). Note that when a homogeneous proportion holds between 4 Boolean values, there is no intruder among them. More generally when dealing with Boolean vectors (describing items), we may extend this view by requiring that for at least one component, there is an intruder value in the previous sense. In that case, the corresponding vector (and the associated item) is an *intruder w.r.t. that component*. Then the candidate intruder, if it exists, is the item whose vector is an intruder w.r.t. a maximum number of components. As a consequence, there is no intruder when there is no candidates, or when there are more than one. Besides, note that although in practical quizzes it is often the case that the items are identical on many components, this is not always true in situations where the notion of intruder makes sense.

Let us consider the simple example (*lorry, bus, bicycle, car*) (where the obvious intruder is *bicycle*) shown in Figure 6, with an encoding in terms of 5 binary attributes having a straightforward meaning ('specLicense' is short for 'requires a specific driving license'). When considering the item componentwise, we see that:

| | canMove | hasEngine | specLicense | has4Wheel | canFly |
|---------|---------|-----------|-------------|-----------|--------|
| lorry | 1 | 1 | 1 | 1 | 0 |
| bus | 1 | 1 | 1 | 1 | 0 |
| bicycle | 1 | 0 | 0 | 0 | 0 |
| car | 1 | 1 | 0 | 1 | 0 |

Fig. 6. A simple quiz and its Boolean coding

- for $i = 1, 3, 5$,
 $H_a(a_i, b_i, c_i, d_i) = H_b(a_i, b_i, c_i, d_i) = H_c(a_i, b_i, c_i, d_i) = H_d(a_i, b_i, c_i, d_i) = 0$.
- for $i = 2, 4$, $H_a(a_i, b_i, c_i, d_i) = H_b(a_i, b_i, c_i, d_i) = H_d(a_i, b_i, c_i, d_i) = 1$.
- for $i = 2, 4$, $H_c(a_i, b_i, c_i, d_i) = 0$.

The indices 1, 3 and 5 are not useful to pick up the intruder because all the heterogeneous proportions have the same truth value. This is not the case for the indices 2 and 4: H_a for instance, being equal to 1, insures that there is an intruder (which is not the first element). The intruder is then given by the proportion having the value 0: for instance, $H_a(a_j, b_j, c_j, d_j) = 1$ together with $H_c(a_j, b_j, c_j, d_j) = 0$ mean that the assertion *c is not an intruder* is false, which implies that *c* is the intruder w.r.t. component *j*. In our example, *c = bicycle* is the unique intruder w.r.t. components 2 and 4, and is thus the global intruder. Note that it could be the case that the values of the proportions refer to *distinct* items as potential intruders. Then, a special procedure is needed to identify a global intruder, when possible, as explained now.

Let us assume that each item *a* is represented as a Boolean vector (a_1, \dots, a_n) where *n* is the number of attributes and $a_i \in \{0, 1\}$. For each $x \in \{a, b, c, d\}$, we compute the number \mathcal{N}_x of components where H_x is equal to 0 (rather than 1), namely

$$\mathcal{N}_x = |\{i \in \{1, \dots, n\} \text{ s.t. } H_x(a_i, b_i, c_i, d_i) = 0\}|$$

In case where $\mathcal{N}_a = \mathcal{N}_b = \mathcal{N}_c = \mathcal{N}_d = n$, it means that homogeneous proportions hold for each component, and obviously, there is no intruder. More generally, in case where

$\mathcal{N}_a = \mathcal{N}_b = \mathcal{N}_c = \mathcal{N}_d = p < n$, it means that on $n - p$ components, heterogeneous proportions hold for each of the items a, b, c, d . Thus, a, b, c, d are equally referred to as potential intruders (they are intruders w.r.t. the same number of components), and there is no intruder. Lastly, x is an intruder only if $\mathcal{N}_x > \mathcal{N}_y$ for each $y \neq x$. Otherwise, $\exists u, v \in \{a, b, c, d\}$ s.t. $u \neq v$ and $\mathcal{N}_u = \mathcal{N}_v$, and there is no clear intruder.

In the case where we have to find the odd one out among more than 4 items, an option is to consider all the subsets of 4 items. For each such subset, we apply the previous method to exhibit an intruder (if any). Then the global intruder will be the one which is intruder for the maximum number of subsets.

Thanks to the multiple-valued extension, we may think of generalizing this method to the non Boolean case where each item a is represented as a real vector (a_1, \dots, a_n) and $a_i \in [0, 1]$. According to the end of section 4.2, we are in position to detect an intruder w.r.t. a component, only if 3 heterogeneous proportions are high, and the remaining one is low. When such a condition is satisfied to some extent on one or several components, and if all these components identify the same intruder x , this x is the global intruder as in the Boolean case. When the considered components refer to different intruders, it would be necessary to take into account the degree to which each of these components points out a candidate as an intruder. This degree may be estimated as an increasing function of the difference between the two smallest truth-values among H_a, H_b, H_c, H_d (the smallest referring to the intruder). Vectors of intruder degrees associated with each candidate are then compared on a leximax basis.

8 Classification by transduction

Transduction [17] is the name given to a form of reasoning that amounts to predict the class of a new piece of data on the basis of a set S of previously observed pieces of data whose class is known, without any attempt at inducing a generic model for the observed data (which would be then applied to the new piece of data in order to determine its class). A simple example of transduction mechanism (also known as lazy learning) is the k -Nearest Neighbors method, where the class that is the most frequent among the k closest neighbors of x is inferred for x .

The application of the transfer pattern to classification provides the basis for another transduction mechanism, proposed in [52] for binary classification and binary-valued features, and then extended to multiple-valued features and to multiple class problems [58]. Each piece of data x is described by a vector (x_1, \dots, x_n) of feature values that are normalized in the interval $[0, 1]$ together with its class $cl(x)$. Then a classification method, quite different from k-NN methods, is applied since the new item d to be classified is not just compared with the classified items on a one-by-one basis. Once chosen some fixed analogy-related proportion T , we look for 3-tuples $(a, b, c) \in S^3$ such that the proportion $T(cl(a), cl(b), cl(c), cl)$ has a solution cl . This requires that $cl(a), cl(b)$, and $cl(c)$ correspond to a *maximum of two* distinct classes: either $cl(d) = cl(a) = cl(b) = cl(c)$, or there are two distinct classes ($cl(a) = cl(b) \neq cl(c) = cl(d)$ or $(cl(a) = cl(c) \neq cl(b) = cl(d))$). Only such triples (a, b, c) are retained as potentially useful. indeed the other triples (a, b, c) are useless since whatever the coming d , they

cannot constitute a logical proportion with d . This processing of the suitable set of triples can be done offline.

Then, we have to look, among the set of suitable triples, for the one(s) that seem(s) the most appropriate to predict the class $cl(d)$. For doing this, each suitable triple we consider is evaluated by means of the following vector:

$$(T(a_1, b_1, c_1, d_1), \dots, T(a_i, b_i, c_i, d_i), \dots, T(a_n, b_n, c_n, d_n))$$

Then the vectors (and thus the triples) are ordered in a lexicographic decreasing manner⁵. Then we may choose for $cl(d)$ the class associated to the triple having the best evaluation, or the most frequent class among the k best triples.

This approach has been tested on different benchmark problems and has given rather good results [58]. This agrees with the results previously obtained in [6, 37], where the authors have developed a binary classifier on the basis of an “analogical dissimilarity” measure AD : for a given tuple (a, b, c, d) , $AD(a, b, c, d)$ is a positive number which is zero if and only if the proportion $a : b :: c : d$ holds. The reasons behind this success are investigated in [9].

9 Interpolation, extrapolation and non-monotonic reasoning

A problem formally similar to transduction is the problem of reasoning from an incomplete collection of parallel if-then rules of the form “if X_1 is A_1 and ... and X_n is A_n then Y is B ” where the A_i ’s and the B ’s are labels belonging to ordered domains; for instance, the labels associated with the domain of X_i could be ‘small’, ‘medium’, and ‘large’. Such a rule base may be incomplete in the sense that for some combination of the labels of the condition variables X_i , there may not be a corresponding rule. Such a problem may be handled by considering that the A_i ’s and the B ’s are represented by fuzzy sets. Another route may take advantage of the idea of analogical proportion, as first suggested in [59]. See [65] for a detailed discussion. The analogical proportions will no longer apply between feature values pertaining to different pieces of data as in classification, but between labels appearing in the expression of generic rules. The approach is based on the assumption that the mapping from the X_i ’s to Y which is partially described by the set of available rules, is sufficiently “regular” for completing missing rules, as in the following example. Assume we know the two rules

[rule 1] “if X_1 is *small* and X_2 is *small* then Y is *large*”

[rule 2] “if X_1 is *small* and X_2 is *large* then Y is *small*”

where the possible labels associated with variables X_1 , X_2 , and Y are *small*, *medium*, or *large*. Then, if we wonder what may be a plausible conclusion for the rule

[rule 3] “if X_1 is *small* and X_2 is *medium* then Y is ...”

we may observe that a kind of analogical proportion of the form *rule 1* : *rule 3* :: *rule 3* : *rule 2* holds. Indeed, one may consider that we have for variable X_1 : *small* : *small* :: *small* : *small*, which certainly holds on the basis of pure identity, and for variable X_2 we get *small* : *medium* :: *medium* : *large*, which holds as much as the

⁵ $(u_1, \dots, u_i, \dots, u_n) >_{\text{lexicographic}} (v_1, \dots, v_i, \dots, v_n)$, once the components of each vector have been decreasingly ordered, iff $\exists j < n \forall i = 1, j u_i = v_i$ and $u_{j+1} > v_{j+1}$.

increase from *small* to *medium* is the same as the increase from *medium* to *large*. What we have here are analogical proportions on a discrete scale. This leads to an equation of the form *large* : x :: x : *small* for the label of Y in rule 3, with *medium* as a solution. In case we would rather know rule 1 and rule 3, and we have to guess the conclusion of rule 2, we would obtain the equation *large* : *medium* :: *medium* : x (with $x = \text{small}$ as a solution). We can thus perform extrapolation as well as interpolation. Such an approach fully agrees with a more cautious treatment of the ideas of “betweenness” and “parallelism” between symbolic situations in conceptual spaces; see [64] for further discussions.

Non-monotonic reasoning and analogical reasoning are very different in nature since one handles generic default rules while the other one parallels particular situations, even if both leads to brittle conclusions. Still there are noticeable connections between them; see [54] for detailed discussions. Let us only mention that in the setting of nonmonotonic consequence relations, one may propose a representation of “ a is analogous to b ”, which agrees with the pattern of inference originally hinted by Polya [46]: “ a is analogous to b , a is true, b plausibly holds”. This may be related to the idea that two situations are analogous if they only differ on *non important* features.

10 Concluding remarks

A Boolean view of the ideas of similarity and dissimilarity has led to the concept of logical proportions, which is a logical generalization of the numerical concept of proportion. 8 proportions are especially remarkable. The four homogeneous ones seem to play an important role in classification and other types of reasoning tasks where particular patterns can be put in parallel. The four heterogeneous proportions can be used to code and solve Find The Odd One Out quizzes. Ultimately, logical proportions offer a uniform approach to deal with large variety of reasoning tasks. In spite of its wide scope, this overview has left aside some worth-mentioning issues such as

- the study of the relation of analogical proportion with formal concept analysis [41],
- the evaluation of natural language analogical proportions in terms of Kolmogorov complexity [5],
- the handling of analogical proportions with respect to situations pervaded with probabilistic uncertainty [51],
- the use of analogies in argumentation processes [4] (see [2] for a preliminary proposal using the modeling of analogical proportions presented in this survey).

References

1. L. Akoglu, M. McGlohon, and C. Faloutsos. Oddball: Spotting anomalies in weighted graphs. In M. Javeed Zaki, J. Xu Yu, B. Ravindran, and V. Pudi, editors, *Proc. 14th Pacific-Asia Conf. Advances in Knowledge Discovery and Data Mining (PAKDD’10), Hyderabad, India, June 21-24, Part II*, volume 6119 of *LNCS*, pages 410–421. Springer, 2010.
2. L. Amgoud, Y. Ouannani, and H. Prade. Arguing by analogy towards a formal view a preliminary discussion. In G. Richard, editor, *Working Papers of the 1st Inter. Workshop on Similarity and Analogy-based Methods in AI (SAMAI’12), Montpellier, Aug. 27*, pages 64–67. Res. Rep. IRIT/RR-2012-20, 2012.

3. S. Ando and Y. Lepage. Linguistic structure analysis by analogy: Its efficiency, 1996.
4. P. F. A. Bartha. *By Parallel Reasoning: The Construction and Evaluation of Analogical Arguments*. Oxford University Press, 2009.
5. M. Bayoudh, H. Prade, and G. Richard. Evaluation of analogical proportions through Kolmogorov complexity. *Knowledge-Based Systems*, 29:20–30, 2012.
6. S. Bayoudh, L. Miclet, and A. Delhay. Learning by analogy: A classification rule for binary and nominal data. *Proc. Inter. Conf. on Artificial Intelligence IJCAI07*, pages 678–683, 2007.
7. D. Bollegala, T. Goto, N. Tuan Duc, and M. Ishizuka. Improving relational similarity measurement using symmetries in proportional word analogies. *Information Processing and Management*, to appear in 2012.
8. A. Cornuéjols. Analogy as minimization of description length. In G. Nakhaeizadeh and C. Taylor, editors, *Machine Learning and Statistics: The interface*, pages 321–336. Wiley and Sons, 1996.
9. W. Correa, H. Prade, and G. Richard. Trying to understand how analogical classifiers work. In E. Hüllermeier et al., editor, *Proc. 6th Inter. Conf. on Scalable Uncertainty Management (SUM'12), Marburg, Germany*, LNAI 7520, pages 582–589. Springer Verlag, 2012.
10. W. Correa, H. Prade, and G. Richard. When intelligence is just a matter of copying. In *Proc. 20th Eur. Conf. on Artificial Intelligence, Montpellier, Aug. 27-31*, pages 276–281. IOS Press, 2012.
11. M. Dastani, B. Indurkhy, and R. Scha. An algebraic approach to modeling analogical projection in pattern perception. *Journal of Experimental and Theoretical Artificial Intelligence*, 15(4):489 – 511, 2003.
12. T. R. Davies and Russell S. J. A logical approach to reasoning by analogy. In J. P. McDermott, editor, *Proc. of the 10th International Joint Conference on Artificial Intelligence (IJCAI'87), Milan*, pages 264–270. Morgan Kaufmann, 1987.
13. D. Dubois and H. Prade. Conditional objects as nonmonotonic consequence relationships. *IEEE Trans. on Systems, Man and Cybernetics*, 24:1724–1740, 1994.
14. T. G. Evans. A heuristic program to solve geometry-analogy problems. In *Proc. A.F.I.P. Spring Joint Computer Conf.*, volume 25, pages 5–16, 1964.
15. K. Forbus, A. Lovett, K. Lockwood, J. Wetzel, C. Matuk, B. Jee, and J. Usher. Cogsketch. In *AAAI'08: Proceedings of the 23rd national conference on Artificial intelligence*, pages 1878–1879. AAAI Press, 2008.
16. R. M. French. The computational modeling of analogy-making. *Trends in Cognitive Sciences*, 6(5):200 – 205, 2002.
17. A. Gammerman, V. Vovk, and V. Vapnik. Learning by transduction. In *Proc. 14th Conf. on Uncertainty in AI*, pages 148–155. Morgan Kaufmann, 1998.
18. D. Gentner. Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2):155–170, 1983.
19. D. Gentner. The mechanisms of analogical learning. In S. Vosniadou and A. Ortony, editors, *Similarity and Analogical Reasoning*, pages 197–241. Cambridge University Press, New York, 1989.
20. H. Gust, K. Kühnberger, and U. Schmid. Metaphors and heuristic-driven theory projection (HDTPT). *Theoretical Computer Science*, 354(1):98 – 117, 2006.
21. A. Hampshire. The odd one out tests of intelligence. 2010. <http://www.cambridgebrainsciences.com/browse/reasoning/test/oddoneout>.
22. D. H. Helman, editor. *Analogical Reasoning: Perspectives of Artificial Intelligence*. Cognitive Science, and Philosophy. Kluwer, Dordrecht, 1988.
23. M. Helms and A. Goel. Analogical problem evolution from inception: Complementing problem-solution co-evolution. *Proc. of 5 International Conference on Design Computing and Cognition (DCC'12)*, 2012 (to appear).

24. D. Hofstadter and M. Mitchell. The Copycat project: A model of mental fluidity and analogy-making. In D. Hofstadter and The Fluid Analogies Research Group, editors, *Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought*, pages 205–267, New York, NY, 1995. Basic Books, Inc.
25. K. J. Holyoak and P. Thagard. Analogical mapping by constraint satisfaction. *Cognitive Science*, 13:295–355, 1989.
26. S. Klein. Culture, mysticism & social structure and the calculation of behavior. In *Proc. 5th Europ. Conf. in Artificial Intelligence (ECAI'82), Orsay, France*, pages 141–146, 1982.
27. S. Klein. Analogy and mysticism and the structure of culture (and Com-ments & Reply). *Current Anthropology*, 24 (2):151–180, 1983.
28. M. Klenk, K. Forbus, E. Tomai, and H. Kim. Using analogical model formulation with sketches to solve bennett mechanical comprehension test problems. *Journal Experimental and Theoretical Artificial Intelligence - Special Issue on "Test-Based AI"*, 23(3), 2011.
29. A. Kulakov, G. Stojanov, and D. Davcev. A model of an expectancy-driven and analogy-making actor. *Proceedings of the 2nd International Conference on Development and Learning*, pages 61 – 66, 2002.
30. P. Langlais, F. Yvon, and P. Zweigenbaum. Analogical translation of medical words in different languages. In *GoTAL*, pages 284–295, 2008.
31. Y. Lepage. Analogy and formal languages. *Electr. Notes Theor. Comput. Sci.*, 53, 2001.
32. Y. Lepage, J. Migeot, and E. Guillerm. A measure of the number of true analogies between chunks in japanese. In Z. Vetulani and H. Uszkoreit, editors, *Human Language Technology. Challenges of the Information Society, Third Language and Technology Conference, LTC 2007, Poznan, Poland, October 5-7, 2007, Revised Selected Papers*, volume 5603 of *LNCS*, pages 154–164. Springer, 2009.
33. A. Lovett, K. Forbus, D. Gentner, and E. Sagi. Using analogical mapping to simulate time-course phenomena in perceptual similarity. *Cognitive Systems Research*, 10, 2009.
34. K. McGregor and A. K. Goel. Finding the odd one out: a fractal analogical approach. In *8th ACM conference on Creativity and cognition*, pages 289–298, 2011.
35. E. Melis and M. Veloso. Analogy in problem solving. In *Handbook of Practical Reasoning: Computational and Theoretical Aspects*. Oxford Univ. Press, 1998.
36. L. Miclet, N. Barbot, and B. Jeudy. Analogical proportions in a lattice of sets of alignments built on the common subwords in a finite language. In H. Prade and G. Richard, editors, *Computational Approaches to Analogical Reasoning - Current Trends*, page : in this volume. Springer, 2013.
37. L. Miclet, S. Bayoudh, and A. Delhay. Analogical dissimilarity: definition, algorithms and two experiments in machine learning. *JAIR*, 32, pages 793–824, 2008.
38. L. Miclet and A. Delhay. Relation d'analogie et distance sur un alphabet defini par des traits. Technical Report 1632, IRISA, July 2004.
39. L. Miclet and H. Prade. Logical definition of analogical proportion and its fuzzy extensions. In *Proc. Annual Meeting of the North Amer. Fuzzy Information Proc. Soc. (NAFIPS), New-York, May, 19-22*. IEEE, 2008.
40. L. Miclet and H. Prade. Handling analogical proportions in classical logic and fuzzy logics settings. In *Proc. 10th Eur. Conf. on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'09), Verona*, pages 638–650. Springer, LNCS 5590, 2009.
41. L. Miclet, H. Prade, and D. Guennec. Looking for analogical proportions in a formal concept analysis setting. In A. Napoli and V. Vychodil, editors, *Proc. 8th Inter. Conf. on Concept Laticces and Their Applications (CLA'11)*, pages 295 – 307. INRIA, Nancy, 2011.
42. D. P. O'Donoghue, A. J. Bohan, and M. T. Keane. Seeing things: Inventive reasoning with geometric analogies and topographic maps. *New Generation Comput.*, 24(3):267–288, 2006.

43. P. K. Paritosh and M. E. Klenk. Cognitive processes in quantitative estimation: Analogical anchors and causal adjustment. In *Proc. 28th Annual Conf. of the Cognitive Science Society, Vancouver*, 2006.
44. J. Piaget. *Essai sur les Transformations des Opérations Logiques: les 256 Opérations Ternaires de la Logique Bivalente des Propositions*. Presses Univ. de France, Paris, 1952.
45. J. Piaget. *Logic and Psychology*. Manchester Univ. Press, 1953.
46. G. Polya. *Mathematics and Plausible Reasoning-Vol.1: Induction and analogy in Mathematics, Vol.2: Patterns of Plausible Inference*. Princeton Univ. Press, 2nd ed. 1968, 1954.
47. H. Prade and G. Richard. Analogy, paralogy and reverse analogy: Postulates and inferences. In *Proc. 32nd Ann. Conf. on Artif. Intellig. (KI 2009), Paderborn*, volume LNAI 5803, pages 306–314. Springer, 2009.
48. H. Prade and G. Richard. Analogical proportions: another logical view. In M. Brammer, R. Ellis, and M. Petridis, editors, *Research and Development in Intelligent Systems XXVI, Proc. 29th Ann. Inter. Conf. on AI (SGAI'09), Cambridge, UK, December 2009*, pages 121–134. Springer, 2010.
49. H. Prade and G. Richard. Logical proportions - Typology and roadmap. In E. Hüllermeier, R. Kruse, and F. Hoffmann, editors, *Computational Intelligence for Knowledge-Based Systems Design: Proc. 13th Int. Conf. on Inform. Proc. and Manag. of Uncertainty (IPMU'10), Dortmund, June 28 - July 2*, volume 6178 of LNCS, pages 757–767. Springer, 2010.
50. H. Prade and G. Richard. Multiple-valued logic interpretations of analogical, reverse analogical, and paralogical proportions. In *Proc. 40th IEEE Int. Symp. on Multiple-Valued Logic (ISMVL'10), Barcelona*, pages 258–263, 2010.
51. H. Prade and G. Richard. Nonmonotonic features and uncertainty in reasoning with analogical proportions. In *Online Proc. 13th Inter. Workshop on Non-Monotonic Reasoning (NMR'10)*, 2010. (T. Meyer, E. Ternovska, eds.), Toronto, May 14-16.
52. H. Prade and G. Richard. Reasoning with logical proportions. In *Proc. 12th Int. Conf. on Principles of Knowledge Representation and Reasoning, KR 2010, Toronto, May 9-13, 2010* (F. Z. Lin, U. Sattler, M. Truszczyński, eds.), pages 545–555. AAAI Press, 2010.
53. H. Prade and G. Richard. Analogy-making for solving IQ tests: A logical view. In *Proc. 19th Inter. Conf. on Case-Based Reasoning, Greenwich, London, 12-15 Sept.*, LNCS, pages 561–566. Springer, 2011.
54. H. Prade and G. Richard. Cataloguing/analogizing: A non monotonic view. *Int. J. Intell. Syst.*, 26(12):1176–1195, 2011.
55. H. Prade and G. Richard. Analogical proportions and multiple-valued logics. In L. C. van der Gaag, editor, *Proc. 12th Eur. Conf. on Symb. and Quantit. Appr. to Reasoning with Uncertainty (ECSQARU'13), Utrecht, Jul. 7-10*, LNAI 7958, pages 497–509. Springer, 2012.
56. H. Prade and G. Richard. Homogeneous logical proportions: Their uniqueness and their role in similarity-based prediction. In G. Brewka, T. Eiter, and S. A. McIlraith, editors, *Proc. 13th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'12), Roma, June 10-14*, pages 402–412. AAAI Press, 2012.
57. H. Prade and G. Richard. Picking the one that does not fit - a matter of logical proportions. In D. Ciucci J. Montero, G. Pasi, editor, *Proc. 8th Conf. Europ. Soc. for Fuzzy Logic and Technology (EUSFLAT), Milano, Sept. 11-13*, pages 392–399, 2013.
58. H. Prade, G. Richard, and B. Yao. Enforcing regularity by means of analogy-related proportions-a new approach to classification. *International Journal of Computer Information Systems and Industrial Management Applications*, 4:648–658, 2012.
59. H. Prade and S. Schockaert. Completing rule bases in symbolic domains by analogy making. In *Proc. 7th Conf. Europ. Soc. for Fuzzy Logic and Technology (EUSFLAT), Aix-les-Bains, July 18-22*, 2011.
60. M. Ragni, S. Schleipen, and F. Steffenhagen. Solving proportional analogies: A computational model. pages 51–55. Institute of Cognitive Science, Osnabrück University, 2007.

61. J. Raven. The Raven's progressive matrices: Change and stability over culture and time. *Cognitive Psychology*, 41(1):1 – 48, 2000.
62. T. Sakaguchi, Y. Akaho, K. Okada, T. Date, T. Takagi, N. Kamimeda, M. Miyahara, and T. Tsunoda. Recommendation system with multi-dimensional and parallel-case four-term analogy. *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3137 – 3143, 2011.
63. U. Schmid, H. Gust, K. Kuhnberger, and J. Burghardt. An algebraic framework for solving proportional and predictive analogies. *Eur. Conf. Cogn. Sci.* 295-300, 2003.
64. S. Schockaert and H. Prade. Interpolative and extrapolative reasoning in propositional theories using qualitative knowledge about conceptual spaces. In G. Richard, editor, *Working Papers of the 1st Inter. Workshop on Similarity and Analogy-based Methods in AI (SAMAI'12), Montpellier, Aug. 27*, pages 38–44. Res. Rep. IRIT/RR-2012-20, 2012.
65. S. Schockaert and H. Prade. Completing rule bases using analogical proportions. In H. Prade and G. Richard, editors, *Computational Approaches to Analogical Reasoning - Current Trends*, page : in this volume. Springer, 2013.
66. J. F. Sowa and A. K. Majumdar. Analogical reasoning. In *Proc. Inter. Conf. on Conceptual Structures*, pages 16–36. Springer, LNAI 2746, 2003.
67. N. Stroppa and F. Yvon. An analogical learner for morphological analysis. In *Online Proc. 9th Conf. Comput. Natural Language Learning (CoNLL-2005)*, pages 120–127, 2005.
68. N. Stroppa and F. Yvon. Analogical learning and formal proportions: Definitions and methodological issues. Technical Report ENST-2005-D004, June 2005.
69. P. D. Turney. A uniform approach to analogies, synonyms, antonyms, and associations. In *COLING*, pages 905–912, 2008.
70. T. Veale and M. Keane. The competence of sub-optimal theories of structure mapping on hard analogies. *International Joint Conference on Artificial Intelligence (IJCAI97)*, 15(1):232–237, 1997.
71. P. H. Winston. Learning and reasoning by analogy. *Commun. ACM*, 23(12):689–703, 1980.

Analogical proportions in a lattice of sets of alignments built on the common subwords in a finite language

Laurent Miclet ¹, Nelly Barbot ², and Baptiste Jeudy ³

¹ IRISA-Dyliss, Rennes France. miclet@enssat.fr

² IRISA-Cordial, Lannion France. barbot@enssat.fr

³ Université de Saint-Étienne, Laboratoire Hubert Curien.

baptiste.jeudy@univ-st-etienne.fr

Abstract. We define the locally maximal subwords and locally minimal superwords common to a finite set of words. We also define the corresponding sets of alignments. We give a partial order relation between such sets of alignments, as well as two operations between them. We show that the constructed family of sets of alignments has the lattice structure. The study of analogical proportion in lattices gives hints to use this structure as a machine learning basis, aiming at inducing a generalization of the set of words.

Keywords: Locally maximal subwords, alignments, algebraic structure of sets of alignments on a set of words (lattice), analogical proportion.

1 Introduction

Much has been done on finding maximal subwords and minimal superwords to a set of words, when the order relation is based on the length of words. We are interested in this paper in the same problem, but for the finer order relation based on the definition of a subword. Is there a manner to characterize the set of maximal subwords and that of minimal superwords, given a finite set U of words, according to this relation ? More than that, is there an algebraic relation between all these sets of subwords and superwords of U ? An answer to these questions would allow to give a precise definition to what the words of U share, and how this common core is organised.

The first parts of this paper gives a partial answer to these points. We define in section 2 a particular case of the notion of alignment, which will be useful for our construction. Actually, in section 3, we define two operations and an order relation on sets of alignments that leads to the construction of a lattice.

We are also interested in how this structure could be analysed in terms of analogical proportions, which could be used in machine learning. Since we start from a finite set of words, the convenient machine learning framework seems to be grammatical inference (from a positive set of positive samples, in our case). It seems that the lattice structure is particularly adapted to learning by analogy, since some natural analogical proportions can be observed in such a structure. We give in section 4 some hints on these points.

2 Maximal subword, minimal superword, alignment

2.1 Basics

Let Σ be an alphabet, *i.e.* a finite set of letters. A *word* u is a sequence $u_1 \dots u_n$ of letters in Σ . The length of u , denoted $|u|$ is n . The empty word, of null length, is ϵ . A *language* is a set of words. A *subword* of a word u is a word obtained by deleting letters of u at some (non necessarily adjacent) positions⁴ in u . We denote $u \bullet v$ the *shuffle* of the two words u and v .

In Σ^* , the set of all words on Σ , we use the order relation \leq defined by: $(u \leq v \Leftrightarrow u \text{ is a subword of } v)$. When u is a subword of v , v is called a *superword*⁵ of u . For example: $abc \leq aabbcd$.

A word w is a *common subword* to u and v when $w \leq u$ and $w \leq v$. The word w is a *maximal common subword* to u and v if there does not exist any other common subword x to u and v such that $w \leq x$. For example, ab and c are maximal common subwords to $u = cadba$ and $v = fabghc$, while a is a non maximal common subword. Defining a common maximal subword to a finite set of words is a straightforward extension.

A maximal common subword to two words and to a non empty finite set of words is defined in an analog way.

In a partially ordered set S , an antichain is a subset of S composed of pairwise incomparable elements. Any subset T of S can be reduced to a maximal antichain by removing from T every element of T lesser than another element of T .

2.2 Alignments

Definition

Definition 1 An alignment is a finite set of pairs (w, l) where w is a word and l a set of indices between 1 and $|w|$. The set l defines a subword of w denoted $w[l]$. Moreover, an alignment α must satisfy the following properties for all $(w, l) \in \alpha$ and $(w', l') \in \alpha$:

1. $w[l] = w'[l']$
2. $(w = w') \Rightarrow (l = l')$
3. $(w \leq w') \Rightarrow (w = w')$

The set of words on which the alignment is defined is called the *support* and is denoted $word(\alpha) = \{w \mid \exists l \subset \mathbb{N} \text{ with } (w, l) \in \alpha\}$. According to our definition⁶, the support is an antichain of words for \leq .

The set of indices l will be called the *position* of the indexed subword of $w[l]$.

⁴ Other terms for *subword* are *subsequence* and *partial word*. A *factor*, or *substring* is a subword of u built by contiguous letters of u .

⁵ A *superword* of u , also called a *supersequence* must not be confused with a *superstring* of u , in which the letters of u are contiguous. In other words, u is a factor (a substring) of any superstring of u . See [Gus97], pages 4, 309 and 426.

⁶ An alignment (regardless of the third point of our definition), is called a *trace* by Wagner and Fisher [WF74] for two words and a *threading scheme* in Maier [Mai78].

In the following, an alignment will be represented by a set of words in which some letters are boxed. For each element (w, l) of the alignment, the boxed letters represent the subword $w[l]$ (also called the boxed subword of the alignment, which can be also denoted $box(a)$).

For legibility, the n words can be displayed in such a manner that the corresponding letters of w in the n words are in the same column. Some blanks can be added freely to help the reading. For example:

$$a = \begin{pmatrix} & \boxed{a} & \boxed{c} & b & d & e & g \\ & \boxed{a} & \boxed{c} & & e & & h \\ g & \boxed{a} & h & \boxed{c} & d & & \end{pmatrix}$$

denotes the alignment

$$a = \{(acbdeg, \{1, 2\}), (aceh, \{1, 2\}), (gahcd, \{2, 4\})\}.$$

We can write also without ambiguity:

$$a = (\boxed{a} \boxed{c} bdeg, \boxed{a} \boxed{c} eh, g \boxed{a} h \boxed{c} d).$$

Locally maximal alignments and locally maximal subwords Generally speaking, two alignments on the same support $W = \{w_1, \dots, w_n\}$ with the same boxed subword r can be different (having different set of indexes). We could define maximal alignments as those whose boxed letters are maximal subword of W .

However, all interesting alignments would not be maximal with this definition. Consider for example the two words $w_1 = abcd$ and $w_2 = dabcab$. The complete set of common subwords is $\{\epsilon, a, b, c, ab, ac, bc, abc, d\}$ and their set of maximal common subwords is $\{abc, d\}$.

But these two subwords are not sufficient to define the totality of the interesting alignments. Actually the alignment $(\boxed{a} \boxed{b} cd, dab \boxed{c} \boxed{a} \boxed{b})$ is somehow "maximal" since it is not comparable to the only alignment with the boxed subword abc , namely $(\boxed{a} \boxed{b} \boxed{c} d, d \boxed{a} \boxed{b} \boxed{c} ab)$.

This leads to define the following notion of *locally maximal alignment* and of *locally maximal subword*.

Definition 2 An alignment $a = \{(w_1, l_1), \dots, (w_n, l_n)\}$ is locally maximal if there is no other alignment $b = \{(w_1, l'_1), \dots, (w_n, l'_n)\}$ on the same support such that for all i , $l_i \subset l'_i$.

Notice that the empty alignment \emptyset is locally maximal.

Definition 3 The set of boxed subwords associated to all locally maximal alignments between a finite set of words $W = \{w_1, \dots, w_n\}$ is called the set of locally maximal subwords to W and is denoted $\sqcap(W)$. For some $r \in \sqcap(W)$, the set of locally maximal alignments associated to r is denoted $A_r(W)$.

We also define: $A(W) = \bigcup_{r \in \sqcap(W)} A_r(W)$.

For example, let us consider $W = \{ababc, cabd\}$, its sets of locally maximal alignments are given by

$$A_{ab}(W) = \{ (\boxed{a} \boxed{b} abc, c \boxed{a} \boxed{b} d), \\ (\boxed{a} ba \boxed{b} c, c \boxed{a} \boxed{b} d), \\ (ab \boxed{a} \boxed{b} c, c \boxed{a} \boxed{b} d) \}$$

$$A_c(W) = \{(abab \boxed{c}, \boxed{c} abd)\}.$$

$$A(W) = A_{ab}(W) \cup A_c(W).$$

Then, the set of locally maximal subwords of W is

$$\sqcap(W) = \{ab, c\}$$

2.3 Language associated with an alignment

Definition 4 Let $w = w_1 \dots w_p$ be a word, locally maximal subword of two words u and v at only one position (i.e. $|A_w(\{u, v\})| = 1$). Then there exists an unique set of factors of u , denoted (u^1, \dots, u^{p+1}) , and an unique set of factors of v , denoted (v^1, \dots, v^{p+1}) , such that $u = u^1 w_1 \dots u^p w_p u^{p+1}$ and $v = v^1 w_1 \dots v^p w_p v^{p+1}$. We define $L(A_w(\{u, v\}))$ as the following finite language:

$$L(A_w(\{u, v\})) = (u^1 \bullet v^1) w_1 (u^2 \bullet v^2) \dots (u^p \bullet v^p) w_p (u^{p+1} \bullet v^{p+1})$$

The construction of $L(A_w(\{u, v\}))$ is shown in Figure 1, with straightforward graphic conventions.

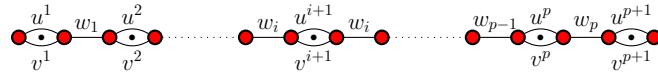


Fig. 1. The construction of $L(A_w(u, v))$ when $|A_w(\{u, v\})| = 1$.

If $|A_w(\{u, v\})| > 1$, $L(A_w(\{u, v\}))$ is defined as the union of all languages associated with all different positions of w as locally maximal subword of u and v . Finally, $L(A(\{u, v\}))$ is defined as the union of the languages $L(A_w(\{u, v\}))$, for all w locally maximal subwords of u and v .

Proposition 1 Let w be a locally maximal subword common to two words u and v and $L(A_w(\{u, v\}))$ constructed as above. We have:

1. All words in $L(A_w(\{u, v\}))$ are (non necessarily minimal) common superwords of u and v .
2. For any word $W \in L(A_w(\{u, v\}))$, we have⁷ $|W| + |w| = |u| + |v|$.

⁷ A consequence of this assertion is : let $LCS(u, v)$ be a longest common subword to u and v and $SCS(u, v)$ be a shortest common superword to u and v . Then we have: $|LCS(u, v)| + |SCS(u, v)| = |u| + |v|$.

Proof. We firstly give an example to show that a word of $L(A_w(\{u, v\}))$ can be a non-minimal superword of u and v .

We take the two words $u = abcabb$ and $v = aabbc$. The associated alignment ($\boxed{a} \boxed{b} ca \boxed{b} b$, $\boxed{a} a \boxed{b} \boxed{b} c$) is locally maximal. The language $L(A_{abb}(\{u, v\}))$ contains the language $aabcab(b \cdot c)$ and, in particular, the word $w = aabcabbc$. The word $w' = abcabbc$ is another superword of u and v , and $w' \leq w$. Thus, w is not a minimal superword of u and v .

Then we demonstrate the proposition.

Let us consider $W \in L(A_w(\{u, v\}))$. By definition of $L(A_w(\{u, v\}))$, there exists (u^1, \dots, u^{p+1}) and (v^1, \dots, v^{p+1}) , respectively sets of factors of u and v , such that the word W can be written as $W = x^1 w_1 \dots x^p w_p x^{p+1}$ where, for every $i \in \{1, \dots, p+1\}$, $x^i \in (u^i \bullet v^i)$.

1. Therefore, for every $i \in \{1, \dots, p+1\}$, $x^i \geq u^i$ and $x^i \geq v^i$. We then have $W \geq u^1 w_1 \dots u^p w_p u^{p+1} = u$ and $W \geq v^1 w_1 \dots v^p w_p v^{p+1} = v$.
- 2.

$$\begin{aligned} |W| &= |x^1| + |w_1| + \dots + |x^p| + |w_p| + |x^{p+1}| \\ &= |w| + \sum_{i=1}^{p+1} |x^i| = |w| + \sum_{i=1}^{p+1} (|u^i| + |v^i|) \\ &= |w| + (|u| - |w|) + (|v| - |w|) = |u| + |v| - |w|. \end{aligned}$$

□

2.4 Constructive algorithms

We have devised an algorithm producing a finite automaton $\mathcal{A}_{\sqcap(\{u, v\})}$ which exactly recognizes the language $\sqcap(\{u, v\})$, the set of locally maximal subwords common to two words u and v . It is based on the transformation of an 2-d array displaying which letters are common to two words into a finite automaton recognizing $\sqcap(u, v)$ (see an example on figure 2(a)). It uses an algorithm of construction of the Hasse diagram of a partial order relation.

Starting from $\mathcal{A}_{\sqcap(\{u, v\})}$, it is then simple to produce a finite automaton that we call $\mathcal{A}_{\sqcup(\{u, v\})}$ which exactly recognizes the language $L(A(\{u, v\}))$ (also denoted $\sqcup(\{u, v\})$). We display an example at figure 2(b).

3 Order relation and operations between alignments

In this section, we are interested in a particular family of alignments, since we want to describe what have in common the subwords and superwords of a finite set U of sentences. We will consider alignments on U , *i.e.* alignments with a support *subset of* U . Moreover, we will assume that U is an antichain according to the order relation \leq .

3.1 Order relation

Definition 5 (Order on alignments on U) *Given two alignments on U $a = \{(w_1, l_1), \dots, (w_n, l_n)\}$ and $b = \{(w'_1, l'_1), \dots, (w'_m, l'_m)\}$, we write $a \sqsubseteq b$ if for all $i \in (1, n)$, it exists $j \in (1, m)$ such that*

1. $w_i = w'_j$
2. $l'_j \subseteq l_i$

Therefore, if $a \sqsubseteq b$, then $\text{word}(a) \subseteq \text{word}(b)$.

It is easy to check that \sqsubseteq is a partial order relation on the set of alignments and that the empty alignment \emptyset is smaller than every other alignment.

Definition 6 (Homogeneous sets of alignments) A set of alignments is homogeneous if it is non empty and all its elements have the same support. The family of homogeneous sets of locally maximal alignments is denoted \mathcal{A}_H .

In order to link this definition with definition 3, we can notice that, for any subset W of U , $A(W) \in \mathcal{A}_H$.

Definition 7 (Order on homogeneous sets of alignments on U) Let A and B be two homogeneous sets of alignments. We have $A \sqsubseteq B$ if for all $b \in B$, there is $a \in A$ such that $a \sqsubseteq b$.

Proposition 2 \sqsubseteq is a partial order on \mathcal{A}_H and the smallest element is $\{\emptyset\}$.

Proof.

Reflexivity and transitivity are immediate. In order to check the antisymmetry, let us consider two homogeneous sets of locally maximal alignments, denoted A and B , such that: $A \sqsubseteq B$ and $B \sqsubseteq A$. Since A and B are homogeneous, all alignments in A have the same support, denoted $\text{word}(A)$, and the same holds for B , with the support denoted $\text{word}(B)$. From the definition of \sqsubseteq , we easily check that $\text{word}(A) = \text{word}(B)$. Let us consider $b_1 = \{(w_1, l'_1), \dots, (w_n, l'_n)\} \in B$: since $A \sqsubseteq B$ and $B \sqsubseteq A$, it exists $a \in A$ and $b_2 \in B$ such that $a \sqsubseteq b_1$ and $b_2 \sqsubseteq a$. By transitivity, we have $b_2 \sqsubseteq b_1$. At last, b_1 and b_2 having the same support and being locally maximal, it implies that $b_1 = b_2$ and then $a \in B$. Hence, $A \subseteq B$. Similarly, we can check that $B \subseteq A$. \square

3.2 Definition and properties of \uplus

Definition 8 Let $a \in A_r(\{u_1, \dots, u_n\})$ and $b \in A_s(\{v_1, \dots, v_m\})$, where $a = \{(u_1, l_1), \dots, (u_n, l_n)\}$ and $b = \{(v_1, l'_1), \dots, (v_m, l'_m)\}$. Firstly, we construct $a + b$, the finite set of alignments $c = \{(w_1, L_1), \dots, (w_p, L_p)\}$ such that

1. $\{w_1, \dots, w_p\} = \text{word}(a) \cup \text{word}(b)$
2. for all (i, k) , if $(w_k = u_i)$ then $(L_k \subseteq l_i)$
3. for all (j, k) , if $(w_k = v_j)$ then $(L_k \subseteq l'_j)$

Secondly, we denote $a \uplus b$ the set of minimal elements of $a + b$ according to \sqsubseteq .

As consequence, if $\sqcap(\{r, s\}) \neq \emptyset$, then the boxed word in $c \in a + b$ is a subword of r and s , else, no letter is boxed in c . In addition, if a and b contain an identical word $u_i = v_j$ such that $l_i \cap l'_j = \emptyset$, no letter is then boxed in c .

Let a and b two alignments, with $a \in A_r(\{u_1, \dots, u_n\})$ and $b \in A_s(\{v_1, \dots, v_m\})$. We construct $\{a\} \uplus \{b\}$, a finite set of alignments, in the following manner:

- Firstly, for every locally maximal subword w common to r and s , and for every position of w an alignment c is created with:
 1. $\text{word}(c) = \text{word}(a) \cup \text{word}(b)$
 2. $\text{box}(c) = w$ in the concerned position.
- Secondly, $a \uplus b$ is the minimal antichain of this set of alignments, according to \sqsubseteq .

In the worst case, computing $a \uplus b$ can have an exponential complexity. Indeed, the number of alignments in $a \uplus b$ can be exponential. For instance, if $a = (\boxed{a} | a)$ and $b = (\boxed{a} | a | a | b, \boxed{a} | a | a | bb, \dots)$. If alignment b contains n words, then $a \uplus b$ will contain 3^n alignments: for each word of b we have to choose two a among the 3 boxed ones, i.e., 3 possible choices for each word.

A possible solution to decrease the complexity would be to use a lazy evaluation: the set of alignments would be computed only if it is really needed. In this example, we could only store the maximum common subword aa and three boxed letters for each word. This representation would mean that the "real" boxed letters are any subset of the boxed ones such that the boxed word is aa .

The operation \uplus is extended to homogeneous sets of alignments by the following definition.

Definition 9 Let A and B be two homogeneous sets of alignments. We define $A \uplus B$ as the set of the minimal elements of $A + B$ according to \sqsubseteq where

$$A + B = \bigcup_{\substack{b \in B \\ a \in A}} (a + b)$$

Proposition 3 The operation \uplus is internal to \mathcal{A}_H , commutative and idempotent.

Proof. Let us consider $A \in \mathcal{A}_H$ and $B \in \mathcal{A}_H$.

1. All the alignments in $A \uplus B$ are locally maximal by definition and have the same support, namely $\text{word}(A) \cup \text{word}(B)$.
2. The commutativity is straightforward.
3. Let a be an element of A , it is immediate that $a \in (a + a) \subseteq A + A$. Moreover, since $A \in \mathcal{A}_H$, a is a locally maximal alignment, and so $a \in A \uplus A$. Consequently, $A \subseteq A \uplus A$. Reciprocally, let c be an element of $A \uplus A$. Then it exists a couple $(a, b) \in A^2$ such that $c \in a + b$. Since $A \in \mathcal{A}_H$ and $\text{word}(c) = \text{word}(a) \cup \text{word}(b)$, a, b and c have the same support. Moreover, from definitions 5 and 8, $a \sqsubseteq c$ and $b \sqsubseteq c$. c being a minimal element of $A + A$ according to \sqsubseteq , and a and b belonging to $A + A$, it turns out that $a = b = c$. At last, $c \in A$. Hence $A \uplus A \subseteq A$. \sqsubseteq is then idempotent on \mathcal{A}_H . \square

3.3 Construction of \bowtie

Definition 10 Let $a \in A_r(\{u_1, \dots, u_n\})$ and $b \in A_s(\{v_1, \dots, v_m\})$ where $a = \{(u_1, l_1), \dots, (u_n, l_n)\}$ and $b = \{(v_1, l'_1), \dots, (v_m, l'_m)\}$. We construct $a \bowtie b$, the finite set of alignments $c = \{(w_1, L_1), \dots, (w_p, L_p)\}$ such that

1. $\{w_1, \dots, w_p\} = \text{word}(a) \cap \text{word}(b)$

2. Either, for all (i, k) such that $w_k = u_i$ we have $l_i \subseteq L_k$, or for all (j, k) such that $w_k = v_j$ we have $l'_j \subseteq L_k$.
3. c is a locally maximal alignment.

An alignment in $a \oplus b$ is thus based either on a restriction of a to the support $\text{word}(a) \cap \text{word}(b)$ or on a restriction of b to the same support. For instance, if $a = \{(\boxed{a}cd, ab\boxed{a}c, \boxed{a}ba)\}$ and $b = \{(a\boxed{c}d, aba\boxed{c}, \boxed{c}a)\}$, then
 $a \oplus b = \{(\boxed{a}\boxed{c}d, \boxed{a}ba\boxed{c}), (\boxed{a}\boxed{c}d, ab\boxed{a}\boxed{c})\}$.

Definition 11

$$A \oplus B = \bigcup_{\substack{b \in B \\ a \in A}} (a \oplus b)$$

Proposition 4 *The operation \oplus is internal to \mathcal{A}_H , commutative and idempotent.*

Proof. The commutativity is straightforward (definition 10 is symmetric wrt a and b). For idempotence, we use the fact (direct consequence of the definition) that if a and b are locally maximal alignments on the same support, then $a \oplus b = \{a, b\}$. Let us consider $A \in \mathcal{A}_H$: if $a \in A$ then $a \in (a \oplus a) \subseteq (A \oplus A)$ and therefore $A \subseteq A \oplus A$. If $c \in A \oplus A$, then there exists $(a, b) \in A^2$ such that $c \in a \oplus b$. Since a and b have the same support, either $c = a$ or $c = b$, therefore $c \in A$ and $A \oplus A \subseteq A$. \square

3.4 Structure of homogeneous sets of alignments on U

We define $\sup_{\sqsubseteq}(A, B)$ as the minimal set of alignments larger than A and B (if it exists) according to \sqsubseteq . Similarly, $\inf_{\sqsubseteq}(A, B)$ is the maximal set of alignments smaller than A and B .

Proposition 5 *Let A and B be finite homogeneous sets of alignments. Then $\sup_{\sqsubseteq}(A, B)$ exists and:*

$$\sup_{\sqsubseteq}(A, B) = A \uplus B$$

Proof.

- First, we show that $A \uplus B$ is greater than A and B for \sqsubseteq . Let $c \in A \uplus B$. By construction, there exist $a \in A$ and $b \in B$ such that $c \in a \uplus b \subseteq a + b$. By the first item of definition 8, $\text{word}(a) \subseteq \text{word}(c)$ and by the two other items, we can conclude that $a \sqsubseteq c$. Thus for every $c \in C$ there is $a \in A$ such that $a \sqsubseteq c$. Thus $A \sqsubseteq A \uplus B$ and $B \sqsubseteq A \uplus B$.
- Let C be a set of alignments greater than A and B , and let $c \in C$. There are $a \in A$ and $b \in B$ such that $a \sqsubseteq c$ and $b \sqsubseteq c$. We need to find $c' \in A \uplus B$ such that $c' \sqsubseteq c$. Remove from the support of c all words not in the support of a or b . The obtained alignment may not be locally maximal, so we add more boxed letters to make it locally maximal. The result alignment c' satisfies all conditions of Definition 8, thus $A \uplus B \sqsubseteq C$ and therefore $\sup_{\sqsubseteq}(A, B) = A \uplus B$. \square

There is no equivalent relation between \oplus and \inf for all homogeneous sets of alignments, we must restrict to sets of all alignments built on a given set of words.

Definition 12 If U is a finite collection of words, we define the collection of sets of alignments $\mathcal{A}(U) = \{A(V) \mid V \subseteq U\}$.

Proposition 6 Let A and B be sets of alignments in $\mathcal{A}(U)$. Then, in $\mathcal{A}(U)$, $\inf_{\sqsubseteq}(A, B)$ exists and:

$$\inf_{\sqsubseteq}(A, B) = A \oplus B$$

Proof.

- First, we show that if $A = A(V)$ and $B = A(W)$ with $V \subseteq U$ and $W \subseteq U$ then $A \oplus B = A(W \cap V)$. Let $c \in A \oplus B$. c is a locally maximal alignment on its support $\text{word}(A) \cap \text{word}(B) = W \cap V$, thus $c \in A(W \cap V)$. Let $a \in A(W \cap V)$. Let a be an alignment on W such that $c \sqsubseteq a$, then c is obtained from $a \in A$ using the definition of $A \oplus B$ and $c \in A \oplus B$.
- Let $C \in \mathcal{A}(U)$ be a set of alignments smaller than A and B . We show that C is smaller than $A \oplus B$. Some alignments of C are smaller than alignments of A and others are smaller than alignments of B . Since C is homogeneous, its support $\text{word}(C)$ must be included in $\text{word}(A) \cap \text{word}(B)$ and since $C = A(T)$ for some $T \subseteq U$, then $T \subseteq V \cap W$. Therefore $A(T) \sqsubseteq A(V \cap W)$ which is exactly $C \sqsubseteq A \oplus B$. \square

Proposition 7 Let $U = \{u_1, u_2, \dots, u_n\}$ be a finite set of words, the operations \oplus and \sqsubseteq are internal to $\mathcal{A}(U)$.

Proof. For \oplus it is a consequence of the previous definition. For \sqsubseteq , it is not difficult to see it from the definition of \sqsubseteq . \square

Proposition 8 Let $U = \{u_1, u_2, \dots, u_n\}$ be a finite set of words, antichain for \leq . Then $\mathcal{U} = (\mathcal{A}(U), \sqsubseteq, \oplus)$ is a lattice. This lattice is said to be built on the finite language U .

Proof. This is a direct consequence of the three previous propositions. \square

4 Analogical proportions in the lattice \mathcal{U}

4.1 The axioms of analogical proportion

Definition 13 (Analogical proportion) An analogical proportion on a set \mathbb{E} is a relation in \mathbb{E}^4 such that, for all 4-tuples A, B, C et D in relation in this order (denoted $A : B :: C : D$):

1. $A : B :: C : D \Leftrightarrow C : D :: A : B$
2. $A : B :: C : D \Leftrightarrow A : C :: B : D$

For every 2-tuple, one has : $A : B :: A : B$

It is easy to show that five other proportions are equivalent:

$$\begin{array}{cccc} B : A :: D : C & D : B :: C : A & D : C :: B : A \\ B : D :: A : C & C : A :: D : B \end{array}$$

These requirements are often called the *axioms* of analogical proportion (see [Lep03]).

4.2 Analogical proportions between words

A first definition using factorization. According to Yvon and Stroppa [SY05] a general definition of analogical proportion, conform to the axioms, can be given in many different cases thanks to the notion of *factorization*. We show here how it applies in Σ^* , and we will come back later to its use in general lattices.

Definition 14 (Analogical proportions between words.)

$(x, y, z, t) \in \Sigma^*$ are in analogical proportion, which is denoted $x : y :: z : t$, if and only if there exists a positive integer n and two sets of words $(\alpha_i)_{i \in [1, n]}$ and $(\beta_i)_{i \in [1, n]} \in \Sigma^*$ such that:

$$x = \alpha_1 \dots \alpha_n, \quad t = \beta_1 \dots \beta_n, \quad y = \alpha_1 \beta_2 \alpha_3 \dots \alpha_n, \quad z = \beta_1 \alpha_2 \beta_3 \dots \beta_n$$

ou

$$x = \alpha_1 \dots \alpha_n, \quad t = \beta_1 \dots \beta_n, \quad y = \beta_1 \alpha_2 \beta_3 \dots \alpha_n, \quad z = \alpha_1 \beta_2 \alpha_3 \dots \beta_n$$

and $\forall i, \alpha_i \beta_i \neq \epsilon$.

Example. $reception : refection :: deceptive : defective$ is an analogical proportion between sequences, with $n = 3$ and the factors : $\alpha_1 = re$, $\alpha_2 = cept$, $\alpha_3 = ion$, $\beta_1 = de$, $\beta_2 = fect$, $\beta_3 = ive$.

| | β_1 | α_1 | β_2 | α_2 | β_3 | α_3 |
|-------|-----------|------------|-----------|------------|-----------|------------|
| $x :$ | | re | | cept | | ion |
| $y :$ | | re | fект | | | ion |
| $z :$ | de | | | cept | ive | |
| $t :$ | de | | fект | | | ive |

The authors have shown that this definition is conform to the axioms.

Another definition using alignments. This second definition, with the associated algorithms, is given in [MBD08]. The axioms of analogical proportion are verified as well.

Definition 15 Let u, v, w and x four words in Σ^* . We assume that an analogical proportion is defined on Σ . We extend this relation to $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$, adding the proportions $a : \epsilon :: a : \epsilon$ for all $a \in \Sigma$. Then u, v, w and x are in analogical proportion in Σ^* if there exists an alignment between the four words such that every column of the alignment is an analogical proportion in Σ_ϵ .

Example Let $\Sigma = \{a, b, c, A, B, C\}$ an alphabet with the analogical proportions $a : b :: A : B$, $a : c :: A : C$, $c : b :: C : B$. The following alignment shows that there is an analogical proportion in Σ^* between the four words $CaCA, CcbBA, bAc$ and $bCbb$.

$$\begin{pmatrix} C & a & C & A \\ C & c & b & B \\ b & A & c & A \\ b & C & b & B \end{pmatrix}$$

Note that there is no boxed letter in this alignment. It can happen anyway in the case of a column such that $a : a :: a : a$.

Links between the two definitions. The second definition using alignments is shown to imply the first one (not the reverse). However, a straightforward modification of the first one lead to a complete equivalence [Has11].

4.3 Analogical proportions in a lattice

Using the factorization technique, Stroppa and Yvon [SY05] have found that a general definition of an analogical proportion can be given in a lattice. Unfortunately, his definition was incomplete. We give here the complete one.

Definition 16 *For four elements x, y, z and t of the lattice (L, \vee, \wedge) , the analogical proportion denoted $(x : y :: z : t)$ is true if and only if:*

$$\begin{aligned} x &= (x \wedge y) \vee (x \wedge z) \text{ and } x = (x \vee y) \wedge (x \vee z) \\ y &= (x \wedge y) \vee (t \wedge y) \text{ and } y = (x \vee y) \wedge (t \vee y) \\ z &= (t \wedge z) \vee (x \wedge z) \text{ and } t = (t \vee z) \wedge (t \vee y) \\ t &= (t \wedge z) \vee (t \wedge y) \text{ and } z = (t \vee z) \wedge (x \vee z) \end{aligned}$$

The geometry of this definition is displayed in figure 3.

A simple example of proportion in a lattice is given by the following property:

Proposition 9 *Let y and z be two elements of a lattice. Then the following analogical proportion holds:*

$$(y : y \vee z :: y \wedge z : z)$$

4.4 Learning from \mathcal{U}

After having given the basis in the previous sections, we give preliminary here remarks and hints concerning some possible extensions of this work to applications, via machine learning, in connexion with analogical proportions and lattice structure.

Firstly, when investigating the connexions between locally maximal subwords, locally minimal superwords and analogical proportions, a first property is easy to show from definition 15 and proposition 1.

Proposition 10 *Let $w = w_1 \cdots w_p$ be a word, locally maximal subword of two words u and v . Then we have:*

$$t \in L(A_w(\{u, v\})) \Rightarrow t : u :: v : w.$$

Take $u = abcabb$ and $v = aabbc$ with the maximal subword $w = abb$. The alignment ($\boxed{a} \boxed{b} ca \boxed{b} b, \boxed{a} \boxed{a} \boxed{b} \boxed{b} c$) is locally maximal. The language $L(A_{abb}(\{u, v\}))$ contains the word $t = abcabbc$. The facing figure displays the analogical proportion $t : u :: v : w$.

$$\begin{pmatrix} t = & \boxed{a} & a & \boxed{b} & c & a & \boxed{b} & b & c \\ u = & \boxed{a} & & \boxed{b} & c & a & \boxed{b} & b \\ v = & \boxed{a} & a & \boxed{b} & & & \boxed{b} & c \\ w = & \boxed{a} & & \boxed{b} & & & \boxed{b} & \end{pmatrix}$$

However, what we are really interested in is to find how using the lattice \mathcal{U} and its analogical properties to generalize U . As a second remark, we note that any homogeneous set of alignments A in \mathcal{U} represents an intensional definition of the finite language $\sqcup(A)$, the set of locally minimal superwords common to all the words in the support⁸ of A . We can also construct, as indicated in section 2.4, a finite automaton as an intensional representation of this language, with the syntactic analysis facility. Therefore, we have potentially at our disposal a lattice of finite automata, in connection with the lattice of subsets of U : each automaton recognizes a finite language which is a particular generalization of the associated support, itself a subset of U .

We denote hereafter \leq the order relation between finite set of words derived from the subword relation \leq , defined by: $M \leq N$ iff $\forall m \in M, \exists n \in N$ such that $m \leq n$. For example, $\{ab, c\} \leq \{abcd, e\}$. There is an partial inclusion relation between the languages recognized by this lattice of automata, compatible with that of the subsets, since the following property holds.

Proposition 11 *For any subsets J and K of U , the three following relations are equivalent: $L(A(J)) \leq L(A(K))$, $J \subset K$ and $\sqcap(K) \subset \sqcap(J)$.*

Note that the exploration of such a lattice of automata, constructed on a finite set of positive examples, is the basis of the efficient finite automata inference, see [dIH10]. This could be one basis for the use of our lattice in machine learning.

Another threads to follow could be the idea of analogical closure of a finite language, as described in [Lep03] and that of analogical generation, see [BMMA07]. In both, a triple of words is taken in the learning sample and a fourth sentence is generated, under the constraint that the four sentences are in analogical proportion. It is not yet clear to the authors how this technique can be combined with the lattice structure, but this could be a connection with the area of machine learning on the basis of formal concepts, as in [Kuz01].

5 Conclusion and related work

The problem of finding one longest common subsequence (subword) or one shortest common supersequence (superword) to two or more words has been well covered (see e.g. [Gus97], pp 287-293 and 309, [IF92]). However, to the best of our knowledge, the problem of finding an intentional definition to the sets of maximal subwords and minimal superwords of a set of words has not been explored yet. In this, we have produced, via the construction of a lattice of alignment sets, an interesting subset of minimal superwords and maximal subwords to a set of words. We have not worked yet neither

⁸ Remember that this support, that we have denoted $word(A)$, is a subset of U .

on the theoretical complexity of the construction of the lattice of alignments, nor on its practical complexity and applications. Hereafter we give some bibliographical hints to this problem.

A complexity result (sometimes misinterpreted) is given by Maier [Mai78] who has demonstrated that the "yes/no longest common subsequence problem" and the "yes/no shortest common supersequence problem" are NP-complete for alphabets of sufficient size. These problems are defined as follows: "Given an integer k and a set of sequences R , is $|LCS(R)| \leq k$? Is $|SCS(R)| \geq k$?" where $|LCS(R)|$ and $|SCS(R)|$ are the length of a longest common subsequence and the length of a shortest common supersequence of R .

It is also true that finding the length of a shortest (longest) super(sub)sequence common to a set of k sequences is in⁹ $\mathcal{O}(m_1 \cdot \dots \cdot m_k)$, with m_i the size of the i -th of the k sequences, hence exponential in k .

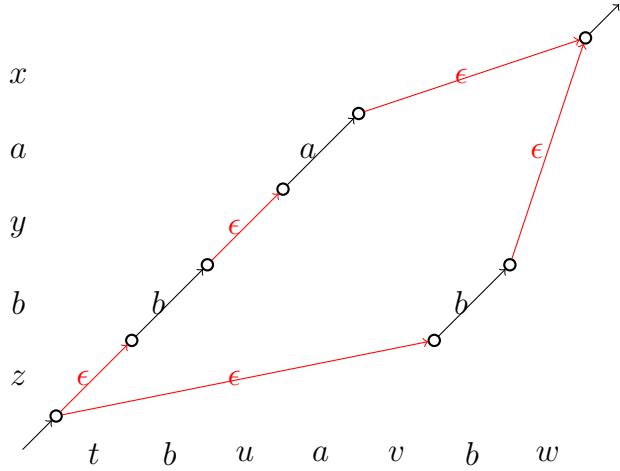
The works of Fraser and Irving [FIM96] have produced algorithms to find the *longest* minimal common supersequence (superword) and the *shortest* maximal common subsequence, according to the order relation \leq .

Yvon and Stroppa [SY05] give a definition of an analogical proportion between words and also within lattices. Our objective is to use the properties of the lattice structure on alignment sets to solve the associated analogical equations.

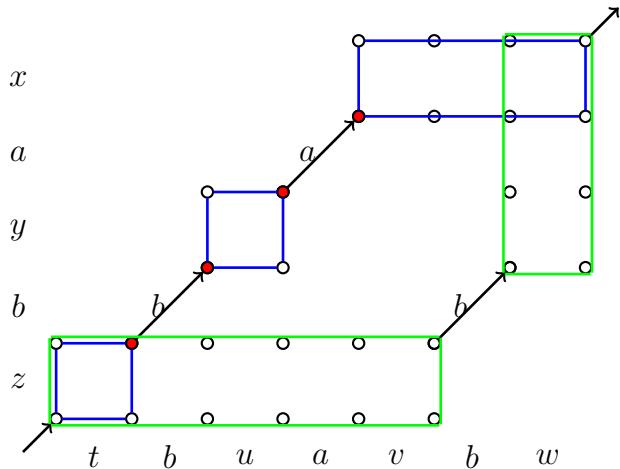
⁹ The elementary operation is the comparison.

References

- [BMMA07] S. Bayoudh, H. Mouchère, L. Miclet, and E. Anquetil. Learning a classifier with very few examples: analogy based and knowledge based generation of new examples for character recognition. In *European Conference on Machine Learning, Springer LNAI 4701*, 2007.
- [dlH10] C. de la Higuera. *Grammatical Inference*. Cambridge University Press, 2010.
- [FIM96] C. Fraser, R. Irving, and M. Middendorf. Maximal common subsequences and minimal common supersequences. *Information and Computation*, 124:145–153, 1996.
- [GHE89] D. Gentner, K. Holyoak, and B. Kokinov (Editors). *The Analogical Mind: Perspectives from Cognitive Science*. MIT Press, 1989.
- [Gus97] D. Gusfield. *Algorithms on Strings, Trees, and Sequences*. Cambridge Univ. Press, 1997.
- [Has11] A. Ben Hassena. *Apprentissage par analogie de structures d'arbres*. PhD thesis, Université de Rennes 1, 2011.
- [IF92] R. Irving and C. Fraser. Two algorithms for the longest common subsequence of three (and more) strings. In *Proc. 3rd Symp. on Combinatorial Pattern Matching, Springer LCNS 644*, pages 214–229, 1992.
- [IF94] R. Irving and C. Fraser. Maximal common subsequences and minimal common supersequences. In *Proc. 5rd Symp. on Combinatorial Pattern Matching, Springer LCNS 807*, pages 173–183, 1994.
- [Kuz01] O. Kuznetsov. Machine learning on the basis of formal concept analysis. *Automation and Remote Control*, 62, Issue 10:1543 – 1564, 2001.
- [Lep03] Y. Lepage. *De l'analogie rendant compte de la commutation en linguistique*. Université de Grenoble, Grenoble, 2003. Habilitation à diriger les recherches.
- [Mai78] D. Maier. The complexity of some problems on subsequences and supersequences. *JACM*, 25:332–336, 1978.
- [MBD08] L. Miclet, S. Bayoudh, and A. Delhay. Analogical dissimilarity: Definition, algorithms and two experiments in machine learning. *Journal of Artificial Intelligence Research*, 32:793–824, 2008.
- [MBJ12] L. Miclet, N. Barbot, and B. Jeudy. The construction of a finite automaton recognizing exactly the maximal subwords common to two (and more) words. *In submission.*, 2012.
- [Mit97] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.
- [SY05] N. Stroppa and F. Yvon. Analogical learning and formal proportions: Definitions and methodological issues. Technical Report ENST-2005-D004, École Nationale Supérieure des Télécommunications, June 2005.
- [WF74] R. Wagner and M. Fisher. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173, 1974.



(a) An automaton which recognizes the language $\sqcap(r, s)$. We have $r = zbyax$ and $s = tbuavbw$; a and b are letters, while t, u, v, w, x, y and z are factors on $\Sigma \setminus \{a, b\}$.



(b) An automaton which recognizes $\sqcup(r, s) = (z \bullet t)b(u \bullet y)a(vbw \bullet x) \cup (tbuav \bullet z)b(w \bullet yax)$. A rectangle holds for the shuffle of the factors on its sides

Fig. 2. Constructing the languages $\sqcap(r, s)$ and $\sqcup(r, s)$

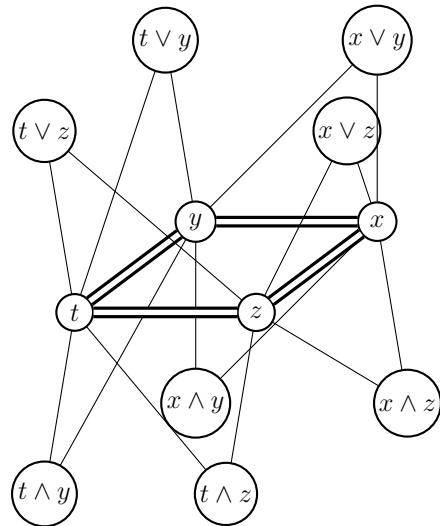


Fig. 3. The general analogical proportion in a lattice.

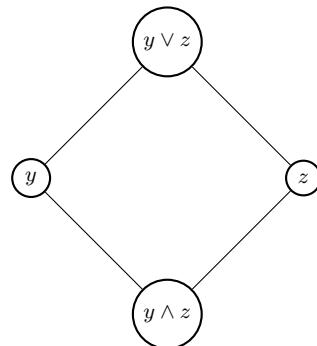


Fig. 4. A canonical proportion in a lattice: $(y : y \vee z :: y \wedge z : z)$.

Part3: From cognition to computational experiments

How similar is what I get to what I want – Matchmaking for Mobility Support

Ute Schmid, Lukas Berle, Michael Munz, Klaus Stein, and Martin Sticht

Faculty Information Systems and Applied Computer Science, University of Bamberg,
Germany
ute.schmid@uni-bamberg.de

Abstract. We introduce matchmaking as a specific setting for similarity assessment. While in many domains similarity is assessed between pairs of entities with equal status, in matchmaking there exists a source request which triggers search for the most similar set of available entities. We describe a specific scenario where elderly people request support or companionship for outdoor activities in the neighbourhood. The scenario is used to formulate requirements for a matchmaking framework. Similarity matching for support requests is based on hard criteria such as gender and on spatio-temporal constraints. Matching companions for outdoor activities needs a more sophisticated method taking into account semantic similarity of requested and offered activities.

1 Introduction

Cognitive scientists consider similarity to play a crucial role in most cognitive processes such as concept acquisition, categorisation, reasoning, decision making, and problem solving [1,2]. Major approaches to similarity in cognitive science as well as in artificial intelligence can be characterised on two dimensions: First, whether basic information about objects is metrical or categorial and second, whether objects are characterised by feature vectors or structural information [3,1]. The majority of cognitive and computational models of similarity are based on feature vector representations [1]. In contrast, many models of analogical reasoning are based on structure mapping since analogy making is per definition a transfer of relations from a base domain to a target domain [4,5].

In psychological research of similarity judgement, the typical task under investigation is that subjects are asked to rate similarity of two objects [1,6]. In this setting, the entities for which similarity is assessed play equivalent roles and can occur as first or second position during evaluation. Furthermore, entities are dissociated from the person who does the rating. However, there are many scenarios, where similarity between a “driver” entity and a series of candidates needs to be assessed. This type of similarity assessment is to the core of information retrieval research. Likewise, in analogical reasoning, a new target problem is the driver for retrieval of a suitable base problem. While analogical matching and transfer is based on structural similarity, for retrieval of a base problem many approaches propose feature based algorithms [7,8]. That is, for retrieval

problems, be it in analogy or information retrieval domains, the leading question is *how similar is what I get to what I need or want?*

In this paper we introduce matchmaking as a special domain of similarity-based retrieval. In general, matchmaking is the process of identifying similar or compatible entities. Requirements stated as a query by a user are matched with descriptions (e.g., of services or social events) provided by other users. We discern two different types of matchmaking – asymmetric and symmetric request relations. An *asymmetric relation* is constituted by a request for service. For example, somebody might look for a person who can drive her or him with a car to some destination. In this case, matching is based on features or constraints which are *complementary* for request and candidate entities. Complementary or fitting features are defined by a request/provides relation. A *symmetric relation* is constituted by a request for a person or entity with similar features or constraints. For example, somebody might look for a person who accompanies her or him to the cinema. In this case, matching is based on corresponding constraints, e.g. with respect to time and location, and similarity of interests.

In the following, we will first present some approaches to matchmaking which were proposed in different application domains. Afterwards, we introduce the domain for our matchmaking approach, that is assistance of outdoor mobility for senior citizens. The requirements are formulated and the matchmaking framework is described. Finally, we present our approach to activity matching based on self-extending ontologies.

2 Approaches to Matchmaking

There exists a wide range of application to matchmaking, such as (online) dating, sports, and business [9,10]. Approaches to matchmaking differ with respect to representation of data, reference to background knowledge and used similarity measures – resulting in different matchmaking algorithms.

In the following, we discuss three approaches to matchmaking. When formulating the requirements of a matchmaking system in the context of our application scenario, we refer back to these approaches – identifying concepts which can and which cannot be used for our specific setting.

2.1 Matching Resources With Semistructured Data

The classad matchmaking framework [11] is a centralised resource management system for managing distributed resources. It allocates information such as availability, capacity, constraints, and other attributes describing a resource. To describe a resource request or to announce a resource to the system, the information is represented as a so called classad (classified advertisement) – a *semi-structured data* model [12] comparable to records or frames in procedural programming languages. Classads are modelled via lists of pairs, each containing an attribute name and one or more values. Data pairs are used to describe offered and requested services. For example, when considering to use a workstation, a requester

would probably store information about the CPU's capabilities or the disk space, while a provider offering a printing service would describe the printer's throughput. In addition, constraints such as restricted user groups can be defined. Finally, ranking rules can be provided.

For a given request, the matchmaker tries to match the classad of the request to a resource with respect to any constraints given in the classads. The rule-based process evaluates expressions like $other.memory \geq self.memory$. The authors focus on the data structure and do not specify a specific matching algorithm. They state that the profiles can be matched in "a general manner" using the specified constraints. Additionally, as goodness criteria, the ranking rules can be applied to find out which classads fit better than others. Finally, the matchmaker informs the matched entities by sending the classads of each other. The resource provider decides whether it accepts or declines the request.

2.2 Matching Web Services Using Clustering

Fenza et al. [13] propose an agent-based system to match semantic web services. There are two different kinds of agents in the system: a broker agent (*mediator*) and one or more advertiser agents. A request for a service is handled only by the broker. When it encounters a request it converts it into a *fuzzy multiset* [14] representation. With these multisets a *relevance degree* is assigned to each possible ontology term that describes a web service according to the place, where the term occurs. For example, if the term occurs in the input specification of the service then it will get a relevance degree of 1. If it occurs in the textual description, then it will get a degree of 0.3 and so on. In this way, it is possible to weight the term for different occurrences via categories.

Advertiser agents interact with web services and with a single broker agent. Each web service description¹ is converted into a fuzzy multiset representation. Note, that the broker does the same with the user's request. So in the end, a broker has a fuzzy multiset of a request and advertiser agents have a fuzzy multiset for each registered service. The broker sends the fuzzy multiset of the request to the advertiser agents to find an appropriate web service. If a web service matches with a request then the matched service is returned by the broker, the corresponding fuzzy multiset is stored to a central cluster and its job is done. Otherwise, the broker tries to find an approximate service by using a knowledge base which is divided into two distinct sets of knowledge: *static knowledge* and *dynamic knowledge*.

There are several ontologies modelled to specific domains in the static part of the background knowledge. To calculate an approximate match, the broker modifies the original request by utilizing the domain ontologies. The dynamic part of the knowledge consists of a cluster of fuzzy multisets where the web service descriptions of the known providers are stored (encoded as fuzzy multisets). It compares the fuzzy multiset of the modified request with the fuzzy multiset of each cluster center and selects the services most similar to the request. That is,

¹ A specific ontology for describing web-services named OWL-S is used.

services with the minimal distance to the request are candidates for an approximation. The similarity is therefore measured using the distances in the fuzzy cluster.

2.3 Matching Activities Using Ontologies

R-U-In? [15] is a social network primarily based on activities and interests of users. A user looking for company for an activity (e.g. going to the cinema or to a jazz club etc.) queries the system with a short description, including time and place. The matchmaker returns contacts found by the users' social network profile, who have similar interests and are located in close proximity. The found contacts need not be known by the querying user yet. For example, the new person might be a social-network "friend" of a "friend" identified by some social network service. Users can post their interests and planned activities on the platform in real-time, i.e. planned activities are dynamic and can often change at the last minute. As a result of this, participants in an activity get updates about changes immediately.

An ontology is used to realise the matching process. There are reasoning mechanisms for ontologies based on description logic [16] and therefore for ontologies based on OWL [17]. Banerjee et al. used an OWL-based *context model* for their activity-oriented social network. Interests are provided by the user itself and are based on tags. Each interest can be tagged via the dimensions *location*, *category* and *time*. Similar interests can be identified by matching on all dimensions: the time (e.g. *evening*, *8 pm*, ...), the category (*horror movie*, *skating*, *jazz*, ...) and the location (*Bamberg*, *jazz-club*).

Tags entered by the user for describing or querying an activity are considered as concepts of the ontology. The matchmaker queries the context model which in return gives a set of similar tags. Those tags are then matched with the tags specified in the user profile. Based on the search criteria of a user, activities might match exactly or just partially. The search result of any match is then ranked by its geographical distance to the current location of the requesting user. Suppose, a user stores the activity (Park, skating, 3 pm) and a second user searches for (skating, afternoon). While the activity *skating* is an exact match, *afternoon* matches only partially with *3pm*. As *afternoon* subsumes *3 pm* it is still possible to match the activity.

In general, the ontology is used to store background knowledge by modelling concepts and relations. For the presented prototype, this is done manually. After a query, the matching process is performed in two steps. First, the context-model is used to get semantically similar tags which are then compared to the tags of the other user's activity descriptions. However, details on how the tags are compared and matched and how the results are ranked (beside of the geographical distance) are not discussed by the authors.

3 SCENARIO

Maintenance of mobility in old age is a crucial factor of quality of life because many activities of daily living require being mobile [18]. Indoor mobility is important for independent living in one's own home, outdoor mobility is a prerequisite for activities such as shopping, visiting friends, and participating in social events [19]. Impairments of mobility result from different causes: old age often entails bodily handicaps such as restricted ability to walk or reduced eye-sight. However, mobility can also be affected by monetary or mental constraints such as the fear of falling [20] or the fear to travel alone with public transport [21].

Our research goal is to improve outdoor mobility and to establish social connections of (older) people. To improve mobility plus covering social aspects we have to distinguish between two domains of interest the matchmaker has to deal with: *mobility assistance* and *outdoor activities*, see also figure 1. Matchmaking in those two domains is either accomplished by a *best fit* approach or by a *similarity* approach. The idea is to build a platform mainly based on collaborative help, but also includes service providers.

In the best fit approach the matchmaker needs to determine the best 'help response' to a 'help request'. In the similarity approach the matchmaker needs to find related or *similar* content or activities.

3.1 Mobility Assistance

Mobility assistance tries to improve mobility of older people. Often, older people are in the position of needing temporal help. One can think of going to the supermarket, going to the medical doctor, using public transports, carrying groceries, going for a walk around the block etc.. To improve mobility of those people the system can be used to request assistance. Volunteers then, can provide assistance and support in their mobility. We call the relationship between seniors and volunteers in the domain of mobility assistance *asymmetric*, because seniors depend on someone else (volunteer) who give support and help. Without the help of a volunteer a senior wouldn't be able to be as mobile as she likes to be or in the worst case couldn't realise it at all.



Fig. 1. *left: asymmetric.* A helping situation is an asymmetric relationship. The older person depends on someone who is willing to help, because she can't do it on her own. *right: symmetric.* The relationship of activities is symmetric. Involved parties are at an equal level, no one depends on someone else to do the activity.

In general, volunteers should be certified over some agency to guarantee they are trustworthy and qualified to deal with the specific needs of older people. Screening and training of volunteers implies some costs, seniors typically would be paying some small fee to be registered at the agency.

Scenario 1 - mobility assistance: asymmetric relationship. Mrs. Peters is a 82 years old lady who needs attendance in taking the public bus lines. She thinks it's too complicated for her, because she has to know how to buy a ticket, where to change bus lines, and the bus stations to get off. Lisa Gustafsson has an appointment at the medical doctor on Wednesday at 12.15 pm. She has a zimmer frame and needs to use the public bus to go to the doctor. Aylia Özdan is 61 and a volunteer. She is willing to help someone else on Monday, Tuesday, and Thursday from 11 am to 1 pm. Mr. Weber is a 58 years old invalidity pensioner. He has committed himself to work as a volunteer and is usually available from Monday until Friday from 12 pm to 16 pm.

In the situations above there are some information explicitly given and some implicitly. There are two people who have themselves committed to work as volunteers and there are two older people who need assistance. Naturally, one would match the help request of Lisa Gustafsson to Mr. Weber who is a volunteer. The time slot of Lisa Gustafsson is fixed, because she has an appointment at the medical doctor, and Mr. Weber's time slot as a volunteer matches in time and day. But the problem here is implicitly given. Lisa Gustafsson is a woman who is going to the doctor and she might be embarrassed about getting assistance from a man. Of course, Lisa Gustafsson could explicitly say she doesn't want a man as an assistance. In that case, the best fit would be Aylia Özdan, although the time slot doesn't fit.

The request of Mrs. Peters is less uncomplicated. She is afraid of taking the public bus, but whether a man or a woman can give her assistance in riding the bus is here of no importance. Therefore, possible matches are both Aylia Özdan and Mr. Weber, because there is no more information given to constraint the matching.

3.2 Outdoor Activities

With outdoor activities not only the mobility of older people can be improved, but also social life. Here, the system can be used to do activities together with other people. Seniors can search for other seniors who have *similar* interests in activities. An important difference between outdoor activities and mobility assistance (described in previous section) is, there are not necessarily volunteers needed. Instead, the focus here lies in meeting people about the same age and with similar interests. The relationship between seniors participating in an activity is *symmetric*, because no one depends on someone else to realize the activity. In the worst case, an activity could be done alone.

Scenario 2 - outdoor activity: symmetric relationship. Mr. Beck is 70 years old and interested in walking. He has also done Nordic walking in the past. None

of his acquaintances is interested in it, and he doesn't know any other person who has the same interests. Mrs. Novak is 73 years old and she likes all kinds of outdoor activities and to be outside and enjoy the nature. Mr. Miller, 81 years old, lives near a park. He loves to go for a walk in the park. He has a walking stick.

In this scenario there are three older people who are looking for other activity partners. When Mr. Beck searches for someone else who also likes walking, he finds that Mr. Miller likes it as well, because both activities are similar. But the system could also offer the activity of Mrs. Novak in the resulting list, because she likes to be in the nature and doing all kinds of activities. She hasn't provided any more information to the system, so the matchmaker can't constrain it any further. But all three activities are - at a higher level - similar and therefore match.

4 Requirements

From the scenarios described above various requirements arise which have to be considered in a matchmaking framework.

Activities A matchmaking framework has to deal with different kinds of requests when it comes to a matchmaking. The essence of scenario 1 is that someone is searching for help and someone else is offering a helping hand. Here, a request for help and an offer to help should be matched. In scenario 2 the situation is different. Users search for other users with similar interests. A matching service should match users with same or similar interests. All requests have in common that they concern "activities". As a result, requests are essentially a search for activities. Therefore, one requirement to a matching framework is to handle those activities properly.

While R-U-In? (Sec. 2.3) matches users with similar interests, it does not provide a fitting algorithm to match offers and requests of users. Classads (Sec. 2.1), on the other hand, supports different roles. It provides the data structure to model 'requests' and 'offers' but no matching algorithm. The fuzzy multiset approach (Sec. 2.2) supports the different roles of requester and provider as long as all parameters can be expressed in fuzzy multisets.

Constraints Activities are essentially sets of constraints. We distinguish between hard ("must") and soft ("should") constraints. If only one hard constraint can't be satisfied the whole activity won't be satisfied at all, as it is the case in scenario 1 not wanting a man as an assistance. If one soft constraint can't be satisfied, the activity will still be available as a possible match, as with Lisa Gustafsson and Aylia Özdan. In the context of matching similar activities, a soft constraint evaluated to false means matched activities do not fit so well. Note, what is seen as hard and soft constraints depends highly on user expectations. Scenario 2 (walking) is an example where a lot of soft constraints exist: time, place, day of week, and even the activity itself (walking, Nordic walking, doing an outdoor

activity). Whereas the examples of scenario 1 have a lot of hard constraints, like time, place, bus line, gender, and day of week.

The classad approach supports modelling of constraints, but the authors do not distinguish between hard and soft constraints. To overcome this, one could think of utilising annotations to distinguish categorically between hard and soft constraints. Both of the other approaches do not have explicit constraining mechanisms. The activity-oriented network R-U-In? models interests of users via three dimensions: “time”, “category”, and “location”. The model could be extended by an additional dimension specifying constraints. The downside of this approach is all dimensions of the model are represented by an ontology. That means, only the concepts of hard and soft constraints could be modelled, but no instances. Otherwise, constraints would be predefined and too inflexible. The fuzzy multiset approach directly supports (weighted) matching of soft constraints, but the data-structure has to be adapted to directly support hard constraints.

Roles The scenarios (Sec. 3) present two basic situations. On one hand, there are people needing help or searching for other people with same interests. On the other hand, there are people offering their help. But there are differences in the degree of helping someone. Some users do volunteering work and other users just do someone a favour. In the context of neighbourly help it’s important to distinguish between these, because there is a difference in the social commitment. the degree of social commitment can be modelled via roles. Roles can represent the different expectations users have, when searching for activities. For example, users searching for help expect to find someone offering help. The same is true vice versa. That is, a volunteer who is looking for users needing help expects to find posted activities.

As already stated, the goal of the proposed framework is to improve social life of older people by focusing on mutual assistance and neighbourly help. Therefore, the role of seniors represents those users who are searching for help or looking for company. The difference in helping is taken into account by other two roles. That is, volunteering work and doing favours are represented by separate roles, because they have different characteristics. People doing volunteering jobs do it on a regular basis and offer assistance explicitly. Usually, they have weaker conditions under which they are willing to help and try to be more flexible in scheduling an appointment with someone who needs help. Furthermore, they are willing to spend some of their spare time in helping others. Users doing favours don’t help out regularly and have stronger conditions under which they are willing to help someone. Doing someone a favour is usually a very time-limited act, so time is a hard constraint. Another characteristic of a favour is most people will do it only if it doesn’t interfere with their own plans and they don’t have to change their schedules.

Classads and fuzzy multisets are designed to match available resources to requests of resources. In those situations there exist only the two roles of service providers and service requesters, and no further differentiation is needed. An activity-oriented social platform like R-U-In? does only have one group of users. All users are interested in doing activities in their spare time. This leads to clear

expectations when using the platform. They either search for or post activities. Because of an activity-oriented user group only one role is needed to represent them.

Knowledge The matchmaking process can be improved by providing knowledge. It is suitable for matchmaking based on similarity to access background knowledge and user profiles.

Background knowledge is the general knowledge available in the system. Activities are represented by it and the knowledge is used to tell how similar different activities are. The matching service can offer similar activities by evaluating it (e.g. by taxonomic relationships). Suppose, someone searches for Nordic walking, but there is no direct match (as described in scenario 2). The matchmaking service can offer activities similar to walking instead and ignore other available activities (e.g. playing golf). Background knowledge has a disadvantage, though. It is often static and explicit. It doesn't change often and represents knowledge to a specific time. Moreover, updating static knowledge is often time consuming. To overcome this we consider to utilise user input as an additional dynamic source. It gives extra information to the matching process to examine the search query of a user and the chosen activity. Considering this, background knowledge is more dynamic and converges to requested user activities.

A user profile is also helpful in the matching process. It has two advantages: first, in the profile are those information stored a user normally doesn't want to re-enter every time a search is submitted. Second, information stored in the profile can be used to filter off matched activities which do not fit. In this way, the result list can be improved. Information in the profile could be among other things: interests of a users, trust to other users, constraints, and a user rating. Activities of other users should be withheld in the result list, if a user marked others as disliked or even untrusted. Trust and user ratings are really important in the context of neighbourly help and are valuable information in the matching process. A matching will get a much higher rating, if there already exists a relationship of trust between users. The implication is, they did some activities in the past, know each other and would like to do future activities together.

Classads [11] have in some extend a user profile, but they do not have any background knowledge. In classads only a resource can specify a list of trusted and untrusted requesters, so the relationship here is unidirectional. The activity network R-U-In? [15] uses both background knowledge and user profiles for a matching. While user profiles are updated in real-time, the background knowledge has no dynamic update mechanism so far. Moreover, there exists a policy repository where a user can define policies for participants when attending an activity. The downside of the platform is one can't rate users, can't mark them as liked or unliked, and it's not possible assigning any status of trust. In the fuzzy multiset approach [13] there is a distinction between background knowledge and fuzzy multisets. The background knowledge is realised in the form of domain ontologies and is static, according to the paper [13]. Whereas, fuzzy multisets are dynamic and are updated according to changes of services.

Requests The matching framework should be able to differentiate between two different classes of requests, *immediate request* and *stalled request*. They represent different searches of activities. Suppose, a user wants to do Nordic walking and issues a search. There aren't defined any preferences in the profile, like hard constraints. Further assume no exact match is possible, but there are two other activities stored (walking and enjoying nature), as the situation in scenario 2. As a result, the best matches are those two. The user has now the choice of either choosing any of the matches he or she is interested in by contacting the other person or to store the request in the system. A user should have the opportunity to store it, if he or she doesn't like any of the activities found or the results are not as expected.

Every time a user initiates a new search for activities to the system he or she immediately receives all matching results best fitting the search. It is an immediate request. The result list is ordered according to a weighting so the best fitting activities are on top. In case the user isn't happy about the found matching results, he or she has the opportunity to initiate a stalled request. The request of the user is stored in the system's activity database and is from now on in monitoring mode. Depending on the preferences stored in the corresponding user profile the user will be notified about new activities of other users similar to his or her activity request. By utilising a stalled request one can find a match that best fits over a period of time. An immediate request returns matches that best fit to all currently available activities.

Classads [11] and the fuzzy multiset approach [13] match a request to the current available set of services only. They do not have to distinguish between different kinds of requests in their systems. In R-U-In? [15] users can search for and post activities. Activities are stored in a so-called activity group repository. A difference to our proposed matchmaking system is, stored activities are not in any monitoring mode, so users are not being informed about searches of other users. Rather, in R-U-In? a user will only be informed, if the requester is interested explicitly in an activity by sending him or her a message.

For the proposed system based on neighbourly help the described requirements are mandatory to the process of matchmaking. Because none of the approaches is appropriate for our needs we propose a matching framework with the required components.

5 COMPONENTS OF A MATCHMAKING FRAMEWORK

We introduce a framework with respect to the requirements identified in chapter 4. Figure 2 depicts all components of the proposed matchmaking framework. It shows the interaction between the components in which the matchmaker is the key component. A user searching for activities initiates a request to the system. All interaction between a user and the system happens via a mediator. The mediator decides whether it is an immediate request or a stalled request. If it's an immediate request the matchmaker will be called. For finding similar activities or

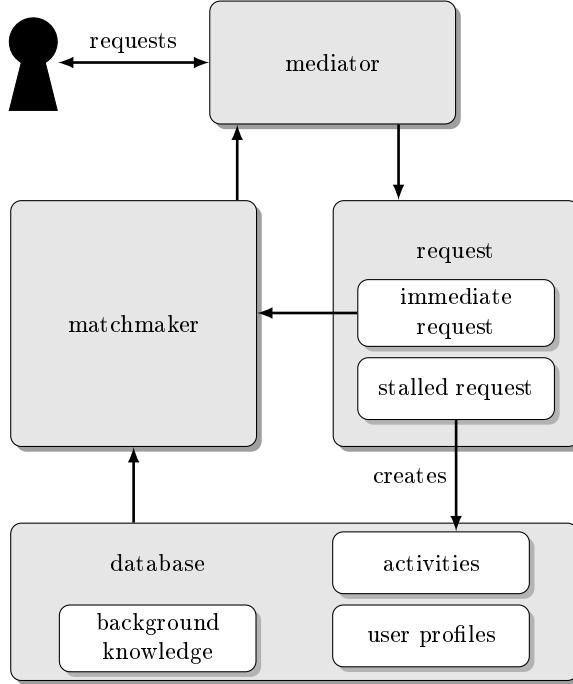


Fig. 2. Components of the matching framework and their interaction. The matchmaker is the key component of the system. It uses the underlying database for a matching and propagates the results to the mediator.

activities which fit to a given request the matching algorithm uses the underlying databases. That is, the background knowledge, the user profiles and the stored activities. A result list is then returned in response to the mediator. If the request is a stalled request an activity will be created in the activities database. There are two things to be aware of: first, the activity is in a monitoring mode. Second, a stalled request can only follow up on an immediate request. Whenever there is a new match for a stalled request the user will be informed.

5.1 Representing Constraints

Descriptions of activities as those mentioned in chapter 3 consist of features, such as gender, time, location, and the name of the activity itself. These features describing an activity are viewed as constraints for a matching and are classified by two dimensions:

$$\begin{array}{ccc}
 \text{similarity} & \leftrightarrow & \text{complement} \\
 \text{hard constraints} & \leftrightarrow & \text{soft constraints}
 \end{array}$$

Some features need to be similar to the activity. Here, reflexivity of mapping holds. On the other hand, some features need to be complementary. For example,

the relationship between *needs car/offers car*. Here we speak of fitting and not of similarity. The matching can be modelled in such a way that the resulting scale corresponds to a similarity mapping. So both similarity and fitting can be processed together.

Hard constraints can be encoded using arbitrary complex boolean formulas on object properties while sets of *weighted* propositions are used for soft constraints. For example, let's assume that Mrs. Peters from scenario 1 only wants help from women who are at least 30 years old (hard constraint). This can be formalised as:

$$\text{other.gender} = \text{female} \quad \wedge \quad \text{other.age} \geq 30 \quad (1)$$

where *other* is a reference to a potential activity partner (similar to [11]). Consider the request of Mrs. Peters finding someone assisting her in riding the public bus as activity a_1 :

$$\text{requires}(a_1, \text{assistance}) \quad (2)$$

Requires relations are matched to corresponding *provides* relations of other activities. Say a_2 is given by another user, namely Aylia Özdan. Both relation will be used to check, if the activities fit, as:

$$\text{provides}(a_2, \text{assistance}) \quad (3)$$

The matchmaker must know that the relations *requires* and *provides* are matchable. However, matching two *requires* relations would not solve any problems. Whereas, relations of the same type (e.g. *likes*) would match in a similarity check:

$$\text{likes}(\text{other}, \text{nordic walking}) \simeq \text{likes}(\text{other}, \text{walking}) \quad (4)$$

Using constraints, time- and location-related restrictions can also be modelled. For time-related restrictions it's necessary to handle intervals to check temporal overlaps. Location-related restrictions calculate and weight distances. The distances are used for ranking purposes. Matches which have a shorter distance are better matches as similar pairs of matches but with greater distance.

5.2 Matching on Constraints

Checking hard constraints can be done by comparing the *requires* and *provides* relations of both, activities and the user profiles. If a hard constraint is violated by an activity description or an involved user profile, the activity will not be considered further in this query. Matching hard constraints should be done before soft constraints are considered. In this way hard constraints are used as filters to omit activities that are violated. Soft constraints have to be checked only on the remaining set of activities to calculate values of the matching quality.

Soft constraints have different weights, i. e. a value between 0.0 and 1.0 representing its importance to a user. These weights are either derived from the user profiles or from the user's query where the requester can specify the importance of each constraint.

If a soft constraint doesn't match, the matchmaker can

1. check the *severity* of the violation (e.g. the other's age is 38 while the claimed age is 40; this violation would not be as strong as if the other's age was 12). Note that this is only possible if a distance between the claimed and the actual value can be obtained (here difference in ages).
2. combine the severity of the violation with the weight of the constraint and find out how severe this violation is for the complete activity. Lower weights of constraints might qualify severe violations and vice versa.

If we assume that the severity can be normalised to a value between 0.0 and 1.0 where 0.0 means no violation and 1.0 represents the hardest possible violation, the *matching violation* V can be obtained by a sum

$$V = \sum_{c \in C} s_c \cdot w_c \quad (5)$$

where s_c represents the normalised severity of the violation of feature c and w_c the weight given by the user. C is the set of all relevant constraints.

In this way, it is possible to calculate for every remaining activity (after checking the hard constraints) a value of how well it fits to a query. A low V means a better fitting. According to these values, target activities can be ranked and presented in the corresponding order.

5.3 Knowledge from User Profiles and Missing Knowledge

We do not only distinguish hard and soft constraints, but also *profile constraints* and *on the fly constraints*. These constraints refer to where they are defined. Profile constraints are defined in user profiles and are used for recurring constraints only. If a user has defined constraints via the profile, the system will take them into account automatically when initiating a request. It's a way of constraining the search implicitly. On the other hand, it should be possible to define constraints manually when doing a search. Those constraints are specified on the fly and are valid only for a specific request. Constraining the search manually should have higher priority as constraints specified in a user profile. For this reason, different knowledge has different priority. Information given in profiles has higher priority as background knowledge. A request has in turn higher priority as profiles. So knowledge with higher priority overwrites lower priority. As a result, constraints defined in the profile influence the search results implicitly, whereas constraints defined on the fly influence it explicitly.

Suppose, a user has the constraint *ignore(fitness walking)* in his profile and searches for a stroll in the city. Walk in the park, Nordic walking, and trekking pole are in the system as available activities. Because walking isn't available, the only similar activities are a walk in the park, Nordic walking, and trekking pole. The matching service just offers a 'walk in the park' as an alternative activity and discards the Nordic walking and trekking pole, because they are on the ignore list via *ignore(fitness walking)*. Now suppose, the same user initiates an explicit request for Nordic walking. The request has higher priority as the

constraint in the profile and overwrites it. This approach allows users to find still activities explicitly, even when the profile states otherwise.

Further, it is also important not to treat unmatched constraints as fails because of missing information about the feasibility on the other side. Assume Lisa Gustafsson wants to go to an opera, but needs someone with a car to go there. So the car is a requirement that can be modelled as a (hard) constraint: *requires(car)*. Someone in her neighbourhood also wants to go to the opera. However, he doesn't mention in his stalled activity that he's going to drive with his own car. The problem here is, Lisa Gustafsson wouldn't find him although the activities would match. In this case, the matchmaker should identify the match and the missing fulfiller (car). Then, inform Lisa Gustafsson about the possible match and propose her to contact the person. After contacting him, Lisa Gustafsson is going to be able to go with him to the opera by car.

Missing information can be treated as *wild cards* which match everything. The matchmaker doesn't know if someone possesses a car, but the requirement is assumed to be fulfilled. However, the activity is going to be marked as *uncertain* until the person in question confirms he's going to the concert by driving his own car.

5.4 Presentation of Results

The approach we're going to use here is to present *all* matching results to the user. For this purpose, the result list is divided into three subsets: matches with complete information, matches with incomplete (missing) information, followed by matches violated by hard constraints. The results within the first subset are ranked by violation of soft constraints. Matches with no violations come first, then matches with low violation and finally matches with high violation. To improve the subset of matches with missing values the user is asked to provide additional information.

6 Activity Mapping

In the following we present details for symmetric matching of leisure activities. Since interaction with the matchmaking service should be as simple as possible, we decided against presenting selection menus for activities in – possibly hierarchical – categories and allow the user to input simple natural language descriptions instead.

Figure 3 shows a model that describes the process of activity suggestions. If a person searches for a partner to join some outdoor leisure activity, the database needs to be searched for a person who is interested in the same or a similar activity. As described in the section above there are further constraints to obey such as local nearness and similar time frames. For matching natural language descriptions we introduce a simple ontology which represents a hierarchy of leisure activities. That is, natural language descriptions are mapped either to a category

to which the current description is similar or to which the current description belongs with a certain probability.

The converter transforms the natural language activity description, the user profile and the restrictions to a Java object and sends this object to the category suggester. The category suggester figures out categories the description belongs to and sends them to the activity weight calculator. If the category suggester cannot find out well matching categories it provides some category suggestions and the user returns his desired category. Finally, the activity weight calculator returns the best matching activities that are stored in the database to the user.

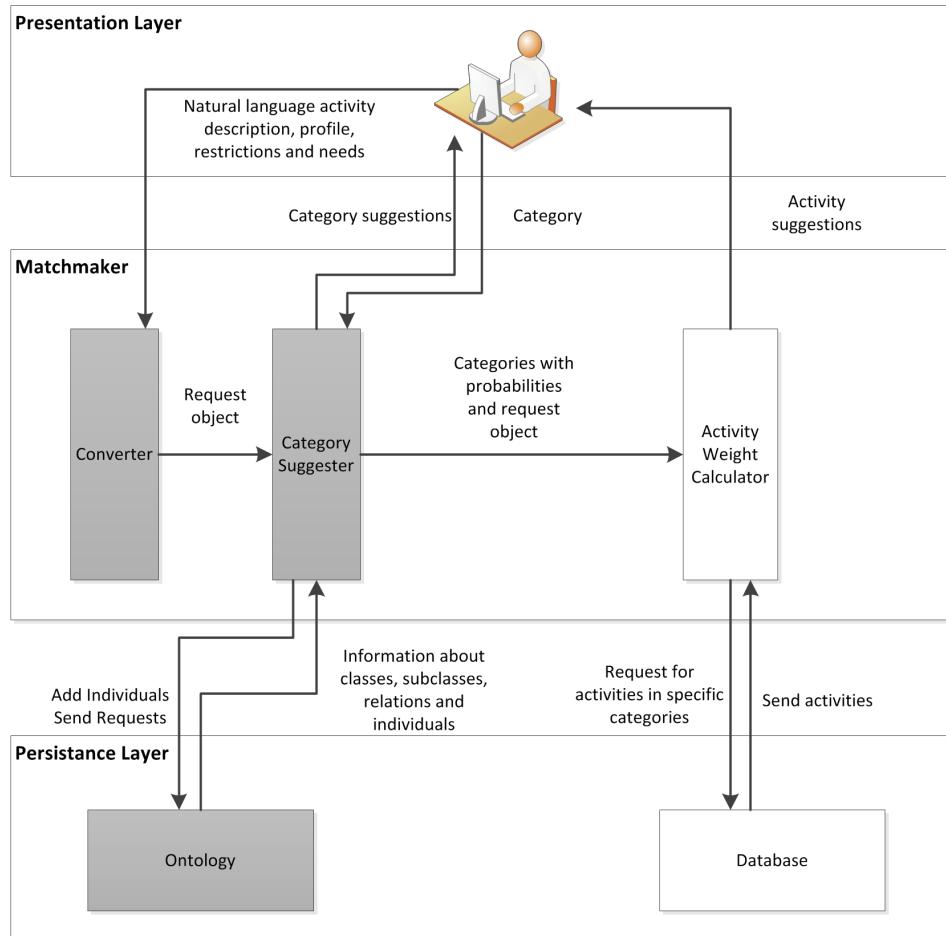


Fig. 3. The process of identifying activity suggestions (aspects discussed in this paper highlighted in grey)

The functional requirements with respect to the category suggester and the ontology are:

Mapping Natural Language Activity Descriptions The application must be able to map natural language activity descriptions to activities. In addition it must be able to return a probability that a description fits to an activity and return how reliable the mapping is.

Semantic Matching The application must be able to deal with generalisations and similarities between different kinds of activities.

Self-Learning The application must learn from user inputs in order to improve future outputs.

6.1 Ontology

One working definition for an ontology is “a formal, explicit specification of shared conceptualisation” [22]. A conceptualisation can be defined as “an abstract, simplified view of the world that we wish to represent for some purpose” [23]. Thus it is possible to model a part of the world by using an ontology. By contrast, a taxonomy is a hierarchical structure in which the objects are just related with one type of relation. This relation can be a sub-concept or a part-of relation [24]. In many ontologies there is a taxonomy which functions as backbone in order to create hierarchical structures [22]. The taxonomy is also a large and important part of the ontology for the desired matchmaker.

There are many different formal ontology languages that describe ontologies. We use OWL2, however making use only of a small part of the OWL2 features. These features are classes, individuals, self defined flags and self defined “has_Relation” annotations. In table 1 all used ontology elements of the matchmaker are described.

Table 1. Used elements in the matchmaker’s ontology

| Representation | Definition | Example |
|----------------|-----------------|-------------------------------|
| | Direct subclass | “Other Activities” → “Zoo” |
| | Individual | “Zoo” → “Zoo of Berlin” |
| | has_Relation | “Stroll” → “Jogging” |
| | Flag | “Zoo of Berlin” → flag: false |

Figure 4 depicts all elements which are used in the ontology for this matchmaker. The following definitions of ontology elements may differ from other definitions because the ontology elements are defined referring to their usage in this work.

Definition 1 (Class). A class is a collection of individuals. There are two kinds of relations between classes: A subclass relation and the inverse parent class relation. Here, a class represents an activity or a collection of activities.

Definition 2 (Parent Class). The parent class P of a class C is $P \mid [C \neq P \wedge C \sqsubseteq P \wedge \neg \exists X \mid (X \sqsubseteq P \wedge C \sqsubseteq X)]$.

Definition 3 (Subclass). The set of subclasses SC of the parent class C are all classes $S \mid (S \sqsubseteq C)$.

Definition 4 (Direct Subclass). The set of direct subclasses SC of the parent class C are all classes $S \mid (\text{parentClass}(S) \equiv C)$.

Definition 5 (Top-Level Class). A top-level class is a class that is a direct subclass of the class “owl:Thing” which is the root class in ontologies described in OWL.

Definition 6 (Individual). An individual is a natural language activity description that has been entered by a user and has been subordinated to a class. Individuals can hold a flag. When “ a ” is an individual that belongs to the class “ C ” then the formal description is $a : C$.

Definition 7 (Flag). A flag is a boolean value an individual can hold. If the flag is true then the individual must be further subordinated in the class hierarchy. If the flag is false, it already belongs to the correct class from the system’s view.

Definition 8 (has_Relation). Between two classes there can be a has_Relation connection. If activities in class B and their subclasses could be relevant for a user if class A is relevant, then there must be a “has_Relation” connection from class A to class B .

6.2 The Algorithm

Definitions for the Algorithms

Definition 9 (Similarity Weight). A similarity weight is a global weight between 0.0 and 1.0. After each class has received a weight, the weight of the classes and their subclasses that are connected with a has_Relation connection from a class that have the highest weight become increased to similarity weight * highest weight.

Definition 10 (Strictness Factor). A strictness factor is a global factor between 0.0 and 1.0. A strictness factor x means that at least the $100 * x \%$ best matching classes are returned. A high strictness factor could cause more potential matching classes than a low strictness factor.

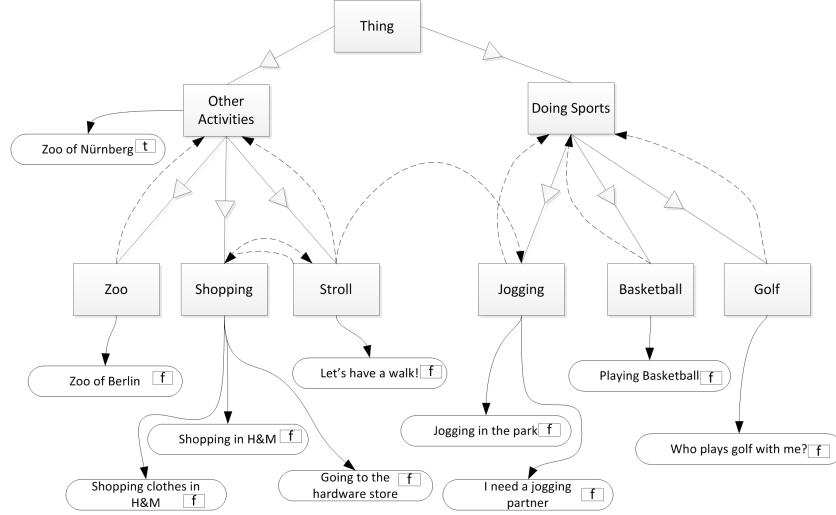


Fig. 4. An exemplary ontology

Definition 11 (Quality Factor). A *quality factor* is a global factor between 0.0 and 1.0. It is used to define whether a class suggestion for a specific natural language activity description is good or bad. Considering a quality factor x , a result is good if at least $100 * x \%$ of the suggested classes are subordinated to one top level class, otherwise the result is bad.

Definition 12 (Direct Match). A *direct match* is a match that occurs if a natural language activity description is a substring of a class name or vice versa.

The technical definitions used in the algorithms are given in table 2.

Weight Calculating The algorithm calculates a normalised weight for the classes in the ontology. The normalised weights indicate the probability that a natural language activity description matches a specific class.

Algorithm 1 *calculateWeights(input_a: natural language activity description)*

- 1: $\text{input_o} \leftarrow$ set of all classes in the ontology
- 2: $\text{input_i} \leftarrow$ set of all individuals in the ontology
- 3: $\text{input_k} \leftarrow$ similarity weight
- 4:
- 5: $\text{dirctMatch} = \text{false}$
- 6: $\text{input_a} = \text{clean}(\text{input_a})$ //eliminates stop words
- 7: **ForEach** $\text{class} \in \text{input_o}$ **Do**
- 8: **If** $\text{input_a} \geqslant \text{class.n}$ **Then**

Table 2. Technical Definitions

| | |
|---------------------------------|---|
| C.n | If C is a class, C.n is its name |
| C.w | If C is a class, C.w is its weight |
| i.n | If i is an individual, i.n is its name |
| i.flag | If i is an individual, i.flag is its flag |
| i.class | If i is an individual, i.class is the class i is subordinated to. Formal: $i.class \equiv C \mid [i : C \wedge \neg\exists X \mid (X \sqsubseteq C \wedge i : X)]$ |
| equals(String s1, String s2) | Is true if s1 is exact the same String like s2 |
| clean(String s) | Returns s without stop words |
| getWords(String a) | Returns a set that contains all words of a |
| parentClass(Class C) | Returns $P \mid [C \not\equiv P \wedge C \sqsubseteq P \wedge \neg\exists X \mid (X \sqsubseteq P \wedge C \sqsubseteq X)]$ |
| subClass(Class C) | Returns all S $\mid (\text{parentClass}(S) \equiv C)$ |
| Class D \equiv Class F | $C \sqsubseteq D \wedge D \sqsubseteq C$ |
| individualsOf(Class C) | Returns all i $\mid [i : C \wedge \neg\exists X \mid (X \sqsubseteq C \wedge i : X)]$ |
| hasRelation(Class C) | Returns a set that contains all classes to which C is connected with a has_Relation connection |
| String s1 \geqslant String s2 | Returns true if s1 is a substring of s2 or if s2 is substring of s1. Otherwise it returns false |
| a++ | Adds 1 to the variable a. More formal: $a = a + 1$ |
| Set.add(c) | Adds c to the set |
| Set.delete(i) | Deletes i from the set |
| C.addIndividual(i) | Adds individual i to the class C |
| getTopLevelClasses | Returns a set containing all C $\mid C \in \text{subClass}(\text{"Thing"})$ |

```

9:   /* There is a direct match. Set class weight to one and assign weights
10:  to subclasses and related classes */
11:  directMatch = true
12:  class.w = 1
13:  If subClass(class)  $\neq \emptyset$  Then
14:    setWeightToClassAndSubClasses(class, 1)
15:  EndIf
16: EndIf
17: EndFor
18: If directMatch = false Then
19:  /* There is no direct match. Check for every word in the natural language
description if it is a substring of a word in an individual. If it is a
substring, increase the weight of the corresponding class */
20: ForEach word  $\in$  getWords(input_a) Do
21:  ForEach class  $\in$  input_o Do
22:    ForEach individual  $\in$  individualsOf(class) Do
23:      ForEach individual_word  $\in$  getWords(individual.n) Do
24:        If word  $\geqslant$  individual_word Then
25:          class.w++
26:        EndIf
27:      EndFor
28:    EndFor
29:  EndFor
30: EndFor

```

```

31: highestWeight =  $\max_{\forall \text{classes} \in \text{input\_o}}$  (class.w)
32: ForEach class | class  $\in$  input_o && class.w = highestWeight Do
33:   /* Increase the weight of the subclasses and the related classes of the
      highest-weight-class(es) */
34:   setWeightToClassAndSubClasses(class, highestWeight * input_k)
35:    $\forall \text{relClasses} \in \text{hasRelation}(c)$ : setWeightToClassAndSubClasses(relClasses,
      highestWeight * input_k)
36: EndFor
37: EndIf
38: /* Normalise the weights */
39: totalWeight =  $\sum_{\text{class} \in \text{input\_o}} \text{class.w}$ 
40:  $\forall \text{class} \in \text{input\_o}$ : class.w = class.w / totalWeight

end

```

Algorithm 2 *setWeightToClassAndSubClasses(input_c: class, input_w: weight)*

```

1: If input_c.w < input_w Then
2:   input_c.w = input_w
3: EndIf
4: ForEach class  $\in$  subClass(input_c) Do
5:   setWeightToClassAndSubClasses(class, input_w)
6: EndFor

end

```

The direct match (lines 7–17 in algorithm 1) is mostly relevant in the initial phase of the software usage because the direct match alleviates the problem of the cold start. The reason for this is that the ontology holds very few individuals in this phase and thus the algorithm has rather limited background knowledge. The algorithm must make use of the class names in the ontology.

If there is not a direct match for a given natural language activity description, the matchmaker will try to find out the related category by using the individuals (lines 18–30 in algorithm 1). Figure 4 illustrates an ontology with individuals.

Assumed that somebody enters the natural language activity description “I need to buy some modern clothes from H&M”. At first, all stop words of the description are deleted. So the cleaned description could be “Need buy modern clothes H&M”. Then the algorithm checks if a direct match is possible. Considering the direct match definition (definition 12), there is no direct match in the ontology in figure 4.

The next step is to start the individual based matching: It assigns weights to potential matching classes. If a word in the description is a substring of a word in an individual or vice versa, the class weight of the individual’s class is increased by one. This is done for each word in the description and for each word in the name of every individual. After finishing this process, all potentially

relevant classes have a weight. The result of this process can be seen in figure 5. Above each individual there is an addition of 5 numbers. The first number of each addition is one if the first word of the cleaned description (here: “Need”) is a substring of a word in the individual or vice versa, the second number is one if the second word of the cleaned description is a substring of a word in the individual or vice versa and so on. The number above each class is the sum of the sums of its belonging individuals.

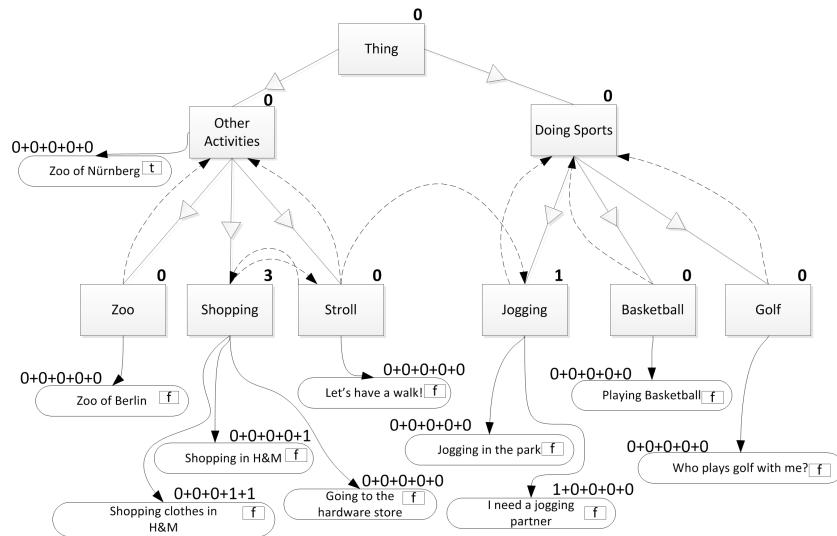


Fig. 5. Ontology with weights for the natural language activity description “I need to buy some modern clothes from H&M” (“has_Relation” connections are not considered)

In the next step, the class(es) that have the highest weight (here: “Shopping”) are in the focus. The weight of all classes and their subclasses that are connected with a “has_Relation” connection from the highest weight classes increase, if possible, their weight to *highest weight * similarity weight*. Considering a similarity weight of $\frac{1}{3}$, the weight of all related classes and their subclasses (here it is just “Stroll”) is increased by one. Besides the weight of all subclasses of the highest weight classes is increased to *highest weight * similarity weight*. In this example, “Shopping”, however, has no subclasses.

Now all classes that are relevant from the system’s view have a weight (Shopping: 3, Stroll: 1, Jogging: 1). In the last step the class weights are normalised. This is done by dividing each class weight by the sum of all class weights ($3 + 1 + 1 = 5$). So the final weights for the classes are: Shopping: 0.6, Stroll: 0.2 and Jogging: 0.2.

Extending the ontology Algorithm 3 is invoked by the presentation layer when a user wants to publish a new activity. It returns class suggestions for a specific activity description. The algorithm considers the strictness factor, see definition 10. Thus only classes with a high weight are returned.

Algorithm 3 *getBestMatchingClasses(input_a: natural language activity description)*

```

1: input_s ← strictness factor
2:
3: classes ← set of classes (empty at the beginning)
4: matchingClasses = getClassesAndWeights(input_a)
5: classesCompleted = false
6: While  $\sum_{class \in classes} class.w \leq input_s$  Do
7:   highestWeight =  $\max_{class \in matchingClasses} (class.w)$ 
8:   ForEach class | class ∈ matchingClasses && class.w = highestWeight
      Do
9:     classes.add(class)
10:    matchingClasses.delete(class)
11:   EndFor
12: EndWhile
13: Return classes
end

```

Algorithm 4 is invoked by the presentation layer in order to subordinate an individual to a specific class. The variable *flag* predicates whether the individual must be further subordinated (true) or if the individual has already been subordinated to a correct class (false).

Algorithm 4 *setIndividual(input_individual: individual to add, input_c: class of individual, input_flag: flag)*

```

1: input_individual.flag = input_flag
2: input_c.addIndividual(input_individual)
end

```

If a user (“U1”) enters a natural language activity description in order to publish an activity and the system cannot return satisfying results, the user can subordinate this description (and in this way an individual) to one top level class, for example to the class “Other Assistances”. Then the flag will be set to “true”. Assumed another user (“U2”) enters, in order to search for an activity, a description that matches the class “Zoo”. It matches the class “Zoo” if U2 chooses this class manually or if algorithm 3 just returns this class. Supposed that U2 contacts U1 by using an internal message service, algorithm 5 is invoked. Because U2’s description belongs to a subclass (“Zoo”) of U1’s “Other Activities”,

the entered individual of U1 is moved to the class “Zoo” and the flag is set to “false”. A formal description of this procedure is described in algorithm 5.

Algorithm 5 *classifyIndividual(input_individual: individual to classify, input_c: classified class)*

```

1: input_i ← set of all individuals in ontology
2:
3: If input_individual.flag = true && input_c ⊑ input_individual.class
   Then
4:   input_i.delete(input_individual)
5:   setIndividual(input_individual, input_c, false)
6: EndIf

end

```

Improving the ontology To increase the probability of a direct match, a possible improvement is to store other words as labels in a class. Accordingly a class could have the name “watching football” and this class has labels like “football stadium”, “Champion’s League” and so on. The algorithm can easily be improved in order to handle labels. Considering the mentioned example, a natural language activity description that contains “football stadium” will lead to a direct match with the class “watching football”. This will definitely improve the matching quality. The creation of labels could be done automatically. An algorithm can be developed that recognises that numerous individuals of a certain class contain a specific substring which cannot be found in the individuals of other classes. Then the algorithm can add the substring automatically as label to the corresponding class. Figure 6 shows such an automated ontology transformation.

7 CONCLUSION AND FUTURE WORK

We proposed a framework for matchmaking for asymmetric support requests and for joint activity requests. This framework constitutes the core of a community-based platform for elderly people. It is part of a larger web-application developed in the interdisciplinary research project EMN-Moves. While asymmetric matchmaking of support requests can be dealt with in a rather simple manner, activity matching relies on a more complex approach based on ontologies.

The proposed ontology-based matching algorithm could also be used as retrieval component in specific settings of analogy making to incorporate background knowledge not only for mapping [?,?] but also for finding a suitable source problem.

Currently, the first prototype of our matchmaking framework is tested in a social service company.

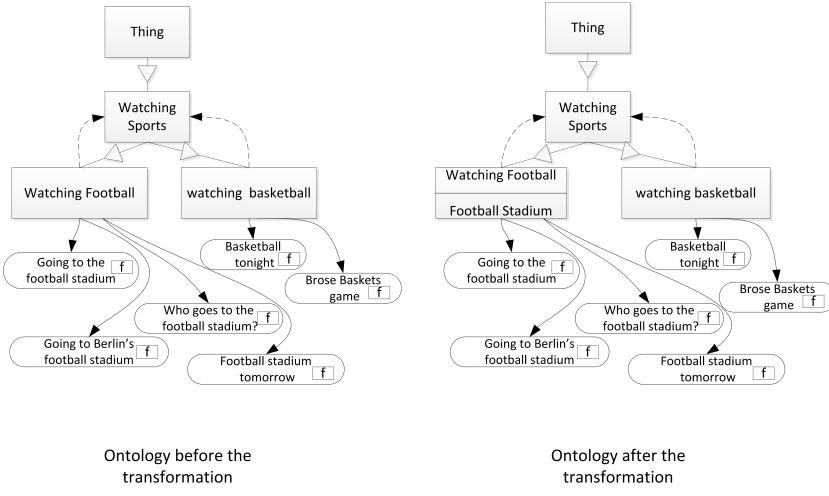


Fig. 6. The automated creation of labels

Acknowledgement. This work is funded by BMBF grant 16SV5700K (Technology and Innovation), Cooperation project “Europäische Metropolregion Nürnberg macht mobil durch technische und soziale Innovationen für die Menschen in der Region” (EMN-MOVES). We thank members of the senior citizen councils of Bamberg, Erlangen, and Nürnberg. We also thank the reviewers for their helpful comments.

References

1. Goldstone, R.L., Son, J.Y.: Similarity. *Psychological Review* **100** (2004) 254–278
2. Gentner, D., Markman, A.: Defining structural similarity. *The Journal of Cognitive Science* **6** (2006) 1–20
3. Schmid, U., Siebers, M., Folger, J., Schineller, S., Seufz, D., Raab, M., Carbon, C.C., Faerber, S.: A cognitive model for predicting esthetical judgements as similarity to dynamic prototypes. *Cognitive Systems Research* **24** (2013) 72–79
4. Gentner, D.: Why we're so smart. In Gentner, D., Goldin-Meadow, S., eds.: *Language in Mind*. MIT Press (2003) 195–235
5. Schmid, U., Wirth, J., Polkeln, K.: A closer look on structural similarity in analogical transfer. *Cognitive Science Quarterly* **3**(1) (2003) 57–89
6. Tversky, A.: Features of similarity. *Psychological Review* **85** (1977) 327–352
7. Thagard, P., Holyoak, K., Nelson, G., Gochfeld, D.: Analogical retrieval by constraint satisfaction. *Artificial Intelligence* **46** (1990) 259–310
8. Forbus, K., Gentner, D., Law, K.: MAC/FAC: A model of similarity-based retrieval. *Cognitive Science* **19**(2) (1995) 141–205
9. Whitty, M.T., Baker, A.J., Inman, J.A., eds.: *Online matchmaking*. Palgrave Macmillan, Basingstoke (2007)

10. Shaw, D., Newson, P., O'Kelley, P., Fulton, W.: Social matching of game players online (2005)
11. Raman, R., Livny, M., Solomon, M.: Matchmaking: Distributed resource management for high throughput computing. In: High Performance Distributed Computing, 1998. Proceedings. The Seventh International Symposium on, IEEE (1998) 140–146
12. Abiteboul, S.: Querying Semi-Structured Data. In: Database Theory - ICDT '97, 6th International Conference, Delphi, Greece, January 8-10, 1997, Proceedings. (1997) 1–18
13. Fenza, G., Loia, V., Senatore, S.: A hybrid approach to semantic web services matchmaking. *International Journal of Approximate Reasoning* **48**(3) (2008) 808–828
14. Miyamoto, S.: Information clustering based on fuzzy multisets. *Inf. Process. Manage.* **39**(2) (March 2003) 195–213
15. Banerjee, N., Chakraborty, D., Dasgupta, K., Mittal, S., Nagar, S., et al.: R-U-In?-exploiting rich presence and converged communications for next-generation activity-oriented social networking. In: Mobile Data Management: Systems, Services and Middleware, 2009. MDM'09. Tenth International Conference on, IEEE (2009) 222–231
16. González-Castillo, J., Trastour, D., Bartolini, C.: Description Logics for Matchmaking of Services. In: IN PROCEEDINGS OF THE KI-2001 WORKSHOP ON APPLICATIONS OF DESCRIPTION LOGICS. (2001)
17. Horrocks, I., Patel-Schneider, P.: Reducing OWL Entailment to Description Logic Satisfiability. In Fensel, D., Sycara, K., Mylopoulos, J., eds.: The Semantic Web - ISWC 2003. Volume 2870 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2003) 17–29
18. Gagliardi, C., Marcellini, F., Papa, R., Giuli, C., Mollenkopf, H.: Associations of personal and mobility resources with subjective well-being among older adults in italy and germany. *Archives of Gerontology and Geriatrics* **50** (2010) 42–47
19. Mollenkopf, H.: Enhancing Mobility in Later Life: Personal Coping, Environmental Resources and Technical Support, the Out-Of-Home Mobility of Older Adults in Urban and Rural Regions of Five European Countries. Ios Press (2005)
20. Scheffer, A.C., Schuurmans, M.J., van Dijk, N., van der Hooft, T., de Rooij, S.E.: Fear of falling: measurement strategy, prevalence, risk factors and consequences among older persons. *Age and Ageing* **37**(1) (2008) 19–24
21. Haustein, S., Kemming, H.: Subjektive Sicherheit von Senioren im Straßenverkehr. *Zeitschrift für Verkehrssicherheit* **54**(3) (2008) 128–133
22. Staab, S.: Handbook on ontologies. Springer Verlag (2009)
23. Genesereth, M., Nilsson, N.: Logical foundations of artificial intelligence. Volume 9. Morgan Kaufmann Los Altos, CA (1987)
24. Daconta, M., Obrst, L., Smith, K.: The Semantic Web: a guide to the future of XML, Web services, and knowledge management. Wiley (2003)

A Computational Strategy for Fractal Analogies in Visual Perception

Keith McGreggor and Ashok Goel

Design & Intelligence Laboratory, School of Interactive Computing
Georgia Institute of Technology, Atlanta, GA 30332, USA
keith.mcgreggor@gatech.edu, goel@cc.gatech.edu

Abstract. A theory of general intelligence must account for how an intelligent agent can map percepts into actions at the level of human performance. We sketch the outline of a new approach to this perception-to-action mapping. Our approach is based on four ideas: the world exhibits fractal self-similarity at multiple scales, the structure of representations reflects the structure of the world, similarity and analogy form the core of intelligence, and fractal representations provide a powerful technique for perceptual similarity and analogy. We divide our argument into three parts. In the first part, we describe the nature of visual analogies and fractal representations. In the second, we illustrate a technique of fractal analogies and show how it gives human-level performance on an intelligence test called the Odd One Out. In the third, we describe how the fractal technique enables the percept-to-action mapping in a simple, simulated world.

1 Introduction

Russell & Norvig (2003) characterize an intelligent agent as a function (f) that maps a perceptual history (P^*) into an action (A). If we accept $f: P^* \rightarrow A$ as a useful characterization of intelligence, it follows that a theory of general intelligence must account for how the intelligent agent maps percepts into actions. Although Russell & Norvig do not delve into it, we believe that a theory of general intelligence must also account for agent's performance at the level of human intelligence. In this chapter, we sketch the outline of a novel approach to addressing the $f: P^* \rightarrow A$ mapping at the level of human intelligence.

Our approach is based on four ideas: (1) the world exhibits fractal self-similarity at multiple scales (Mandelbrot, 1982); (2) the structure of representations reflects at least in part the structure of the world (Haugeland, 1981); (3) similarity and analogy form the core of intelligence (Hofstadter, 1995); and (4) fractal representations provide a powerful technique for similarity and analogy (McGreggor and Goel, 2012). The first three of these ideas are old and familiar in theories of nature and intelligence; however, it is the fourth idea which is new. We claim that analogy initiates with an act of being reminded, and that fractally representing both that triggering percept as well as all prior percepts affords unprecedented similarity discovery, and thereby analogy-making. Additionally, our technique builds fractal representations of

the perceptual stimulus in real-time, instead of assuming the representations as a given.

We divide the argument in this paper into three parts. In the first part, we describe the nature of visual analogies and fractal representations. In the second part, we illustrate the general technique of fractal analogies and show how it gives human-level performance on an intelligence test called the Odd One Out. In the third part, we describe how the same fractal technique enables the $f: P^* \rightarrow A$ mapping in a simple, simulated world, in which intelligent agents recognize one another and flock together.

2 Visual Analogies and Fractal Representations

Suppose we have a visual analogy, expressed symbolically as $A : B :: C : D$, with the symbols representing images, as shown in Figure 1. We can interpret this as suggesting that some operation T exists which captures the relationship between image A and image B (“ A is to B ”). Likewise, some other operation T' is proposed which captures the relationship between image C and image D (“ C is to D ”).

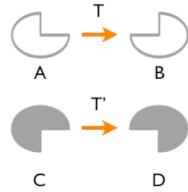


Figure 1. An Example of Visual Analogy.

In this manner, we may see that the central analogy in such a problem rests not with the images themselves, but in the degree to which the two operations T and T' are analogous or similar. We can express the problem to make plain this distinction thusly:

$$A : B :: C : D \rightarrow T(A,B) :: T'(C,D)$$

The nature of this similarity may be determined by a number of means, many of which might associate visual or geometric features to points in a coordinate space, and compute similarity as a distance metric (Tversky, 1977). Tversky developed an alternate approach by considering objects as collections of features, and similarity as a feature-matching process.

2.1 Fractals are Operations Over Images

We adopt Tversky’s interpretation of similarity, and thus seek to express these operations T and T' in some representation which both is robust and affords sufficient feature production to permit Tversky’s feature-matching to occur without reliance on any additional information. A particular nuance of Tversky’s approach, however, is that

either the representation or the features derived from the representation must be formable into sets, as the calculation for similarity employed requires the counting of elements in sets (and their union and intersection). We previously have shown that the fractal representation is one such representation (McGreggor et al. 2011; McGregor and Goel, 2012).

Thus, we may revisit the typical visual analogy $A : B :: C : D$, where T and T' are now fractal representations. To make a comparison between the two representations, we first derive features from each, and then calculate a measure of similarity based upon those features.

We desire a metric of similarity which is normalized, one where the value 0.0 means entirely dissimilar and the value 1.0 means entirely similar. Accordingly, we use the ratio model of similarity as described in (Tversky 1977), wherein the measure of similarity S between the two representations T and T' is calculated thusly:

$$S(T, T') = f(T \cap T') / [f(T \cap T') + \alpha f(T - T') + \beta f(T' - T)]$$

where the operator $f(X)$ denotes the number of features in some set X . The particular sets involved may be considered as indicating, respectively, those features the two operations share ($T \cap T'$), those features in T but not in T' ($T - T'$), and those features in T' but not in T ($T' - T$).

Tversky (1977) notes that the ratio model for matching features generalizes several set-theoretical models of similarity proposed in the psychology literature, depending upon which values one chooses for the weights α and β . To favor features from either T or T' equally and thereby avoid the introduction of bias, we have chosen to set $\alpha = \beta = 1$ (the Jaccard similarity).

2.2 The Mathematical Basis for Fractals as Operations

As we have noted previously (McGreggor et al. 2011a), the mathematical derivation of fractal representation as an operation over images expressly depends upon the notion of real world images, i.e. images that are two dimensional and continuous (Barnsley & Hurd, 1992). We hasten to point out that every image received by the human visual system may be construed as meeting this requirement, with the proviso that the notion of continuity has a resolution limit, and that limit plays an important role in abstraction. Of particular importance are two observations about images themselves, made by Mandelbrot (1982). The first is that all naturally occurring images appear to have similar, repeating patterns. The second observation is that no matter how closely one examines the real world, one may find instances of similar structures and repeating patterns. These observations suggested to Barnsley and Hurd (1992) that it is possible to describe images in terms that capture the observed similarity and repetition alone, without regard to shape or traditional graphical elements.

2.3 Fractal Encoding Algorithm

Computationally, determining fractal representation of an image requires the use of the fractal encoding algorithm. The collage theorem (Barnsley & Hurd, 1992) at the heart of the algorithm can be stated concisely:

For any particular real world image, there exists a finite set of affine transformations which, if applied repeatedly and indefinitely to any other real world image, will result in the convergence of the latter into the former.

It is important to note that the collage theorem is describing a set of transformations which are derived by mapping an image onto itself. In other words, fractal encoding determines an iterated function system which is applied repeated to some source image, with the result that the encoded image emerges. Suppose $F()$ is the fractal encoding of image B. Then, given any other image A:

$$F(A) = A_1, F(A_1) = A_2, F(A_2) = A_3 \dots \text{and so on, until } F(A_\infty) \doteq B$$

$F()$ is itself a finite set of affine transformations T which describe how to modify portions of an image such that convergence is assured. Each affine transformation may affect some or all of the given image, but the union of their actions comprises the resultant image. Thus:

$$F() \equiv T \equiv \{ T_1, T_2, T_3, \dots, T_n \}$$

$$F(A) = T(A) = \bigcup T_i(A)$$

This fractal encoding is dependent not only upon the destination image D, but also upon the source image S, from which the set of affine transformations T are discovered.

Thus, for a target image D and a source image S, the fractal encoding algorithm seeks to discover this particular set of transformations T.

**First, systematically partition D into a set of smaller images, such that
 $D = \{d_1, d_2, d_3, \dots\}$.**

For each image d_i :

- Examine the entire source image S for an equivalent image fragment s_i such that an affine transformation of s_i will likely result in d_i .
- Collect all such transforms into a set of candidates C.
- Select from the set C that transform which most minimally achieves its work, according to some predetermined metric.
- Let T_i be the representation of the chosen transformation associated with d_i .

The set $T = \{T_1, T_2, T_3, \dots\}$ is the fractal encoding of the image D.

Algorithm 1. Fractal Encoding of D from S

The fractal encoding of the transformation from a source image S into a destination image D is tightly coupled with the partitioning scheme P of the destination image D , which may be achieved through a variety of methods. In our present implementation, we merely choose to subdivide D in a regular, gridded fashion. Alternatives could include irregular subdivisions, partitioning according to some inherent colorimetric basis, or levels of detail.

Thus, a stronger specification of the fractal encoding T may be thought of as a function:

$$T(S, D, P) = \{ T_1, T_2, T_3, \dots, T_n \}$$

where the cardinality of the resulting set is determined solely by the partitioning P . That is, each subimage d_i that P extracts from D will be represented by exactly one element of the set T .

2.3.1 Searching and Encoding

The partitioning scheme P extracts a set of smaller images d_i from the destination image D , but this partitioning does not apply to the source image. The entire source image S is examined for a fragment that most closely matches that fragment d_i . The matching is performed by first transforming d_i as described below, and then comparing photometric (or pixel) values. A simple Euclidean distance metric is used for the photometric similarity between the two fragments.

The search over the source image S for a matching fragment is exhaustive, in that each possible correspondence s_i is considered regardless of its prior use in other discovered transforms. By allowing for such reuse, the algorithm ensures the first key fractal observation, the notion of repetition.

2.3.2 Similitude Transformations

For each potential correspondence, the transformation of d_i by a restricted set of similitude transformations is considered. A similitude transformation is a composition of a dilation, orthonormal transformation, and translation. Our implementation presently examines each potential correspondence under eight transformations, specifically dihedral group D_4 , the symmetry group of a square. We fix our dilation at a value of either 1.0 or 0.5, depending upon whether the source and target image are dissimilar or identical, respectively. The translation is found as a consequence of the search algorithm.

| <i>Spatial</i> | | <i>Photometric</i> | |
|----------------|-----------------------------|--------------------|--------------------------|
| s_x, s_y | Source fragment origin | C | Colorimetric contraction |
| d_x, d_y | Destination fragment origin | Op | Colorimetric operation |
| T | Orthonormal transformation | | |
| S | Size/shape of the region | | |

Table 1. Elements of a Fractal Code

2.3.3 Fractal Codes

Once a transformation has been chosen for image fragment d_i , we construct a compact representation of that transformation into a tuple called a fractal code, which contains the spatial and photometric properties necessary to transform a portion of a source image into a portion of the destination image.

2.3.4 Arbitrary selection of source and coding

The choice of source image S is arbitrary. Indeed, the image D can be fractally encoded in terms of itself, by substituting D for S in the algorithm. Although one might expect that this substitution would result in a trivial encoding (in which all fractal codes correspond to an identity transform), this is not the case, a fractal encoding of D will converge upon D regardless of chosen initial image. For this reason, the size of source fragments considered is taken to be twice the dimensional size of the destination fragment, resulting in a contractive affine transform. Similarly, color shifts are made to contract. This contraction, enforced by setting the dilation of spatial transformations at 0.5, provides the second key fractal observation, that similarity and repetition occur at differing scales.

The ordinality of the set of fractal codes which comprise a fractal representation is similarly arbitrary. The partitioning P may be traversed in any order during the matching step of the encoding algorithm. Similarly, once discovered, the individual codes may be applied in any order, so long as all are applied in any particular iteration.

Thus, the fractal encoding algorithm, while computationally expensive in its exhaustive search, represents the relationship between two images (or between an image and itself) as a much smaller set of fractal codes, an instruction set for reconstituting the relationship, with inherently strong spatial and photometric correspondence.

2.4 Features from Fractals

The fractal representation of an image is an unordered set of fractal codes, which compactly describe the geometric alteration and colorization of fragments of the source image that will collage to form the destination image. While it is tempting to treat contiguous subsets of these fractal codes as features, we note that their derivation does not follow strictly Cartesian notions (e.g. adjacent material in the destination might arise from non-adjacent source material). Accordingly, we consider each of these fractal codes independently, and construct candidate fractal features from the individual codes themselves, and not from clusters of codes.

Each fractal code yields a small set of features, formed by constructing subsets of its underlying six-tuple. These features are determined in a fashion to encourage both spatial- and photometric-agnosticism, as well as specificity. Our algorithm creates features from fractal codes by constructing almost all possible subsets of each of the

six members of the fractal code's tuple (at present we ignore singleton sets as well as taking the entire tuple as a set). Thus, in the present implementation of our algorithm, we generate $C(6,2)+C(6,3)+C(6,4)+C(6,5) = 106$ distinct features for each fractal code, where $C(n, m)$ refers to the combination formula ("from n objects, choose m ").

We further chose to represent each feature as a concatenated string in memory. We form these strings by attaching a character tag to each field in the fractal code and then converting that field into string format prior to concatenation, like so:

$$s_x, s_y \text{ (source fragment origin)} \rightarrow Ss_xs_y \text{ (string representation)}$$

The choice of the particular tag is arbitrary, but tagging itself is not: tagging is necessary to avoid in-string matching between the different kinds of fields (e.g. a numerical value may appear in multiple fields of a fractal code). Doing so attributes a world grounding to each field, and collectively to the entire fractal code.

2.5 Mutuality

The analogical relationship between source and destination images may be seen as mutual; that is, the source is to the destination as the destination is to the source. However, the fractal representation is decidedly one-way (e.g. from the source to the destination). To capture the bidirectional, mutual nature of the analogy between source and destination, we now introduce the notion of a mutual fractal representation. Let us label the representation of the fractal transformation from image A to image B as T_{AB} . Correspondingly, we would label the inverse representation as T_{BA} . We shall define the mutual analogical relationship between A and B by the symbol M_{AB} , given by this equation:

$$M_{AB} = T_{AB} \cup T_{BA}$$

By exploiting the set-theoretic nature of fractal representations T_{AB} and T_{BA} to express M_{AB} as a union, we afford the mutual analogical representation the complete expressivity and utility of the fractal representation.

2.6 Extended Mutuality

We note that the mutual fractal representation of the pairings may be employed to determine similar mutual representations of triplets, quadruplets, or larger groupings of images. As a notational convention, we construct these additional representations for triplets (M_{ijk}) and quadruplets (M_{ijkl}) in an analogous manner:

$$M_{ijk} = M_{ij} \cup M_{jk} \cup M_{ik}$$

$$M_{ijkl} = M_{ijk} \cup M_{ikl} \cup M_{jkl} \cup M_{ijl}$$

Thus, in a mutual fractal representation, we have the necessary apparatus for reasoning analogically about the relationships between images, in a manner which is dependent upon only features which describe the mutual visual similarity present in those images.

3 Fractal Analogies and Novelty Detection

To deem some apprehended object as novel involves the complex interplay of at least two relationships (Wagemans et al, 2012a): the relationship between the observer and the observed, and the relationship between the observed and its context. The relationship between the observing agent and the observed object may vary depending upon some act taken by the observer. For example, if one wishes to appreciate an object at a higher level of detail, one might move closer to the object, or bring the object closer, resulting in the object occupying a larger expanse of the observer's field of view. This action modifies the resolution of the object: at differing levels of resolution, fine or coarse details may appear, which may then be taken into the consideration of the novelty of the object. The observed object also is appreciated with regard to other objects in its environment. Comparing an object with others around it may engage making inferences about different orders of relationships. We may begin at a lower order but then proceed to higher orders if needed. The context also sanctions which aspects, qualities, or attitudes of the objects are suitable for comparison.

Given the importance of perceptual novelty detection, there has been quite a bit of work on the topic. Markou & Singh (2003a, 2003b) review statistical and neural network techniques for novelty detection. Neto & Nehmzow (2007) illustrate the use of visual novelty detection in autonomous robots. Work on spatial novelty and oddity by Lovett, Lockwood & Forbus (2008) centered on qualitative relationships in visual matrix reasoning problems. They showed that by applying traditional structure-mapping techniques (Gentner, 1983) to qualitative representations, analogical reasoning may be used to address problems of visual oddity; however, they did not show where the representations come from (Indurkhyia, 1998). More recently, Prade and Gilles (in press) present a logical axiomatic approach to the problem.

Analogies in a general sense are based on similarity and repetition (Hofstadter, 1995), and so we seek to employ a suitable representation, one which affords the capture of these qualities as well as sanctions reasoning over them. Fractals capture self-similarity and repetition at multiple scales (Mandelbrot, 1982). Thus, we believe fractal representations to be an appropriate choice for addressing some classes of analogy problems.

The strategy we employ addresses both aspects of novelty detection we described above. We model the relationship between the observer and the observed by starting with fractal representations encoded at a coarse level of resolution, and then adjusting to the right level of resolution for addressing the given problem. We model the relationship between the observed and its context by searching for similarity between simpler relationships, and then shifting its searches for similarity between higher-order relationships. In each aspect, these adjustments are made automatically by our strategy, by characterizing the ambiguity of a potential solution.

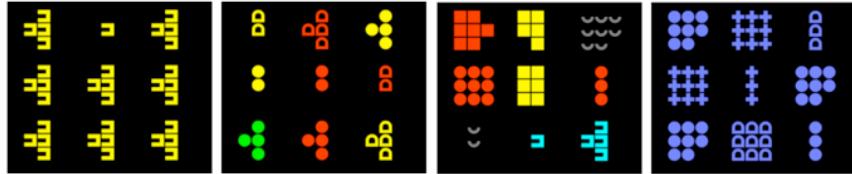


Figure 2. Odd One Out problems.

3.1 Odd One Out Problems

General one-one-out (or odd-man-out) tasks can be presented with many kinds of stimuli, from words, colors, and images, to sets of objects. Minimal versions of these tasks are presented with three items, from which the “odd” one must be selected. Three item one-one-out tasks, in contrast to two-item response tasks, evaluate a participant’s ability to compare relationships among stimuli, as opposed to just comparing stimuli features. It has been shown that these relationship-comparison tasks track general IQ measure more closely than do two-item tasks, and this tracking of IQ increases with the number of relationships to be considered (Diascro and Brody, 1994).

In our present work, we have chosen the Odd One Out test developed by Hampshire and colleagues at Cambridge Brain Sciences (Hampshire, 2010). This particular test consists of almost 3,000 3x3 matrix reasoning problems organized in 20 levels of difficulty, in which the task is to decide which of the nine abstract figures in the matrix does not belong (the so-called “Odd One Out”). Figure 2 shows a sampling of the problems, illustrating the nature of the task, and several levels of difficulty.

We note that the matrix like arrangement in the Odd One Out problem is arbitrary; that is, the “Odd One Out” is odd no matter the configuration. We present the problems in that arrangement because the original data set had them so.

3.2 A General Strategy for Odd One Out

A general strategy for determining which figure doesn’t belong begins with the initial segmentation of the given image into the constituent figures. Each of these figures must be somehow represented in a fashion which affords its comparison to its fellow figures. Next, reflection occurs over those comparisons, to see if one of the figures’ comparisons might be exceptional. If no one figure thus stands out, perhaps reexamining some or all of the figures might be in order. Upon the conclusion of this iteration of reflection and reexamination, the Odd One Out may be indicated.

In this manner then, the keys to solving such a problem lie in addressing these issues:

- how the figures and their relationships may be represented;
- what constitutes an exceptional comparison; and,
- how might the figures and their relationships be reexamined.

3.3 Finding the Odd One Out, Fractally

Our algorithm for tackling the Odd One Out problems consists of three phases: segmentation, representation, and reasoning. We shall illustrate our technique by working through an example problem, shown in Figure 3.

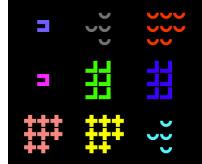


Figure 3. Example of an Odd One Out problem

3.3.1 Segmentation Phase

First, we must segment the problem image into its nine constituent subimages, which we shall label I_1 through I_9 . In the present implementation, the problems are given as a 478x405 pixel JPEG image, in the RGB color space. The subimages are arrayed in a 3x3 grid within the problem image. At this resolution, we have found that each subimage fits well within a 96x96 pixel image, as may be seen in Figure 4.

Note that even though these subimages appear to contain regular geometric shapes, we are not interpreting them as such, and due to the nature of the JPEG compression algorithm, each subimage contains a certain quantity of noise and image artifacts. We do not process these images in any fashion to remove these artifacts: we address the pixels as received.

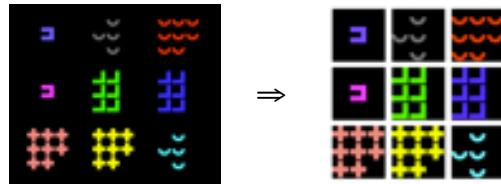


Figure 4. Segmentation of an Odd One Out problem

3.3.2 Representation Phase

Given these nine subimages, we group subimages into pairs, such that each subimage is paired once with the other eight subimages, forming 36 distinct pairings. We then calculate the mutual fractal representation M_{ij} for each pair of subimages I_i and I_j , as described above.

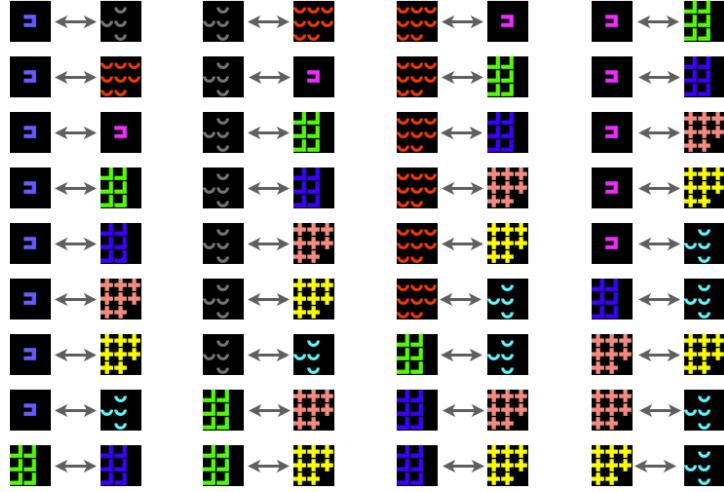


Figure 5. The 36 pair-wise relationships to be represented fractally

The block partitioning we use initially is identical to the largest possible block size (in this case, 96x96 pixels). In the present implementation, we conduct the finer partitioning by uniform subdivision of the images into block sizes of 48x48, 24x24, 12x12, 6x6, and 3x3 pixels.

3.3.3 Reasoning Phase

To determine the Odd One Out, we must determine how dissimilar each subimage is from its fellows. As we have placed each of the nine subimages into a variety of fractal representations, we must first determine how similar each of these representations is to all of the others, and then distribute that similarity to the subimages.

To calculate a measure of similarity, we use the Tversky metric described above, which provides a comparison of the number of fractal features shared between each pair member (Tversky, 1977). To favor features from either image equally, we choose to set the weights α and β equal to unity. Thus, we calculate similarity as a Jaccard similarity:

$$S(A,B) = f(A \cap B) / f(A \cup B)$$

where $f(X)$ denotes the number of features in the set X .

3.3.3.1 Relationship Space

As we perform this calculation for each pair A and B taken from the grouping, we determine a set of similarity values for each member of this collection of fractal representations. We consider the similarity of each analogical relationship as a value

upon an axis in a large “relationship space” whose dimensionality is determined by the size of the collection: for pairings of images, the space is 36 dimensional. For collections of relationships formed between triplets of images, the space would be 84 dimensional; for relationships formed between quadruplets of images, the space is 126 dimensional.

3.3.3.2 Maximal Similarity as Distance

To arrive at a scalar similarity score for each member of the collection, we construct a vector in this multidimensional relationship space and determine its length, using a Euclidean distance formula. The longer the vector, the more similar two members are; the shorter the vector, the more dissimilar two members are. As the Odd One Out problem seeks to determine, literally, “the odd one out,” we seek to find the shortest vector, as an indicator of dissimilarity.

3.3.3.3 Distribution of Similarity

From the similarity score for a member of the given collection of fractal representations, we determine subimage scoring by distributing the similarity value equally among the participating subimages. For each of the nine subimages, a score is generated which is proportional to its participation in the grouping. If a subimage is one of the two images in a pairing, as an example, then the subimage’s similarity score receives one half of the pairing’s calculated similarity score. Once all similarity scores of the grouping have been distributed to the subimages, the similarity score for each subimage is known. Algorithm 2 provides an overview of this similarity distribution.

Given a set of 9 subimages $I = \{ I_1, \dots, I_9 \}$ and a set of N representations $R = \{ R_1, \dots, R_n \}$, where each R_i is the mutual fractal representation between two (or more) images.

Let Q be a vector of cardinality 9, initialized to 0: $Q \leftarrow 0, |Q| = 9$

For each representation $R_i \in R$:

- Let S be an vector of cardinality N , initialized to 0: $S \leftarrow 0, |S| = N$
- **For each representation $R_k \in R$:**
 - If $i = k$, then $S_k = 1 \because R_i$ is identical to itself
 - If $i \neq k$, then calculate S_k using the Tversky formula:

$$S_k \leftarrow f(R_i \cap R_k) / [f(R_i \cap R_k) + \alpha f(R_i - R_k) + \beta f(R_k - R_i)]$$
- **Let V be a scalar value, set to the normalized magnitude of S : $V \leftarrow \|S\| / |S|$**
- **For each subimage I_j which is represented by R_i :**
 - $Q_j \leftarrow Q_j + V$

The vector Q then contains the distributed similarity of each subimage to one another.

Algorithm 2. Similarity distribution

Using our example problem, and following this algorithm, we can derive these distributed similarity values for the subimages, shown in Table 2.

| | | | | | | | | |
|---|---|---|---|---|---|--|---|---|
|  |  |  |  |  |  |  |  |  |
| 21.301 | 20.206 | 20.384 | 21.301 | 21.985 | 21.814 | 20.639 | 20.632 | 20.198 |

Table 2. Example similarity distribution for the initial 96x96 partition

3.3.4 Concluding the example

Once that distribution is accomplished, we examine the resulting similarity values to see if any subimage has a substantially lower score than the others. A tempting way to accomplish this would be to simply select the subimage with the lowest similarity distribution, but as this example illustrates, the distinction between the lowest and next-lowest score may be quite fine, and it is difficult to determine whether that distinction is substantial enough to warrant the decision.

The way we make this assessment is to calculate the mean of the similarity values and the standard deviation of each of the scores from this mean, and then check whether any of these standard deviations falls below the mean sufficiently to indicate a desired confidence interval (assuming a normal distribution). For this example, at a 96x96 partition, the mean score is 20.94 and the standard deviation is 0.68. If we desire a confidence of 90%, then some score must be less than 20.15 to be sufficiently “odd.” At this partitioning, none of the subimages manage to achieve this level. If we relax our confidence to 80%, then the score must be less than 20.24, a value which two of the subimages are below.

3.4 Ambiguity, Abstraction, and Refinement

As our example illustrates, we have found it not quite so simple to select the Odd One Out, as ambiguity may be present. Let us now be much more precise.

3.4.1 Ambiguity

Similarity scores for subimages may vary widely. If the score for any subimage is “unambiguously smaller” than that of any other subimage, then we may deem that subimage to be the Odd One Out. By unambiguous, we mean that there is no more than one score which is less than some ϵ , which we may

vary as a tuning mechanism for the algorithm. We see this as a useful yet coarse approximation of the boundary between the similar and the dissimilar in feature space. As we have shown, one way to characterize ϵ is as a confidence interval, and indeed we may choose this value arbitrarily. We believe it to be rigorous to report the Odd One Out as “subimage X, with Y% confidence.” In our implementation, we use 90% as our de-facto confidence interval.

How might we characterize these ambiguities? If no single subimage’s similarity value is sufficiently lower than the mean to fall below our confidence threshold, then perhaps the value itself is derived from a set of data which is too homogenous or too sparse. If more than one subimage’s similarity value meets the criteria, then we may also remark that the data used was too sparse or too consistent.

3.4.2 Abstraction

We argue that the ambiguity arises due to a data problem, but it is more: it is a problem with the representation itself, from whence the data arise. If the data is sparse, we can create more of it; if it is too homogenous, we can change how we create the data, potentially affording variance.

Since we are performing reasoning afforded by the fractal representation of the relationship between subimages, we are limited in our mechanisms to those which the representation sanctions. There are two primary sanctions of the representation: the number of fractal codes which constitute the representation, and the creation of features from those fractal codes.

The number of fractal codes in a particular fractal representation is determined solely by the partitioning scheme chosen when constructing the representation. The twin key observations of images which entailed fractal encoding (repetition and similarity at different scales) may be exploited here. In essence, partitioning is a modeling of how coarsely or finely we receive or regard an image, and that granularity determines the algorithm’s ability to capture within the representation any present repetition or inherent similarity at that limit. The partitioning affords a level of visual abstraction. Figure 6 illustrates how changes in partition may be thus interpreted.

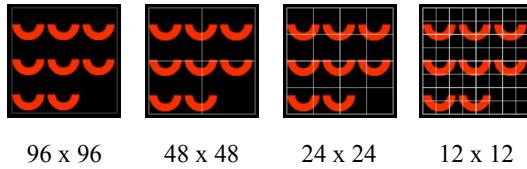


Figure 6. Visual abstraction as partitioning

Increasing the partitioning accomplishes two acts: more fractal codes are created, and the possible variety of features arising from those codes increases. Both of these may address the ambiguity we illustrated in the data.

3.4.3 Refinement

Let us revisit our example, and illustrate the effect of partitioning as abstraction. We redetermine the fractal representation at partitioning levels of 96x96, 48x48, 24x24, 12x12, 6x6, and 3x3 pixels successively. After each partitioning, we run Algorithm 2, and determine the similarity distribution and the attendant confidence scores. Table 3 illustrates the result of these iterations.

| 96x96 | 0.404 | -0.72 | -0.59 | 0.404 | 0.876 | 0.801 | -0.34 | -0.35 | -0.73 |
|-------|-------|-------|--------------|-------|-------|-------|-------|-------|-------|
| 48x48 | 0.840 | 0.304 | -0.74 | 0.820 | -0.48 | -0.24 | -0.74 | -0.51 | 0.546 |
| 24x24 | 0.731 | 0.307 | -0.91 | 0.853 | -0 | 0.202 | -0.65 | -0.58 | 0.182 |
| 12x12 | 0.780 | 0.506 | -0.66 | 0.810 | -0.04 | 0.026 | -0.78 | -0.82 | 0.246 |
| 6x6 | 0.770 | 0.635 | -0.27 | 0.722 | -0.44 | 0.033 | -0.81 | -0.86 | 0.366 |
| 3x3 | 0.742 | 0.645 | -0.02 | 0.687 | -0.46 | -0.14 | -0.85 | -0.85 | 0.458 |

Table 3. Confidence scores for various partitions

There is one value, at a partitioning of 24x24, which yields a single answer whose confidence value exceeds 90%. The Odd One Out for this example is therefore subimage 3.

A closer examination of this table reveals further nuances in abstraction. As the partitioning becomes finer, there is a moment at which the ambiguity is resolved. However, as the partitioning surpasses that point, the answer becomes ambiguous once again. In this example, other subimages arise as potential candidates (note subimages 7 and 8 at 6x6 and 3x3 in particular), but none exceed the confidence threshold. As the resolution reaches its limit (for our purposes, the 3x3 partitioning), there are two candidate answers, and thus ambiguity remains, even though the confidence for either candidate approaches 85%.

3.4.4 Ambiguity, Fractal Representations, and the First Aspect of Visual Perception

As the level of abstraction becomes finer, the number of fractal codes, and thereby the number of features, rises. It is reasonable to presume that not all of these features will be unique at any particular level. At 96x96, there is but a single fractal code per subimage, and 106 features. At 24x24, there are 16 fractal codes (1,696 features). At the finest level, there are 1,024 fractal codes (108,544 features). Why is it that ambiguity appears to resolve at a certain level of granularity, only to retreat at others?

As resolution increases, the fractal codes which represent areas in the subimage are covering ever smaller areas. These areas become increasingly more homogenous, and therefore the fractal codes become more similar to one another (that is, their features become more consistent). As the abstraction grows finer, more codes are devoted to representing areas of consistent color and texture. Even though the number of codes and features is increasing, the ability to discriminate based on features is decreasing. We believe this equates to a frequency apprehension of the image, with coarse resolution corresponding to low frequencies (fundamentals), and fine resolution corresponding to high frequencies (overtones, and then noise).

Thus, the disappearance and reemergence of ambiguity is an emergent characteristic of the fractal representation itself. The strategy to determine novelty, the Odd One Out, is determined solely by data arising from reasoning sanctioned by the representation. In doing so, this strategy expresses the first aspect of visual perception: the relationship between the observer and the observed.

3.4.5 Ambiguity, Extended Mutuality, and the Second Aspect of Visual Perception

We have shown, through this example, that ambiguity may be resolved through repartitioning, and that a strategy may be derived which notices the need for repartitioning in an automatic fashion. There exists a case which bears brief further discussion: what if every level of detail or repartitioning results in continued ambiguity?

We believe that a second strategy is to use not just pairs of subimages in the calculation of similarity, but to extend that grouping to triplets or even quadruplets of subimages. Through the use of extended mutual fractal representations as proscribed above, the algorithm for similarity distribution readily extends to accommodate any degree of groupings of subimages without loss of generality or modification.

It is this second strategy, of extended mutuality, which captures the second aspect of visual perception: the relationship between the observed and its

context. Like the first strategy, this strategy also follows directly as an emergent consequence of the fractal representation.

3.5 Analysis and Discussion

As we previously reported (McGregor & Goel 2011a), we have run our similarity distribution algorithm and abstraction refinement strategy against 2,976 problems of the Odd One Out. These problems span a range of difficulty in 20 levels, from the very easiest up to the most difficult. For example, the problem in Figure 4 which we use to illustrate our algorithm is a level 16 problem.

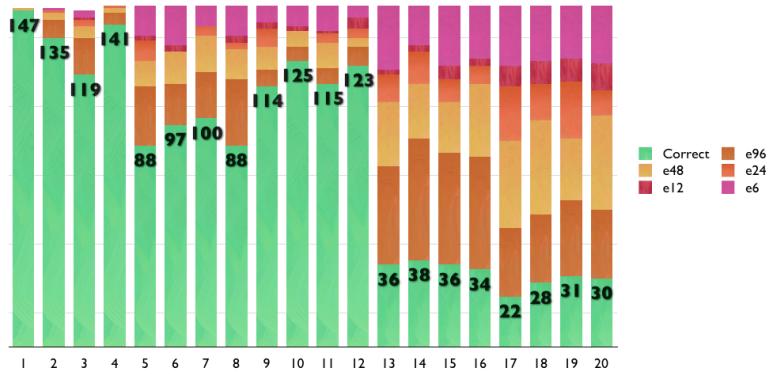


Chart 1. Performance of the similarity distribution algorithm

In those initial runs, we restricted the partitioning to progress from a coarse partition of 96x96 down to a fine partitioning of 6x6 blocks. We were focused only on testing partition shifting. Chart 1 illustrates the overall performance of the algorithm, with 1,647 answered correctly, and subsequent error patterns for the various partitionings.

These results led us to reexamine that data in light of abstraction. From the data, we developed the theory of abstraction emergence that we present for this first time in this paper, as outlined in the previous section. In addition, the previous analysis of errors made led us to the conclusion that too coarse a level of abstraction may lead to reasoning errors, a result we now realize (and argue above) is attributable to too sparse or too homogenous a set of features. Lastly, errors which occurred at the finest level of partitioning we now know are attributable to ambiguity, and this led us to develop and refine the use of extended mutuality in the similarity distribution algorithm, which gave rise to satisfaction of the second aspect of visual perception.

This work is ongoing, and future work centers on the interplay between the first and second strategies. We believe that it is practical, and perhaps even desirable, to explore both strategies in parallel, and allow the first unambiguous result from either strategy to be deemed the Odd One Out. Other activities we are pursuing involve the use of the onset or the width of range of unambiguous partitionings as a mechanism

for characterizing the ease with which a human might solve such problems: a wide range of successful partitionings might suggest an easy problem, but a narrow range, or the onset of such a range at a fine partitioning might suggest that the problem would be considered difficult.

4 Fractal Perception and Action

We have introduced fractal analogies and demonstrated how fractal analogies lead to human-level performance on the Odd One Out test of intelligence. In order to demonstrate that fractal analogies may form the basis of a theory of general intelligence, we need to describe how they can address the $f: P^* \rightarrow A$ mapping. To illustrate this we will construct an intelligent agent that lives in a simple simulated world similar to Reynolds's (1987, 1999) boid worlds.

4.1 The Familiar and the Novel

Among the variety of processes, there are two aspects of visual reasoning that seem especially powerful: the ability to recognize the familiar, and to notice novelty.

Consider that childhood ritual known as the Easter Egg hunt. How straightforward a task, to find those eggs hidden among the grasses and bushes, and yet, how remarkably difficult would it be to program a robot (even a remarkable one) to do so. The incoming visual signal, the messy and ever-shifting scene, confounded by the whirlwind of other children engaged in their own hunt, would be very complex indeed. Still, from this chaotic signal, children are able quite readily to pick out the elusive eggs. They are familiar with what an egg looks like, and notice the novelty of an egg amongst the grasses.

Novelty and familiarity are related and intertwined ideas—almost, but not quite purely opposite sides of the same mental coin. For one might be very familiar with some visual object, yet one may not consider it to be novel unless that object is encountered at a time when one least expects it. Novelty implies a context in which the visual signal is appraised; familiarity does not necessarily suggest this. As an example, one may be entirely familiar with what an egg looks like, but that egg would be otherwise unremarkable and lack novelty without some context.

It is within the implicit contextualization of novelty that we find the bridge to that central core of cognition, the ability to make analogies. Our experiences provide a rich, ever changing context in which to situate, to compare, and to remember the infalling visual world. This textural lexicon is the structure unto which we lay the newly arriving world for judgment. Something we regard as familiar (or rather, similar, or analogous) must agree, in some sufficient number of aspects or ways or degrees, to that expectant tapestry. For something to be novel, though, we need only note a single aspect or way or degree that doesn't match. Finding similarities within what we see is a method by which we make analogies; noticing novelty is a way by which we break analogies, and make newer ones.

4.2 Representing a Complex World

A distinction must be drawn between what the world is and what the world affords. Some object in the world may be labelled as novel by a particular observer, but that is not sufficient to suggest that the object in question would be labelled as novel by every conceivable observer. Novelty depends upon context, and each observer's context will vary.

Similarly, while the world is continuous, it does not directly offer a notion of an appropriate level of abstraction, merely offering an opportunity for an observer to receive the world in differing manners through some enactment of the observer upon or within the world (changing the nature of the light which falls upon an object or manipulating the object somehow) or through some modification of the observer as an entity within the world (moving closer or further to an object, or changing the visual system mechanically via squinting, and the like).

4.2.1 Requirements

The acts of noting novelty and recognizing the familiar are cognitive acts which occur entirely within the mind of the observer. The world affords them, but it is the observer that performs them. That is, some set of cognitive processes occurs within the observer to accomplish these feats. These mental acts are available to be performed because the mind somehow must possess a sufficient representation of the received world which affords them.

4.2.2 Representations

In their 1993 paper, Davis, Shrobe, and Szolovits note that representations play five distinct, critical roles:

- as a surrogate;
- as a set of ontological commitments;
- as a fragmentary theory of reasoning;
- as a medium for efficient computation; and
- as a medium of expression.

Each of these aspects matters when regarding visual reasoning. The fidelity of the correspondence between the representation as surrogate and the received world affects and informs the possible levels of abstraction. The ontological commitment of what within the received signal to represent (and what to leave out) contribute to the constraints the representation may impose upon reasoning. The fragmentary reasoning that a representation affords stems from what inferencing it allows, and how that set of allowed inferences may be constrained. The guidance a representation gives for computation arises from its role as an organizational mechanism for the corresponding received information, and reflects upon the adequacy with which that information is captured. The utility of the representation for communicating information directly affects our ability to mix new data with old into newer data, and provides the way in which comparison arises.

4.3 Agents of a World

While we may not be able to ascertain the workings of visual reasoning by direct interrogation, we may observe the interaction of humans and animals as they interact with each other and their environment. We might construct artificial agents, endow them with our models of such reasoning, place them into virtual worlds, and observe the correlation of their acts with their reality companions.

In nature, highly complex interactions between agents are common. Murmurations of starlings, schools of fish, and stampedes of wildebeest are at once stunning and remarkable in appearance. Even though these groups are made up of discrete individuals, the overall group splits and combines with extraordinary fluidity and grace. The collection of agents, taken together, appear to be acting under some organized control system. Yet, as Craig Reynolds, a pioneer in computer graphic flocking observes, “all evidence indicates that flock motion must be merely the aggregate result of the actions of individual animals, each acting solely on the basis of its own local perception of the world.” (Reynolds, 1987) He reinforces the distinction between his work on boids and prior particle system research by remarking that to flock realistically, boids (and birds) must interact strongly with one another, and rely computationally upon both an internal state and a received external state. A simulation of flocking consists of having each agent adjust itself (modulated by internal and external state) and then rendering each agent in the simulated environment.

Reynolds' work in flocking has inspired a generation of computer graphics artists simulating natural flocking (Tu and Terzopoulos, 1994), including some of the most stunning examples ever presented on film (Allers and Minkoff, 1994; Jackson, 2003). The initial work has been extended to provide mimicry of other natural, commonplace steering mechanisms (Reynolds 1999). In each of these systems, however, Reynolds' initial proscription for how agents interpret their environment has remained essentially intact. We now briefly introduce this proscription, as a prelude to our departure from it.

4.4 Boids, and the Three Laws of Flocking

Reynolds' boids are agents with an internal state which describes their current heading (which can be modeled by a velocity vector in two or three dimensions) and an awareness of those agents to whom they should attend (their flock mates). They also have a minimum set of intrinsic behaviors that drive them to coordinate their actions with those flock mates.

As shown in figure 7, the minimum set of behaviors required to produce realistic facsimiles of flocks in nature are three: stay close together, don't collide, and mimic the motion of others.

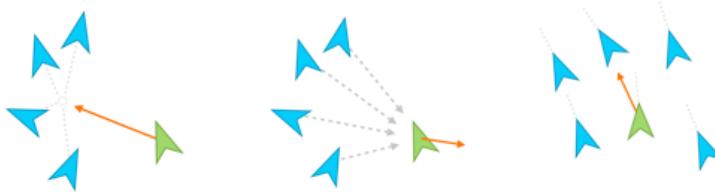


Figure 7. Flocking Behaviors: Cohesion, Separation, and Alignment

4.4.1 Cohesion

Flocking animals appear to want to be close to others like themselves. In simulations, this is achieved by calculating a centroid of the apparent position of flock mates, and adjusting a boid's heading to aim in that direction.

4.4.2 Separation

Animals generally do not want to collide with one another. The separation behavior balances the cohesion behavior by forcing a boid's heading away from the apparent direction of each individual flock mate.

4.4.3 Alignment

Animals mimic one another. A way to provide this mimicry to agents in a flock is to have each agent attempt to match the movement of its flock mates. In practice, this is accomplished by having a boid adjust its heading to align with the aggregate direction of its flock mates.

4.4.4 Interaction

These three behaviors interact with one another in specific ways. The separation behavior affords static collision avoidance, in that the position of flock mates is perceived at every new moment, and thus may be considered static. In contrast, the alignment behavior is dynamic, in that the heading (and not the position) of flock mates is considered. As Reynolds points out, this is a simplified predictive version of collision avoidance, complementary to the separation behavior, in that boids that mimic the motion of their flock mates are less likely to collide with them than would boids which moved freely (Reynolds 1987). The cohesion behavior drives a boid to become the center of its flock, with the urge to move to the center modulated by its distance from the centroid of its mates. This movement to the center is localized, and allows a flock to split around obstacles (or other portions of the flock) with natural fluidity.

4.4.5 Perception

A flock in nature (a murmuration of starlings, for example) may be composed of many thousands of individuals. It would seem an improbable computational load to place upon each agent within the flock the attempt to ascertain aspects of each member of the flock prior to making modifications to its own behavior. Some restriction of which individuals to consider must occur. Reynolds characterizes this as considering each agent to have a local perception. In computer simulations of flocks, the local perception each agent has of the world typically is provided to the agent by a godlike view of the entire environment, and a superimposed restriction of individuals by culling those deemed too distant to consider. This distance is usually referred to as a range of influence.

4.5 The Froid World

The oraclesque decision of whom to consider is only practical in computer simulations. In natural flocks, clearly no such ability is afforded, and each animal must make its decisions based upon some combination of what it is perceiving of or thinking about its world. The choice of flock mate is crucial for the remaining behavior to succeed.

For explorations of visual reasoning, affording agents with models of perception based on familiarity and novelty and observing those agents as flocks seems ideal. Prior research has employed fractal representations to model and discover similarity and novelty in visual analogy tasks such as intelligence tests (references omitted). In our system, we wish to endow our agents with a visual reasoning apparatus that embodies precisely these characteristics. Thus, for our flocking simulation, we created agents possessed with the ability to receive their local environment by localized observation only, and to perceive this received world via manipulations of fractal representations. We call our agents **froids** (“fractal boids”).

4.6 Froids versus Boids

The difference between our froids and typical Reynolds boids is twofold: froids sense and then classify their environment, whereas boids are told explicitly about their surrounds. Both boids and froids manifest the same behaviors, and thus participate in flocking with their mates, but only froids perceive and reason about their environment prior to enacting those behaviors. Figure 8 illustrates the visual reasoning pipeline of a froid, from the reception of the world, through perceiving individuals and objects in the world, reasoning about those perceptions, and finally to enacting a decided upon course of action.

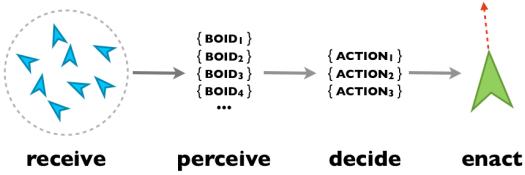


Figure 8. Visual reasoning pipeline

The perceptive system of a froid must be computationally efficient, to permit sufficient time to select and enact a behavior based upon the arriving stimulus. In animals, the perception system, while informed by the decision and control system, appears to operate concurrently with those systems, providing a real-time appraisal of a continuously shifting world. For the purposes of our experiment, and in the general case for systems based upon Reynolds' boids, the simulation of the agents within the world proceeds in discrete steps. Thus, the available stimuli from the world changes at a known pace, and we need not provide for parallel processing with the visual reasoning pipeline, nor for the need to interrupt an action in process to accommodate new information.

We therefore made two simplifying architectural decisions for our initial experiment. First, the perception stage occurs in a serial fashion with the behavior decision stage, since the world of the simulation will not have changed until all the agents have moved themselves. Second, the perception stage would act only upon newly arriving stimuli, and not be influenced by prior decisions. This variety of architecture is deemed "reactive control" in robotics (Arkin 1998) and "situated action" in cognitive science (Norman 1993). We make these simplifications so that we may better compare the effect of perception on the subsequent behavior, without having our analysis take into account any perceptual hysteresis or other internal state.

We now shall describe each stage of the visual reasoning pipeline in some detail.

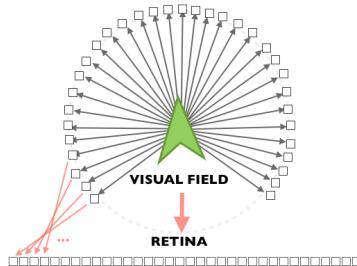


Figure 9. Visual field to retina mapping

4.7 How a Froid Sees

We image a froid as having a single “eye” with a broad field of view. The froid’s eye consists of a simulated retina, an arrangement of sensors. A froid sees its environment by receiving photometric stimulation upon this retina. The light entering each of these sensors is combined to form a visual field, as shown in figure 8. In our simulation, we use ray-casting to send a ray out through each of the sensors into the simulated world, and note whether that ray intersects anything. We illustrate this in Figure 10.

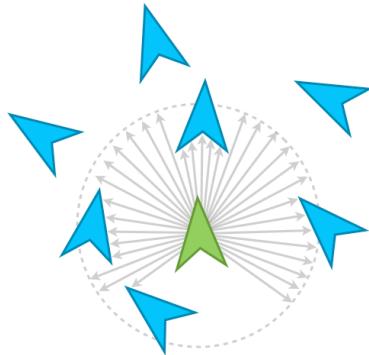


Figure 10. Seeing via ray casting

We interpret the “light” falling upon the sensor is a function of the distance of the intersected object from the froid, where objects which are distant are fainter than close objects. No characterization is made regarding what object has been intersected, only that an intersection has occurred at some distance. Figure 11 shows an example of how objects within the froid’s immediate environment may be mapped by this visual system onto its retina.

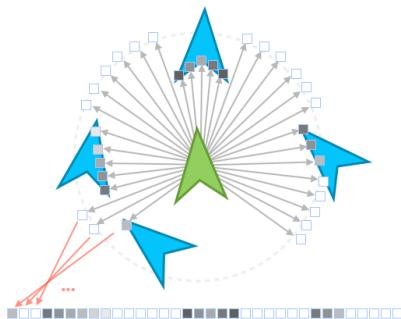


Figure 11. Objects in the environment, retinal image

4.8 Fractal Perception

The photometric values arriving via the froid's retina next are interpreted by the froid's perception stage. While there may be many possible objects one may wish to divine from this visual stimuli, for our present implementation we restrict the intentionality of the perception to only those tasks which will drive the flocking behavior. Accordingly, the primary task of the perception system is to determine flock mates.

This, however, raises an immediate question: what does a flock mate look like to a froid? Our froids are rendered into the simulated environment as chevrons whose orientation, color and physical size may vary. The visual environment, as transduced onto the retinal image, will show only an arranged set of values, roughly corresponding to visual distance to whatever object happened to intersect the ray from the sensor.

4.9 Filial Imprinting

We were inspired to approach this problem using techniques from neurological and biological research. Certain baby animals acquire behaviors from their parents, via a process called filial imprinting. Implicit in the imprinting is the ability to identify a parent. There is evidence that certain species have innate or rapidly develop through acclimation visual prototypes which allow young members to accurately identify their parents (O'Reilly 1994).

There are many possible visual arrangements between a froid and a prototypical "other" in its environment. We chose to restrict our prototypes to six, four corresponding to points on the compass (north, south, east and west), and two corresponding to specific situations which would seem useful for behavior selection (close and empty). Figure 12 illustrates these filial imprints, along with their corresponding retinal impressions. These imprints are given as innate knowledge to each froid.

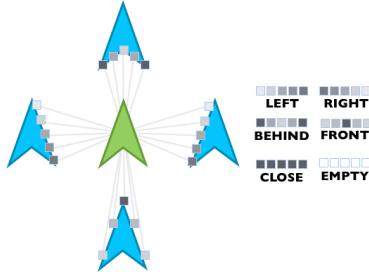


Figure 12. Filial Imprinting

4.10 Fractal Imprinting

A froid encodes all its visual information, its retinal data, as a fractal representation. Accordingly, each imprinted prototype is encoded into a fractal representation, and placed, indexed by derived fractal features, into the froid's memory system. *The technique for transforming the image into a fractal representation and the derivation of fractal features from that representation is the same as in Part I of this paper.*

This imprinting, encoding, and memorization provides each froid with a static knowledge base. From this foundational base, a froid can receive new retinal images and seek within those arriving images what it believes to be familiar.

4.11 Finding the familiar by visual analogy

The arriving retinal image is an otherwise undifferentiated collection of photometric information, with each value corresponding to a particular direction and distance. From this retinal image, flock mates that might be within the visual range of the froid may be identified.

As shown in figure 13, we begin by segmenting the retinal image into varying sets (collections of adjacent sensors), and then encoding each of these segments into fractal representations. We note that no attempt is made to interpret the retina image for edges or other boundary conditions: the segments are treated merely as they are found. Additionally, the segment size itself is arbitrarily chosen. For our experiment, we selected a segment size corresponding to 10 retinal sensors, with the entire retina being 90 sensors in size, encompassing a field of view roughly 135° , oriented to the froid's forward motion. Thus, each retinal image yielded nine segments for analysis.

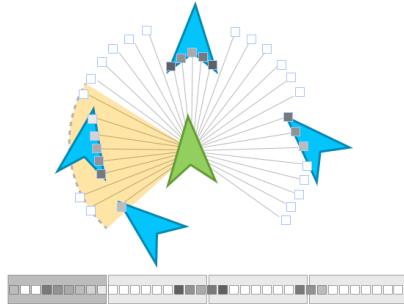


Figure 13. Segmenting the retinal image

The process of perceiving a segment and possibly selecting a familiar prototype is given in Algorithm 3. We first encode the segment into its fractal representation, exploiting its self-similarity. Next, we interrogate the froid's memory for similarity with imprinted prototypes, using a scoring system for featural similarity as described by Tversky (1977). The most similar imprinted prototype is chosen as the interpretation of that segment of the froid's retinal image.

To determine the prototype P' which is most analogous to the retinal segment R from a set of fractal prototypes $P := \{ P_1, P_2, \dots, P_n \}$:

```
F ← Fractal( R, R )
Set M ← 0 and P' ← unknown
For each prototype  $P_i \in P$ :
    · Calculate the similarity of F to  $P_i$  :  $S \leftarrow \text{Sim}( F, P_i )$ 
    · If  $S > M$ , then  $M \leftarrow S$  and  $P' \leftarrow P_i$ 
```

P' is therefore that prototype $P_i \in P$ which corresponds to the maximal similarity S , and is deemed the most analogous to retinal segment R .

Algorithm 3. Selecting the familiar

If a segment appears to correspond to an imprinted prototype (and not to empty space), then we may make several inferences. The first is that an individual flock mate exists in that direction of view, which corresponds to the segment's retinal constituents. Secondly, we may infer that the flock mate lies at a distance which corresponds to a function of the faintness of the photometric readings of that portion of the froid's retinal image. By systematically examining each segment of the retina, the froid's flock mates thus may be inferred by visual analogy.

4.12 The Three Laws for Froids

Once the flock mates have been discovered, the Reynolds rules for flocking may be invoked. Since the perception system has inferred the existence of a flock mate at a particular distance and direction, the **separation** and **cohesion** rules may be enacted directly. However, the **alignment** rule's application requires further inference.

To align with a flock mate, the froid must infer the heading from the visual classification of the mate. This classification depends explicitly upon which of the filial prototypes has been selected as most representative of the retinal segment. Algorithm 4 provides the following five rules of heading inference.

To determine the heading H for an identified flock mate with classification C and apparent direction D :

```
If C = LEFT, then H ← D - 90°
If C = RIGHT, then H ← D + 90°
If C = BEHIND, then H ← D
If C = FRONT, then H ← D + 180°
If C = CLOSE or C = EMPTY, then H ← unknown
```

Algorithm 4. Inferring flock mate heading

Figure 14 shows an example of these inference rules at work. In this example, the retinal image is classified as most similar to the RIGHT filial prototype. The heading of this identified object is inferred to be at a 90° angle to its apparent direction. Note that the partially viewed individual does not sufficiently cover enough retinal space to be identified.

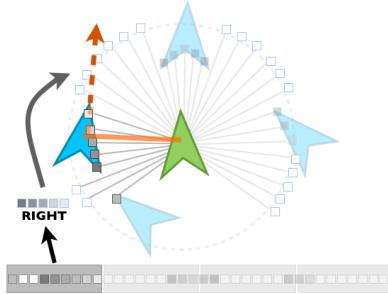


Figure 14. Inferring heading from a retinal segment

Once the heading is inferred, the **alignment** rule of Reynolds may be used to adjust the motion of the froid.

4.13 Froids and Boids

To test our belief that a froid could behave as naturally as its boid counterparts, we created a traditional Reynolds- style boid system, written in Java, running on a conventional computer system. We first placed into the environment several thousand standard boids, and observed that their aggregate motion was as expected: a realistic simulation of natural flocking behavior.

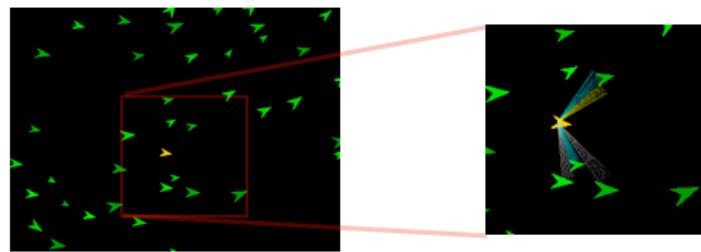


Figure 15. A froid flocks with boids, a closeup of the froid perceiving its environment

We then introduced one froid into the environment with the boids. Figure 15 shows a view of this simulation, with traditional boids in green, and the froid in gold. We observed that the froid, whose identification of flock mates was based solely upon its

fractal perception system, behaved in the same manner as those boids whose identification of flock mates was given in the traditional oracle manner. We subsequently added several more froids into the mix, and found that the overall flocking behavior remained consistent and realistic. Figure 15 also shows a closeup of the froid, in the company of several boids, with its perception system visualized, perceiving and classifying its flock mates.

We note that, unlike the boids, the froids appeared to suffer from uncertainty (manifested by a stuttering motion) when in the proximity of a large number of other boids. We surmised that this is due to the inability of the segmentation system using within the retina to accommodate or otherwise classify large amounts of overlapping or confounding visual data. Another possibility concerns the enactment itself. Let us suppose that two action vectors arising due to two received perceptual signals almost exactly cancel each other. In this case, small fluctuations in the perceptual signal can cause a significant change in the action vector, which may result in stuttering.

5 Conclusion

In earlier work (Davies and Goel, 2001; Davies, Goel, and Yaner, 2008), we showed that visual knowledge and reasoning alone could address some classes of analogy problems that had been assumed to require causal knowledge and reasoning. We also showed how visual analogies could account for several aspects of creative problem solving in scientific discovery (Davies, Nersessian, and Goel, 2005) and engineering design (Davies, Goel, and Nersessian, 2009). However, this work still used propositional representations: while the content of knowledge was visuospatial, the form of representation was propositional.

Previously (Kunda, McGregor, and Goel, 2010a, 2010b, and 2012), we showed how visual knowledge represented iconically can address analogy problems on the Raven's Progressive Matrices test of intelligence. Previously, the visual analogy problems on the Raven's test had been assumed to require propositional representations. The Raven's test also formed the context of our first development of fractal representations for addressing visual analogy problems (McGregor, Kunda and Goel, 2011). The fractal method on the Raven's test performs about as well as typically human teenager. Hertzmann et al (2001) have used a different fractal technique for comparing texture in two images.

In this paper, first we showed that an improved fractal technique can address visual analogy problems on the Odd One Out test of intelligence at the level of most adult humans. Further, the fractal technique imitates two important features of human performance: starting with low-level relationships and moving to higher relationships if and as needed, and automatic adjustment of the level of resolution to resolve ambiguities. We posit that fractal representations are knowledge representations in the sense of Biederman (1987) in that they encode the relationship between non-accidental perceptual constructs within an image. We posit further that fractals are knowledge representations in the deep sense of Davis, Shrobe & Szolovits (1993) in which representation and reasoning are closely intertwined.

Then, in this paper we that showed the fractal technique for visual analogies can be used for perception. We constructed an artificial boid like world that is popular in graphics and games, and demonstrated that froids (fractal-based boids) can use the

fractal technique for mapping percepts into actions in real time and manifest flocking behavior.

While the use of fractal representations is central to our technique, the emphasis upon visual recall in our solution afforded by features derived from those representations is also important. There is evidence that certain species have innate or rapidly develop through acclimation visual prototypes which allow young members to accurately identify their parents (O'Reilly and Johnson, 1994). We hold that placing imprints into memory, indexed via fractal features, affords a new and robust method of discovering image similarity, and that images, encoded and represented in terms of themselves, may be indexed and retrieved without regard to shape, geometry, or symbol. We also hold that the representations of the perceptual stimuli, as in our fractal technique, need to be built at run-time and in real-time.

The relationship of our computational approach to visual analogy using fractal representations and the gestalt view of visual perception in cognitive psychology (Wertheimer 1954; Wagemans 2012a, 2012b) requires examination. The gestalt view of visual perception recently has received attention in computational models of visual analogy (e.g., Schwering et al. 2007). As in gestalt methods in general, our fractal approach to visual analogy constructs different interpretations of the input dynamically and re-represents the problem as needed.

6 Acknowledgements

This work has benefited from many discussions with Maithilee Kunda. We thank the US National Science Foundation for its support of this work through IIS Grant #1116541. We also thank Henri Prade and Gilles Richard for assembling this volume on similarity and analogy in AI and for inviting us to write this chapter for it. An earlier version of this chapter was presented to the Fifth International Conference on Artificial General Intelligence, Oxford, United Kingdom, in December 2012.

7 References

- Allers, R., and Minkoff, R., directors. 1994. *The Lion King*. Walt Disney Pictures.
- Arkin, R. 1998. *Behavior-Based Robotics*. Boston, MA: The MIT Press.
- Barnsley, M., and Hurd, L. 1992. *Fractal Image Compression*. Boston, MA: A.K. Peters.
- Biederman, I. 1987. Recognition-by-Components: A Theory of Human Image Understanding. *Psychological Review*, 94, 115-147.
- Davis, R. Shrobe H., and Szolovits, P. 1993. What is a Knowledge Representation? *AI Magazine* 14.1:17-33.
- Davies, J., and Goel, A. 2001 Visual Analogy in Problem Solving. In *Proc. 17th International Joint Conference on Artificial Intelligence* (IJCAI-01), August 2001, pp. 377-382. Morgan Kaufmann.
- Davies, J., Goel, A., & Yaner, P. 2008. Proteus: Visuospatial Analogy in Problem Solving. *Knowledge-Based Systems* 21(7):636-654, October 2008.
- Davies, J., Goel, J., & Nersessian, N. 2009. A Computational Model of Visual Analogies in Design. *Cognitive Systems Research*, 10:204-215.

- Davies, J., Nersessian, N., & Goel, A. 2005 Visual Models in Analogical Problem Solving. *Foundations of Science*, 10(1):133-152, 2005.
- Diascro, M. N., & Brody, N. 1994. Odd-man-out and intelligence. *Intelligence*, 19(1), 79-92.
- Gentner, D. 1983. Structure-Mapping: A Theoretical Framework for Analogy. *Cognitive Science* 7(2), 155-170.
- Hampshire, A. 2010. The Odd One Out Test of Intelligence.
<http://www.cambridgebrainsciences.com/browse/reasoning/test/oddoneout>
- Haugeland, J. (editor, 1981) *Mind Design: Philosophy, Psychology and Artificial Intelligence*, MIT Press.
- Hertzmann, A., Jacobs, C. E., Oliver, N., Curless, B., & Salesin, D. 2001. Image analogies, in *Computer Graphics*, 25(4) (*SIGGRAPH '01 Conference Proceedings*): 327-340.
- Hofstadter, D., & Fluid Analogies Research Group (Eds.). 1995. *Fluid concepts & creative analogies: Computer models of the fundamental mechanisms of thought*. New York: Basic Books.
- Indurkhy, B. (1998). On creation of features and change of representation. *Journal of Japanese Cognitive Science Society*, 5(2), 43-56.
- Jackson, P., director. 2003. *Lord of the Rings: Return of the King*. New Line Cinema.
- Kunda, M., McGregor, K. and Goel, A. 2010a. Taking a Look (Literally!) at the Raven's Intelligence Test: Two Visual Solution Strategies. In Proc. 32nd Annual Meeting of the Cognitive Science Society, Portland, August 2010.
- Kunda, M., McGregor, K. and Goel, A. 2010b. Two Visual Strategies for Solving the Raven's Progressive Matrices Intelligence Test. In Proc. Twenty Fifth National Conference on AI (AAAI-2011), San Francisco, August 2011.
- Kunda, M., McGregor, K. and Goel, A. 2013. A computational model for solving problems from the Raven's Progressive Matrices intelligence test using iconic visual representations. *Cognitive Systems Research* 22-23: 47-66.
- Lovett, A., Lockwood, K., & Forbus, K. 2008. Modeling cross-cultural performance on the visual oddity task. *Proceedings of Spatial Cognition* 2008.
- Mandelbrot, B. 1982. *The fractal geometry of nature*. San Francisco: W.H. Freeman.
- Markou, M., & Singh, S. 2003a. Novelty Detection: A Review- Part 1: Statistical Approaches. *Signal Processing*, 83(12): 2481-2497.
- Markou, M., & Singh, S. 2003b. Novelty Detection: A Review- Part 2: Neural Network Based Approaches. *Signal Processing*, 83(12): 2481-2497.
- McGregor, K., and Goel, A. 2012. Fractal Analogies for General Intelligence. In Proc. the Fifth Annual Conference on Artificial General Intelligence, Oxford, UK. December 2012.
- McGregor, K., Kunda, M., and Goel, A. 2011. Fractal Analogies: Preliminary Results from the Raven's Test of Intelligence. In Proc. International Conference on Computational Creativity, Mexico City, Mexico, April 27-29.
- Norman, D. 1993. Cognition in the Head and in the World: An Introduction to the Special Issue on Situated Action. *Cognitive Science* 17:1-6.
- Neto, H. & Nehmzow, U. 2007. Visual Novelty Detection with Automatic Scale Selection. *Robotics and Autonomous Systems*, 55(693-701).
- O'Reilly, Randall C., and Johnson, Mark H., 1994. Object Recognition and Sensitive Periods: A Computational Analysis of Visual Imprinting. *Neural Computation* 6: 357-389.
- Prade, H., & Richard, G. (in press) Picking the one that does not fit - A matter of logical proportions.

- Reynolds, C. W. 1987. Flocks, Herds, and Schools: A Distributed Behavioral Model, in *Computer Graphics*, 21(4) (*SIGGRAPH '87 Conference Proceedings*): 25-34.
- Reynolds, C. W. 1999. Steering Behaviors For Autonomous Characters, in *Proceedings of Game Developers Conference 1999* held in San Jose, California. Miller Freeman Game Group, San Francisco, California. 763-782.
- Russell, S., & Norvig, P. (2003) *Artificial Intelligence: A Modern Approach*. Prentice-Hall.
- Schwinging, A., Krumnack, U., Kühnberger, K. U., & Gust, H. (2007). Using gestalt principles to compute analogies of geometric figures. In *19th Meeting of the Cognitive Science Society (CogSci07)*.
- Tu, X., and Terzopoulos, D. 1994. Artificial fishes: physics, locomotion, perception, behavior. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques (SIGGRAPH '94)*. ACM, New York, NY, USA, 43-50.
- Tversky, A. 1977. Features of similarity. *Psychological Review*, 84(4), 327-352.
- Wagemans, J., Elder, J. H., Kubovy, M., Palmer, S. E., Peterson, M. A., Singh, M., & von der Heydt, R. (2012a). A century of Gestalt psychology in visual perception: I. Perceptual grouping and figure-ground organization. *Psychological Bulletin*: 138(6): 1172-1217.
- Wagemans, J., Feldman, J., Gerpshtain, S., Kimchi, R., Pomerantz, J., van der Helm, P. van Leeuwen, (2012b). A Century of Gestalt Psychology in Visual Perception II. Conceptual and Theoretical Foundations. *Psychological Bulletin*: 138(6): 1218-1252.
- Wertheimer, M. (1959). *Productive thinking*. M. Wertheimer (Ed.). New York: Harper.

An Information-Processing Theory of Interactive Analogical Retrieval

Swaroop S. Vattam & Ashok K. Goel

Design & Intelligence Laboratory, School of Interactive Computing, Georgia Institute of Technology
Atlanta, GA 30308 USA

Abstract: Analogy is said to be ubiquitous in cognition. Since cognition is situated, it follows that analogical reasoning too is situated in external physical, social and informational environments. An essential first step in analogical reasoning is retrieval of a source case appropriate for the target situation. In this chapter, we study analogical retrieval in the context of biologically inspired design in which new technological designs are generated by cross-domain analogies to biological systems. We focus on the phenomenon of *interactive analogical retrieval (IAR)* wherein the source biological cases are obtained through interaction with online information environments. We first provide a description of IAR based on two *in situ* studies of biologically inspired design in an educational setting. We then describe an information-processing theory called PRISM that provides an explanation of IAR. The PRISM theory builds on the Pirolli's (2007) Information Foraging Theory and Thagard et al.'s (1990) computational model of Analogical Retrieval by Constraint Satisfaction. If we take the boundary of the cognitive system in biologically inspired design as including online information environments, then the phenomenon of IAR becomes an important element of understanding the situatedness of analogical reasoning. By folding in interactions with external information environments, PRISM may provide a starting point for developing a general information-processing theory of situated analogy.

1. Introduction

Analogy is said to be ubiquitous in human cognition (Hofstadter 2001; Hofstadter & Sander 2013). Thus analogical reasoning has received much attention in both cognitive science and AI (e.g., Burstein 1986; Carbonell 1986; Chalmers, French & Hofstadter 1992; Clement 2008; Davies, Goel & Nersessian 2009; Davies, Goel & Yaner 2008; Falkenhainer, Forbus & Gentner 1989; Forbus, Gentner & Law 1995; Gentner 1983; Gentner & Markman 1997; Gick & Holyoak 1983; Hofstadter 1995; Holyoak & Thagard 1996; Indurkhya 1992; Keane 1988; Kokinov & Petrov 2001; Kokinov, Holyoak & Gentner 2009; Kolodner 1993; Kunda, McGregor & Goel 2013; Prade & Richard 2009; Richard 2012; Thagard et al. 1990; Yaner & Goel 2006). The various theories of analogical reasoning differ in many ways and a commonly accepted characterization of it eludes us. In one particular characterization, analogy is the cognitive process of transferring information from a familiar situation (*source case*) to a given situation (*target problem*) (Falkenhainer, Forbus & Gentner 1989; Forbus, Gentner & Law 1995; Gentner 1983; Gentner & Markman 1997; Gick & Holyoak 1983; Holyoak & Thagard 1996; Keane 1988; Thagard et al. 1990). In this view, given a target problem, an essential first step in analogical reasoning is retrieval of a source case appropriate to the problem.

However, cognition is situated in external physical, social and informational worlds (e.g., Brown, Collins & Duguid 1989; Clancey 1997; Greeno 1998; Norman 1993). It follows that analogical reasoning too is situated: analogies are made through interaction with the external environment and representations are constructed as and when needed, rather than simply retrieved from long-term memory. Thus, recently, situated analogy has started receiving significant attention in research on analogical reasoning. Dunbar (1997) describes collaborative analogical making in science research meetings. Nersessian & Chandrasekharan (2009) describe different kinds of analogy in a science research laboratory including situated analogy. Kulinski & Gero (2001) view many analogies in design

as situated. Christensen & Schunn (2007) describe collaborative analogy making by an engineering design team.

In this chapter, we study analogical reasoning in the context of biologically inspired design in which new technological designs are generated by cross-domain analogies to biological systems. In particular, we focus on the phenomenon of *interactive analogical retrieval (IAR)* wherein the source biological cases are obtained through interaction with Web-based online information environments. This is both because (1) retrieval of a source case appropriate to the target problem is the essential first step in analogical reasoning, and (2) the biological case retrieved in biologically inspired design typically has a significant impact on the final design.

We first report on two *in situ* studies of creative, cross-domain analogies in the context of biologically inspired design in an educational setting that identify the IAR phenomenon. Then, we present a descriptive account that identifies the main cognitive challenges of IAR. Next, we develop an information processing theory called PRISM that provides a partial explanation for the challenges of IAR. Our theory builds on Pirolli's (2007) *information foraging theory* of human online information-seeking behavior and Thagard *et al.*'s (1990) computational model of *analogical retrieval by constraint satisfaction* from long-term human memory. Our three goals for the chapter are to (1) identify the IAR phenomenon as an important part of situated analogy, (2) describe the main cognitive challenges of IAR, and (3) propose a preliminary information-processing theory that explains the challenges.

2. Cross-Domain Analogies in Biologically inspired design

Biologically inspired design (also known as biomimicry and biomimetics) is one of the important emerging movements in modern design (Bar-Cohen 2011; Benyus, 1997; Chakrabarti & Shu 2010; Vincent *et al.* 2006; Vincent & Mann, 2002; Shu *et al.* 2011; Yen & Weissburg, 2007). The paradigm espouses use of analogy to biological systems to address design problems. Although biologically inspired design is a familiar and well-known paradigm, over the last generation it has become a movement in technological design, fueled by a growing need for environmentally sustainable development and the desire for creativity and innovation in design.

Consider the following scenario as an illustration of biologically inspired design (Haven 2006): One night in 1944, Swiss engineer and inventor George de Mestral was frustrated trying to free a stuck zipper on his wife's dress and wondered if he can invent a better fastener. Weeks later, after walking his dog through the woods, he paused to remove burrs from his pants and from his dog's fur. He noticed how "clever" and efficient burrs were at hooking onto anything they touched. He then examined burrs under the microscope in his workshop and within minutes he sketched what he called "a locking tape," and the idea of Velcro was born.

Other more recent and well-known examples emerging from this paradigm include the design of energy-efficient wind turbines inspired by the pectoral fins of humpback whales (Ashley 2004; Fish & Battle 1995), drag-reducing underwater surfaces inspired by shark skin (Thilmany 2004), dry adhesives inspired by adhesion mechanism of gecko feet (Lee, Lee & Messersmith 2007), self-cleaning surfaces inspired by the super-hydrophobic effect of lotus leaves (Liu & Jiang 2012), fog harvesting devices inspired by the arrangement of hydrophobic and hydrophilic surfaces on Namibian beetles (Harmon 2010), etc.

From a cognitive standpoint, biologically inspired design is an instance of *cross-domain analogical design* (Goel 1997; Goel & Bhatta 2004): solving a target design challenge in one domain (such as engineering), by transferring elements of a source case from a different domain (such as biology).

However, there are millions of species of biological organisms (Mora *et al.* 2011). If one takes into account different levels of organization of biological organisms, such as molecular-, cellular-, organ-, and ecosystem-levels, then the estimated number of biological systems increases by one or more orders of magnitude. At the same time, designers coming from engineering are typically novices in biology and are not familiar with the extent, scope, and richness of biology: they may be aware of only a tiny fraction of the vast space of biological systems that can be drawn upon in order to develop their design solutions. The near limitless availability of biological systems to draw upon coupled with engineering designers' lack of knowledge of the vast domain of biological systems makes finding biological analogies a cognitively challenging task.

The question then becomes: How do designers situated in one domain (engineering) in fact find relevant cases from the vast space of available systems that belong to a completely different and mostly unfamiliar domain (biology)? In this paper, we present two *in situ* studies of designers engaged in biologically inspired design that examine this question. Our studies indicate that in practice designers often try to retrieve biological cases from their external information environments (e.g., WWW) instead of their long-term memory. We call this phenomenon *interactive analogical retrieval*.

3. The Phenomenon of Interactive Analogical Retrieval (IAR)

We conducted two *in situ* studies of biologically inspired design practice in an educational environment to understand the phenomenon of IAR as it occurs in a real setting. Dunbar (2001) has shown the importance of conducting studies of analogy in natural, everyday settings: he found that people often make analogies in their natural environment more frequently than do in a controlled, laboratory setting. Due to limitations of space, our discussion of the *in situ* studies below is necessarily brief; Vattam (2012) provides more details.

3.1 The Study Context

We conducted both *in situ* studies in the context of a senior-level course on biologically inspired design. ME/ISyE/MSE/PTFe/BIOL 4740 is a project-based course on biologically inspired design that is offered in the Fall semester of every year at Georgia Tech. Typically, the course is taught jointly by biology and engineering faculty. From the outset this course was designed to emulate real world design settings and provide students the opportunity to experience firsthand the actual practices and processes that experts bring to bear on biologically inspired design challenges. Yen *et al.* (2010, 2011) provide details of the teaching and learning in the course.

The most important element of ME/ISyE/MSE/PTFe/BIOL 4740 for us is the semester-long design project. Each design project groups an interdisciplinary team of 4-6 designers together based on similar interests. It is ensured that each team has at least one designer with a biology background and a few from different engineering disciplines. Each team identifies a problem that can be addressed by a biologically inspired solution, explores a number of solution alternatives, and develops a conceptual design solution based on one or more biological sources of inspiration. All teams present their conceptual designs during the last two weeks of class and submit a final design report.

3.2 The Study Methodology

In this section we briefly describe our methodology for data collection and analysis in the two studies conducted in Fall 2006 and Fall 2008 respectively.

3.2.1 Fall 2006 study

In Fall 2006, ME/ISyE/MSE/PTFe/BIOL 4740 attracted 45 students, 41 of whom were seniors. The class was composed of 6 biologists, 25 biomedical engineers, 7 mechanical engineers, 3 industrial engineers, and 4 from other majors. Most students, although new to biologically inspired design, had previous design experience. Out of the 45 students, at least 32 had taken a course in design and/or participated in design projects as part of their undergraduate education. The students were grouped into nine design teams, resulting in nine biologically inspired design projects by the end of this term.

In the Fall 2006 study (Vattam, Helms & Goel 2007), we adopted an observational approach. Two researchers, including the lead author of this article (Vattam), observed participant activities both inside and outside the classroom setting. Inside the classroom, we attended lectures and heard presentations from instructors. We also observed other kinds of classroom activities, including presenting and critiquing of design ideas, discussing practical challenges of engaging in bio-inspired design, having in-class team design meetings, engaging in show-and-tell “found object” exercises, making formal presentations after completing different milestones in the project, participating in other in-class exercises (e.g., analogy exercise), etc. Outside the classroom, we attended many team design meetings. Members of each design team met outside the classroom on a regular basis to make progress on their design projects. During these meetings we were able to observe interactions among members of the design team. We were also able to observe and document various kinds of external representations (e.g., diagrams, equations, graphs) that were created and propagated throughout the design episode. As observers we paid close attention to instructor-designer and designer-designer interactions in the classroom. Each researcher kept detailed field notes consisting of their own observations.

We also collected all the work products generated by all the nine design teams. These were usually milestone documents. After every milestone (e.g., problem definition, biological search, initial design, or design analysis), design teams were required to submit a document detailing their accomplishments leading up to that milestone. These documents gave us snapshots of the progression of the design over the semester-long duration. The final design report, which captured different aspects of their finished design, also provided another data point for our analysis. We also collected their presentation materials that were used to make formal presentations in the class for an audience.

Another source of the data was the design idea journals maintained by each team member in the design teams. As part of the course requirement, every participant was required to maintain a free-form journal in which they were required to not only externalize their design thoughts and the biological systems they encountered, but also reflect on the design process and the activities they were undertaking. In order to motivate students to make effective use of the journal, it was collected and evaluated by the instructor every week from a small set of randomly selected students. At the end of the term, the idea journals from all the participants were collected by the instructor, which were subsequently shared with the researchers. To sum up, the data for this study mainly came from three important sources: 1) direct observations of two researchers captured in field notes, 2) work products produced by design teams, and 3) idea journals maintained by individual design team members.

3.2.2 Fall 2008 study

In Fall 2008, the ME/ISyE/MSE/PTFe/BIOL 4740 course attracted 43 students. The class was composed of 16 biologists, 2 biomedical engineers, 10 mechanical engineers, 7 industrial engineers, and 6 material science engineers, and 2 from other majors. In 2008, students were grouped into eight design teams, resulting in eight design projects by the end of the term.

In the Fall 2008 study (Vattam, Helms & Goel 2010), we adopted a *participatory case study* (Reilly 2010) approach. In Fall 2008, the lead author (Vattam) registered for the course on biologically inspired design and participated in a design project like all other students. With full participation, the researcher

became an integral part of a design team and obtained not only firsthand experience with the task of seeking biological analogies, but also a chance to closely interact with and understand the experiences of fellow team members who were also engaged in this task. While in the Fall 2006 study, we got a big picture view of all the nine design projects, in Fall 2008 we got a much more detailed view of one team's activities.

When the design teams were formed during the initial stages of this course, the lead author became part of a design team called FORO. Upon joining this team, the other team members were made aware of the study and their consent was obtained. Precautions were taken to ensure that their participation in this study did not impact or influence their grades in any way. In terms of data, the participant researcher maintained a field note journal where he noted his observations. His notes included observations of: (1) student-instructor and student-student interactions inside the classroom, (2) team interactions inside the classroom, and (3) team interactions outside the classroom, mostly restricted to design team meetings. With regard to team interactions, he recorded the salient thoughts expressed by members during the team meetings, the concepts that were discussed, the ideas that were thrashed out, the external representations that were constructed (e.g., diagrams on the white boards), etc. Additionally, he noted his own information seeking experiences like the keywords that were used, the results that information environments returned, the problems that he faced, how long it took him to find information sources, etc. Then he discussed some of these issues that he faced with his team members and noted their thoughts on those issues in the field journal. All electronic and non-electronic written communication between team members was also used as a data point for this research.

Similar to the earlier Fall 2006 study, each team member of team FORO maintained his or her own idea journal, which provided another source of data. We also collected and analyzed the major work products generated by team FORO, which were usually milestone documents. The team's final design report, which captured different aspects of the finished conceptual design, also provided another data point for our analysis.

3.3 Data Analysis

The general data analytic strategy used in these studies was development of case descriptions (Eisenhardt 1989; Yin 2009). This qualitative analytic strategy involves developing a descriptive framework for organizing and presenting the data from a study. It is considered an alternative to the approach of starting with theoretical propositions; it is applicable to situations when researchers collect data without having settled on an initial set of research propositions. In Fall 2006, we observed nine design teams and hence we developed nine high-level case descriptions. In Fall 2008, we observed and participated in one design project and hence we developed one detailed and rich case description. All these case descriptions can be found in Vattam (2012).

This general qualitative analytic approach was operationalized using more specific data analytic techniques like time-series analysis and cross-case synthesis. The guidelines for time-series analysis suggest tracing the salient events that occur over time (Yin 2009). In this case, the events related to design problem-solving (from the project beginning to the end) within individual design teams were traced in order to develop design trajectories. These events could have had its origins either in direct observations or in observations gleaned from work products and idea journals or both.

Cross-case synthesis (Yin 2009) is an analytical technique that applies to situations where more than one case description is developed, aggregating findings across those individual cases. Cross-case synthesis is usually performed when the individual case studies have previously been conducted as independent research studies (authored by different researchers). But it can also be performed when

multiple case studies are predesigned as part of the same research study. Cross-case synthesis is considered to yield more robust findings rather than findings obtained from a single case description.

3.4 A Sample Case Description

We present a sample case description from the Fall 2006 study to convey the nature of analogies we encountered in our studies. This case description, which has been necessarily condensed, presents the design trajectory of one of the projects called *InvisiBoard*. The goal of this project was to conceptualize a new kind of surfboard technology that prevents the formation of surfboard silhouette, which typically resembles a potential prey to sharks from underneath. The goal is to prevent “hit-and-run” shark attacks due to mistaken identity. This problem was biologized as: “how do organisms camouflage themselves in water to prevent detection by their predators?” The team researched and identified the following biological systems as potential source cases. (i) Indonesian mimic octopuses are expert camouflage artists. They can mimic various animals based on which predator is close by. Upon studying closely, this source was rejected because the surfboard is a rigid body and does not afford the same flexibility as the body of an octopus. (ii) Bullethead parrotfish uses the principle of pointillism to camouflage themselves. When viewed at close range, the fish appear bright and colorful but when viewed from a further distance, the combination of the complementary colors creates the illusion that the fish is grey-blue. This trick blends the parrotfish into the backlight of the reef, and in essence it disappears. (iii) Pony fish achieves camouflage by producing and giving off light that is directly proportional to the amount of ambient down-welling light for the purpose of counter-illumination.

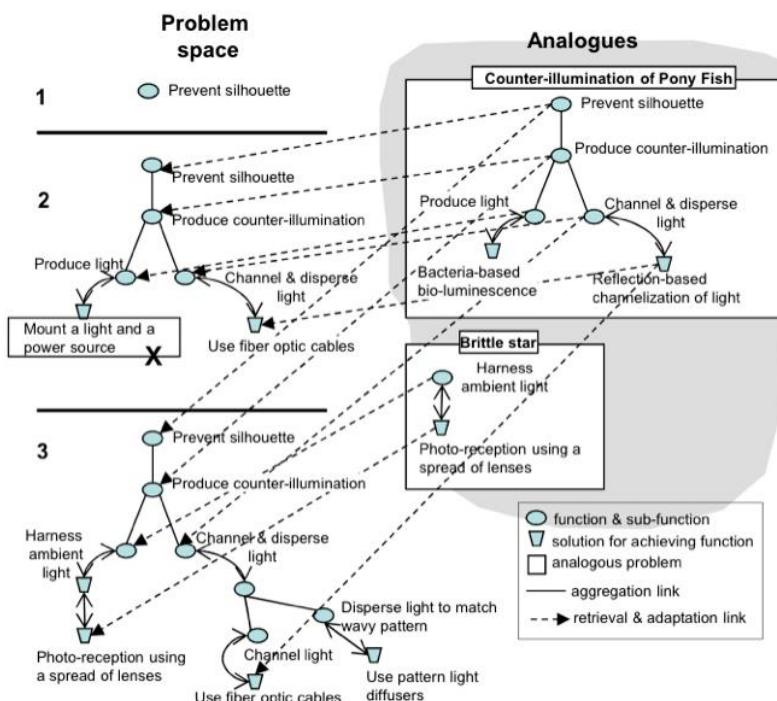


Figure 1: Design trajectory of the InvisiBoard project

Designers chose the pony fish as their source of inspiration. The function of camouflage indicated the sub-function of producing a glow on the ventral side of the surfboard to match the ambient down-welling light in order to prevent the formation of the silhouette. Now the issue became what mechanism of producing the light achieved this function. In the case of pony fish, designers understood that the

light is produced by bioluminescence - the light-producing organ of the fish houses luminescent bacteria *Photobacterium leiognathi*. This light is channeled from the light-producing organ to the ventral side and dispersed by creating rectangular light spots on the ventral side. Therefore, the function of producing ventral glow was decomposed in other sub-functions: produce light, channel and disperse light.

In order to produce light for the surfboard, the traditional means of having an onboard light source and a power source was considered an inferior solution. The search for alternate means of producing light sparked another round of search for biological sources of inspiration, which led them to an organism called Brittle star (a kind of a star fish). This organism implements the mechanism of photoreception. The dorsal side of the Brittle star is covered with thousands of tiny eyes, or microscopic lenses, making the entire back of the creature into a compound eye. This mechanism can be used to collect surrounding light rather than having to produce luminescence as in Pony fish. This suggested a design in which the top of the surfboard would be covered with (suitably distributed) tiny lenses to collect the sunlight incident upon the surfboard.

In order to channel and disperse the light collected to the bottom, their design incorporated embedding optic fibers within the surfboard. One end of these cables would be connected to the lenses on the topside and the other end would be positioned on the bottom side. Although this would channel and disperse light, it would lead to spots of brighter and dimmer light when seen from below the surfboard. This would still produce a silhouette, albeit of a different kind compared to the normal surfboard. To counter this, they had to think of another sub-function: disperse light to mimic the wavy pattern of the ocean surface. In order to achieve this function, their final design included adding a layer of “pattern light diffusers” on the bottom of the surfboard that disrupts the pattern of light (coming from the optical fibers) in controlled ways. This layer could be structured to mimic the wavy pattern of the ocean surface.

Figure 1 depicts the evolution of the final design solution. Step 1 depicts the nature of the problem space early in the design. The main function is the prevention of silhouette. Step 2 shows the retrieval of the pony fish case and the creation of two sub-functions: produce light, and channel and disperse light. For the first sub-function (produce light), Step 2 depicts the following: (i) solution in the source design (bioluminescence) is *not* transferred, and (ii) the simple solution of mounting a light and power source is rejected. For the second sub-function (channel and disperse light), a fiber optic-based solution is also proposed in Step 2.

In Step 3, the search for a solution to the function of producing light has been transformed into “harness ambient light.” We do not have a good explanation of this function transformation. A search based on the transformed function led to the retrieval of the Brittle star case and the transfer of the photoreception mechanism. Step 3 also depicts how the evaluation of partial solution of Step 2 indicated that using fiber optic cables alone for both channeling and dispersing light does not eliminate the silhouette (but merely creates a different kind of silhouette). This led to further decomposition of the original “channel and disperse light” function into two individual sub-functions. The channel light sub-function is still done through fiber-optic cables, but the dispersion is done through specialized “pattern light diffuser” devices. Knowledge about the diffuser devices was based on background domain knowledge (and not gained by analogical transfer insofar as we can tell).

Helms, Vattam & Goel (2009) and Vattam, Helms & Goel (2008, 2010) describe several other observed examples of analogies in biologically inspired design. In the following sections, we briefly describe the main findings from the two *in situ* studies biologically inspired design practice; again Vattam (2012) describes the findings in detail.

3.5 Cross-domain and Compound Analogies

Biologically inspired engineering design by definition is based on cross-domain analogies from biology to engineering. Our case descriptions of biologically inspired design episodes show that a single design solution may sometimes require multiple biological source cases (Vattam, Helms & Goel 2008). One single biological source was not always available to help solve a target problem in its entirety. In such cases, the design solution was generated in a piecemeal or modular fashion by composing the resultants of multiple analogies to smaller sub-problems, while the target design problem evolved with each application of an analogy. This implies that the act of actively seeking biological analogies is not a one-time exercise in a given design episode. In the Fall 2006 study, 6 out of 9 (66%) design solutions were generated by composing the results of multiple biological analogies. Similarly, in the following year (Fall 2007), 4 out of 10 (40%) of the design solutions were compound solutions.

One implication of such cases of compound biological analogies is that a single design episode requires multiple undertakings of the task of finding useful biological cases. Thus, any support for accomplishing the task of finding biological cases can have a multiplier effect, resulting in greater payoff for designers.

3.6 Online Search for Biological Cases

We found that designers in our study searched online for biological cases analogous to the target problems. Based on our observations, this was one of the predominant approaches for finding biological cases that typically were in the form of biology articles. Designers reported using a range of online information environments to seek information resources about biological systems. These included: (1) online information environments that provided access to scholarly biology articles like Web of Science, Google Scholar, ScienceDirect, etc., (2) online encyclopedic websites like Wikipedia, (3) popular life sciences blog sites like Biology Blog, (4) biomimicry portals like AskNature, and (5) general web search engines like Google. But the most frequently used environments were the ones that provided access to scholarly biology literature like Web of Science and Google Scholar. Biology articles, both scholarly and otherwise, were the predominant types of media or information resources consumed during the process of online finding of biological cases.

3.6.1 Strategies for online search for biological cases

The target problem and the biological cases are situated in different domains. If cues from the target problem alone are employed during search, then designers are likely to find information resources that belong to the engineering domain rather than biology. We noted several strategies used by designers in order to bridge the engineering-biology divide.

The first strategy was to “biologize” the problem. Biologizing the problem involved redefining the problem by taking the key concepts in the design problem and substituting them with similar biological concepts. Then the concepts from the biologized problem were used as cues in order to search for biological cases. For instance, in the InvisiBoard project discussed above, the problem of prevention of silhouette formation was biologized to organisms employing camouflage to avoid detection by predator or prey. Then the concept of camouflage was chiefly used to find biological cases that had this feature. Although this process of biologizing the problem was observed in all the design projects, it remains a black box: there were no explicit rules for the process that appeared to rely on the tacit knowledge of designers. The other strategies that were used are shown in Table 1.

Table 1. Strategies for finding biological cases

| Strategy | Description |
|----------------------------------|--|
| Change constraints | If the problem is narrowly defined, such as ‘keeping cool’, change the constraints to increase the search space, for instance to ‘thermoregulation’. |
| Champion adapters | Find an organism or a system that survives in the most extreme case of the problem being explored. For instance, for ‘keeping cool’, look for animals that survive in desert or equatorial climates. |
| Variation within solution family | Where multiple organisms have faced and solved the same problem in slightly different ways, e.g. bat ears and echolocation, look at the small differences in the solutions and identify correlating differences in the problem space. |
| Multi-functionality | Find organisms or systems with single solutions that address multiple problems simultaneously. |
| Inverse functions | If a particular function is not yielding many biological solutions, try the inverse of the function. For instance, if the function is ‘keeping cool,’ look for organisms that achieve the function ‘keeping warm.’ In some cases, the inverted function might yield many potential systems. Learning about the mechanism for the inverse function can sometimes yield insights into accomplishing the original function. |

It should be noted that the core of these strategies involve using cross-domain abstractions to bridge the domain of technology and the domain of biology. These abstractions include *functions* (e.g., strategies like biologizing, inverse functions, multi-functionality involves abstracting and/or transforming the functions), *operating environment* (e.g., the strategy of champion adapters), *mechanisms/physical principles* (e.g., the strategy of variation within solution family involves similarity across mechanisms and principles), and *constraints* (e.g., the strategy of changing constraints).

3.7 Challenges of Online Search for Biological Cases

Our observations indicate that the online information environments on which designers relied upon did not adequately support the online search for biological cases. We noted that it took a long time (several weeks of searching) for designers to find useful biological cases. Designers complained that the search process was frustrating because it consumed a lot of time but yielded very few biology articles containing biological systems that were actually useful in addressing their target problem. We also observed that although designers spent a lot of time searching for novel material, they often ended up using biological systems that they were already exposed to during their class lectures because their search process did not yield any new cases. The difficulty of finding useful biological cases was exasperated due to the fact that the designers often had to undertake this task multiple times in the course of their design episode due to the nature of compound analogies.

Three main challenges were noted in the process of online analogy seeking for the purpose of finding biological sources of inspiration. These difficulties were encountered irrespective of the type of information environment (such as Google), and contributed greatly to the inefficiency of the search process.

The first challenge was *findability*: designers often went for long periods without finding a useful biological case through online search. In our everyday online search experiences, we often find useful information resources efficiently and easily. However, the relative frequency of finding useful biological cases in our studies typically was low. A back-of-the-envelope calculation suggests that designers spent approximately three person-hours of search time in order to find a single relevant article.

Team FORO, consisting of 6 people, collected 39 articles over the 7 week period, where each designer reported spending an average of 2 to 3 hours per week on this task.

The second challenge was *recognizability*: designers were prone to making errors in judgment about the true utility of biological cases that they encountered in the search process. In almost all online environments, search queries brought back a ranked list of search results in the form of a set of information resources. An important aspect of the search process was assessing and selecting promising information resources from this list for further analysis. In many instances, designers picked up on low-utility articles and spent a lot of time and effort trying to understand its contents, only to realize later that it was not actually very useful (false positives). In the Fall 2008 study where the researcher maintained notes about his search experiences, 53 instances of false positives were identified. False positives lead to wasted time and effort (resource cost). False positives also have opportunity costs associated with them: in the time taken to handle less profitable items, the opportunity to go after more profitable items is lost. Conversely, one can imagine situations where designers might dismiss a resource that they encounter during the search as having low utility even though in actuality it might have contained useful information about a potential biological case (false negatives). Unfortunately, even though such situations are highly probable, it is practically impossible to observe them in the field because the rejected items are not tracked and independently assessed for their true utility. Although false negatives do not have resource cost associated with them, they represent lost opportunities. The resource and opportunity cost of recognition errors, coupled with the fact that there is a tendency among designers to fixate on the biological cases that they find initially, can potentially lead to suboptimal choice of source cases.

The third challenge was *understandability*. Developing an understanding of the workings of unfamiliar biological systems from the biology articles that one encounters is an integral part of the online search for biological cases. It is also important to develop the right kind of understanding of the systems: understanding that allows designers to “see” in the ways in which a biological system is similar or dissimilar to the technology that is being designed and how its mechanisms, strategies, principles, etc., can or cannot be transferred and adapted to solve the target design problem. Thus, learning and sensemaking are inextricably intertwined in the context of online search in biologically inspired design. Building a good understanding of systems from online information resources presented one of the biggest challenges for designers (especially for non-biologists). That it was difficult to make sense of biology articles was one of the common complaints expressed. This was in part due to the scholarly nature of the articles that were being sought and used. A majority of such articles are produced by experts and for experts in the biology community, whose focus is on communicating what, why, and how the researchers did their work, presenting key data (often in figures, tables, and charts), and discussing the results of their analysis. More importantly, the focus is not always on providing a step-by-step guide to how a biological system works. The level of abstraction at which the targeted biological systems are discussed in these articles is often too detailed, obfuscating the “big picture” of their workings. This issue, combined with the technical and domain-specific nature of the vocabulary used, and the implicitness or omission of key concepts, often hindered the sensemaking process. One consequence of this difficulty is that the process of sensemaking spawned new information needs and contributed additional cycles of search. This characteristic of sensemaking adds to the complexity and cost of online search for biological cases.

4. PRISM: An Information-Processing Theory of IAR

We now present an information-processing theory of the IAR phenomenon. This theory builds on two existing theories: *Information Foraging Theory* (Pirolli 2007) and *Analogical Retrieval by Constraint Satisfaction* (Thagard *et al.* 1990). Our goal here is to find an explanation for the observed challenges of

IAR. Here we focus on the challenges of findability and recognizability; Vattam (2012) addresses the challenge of understandability of biological cases that requires a different kind of explanation.

Information foraging theory (IFT) (Pirolli, 2007) is a theory that explains how people interactively seek information in their environment, particularly online information environments like the Web. IFT itself is biologically inspired, according to which, the information seeking behavior of people in online environments is analogous to how animals forage for food in their natural environments. Our account borrows the following important concepts from IFT.

Information patches: Information seekers usually navigate the online environment seeking content related to some topic of interest, and the online environment is conceptually and structurally organized into topical localities. The local groups of information are generically referred to as *information patches* - the idea being that it is easier to navigate and process information that resides within the same patch than to navigate and process information across patches. Thus, the term “patch” indicates a locality in which within-patch distances are smaller than between-patch distances. Examples of patches include individual web pages in the hyper-linked World Wide Web, individual documents in online document repositories, individual articles in digital libraries, individual entries in online encyclopedic resources such as Wikipedia, etc. IFT posits that the information seekers adapt their behavior to the structure of information environment in which they operate such that the system as a whole (comprising of the information seeker, the information environment, and the interactions between the two) tries to maximize the ratio of the expected value of the knowledge gained to the total cost of the interaction (Pirolli 2005).

Proximal cues and Information scent: The structure of the online information environments have evolved to exhibit certain regularities in the distribution of information resources and the navigation mechanisms that lead to those resources. One such regularity is that when foragers encounter patches in the online information environment, they cannot perceive the contents of those patches all at once. Rather they perceive snippets of information representative of the distal information patches. These snippets of information are referred to as *proximal cues* - cues that users can perceive in their local information environment to judge the utility of distal information patches and can choose to either navigate towards or away from those patches. Proximal cues are intended to concisely represent the content that an information forager will encounter by choosing a particular patch. Figure 2 shows proximal cues in Google Scholar.

[CITATION] [Analogical retrieval in reuse-oriented requirements engineering](#)
NAM Maiden, AG Sutcliffe - Software Engineering Journal, 1996
[Cited by 24](#) - Related articles - BL Direct - All 2 versions

[PDF] ... [a new role for analogy in problem solving and retrieval: When two problems ...](#)
KJ Kurtz, J Loewenstein - Memory and Cognition, 2007 - mc.psychonomic-journals.org
People often solve problems by recalling similar examples. However, many such reminders are superficially similar, based only on isolated matches at the object or feature level. If a retrieved example does not have the same structure as the target problem, it is unlikely to provide ...
[Cited by 13](#) - Related articles - BL Direct - All 6 versions

[Retrieval and learning in analogical problem solving](#)
RM Jones, P Langley - ... of the seventeenth annual conference of ..., 1995 - books.google.com
Retrieval and Learning in Analogical Problem Solving Randolph M. Jones Artificial Intelligence Laboratory University of Michigan 1101 Beal Avenue Ann Arbor, MI 48109-2110 rjones@eecs.umich.edu Pat Langley Robotics Laboratory Computer Science Department ...
[Cited by 11](#) - Related articles - All 10 versions

[CITATION] [Analogical retrieval within a hybrid spreading-activation network](#)
TE Lange, ER Melz, CM Wharton, ... - ... : proceedings of the ..., 1991 - Morgan Kaufmann Pub
[Cited by 7](#) - Related articles

Figure 2: Proximal cues in Google Scholar

The perception of proximal cues associated with information patches by the information seeker is referred to as *information scent* of a patch. While cues are situated in the environment, scent is associated with the information seeker and can be subjective. The information scent of a patch can be strong or weak based on the perceived relevance of the patch with respect to the information need of the seeker. If proximal cues are perceived to have high information scent, a forager will assess that the patch associated with that scent is likely to lead to information relevant to forager's goals and vice versa. Thus, information scent plays an important role in guiding seekers to or away from the information they seek depending on the accuracy of the information scent perception.

The second influence on our theory of IAR comes from traditional models of analogical retrieval from long-term memory. Although there are several models of analogical retrieval in the cognitive science literature that we can draw upon (e.g., Forbus, Gentner, & Law 1995; Jones & Langley 1995; Kokinov & Petrov 2001; Kolodner 1993; Schank 1983; Wolverton & Hayes-Roth 1994; Yaner & Goel 2006), we chose to build on ARCS in our work because it provides a content account of types of similarity that helps explain our observational data (Vattam 2012). ARCS posits that in order to access source cases stored as schemas in long-term memory that are analogous to a target problem held as schema in short-term memory, the access mechanism should simultaneously consider three kinds of constraints: (1) *semantic similarity* - the overlap in terms of the number of similar concepts between the target problem and the source cases, (2) *structural similarity* - the overlap in terms of the higher-order relationships between the target problem and the source cases, and (3) *pragmatic similarity* - the overlap in terms of the pragmatic constraints or goals surrounding the target problem and source cases. It is these three pressures acting simultaneously that distinguish analogical retrieval from other kinds of routine information access mechanisms.

Thus, on one hand, ARCS explains how sources cases are retrieved from the long-term memory but is silent about analogies situated in external information environments. On the other, IFT explains how people seek information in online information environments in general, but it does not address the pressures of analogical retrieval. Our PRISM theory specializes IFT to online search for biological cases by introducing the pressures of analogical retrieval from the ARCS model into the information foraging process. Hence the name PRISM for PPressurized Information Scent foraging Model.

As Figure 3 illustrates, according to PRISM, the task of retrieving a source case is accomplished by two iterative processes that constitute the general information seeking behavior: *between-patch* processes (on the left side of the vertical dotted line) and *within-patch* processes (on the right side).

4.1 Between-patch foraging

Between-patch foraging explains the navigation process where the analogy seeker browses the information environment looking for high-utility information resources to consume. In the context of IAR, high-utility information patches correspond to information resources describing sources cases that are analogous to the target problem. In this process, numerous information patches (e.g., online articles, etc.) compete for the analogy seeker's attention. These patches may or may not contain information relevant to the information goals of the analogy seeker. Thus, the analogy seeker expends time and effort navigating from one patch to another until one that can be exploited is found. This is captured by the *Formulate Query – Retrieve – Compute Information Scent* cycle depicted in Figure 3.

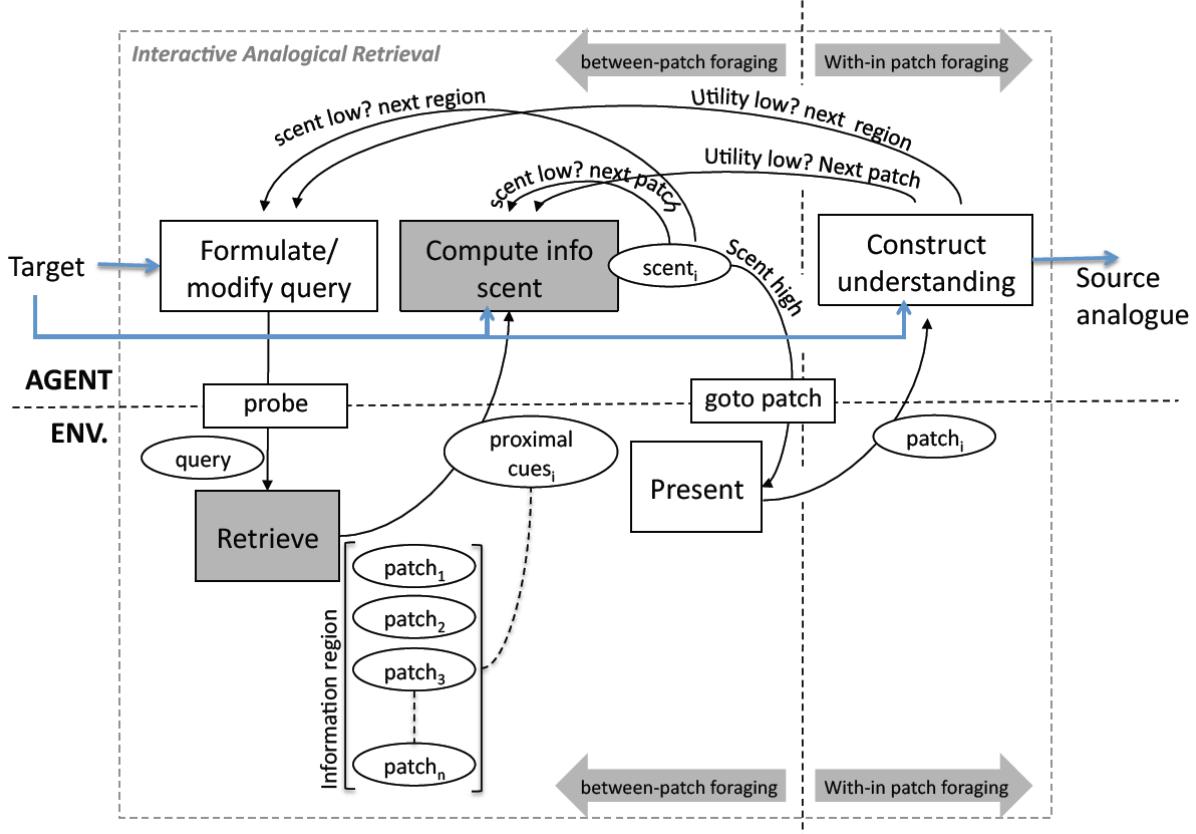


Figure 3: PRISM: An information-processing theory of interactive analogical retrieval

Between-patch foraging using information scent in IAR works as follows. Given a target problem or situation:

1. The analogy seeker probes the environment by formulating and issuing a *query*. This query is context-dependent and represents the target problem.
2. In response, the environment retrieves and conveys an information region consisting of a set of information patches $\{(P_1, \{c_{11}, c_{12}, \dots\}), (P_2, \{c_{21}, c_{22}, \dots\}), \dots\}$, where P_i is an information patch and c_{ij} 's are the proximal cues associated with the patch P_i .
3. The analogy seeker perceives the information scent of the patches based on the proximal cues; the information scent is an estimation of the analogical relevance of different patches the target: $\{(P_1, S_1), (P_2, S_2), \dots\}$, where P_i is an information patch and S_i is the information scent that the analogy seeker associates with the patch P_i .
4. If the information scent of an information patch exceeds a certain threshold, it is considered relevant (high perceived utility). Pirolli (2005; Section 3.3.1) provides details of how the threshold is estimated. In this case, the analogy seeker goes to that patch (by acting on the environment like clicking the associated hyperlink), at which point the environment presents the information patch to the analogy seeker. This initiates *within-patch foraging*.
5. If the scent does not exceed the threshold, it is considered irrelevant (low perceived utility). In this case, one of two things may happen as depicted in Figure 3: (i) the analogy seeker may stay within the same information region but loop back to Step 3 for processing the next patch in the region, or (ii) the analogy seeker may abandon the current information region and loop back to Step 1 in order to look for more fruitful regions.

4.2 Within-patch foraging

Once the analogy seeker picks up the scent of a potentially useful information patch, the seeker goes to that patch and starts consuming information in that patch. In the context of biologically inspired design, this process involves making sense of the contents of an article and constructing a mental model of the biological system described in the article. In the within-patch foraging process, the analogy seeker is also simultaneously evaluating the actual utility of the patch by comparing/aligning/mapping the emerging mental model of the biological system against the target problem. If the evaluation is successful, the agent has obtained a source case. If this evaluation fails, then the between-patch foraging process is again initiated, either within the same information region that led to the current patch or with a search for new information regions as depicted in Figure 3.

4.3 Scent perception in PRISM: Incorporating Pressures of Analogical Retrieval

While the original IFT explains the scent perception mechanism for routine information seeking tasks, it has to be adapted to account for the semantic, structural and pragmatic pressures of analogical retrieval. Hence, as part of the PRISM theory, we developed a new computational model of information scent perception.

According to ARCS model, the three pressures of semantic, structural and pragmatic similarity differentiate routine information retrieval from analogical retrieval. There are two potential places in our model in Figure 3 where these pressures might apply because that is where similarity enters the equation: “Retrieve” and “Compute information scent” that are shaded in gray in Figure 3. The “Retrieve” process may use some notion of similarity to access information patches. But in our model, the “Retrieve” process is implemented in the environment (e.g., the Google Scholar search mechanism) and thus is black-boxed in the scope of this work. “Compute information scent” process, on the other hand, is a cognitive mechanism that requires explanation.

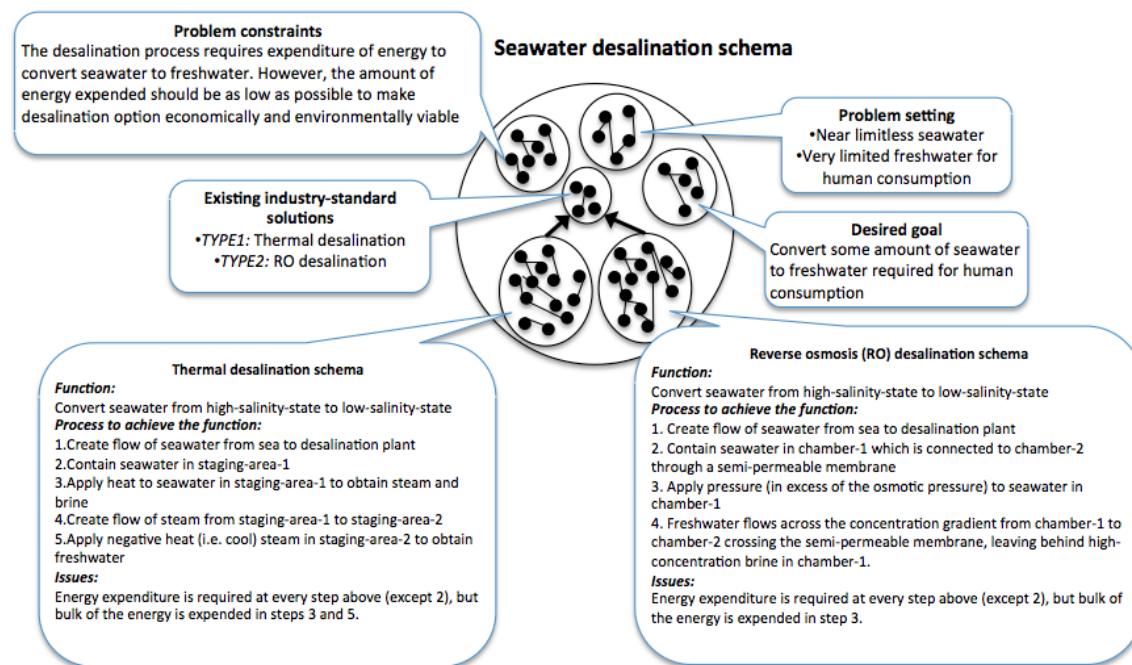


Figure 4: A depiction of a sample target schema related to water desalination problem

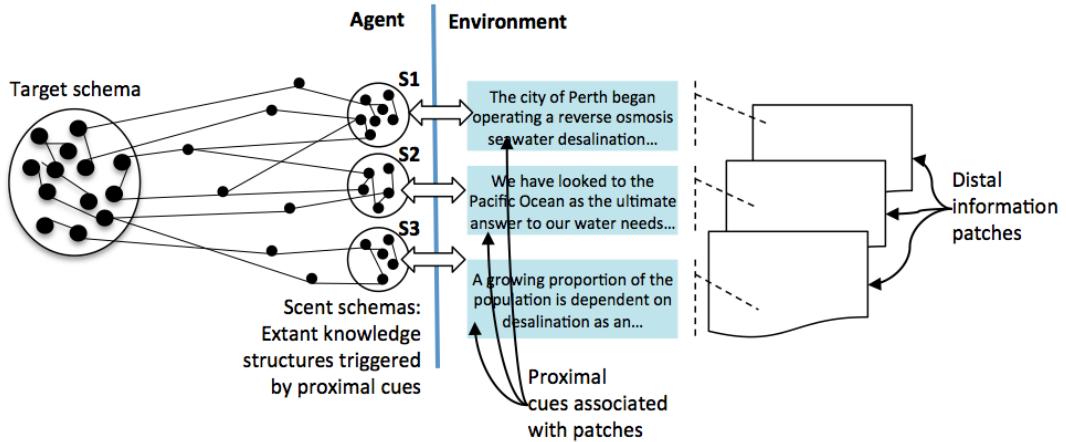


Figure 5: Scent perception in PRISM. (a) Initial state – target schema and proximal cues. (b) Next state – scent schemas from cues are mapped to target schema.

Our computational model of “Compute information scent” assumes that the analogy seeker has represented the target problem as a *target schema*. For example, Figure 4 depicts the target schema of the problem of designing a device for desalination of seawater. With a target problem in mind, the analogist forages the information environment for source cases. Following the between-patch foraging process described above, the analogist encounters a set of information patches with associated proximal cues. When proximal cues of an information patch are encountered in the environment, analogist builds a *scent schema* as depicted in Figure 5. Note that proximal cues are information in the environment and the target and scent schemas are structured knowledge representations in the agent. Our computational model at present does not address the issue of origin of these representations.

Given the target schema and competing scent schemas, the analogist computes the similarity between the target and scent schemas by setting up a multi-constraint satisfaction network and running the network as discussed in the ARCS model. Our illustration of this process in the following section closely follows Thagard et al. 1990.

4.3.1 Illustrative Example

Table 2. Example Target and Scent schemas (adapted from Thagard et al. 1990, pp. 275).

| Target Schema | Scent Schema 1 | Scent Schema 2 |
|---------------|----------------|----------------|
| P1 | S1 | S2 |
| P1-1: A(a, b) | S1-1: M(m, n) | S2-1: M(n, m) |
| P1-2: B(b, a) | S1-2: N(n, m) | S2-2: R(n, m) |

Assume: A and M are semantically similar; B and N are semantically similar;
A(a,b) is most important in this context (dictated by the pragmatics of the context).

Let us suppose that the conceptual structures representing the dots in Figures 4 and 5 consist of predicates. Table 2 above illustrates a target schema (T1) consisting of two predicates (P1-1 and P1-2), and two scent schemas (S1 and S2) consisting of two predicates each (S1-1, S1-2, and S2-1 and S2-2, respectively). Let us also suppose that the concepts *A* and *M* are semantically similar, and likewise concepts *B* and *N* are semantically similar. For example:

$A(a, b)$ is *Regulate(kidney, potassium_ions)*;
 $M(m, n)$ is *Control_Production(pituitary, estrogen)*;
 $B(b, a)$ is *Is_Secreted_By(erythropoietin, kidney)*; and
 $N(n, m)$ is *Is_Released_By(hypothalamic_hormone, pituitary)*

Let us further suppose that $A(a, b)$ is more important than the other predicates as dictated by the pragmatics of the target situation.

4.3.2 Network Setup

In a manner similar to the original ARCS model, PRISM uses information about the semantic similarity of predicates in the target and scent schemas to create a constraint network. Figure 6 depicts the network corresponding to Table 2: units in the network represent the predicates in the target and scent schemas, and the links between units represent correspondences between the predicates. There are two kinds of links between the units in the network, excitatory and inhibitory links. Excitatory links represent positive internal constraints between the units, that is, a unit with an excitatory link to an active unit will gain activation from it. Similarly, inhibitory links represent negative internal constraints where a unit with an inhibitory link to an active unit will have its own activation decreased. In Figure 6, excitatory links are represented using solid lines and inhibitory links are represented using dotted lines.

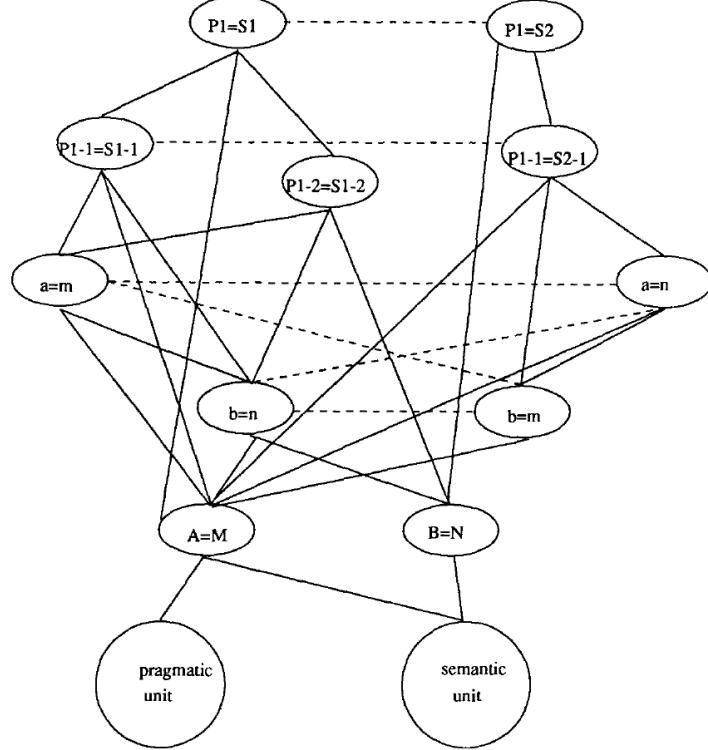


Figure 6: Constraint satisfaction network for computing similarity between Target and Scent schemas shown in Table 2. (Following Thagard *et al.* 1990, pp. 275).

The most important units in the network of Figure 6 are the ones that hypothesize that a scent schema is analogous to the target schema. These units have names of the form TARGET=SCENT, for example, the $(P1=S1)$ unit (“=” here means “corresponds to,” not identity). If the target is $P1$ and the scent is $S1$, then the $(P1=S1)$ unit represents a correspondence between them. If $P1-1$ is a proposition in $P1$ that corresponds to proposition $S1-1$ in scent $S1$, then the $(P1-1=S1-1)$ unit will have an excitatory link with the $(P1=S1)$ unit.

Excitatory links are also set up from a special semantic unit to predicate-predicate units based on the degree of semantic similarity of the predicates (in Figure 6, there are excitatory links from the semantic unit to the (A=M) and the (B=N) units because they are assumed to be semantically similar). Similarly, excitatory links are also set up from a special pragmatic unit to predicate-predicate units that are considered more important than others (in Figure 6, there are excitatory links from pragmatic unit to the (A=M) unit because predicate A is assumed to be more important than others). The activation level of the special semantic and pragmatic units is always kept at the maximum value of 1. Thus, they serve to pump activation to all units that are linked to it.

Inhibitory links are constructed between units representing incompatible hypotheses, for example, between the (P1=S1) and the (P1=S2) units. The inhibitory links make utility calculation competitive, in that choosing one scent will tend to suppress choosing of an alternative.

4.3.3 Running the network

The constraint network constructed above is run by setting the initial activation of all units to a minimal level (typically 0.01), except for the special semantic and pragmatic units for which activation is clamped at 1. Then the activation of each unit is updated by considering the activations of those units to which it has links. In particular, the activation of each unit u is calculated according to the equations suggested by the ARCS model (Thagard *et al.* 1990, pp. 279-280). Thus, the activation of unit j on cycle $t + 1$ is given by:

$$a_j(t + 1) = a_j(t)(1 - d) + enet_j(max - a_j(t)) + inet_j(a_j(t) - min)$$

Here d is a decay parameter, $enet_j$ is the net excitatory input, and $inet_j$ is the net inhibitory input (a negative number), with minimum activation $min = -1$ and maximum activation $max = 1$. Inputs are determined by the equations:

$$enet_j = \sum_i w_{ij} o_i(t) \quad \text{for } w_{ij} > 0;$$

$$inet_j = \sum_i w_{ij} o_i(t) \quad \text{for } w_{ij} < 0.$$

Here, $o_i(t)$ is the output of unit i on cycle t , set by:

$$o_i(t) = max(a_i(t), 0).$$

4.3.4 Analogical similarity and scent of an information patch

When the network settles, the similarity between the target schema, T , and a particular scent schema, S_i , is equal to the activation value of the unit ($T=S_i$) in the constraint network. Higher the activation accumulated by the unit ($T=S_i$) the more similar is the scent schema, S_i , to the target, T . The information scent of a particular information patch, IP_i , which is associated with a set of proximal cues, $\{C_{ij}\}$, is equal to the similarity between the scent schema, S_i , obtained from $\{C_{ij}\}$, and the target schema, T .

5. Using PRISM to Explain the Challenges of IAR

The PRISM theory can be used to reason forwards from changes in the information environment to their observable effects on the IAR process described in Section 3, or backwards from observed IAR effects to the factors in the information environment causing those effects. Reasoning backwards, the PRISM

model provides the following causal explanations for the three observed challenges associated with IAR.

The findability challenge, in which designers often go for long periods without finding a relevant information resource, can be localized to the *Formulate-Retrieve-Compute Information Scent* loop highlighted in the PRISM model shown in Figure 7(a). In current common online information environments, the “Retrieval” process in Figure 7(a) typically is based on keyword matching. This supports access to information based on literal similarity (word-for-word matching) while ignoring semantic, structural and pragmatic similarity. This method does not support access to information resources based on the right kinds of concepts from a designer’s perspective. As a result, each attempt at access can contain a large number of superficially similar cases as opposed to cases that are truly analogous, which entails a lower average information yield per region. This yield is inversely proportional to the number of times the said loop is executed in the PRISM model: a low yield implies more executions of the cycle. Therefore, between-patch foraging time is higher, the period between successive useful finds is longer, and the frequency of encountering useful information resources is lower.

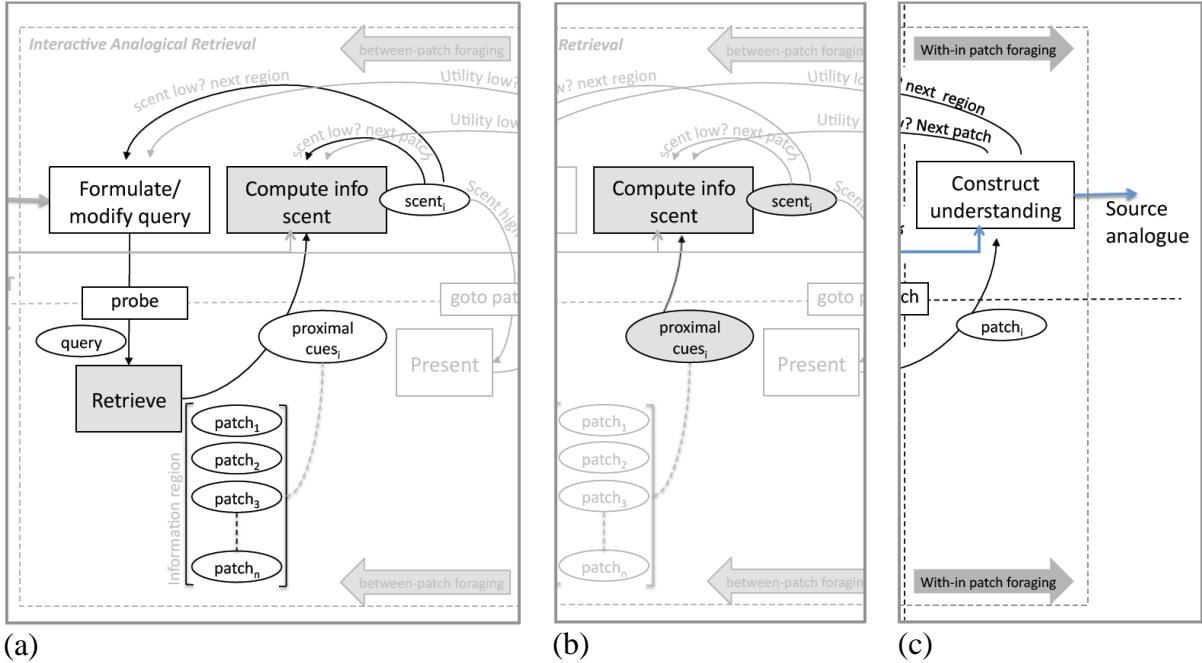


Figure 7: Challenges of IAR. Localizing the issue of (a) findability, (b) recognizability, and (c) understandability in PRISM.

The recognizability challenge can be localized to the information scent computation sub-process highlighted in IAR model shown in Figure 7(b). This issue is attributable to the nature of proximal cues that one encounters in customary online information environments – specifically, their lack of affordance for accurately perceiving the information scent of the resources they represent. Accurately perceiving the scent of an information resource in the context of IAR requires judging the analogical

similarity between that target problem or situation and the source information that can be gleaned from the cues.

The dynamics of analogical similarity calculation and, as a consequence, the perception of information scent in the PRISM theory is adaptive to the different knowledge conditions of target and scent schemas. One condition that can be identified is *sparse* schema condition: refers to scent schemas containing few isolated concepts and relationships. Under this condition, since there is not a lot of higher-order structure in the knowledge, the computed information scent would mostly depend on the semantic similarity between the target and source schemas. If the target and scent schemas were from the same domain (implying significant semantic overlap), then the information scent would be strong. If the target and scent schemas were from different domains (implying low semantic overlap), then the information scent would be weak.

The second condition is the *well-connected* schema condition: this refers to scent schemas containing not only concepts and relationships but also systems of abstract higher-order relationships that connect and organize lower-order relationships and concepts. Under this condition, since there is structure in the knowledge, the computed information scent would depend on both semantic and structural similarity. If the target and scent schemas were from the same domain (implying significant semantic overlap) and the structural similarity was high, the information scent would be strong. If the target and scent schemas were from the same domain (implying significant semantic overlap) and the structural similarity was low, the information scent would be weak. If the target and scent schemas were from different domains (implying low semantic overlap), then the information scent would be weak.

The third condition that can be identified is well-connected and *shared abstraction* schema condition: well-connected schemas can be further classified based on the encoding specificity, which refers to how domain specific or general the content of the schemas are. The encoding specificity is related to the presence or absence of domain general abstractions as part of the higher-order content of the schemas. Research on memory, knowledge representation, and reasoning in analogy indicate that people learn domain general (shared) abstractions that allow them to see commonality between two or more schemas that differ only in specifics but share a deeper commonality. Examples of such shared abstractions include thematic organization packets (Schank 1983), thematic abstraction units (Dyer 1983), and teleological design patterns (Goel & Bhatta 2004). In the third condition, if the target and scent schemas were from the same domain, the information scent would be same as in the simple well-connected knowledge condition discussed above. If the target and scent schemas were from different domains, however, then the domain-bridging abstractions will permit the construction of the constraint network; if the structural similarity between the target and scent schemas were high, the information scent would be strong and vice versa.

Now consider the task of biologically inspired design. The design of proximal cues customarily contains small snippets of information in the domain of biology. Information seekers (engineers) are novices in the domain of biology and typically lack shared abstractions. This will likely lead to the first knowledge condition during information scent perception mentioned above. Consequently, novice designers are likely to make relevancy decisions based on superficial similarity as opposed to deep similarity. This can lead to the rejection of information resources that contain structurally similar source cases (false negatives) and/or selection of information resources that contain superficially or literally similar sources (false positives).

Finally, the third challenge of conceptual understanding can be localized to within-patch foraging process shown in Figure 7(c). The explanation of this issue requires a theory of mental model construction of complex systems and hence is beyond the scope of PRISM's explanation.

6. Conclusions

Cognition is situated in physical and social worlds and so is analogical reasoning. In this chapter, we identified *interactive analogical retrieval (IAR)* as an important phenomenon in situated analogy: in IAR, analogical retrieval is situated in online information environments. We provided a descriptive account of IAR based on our *in situ* studies of designers engaged in biologically inspired design. The descriptive account identified three main challenges associated with IAR: findability, recognizability, and understandability. We posit that IAR is a general phenomenon not limited to biologically inspired design: it occurs whenever people are searching for cross-domain analogies in external online information environments. We posit further that the cognitive challenges of findability, recognizability, and understandability, too are general and likely occur whenever novices are engaged in IAR.

We also developed an information-processing theory of IAR called PRISM that combines Pirolli's (2007) information foraging theory (IFT) and Thagard et al.'s (1990) ARCS model of analogical retrieval. PRISM extends IFT to account for analogy seeking; it expands ARCS into a model of information scent perception that is embedded in the larger context of online analogy seeking. PRISM provides explanations for the findability and recognizability challenges of IAR we observed in our studies of biologically inspired design.

PRISM may help develop new technology for helping designers find biological analogies more efficiently and easily (Vattam & Goel 2011): the model predicts that the findability and recognizability issues could be addressed, respectively, by changing the indexing and access mechanism and enriching the proximal cues in online environments. Finally, if we take the boundary of the cognitive system in biologically inspired design as including online information environments then the phenomenon of IAR becomes an important element of understanding the situatedness of analogical reasoning. By folding in interactions with external information environments, PRISM may provide a starting point for developing a general information-processing theory of situated analogy.

ACKNOWLEDGMENTS

We thank Michael Helms and Bryan Wiltgen for their contributions to this work. We are grateful to the instructors and students of the ME/ISyE/MSE/BME/BIOL 4740 class, especially Professor Jeannette Yen, the main coordinator and instructor of the course. We thank the US National Science Foundation for its support through the CreativeIT Grant (#0855916) "Computational Tools for Enhancing Creativity in Biologically Inspired Engineering Design."

References

- Ashley, S. (2004) Bumpy flying. Scalloped flippers of whales could reshape wings. *Scientific American*, 291(2):18-20.
- Bar-Cohen, Y. (Editor, 2011) *Biomimetics: Nature-Based Innovation*. Taylor & Francis.
- Benyus, J. (1997) *Biomimicry: Innovation Inspired by Nature*. William Morrow.
- Brown, J. S., Collins, A., & Duguid, P. (1989). Situated Cognition and the Culture of Learning. *Educational Researcher*, 18(1), 32-42.
- Burstein, M. (1986) Concept formation by incremental analogical reasoning and debugging. In R. Michalski, J. Carbonell & T. Mitchell (eds.), *Machine Learning 2*, Los Altos: Morgan Kauffman, pp. 351-370.

- Carbonell J. (1986) Derivational analogy: A theory of reconstructive problem solving and expertise acquisition In Michalski, Carbonell & Mitchell (eds), *Machine Learning*, Volume 3, Los Altos, CA: Morgan Kaufmann.
- Chakrabarti, A., & Shu, L. (2010) Biologically Inspired Design. *AIEDAM*, 24:453-454.
- Chalmers, D., French, R., & Hofstadter, D. (1992) High-Level Perception, Representation and Analogy: A Critique of the AI Methodology. *JETAI* 4(3):185-211.
- Clancey, W. (1997) *Situated Cognition: On Human Knowledge and Computer Representations*. Cambridge University Press.
- Clement, J. (2008). *Creative Model Construction in Scientists and Students: The Role of Imagery, Analogy, and Mental Simulation*. Dordrecht: Springer.
- Christensen, B., & Schunn, C. (2007) The relationship of analogical distance to analogical function and preinventive structure: the case of engineering design. *Memory and Cognition*, 35(1): 29-38.
- Davies, J., Goel, A., & Yaner, P. (2008). Proteus: Visuospatial Analogy in Problem Solving. *Knowledge-Based Systems*, 21(7): 636-654.
- Davies, J., Goel, A., & Nersessian, N. (2009). A Computational Model of Visual Analogies in Design. *Cognitive Systems Research*, 10: 204-215.
- Dunbar, K. (1997) How Scientists Think: Online Creativity and Conceptual Change in Science. In T. Ward, S. Smith & J. Vaid (eds.) *Conceptual structures and processes: Emergence, discovery , and change*. Washington D.C.: American Psychological Association Press,
- Dunbar, K. (2001) The Analogical Paradox. In D. Gentner, K. Holyoak & B. Kokinov (editors), *The Analogical Mind: Perspectives from Cognitive Science*, MIT Press.
- Dyer, M. (1983) *In-Depth Understanding: A Computer Model of Integrated Processing for Narrative Comprehension*. MIT Press.
- Eisenhardt, K. (1989). Building theories from case study research, *Academy of Management Review*, 14: 488- 511.
- Falkenhainer, B., Forbus, K. D., & Gentner, D. (1989). The structure-mapping engine: Algorithm and examples. *Artificial Intelligence*, 41(1), 1-63.
- Fish F., & Battle J. (1995). Hydrodynamic design of the humpback whale flipper. *Journal of Morphology*, 225:51–60.
- Forbus, K., Gentner, D., & Law (1995). MAC/FAC: A model of similarity-based retrieval. *Cognitive Science*, 19(2), 141-205.
- Gentner, D. (1983). Structure-mapping: a theoretical framework for analogy. *Cognitive Science*, 23, 155-170.
- Gentner, D. (1989). The mechanisms of analogical learning. In S. Vosniadou & A. Ortony (Eds.), *Similarity and Analogical Reasoning*. New York: Cambridge University Press.
- Gentner, D. & Markman, A. B. (1997) Structural alignment in analogy and similarity, *American Psychologist*, 52:45–56.
- Gick, M., & Holyoak, K. (1983) Schema Induction and Analogical Transfer. *Cognitive Psychology*.
- Goel, A. (1997) Design, Analogy, and Creativity. *IEEE Expert*, 12(3), 62-70.
- Goel, A., & Bhatta, S. (2004). Use of design patterns in analogy-based design. *Advanced Engineering Informatics*, 18(2), 85 –94.
- Greeno, J. (1998). The Situativity of Knowing, Learning, and Research. *American Psychologist*, 53(1), 5-26.
- Harmon, P. (2010). *The Mind of an Innovator: A Guide to Seeing Possibilities Where None Existed Before*. Strategic Book Publishing.

- Haven, K (2005) *100 Greatest Science Inventions of All Time*, Libraries Unlimited, 2005.
- Helms, M., Vattam, S. & Goel, A. (2009) Biologically Inspired Design: Products and Processes, *Design Studies*, 30(5), 606-622.
- Hofstadter, D. (editor, 1995). *Fluid concepts and creative analogies*. New York: Basic Books.
- Hofstadter, D. (2001) Analogy as the Core of Cognition. In Gentner, D., Holyoak, K., Kokinov, B. (editors) *The Analogical Mind: Perspectives from Cognitive Science*, MIT Press.
- Hofstadter, D., & Sander, E. (2013) Surfaces and Essences: Analogy as the Fuel and Fire of Thinking. Basic Books.
- Holyoak, K., & Thagard, P. (1996). *Mental Leaps*. MIT Press.
- Indurkha, B. (1992) On the Role of Interpretive Analogy in Learning. *New Generation Computing*, 8(4): 385-402.
- Jones, R., & Langley, P. (1995) Retrieval and Learning in Analogical Problem Solving. In *Procs. Seventeenth Annual Meeting of the Cognitive Science Society*.
- Keane M. (1988) *Analogical Problem Solving*. Ellis Horwood. Chichester
- Kokinov, B., Holyoak, K., & Gentner, D. (editors, 2009). *New Frontiers in Analogy Research*. Sofia: NBU Press.
- Kokinov, B., & Petrov, A. (2001) Integration of Memory and Reasoning in Analogy-Making: The AMBR Model. In Gentner, D., Holyoak, K., Kokinov, B. (editors) *The Analogical Mind: Perspectives from Cognitive Science*, MIT Press.
- Kolodner, J. L. (1993). *Case-based reasoning*. San Mateo, CA: Morgan Kaufmann Publishers.
- Kulinksi, J., & Gero, J. (2001) Constructive Representation in Situated Analogy in Design. In *Proc. CAADFutures*, Eindhoven, Netherlands.
- Kunda, M., McGregor, K., & Goel, A. K. (2013). A computational model for solving problems from the Raven's Progressive Matrices intelligence test using iconic visual representations. *Cognitive Systems Research*, 22-23, 47-66.
- Lee H, Lee BP, Messersmith PB (2007). A reversible wet/dry adhesive inspired by mussels and geckos. *Nature* 448: 338-341.
- Liu, K. & Jiang, L. (2012), Bio-Inspired Self-Cleaning Surfaces, *Annual Review of Materials Research*, 42, 231–263.
- Mora, C., Tittensor, D., Adi, S., Simpson, A., & Worm, B. (2011) How Many Species are there on Earth and in the Ocean. *PLoS Biology*, 9(8).
- Nersessian, N. (2008) *Creating Scientific Concepts*. MIT Press.
- Nersessian, N. & Chandrasekharan, S. (2009). Hybrid Analogies in Conceptual Innovation in Science, *Cognitive Systems Research Journal*, Special Issue: Integrating Cognitive Abilities. 10: pp. 178-188.
- Norman, D. (1993) Cognition in the Head and in the World. *Cognitive Science*, 17:-6.
- Pirolli, P. (2005). Rational analyses of information foraging on the web. *Cognitive science*, 29(3), 343-373.
- Pirolli, P. (2007) *Information foraging theory: Adaptive interaction with information*. Oxford: Oxford University Press.
- Prade, H., & Richard, G. (2009) Analogy, Paralogy and Reverse Analogy: Postulates and Inferences. *Advances in AI*, LNAI 5803: 306-314.
- Reilly, R. (2010). Participatory case study. In: *Encyclopedia of case study research*. Sage, Thousand Oaks, CA, pp. 658-660.

- Richard, G. (editor, 2012) *Proc. ECAI-2012 Workshop on Similarity and Analogy-Based Methods in AI*, Montpellier, France, August 2012.
- Riesbeck, C., & Schank, R. (1989) *Inside Case-Based Reasoning*. Lawrence Erlbaum.
- Schank, R. (1983) *Dynamic Memory: A Theory of Reminding in Computers and People*. Cambridge University Press.
- Shu, L., Ueda, K., Chiu I., & Cheong, H. (2011). Biologically Inspired Design, *CIRP Annals, Manufacturing Technology*.
- Thagard, P., Holyoak, K. J., Nelson, G., & Gochfeld, D. (1990). Analog retrieval by constraint satisfaction. *Artificial Intelligence*, 46, 259-310.
- Thilmany, J. (2004). Swim like the sharks, *Mechanical Engineering*, 126(5), 68
- Vattam, S. (2012) Interactive Analogical Retrieval: Practice, Theory and Technology, Ph.D. Dissertation, Georgia Institute of Technology, December 2012.
- Vattam, S., & Goel, A. (2011) Foraging for Inspiration: Understanding and Supporting the Online Information Seeking Practices of Biologically Inspired Designers. In Proc. 2011 ASME IDET/DTM Conference, Washington, D.C.
- Vattam, S., Helms, M., Goel, A. (2007) Biologically Inspired Innovation in Engineering Design: A Cognitive Study. Technical Report GIT-GVU-07-07, Graphics, Visualization and Usability Center, Georgia Institute of Technology.
- Vattam, S., Helms, M. & Goel, A. (2008) Compound Analogical Design: Interaction Between Problem Decomposition and Analogical Transfer in Biologically Inspired Design. In *Proc. Third International Conference on Design Computing and Cognition*, Atlanta, June 2008, Berlin:Springer, pp. 377-396.
- Vattam, S., Helms, M. & Goel, A. (2010). A content account of creative analogies in biologically inspired design. *AIEDAM*, 24, pp. 467-481.
- Vincent J., Bogatyreva O., Bogatyrev N., Bowyer A, Pahl A. (2006) Biomimetics: its practice and theory. *Journal of the Royal Society Interface*, 3, 471–482.
- Vincent, J. & Man, D. (2002) Systematic Technology Transfer from Biology to Engineering. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 360(1791): 159-173.
- Wolverton, M., & Hayes-Roth, B. (1994) Retrieving Semantically Distant Analogies with Knowledge-Directed Spreading Activation. In *Procs. National Conference on AI (AAAI-94)*.
- Yaner, P., & Goel, A. (2006) Visual Analogy: Viewing Retrieval and Mapping as Constraint Satisfaction. *Journal of Applied Intelligence*, 25(1):91-105.
- Yen, J., Helms, M., Goel, A., & Vattam, S. (2010) Evaluating Biological Systems for their Potential in Engineering Design. *Advances in Natural Science*, 3(2).
- Yen, J., & Weissburg, M. (2007), Perspectives on biologically inspired design: Introduction to the collected contributions. *Journal of Bioinspiration and Biomimetics*, 2.
- Yen, J., Weissburg, M., Helms, M., & Goel, A. (2011) Biologically inspired design: a tool for interdisciplinary education. In *Biomimetics: Nature-Based Innovation*, Y. Bar-Cohen (editor), Taylor & Francis.
- Yin, R. (2009). *Case Study Research, Design and Methods*, Sage Publications, Thousand Oaks, CA.

Analyzing Raven’s Intelligence Test: Cognitive Model, Demand, and Complexity

Marco Ragni¹ and Stefanie Neubert²

¹University of Freiburg, Center for Cognitive Science,
Friedrichstr. 50, 79098 Freiburg, Germany

²Ludwig-Maximilians-Universität München
ragni@informatik.uni-freiburg.de
stefanie.neubert@campus.lmu.de

Abstract. Identifying hidden functions and patterns in data and reasoning about them is a relevant part of any inference system. Differential psychologists typically measure humans’ ability to reason by intelligence tests such as Raven’s Standard or Advanced Progressive Matrices test. Human reasoning difficulty, however, for IQ-test items is almost always determined empirically: An item’s difficulty is measured by the number of reasoners who are able to solve it. So far this method has proven successful: Nearly all IQ-Tests are designed this way. Still, it is most desirable to have a computational model that captures the inherent reasoning complexity and allows for a predictive classification of problems. In this article, we first analyze and classify fluid intelligence test items as they are common in tests like Raven’s Standard and Progressive Matrices and Catell’s Culture Fair Test. This functional classification allows us to develop a model in the cognitive architecture ACT-R that solves 66 of 72 problems from both Raven tests. We argue that in addition to visual and formal reasoning difficulty the way information is processed in the modular structure of the human mind should be taken into account. A simpler (but functionally equivalent) version of the APM is developed and tested on humans. Implications for developing a cognitive system that can classify human reasoning systems are discussed.

Keywords: Analogical Reasoning, Pattern Recognition, IQ-test Problems

1 Introduction

Following Gardner [1], intelligence is probably the most interesting part of human cognition. But, although intelligence is a major factor of intellectual ability, it has thus far defied any attempts to classify or even define it. Probably the first scientific and systematic approach to test human intelligence by a standardized test was introduced by Binet and Simon [2] in 1905. Today the most prominent intelligence tests are Wechsler’s Measurement of Adult Intelligence [3], Cattell’s Culture Fair Test [4], and Raven’s Progressive Matrices [5]. First steps into characterizing human reasoning difficulty for such tests has been undertaken. An

individual's intelligence is almost always measured by determining the deviation of his or her performance on a given set of reasoning problems, from a particular group of specific age and educational status, etc. Problems, in turn, are classified empirically as simple or challenging, based on whether a given population is able to solve most or only a limited number of similar problems. While it is possible to empirically capture the human reasoning difficulty, it seems more desirable to identify the formal characteristics of such IQ-test problems and to test to what extent they are able to explain human reasoning difficulty. Furthermore, such a formal or algorithmic characterization may elucidate future test development and can then form a formal foundation of human reasoning complexity. A positive side-effect of such a formal classification is that it allows for the creation of new and different reasoning problems with the same reasoning difficulty. A further algorithmic understanding of complexity characteristics provides major steps towards an artificial intelligence system. However, any formal classification of reasoning difficulty requires a computational model. The central paradigm in cognitive science is that human cognition, and especially intelligence, can be captured by computational processes of symbol manipulation. This approach, *cognition as computation*, was coined and introduced by Newell and Simon [6]. Once problems are formally represented and functionally classified, the computational requirements necessary to solve them can be calculated.

Nearly all aspects of analogy-making can already be found in geometrical intelligence tests. Catell [7] formulated the notion of crystallized and fluid intelligence. Crystallized intelligence, which relies heavily on declarative knowledge, is opposed to fluid or analytic intelligence – the ability to solve problems involving new information without relying on background knowledge. Such problems are sometimes classified as culture fair [8] as they require less declarative knowledge than, for instance, word analogy problems. While success at solving word analogy problems can depend on additional knowledge, geometrical reasoning problems can be modeled using mathematical functions exclusively.

Such geometrical analogy problems (cf. Fig. 1) are part of a number of IQ-tests, for example the Hamburg-Wechsler-Intelligence-Test [9]. A significant number of IQ-tests even consist exclusively of such geometrical reasoning problems, e.g., Cattell's Culture Fair Test [10]. Especially Raven's *Standard* and *Advanced Progressive Matrices* [10–12] are well known geometrical analogy tests and broadly used in educational background and research. Performance on Raven's Matrices has implications for planning [13] and other domains of reasoning [14]. An analysis of Raven's IQ-Test problems has been conducted by Carpenter and colleagues [13].

Both problems depicted in Fig. 1 are functionally equivalent¹. The task is to identify the inherent functions and to choose the appropriate answer. In this example, there are at least two explanation principles possible. The first one deals with a so-called Boolean function, i.e., the exclusive or (XOR) over three cells in a row/column: Whenever one object is in the first cell and it is not in

¹ All figures and problems in the following were designed by the authors to protect the security of IQ-tests.

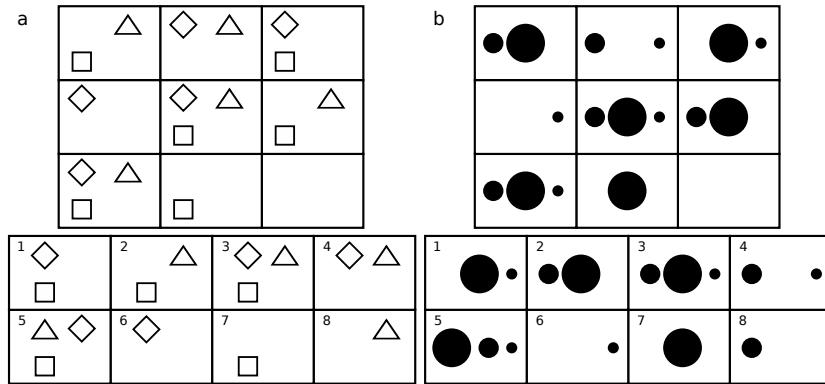


Fig. 1. Two functionally equivalent examples of 3×3 geometric reasoning problems. The task is to fill the empty cell with one of the four answers below. The correct answer for both problems is no. 4.

the second cell then it is in the third cell, the same holds if the object is not in the first cell but in the second cell. In all other cases the object is not in the third cell. A second explanation principle might deal with the number of objects per row/column: In both problems there are exactly two objects with identical attribute values in a row. This depicts a rule, which is applied by the model. [13] coined the term *distribution of two values*. In the examples mentioned there are exactly two associated objects in a row (e.g., two triangles in Problem 1a or two small circles in Problem 1b), the third object is null. Although the latter seems the easier explanation principle, it is not a stable pattern, i.e., if we remove additionally the last cell in the first column, the first explanation pattern allows still to fill in this cell, while the second might not.

Additional examples can be found in Figure 4. The reasoner's task is again to identify an implicitly given geometrical function associating the cells' objects and to apply this function to derive a solution. Does functional equivalence imply an equivalence in difficulty for humans? Two kinds of Raven's Progressive Matrices exist containing similar problems but differing in reasoning difficulty. The first test, the Standard Progressive Matrices (short: SPM), discriminates precisely within two standard deviations around the population's average intelligence. It consists of five sets with twelve problems each. The Advanced Progressive Matrices (short: APM), discriminates beyond these limits, i.e., it is a test for gifted adults. The test consists of two sets with 12 training problems and 36 test problems. The reasoning difficulty of the problems increases within a set; the difficulty of the sets altogether increases as well [15, p. 37].

Empirically the aforementioned Raven's SPM and APM are probably the best analyzed intelligence tests and in the associated manuals an empirical difficulty rating is given, i.e., the percentage of persons who have correctly solved

each problem. Our formal complexity measure is evaluated against these empirical difficulty measures. Another empirical investigation, which was recently performed for Evans Analogy problems [16, 17], serves as an additional benchmark.

Since Marr [18–20] cognitive approaches can be roughly divided into three levels: (1) the *computational level*, (2) the *algorithmic level*, and (3) the *implementational level*. The computational level specifies, in abstract terms, what is computed, i.e., what the characteristics of the problem are and the goal of the computation; the algorithmic level specifies how it is algorithmically realized, i.e., an algorithm transforming the input into the output; and the implementational level specifies how it is biologically realized, i.e., how the algorithm and representation are neurally implemented. In this article we concentrate on the computational and algorithmic levels or, in other words, on symbolic approaches.

Using theoretical background on item complexity and classification [21] generated a set of geometrical analogy test items. Thus, Embretson argues that her “Cognitive Design System Approach” is highly suitable to generate a relatively complete set of test items according to psychometric properties.

The difficulty of Raven’s Matrices items has been heavily studied (e.g., [22–24]. These findings indicate that number and type of transformations can contribute to item difficulty but that the connection is not linear in the number of factors. One of the most systematic analysis of intelligence test items’ complexity was performed by Primi [25]. He identified the following features: (i) The number of figures, (ii) the number and complexity of rules, and (iii) the perceptual complexity of the stimulus. There have been several development studies (e.g., [26, 27]). The latter analyzed children’s errors in Raven’s Progressive Matrices and traced the main cause of error back to the omission of complex transformations.

Solving matrix tasks requires the recognition and computation of similarities between the presented matrix objects and their properties. According to Representational Distortion (RD) theory, object representation similarity is defined by the number of basic transformations necessary to transform one representation into the other [28]. Hahn and colleagues defined the complexity of the similarity computation as a special case of Kolmogorov complexity (e.g., [29] and [30]). Instead of using the shortest program as a complexity measure, they used the transformational closeness, i.e., the length of the sequence of basic transformations [31]. The problem with such a measure is the tractability constraint. A transformation between two object representations can be noted as a binary string. The similarity can then be computed by a Boolean circuit but this results in superpolynomial complexity [32]. Thus, RD models would be psychologically implausible. Müller and colleagues argued for an analysis of restricted problem parameters to identify (in-)tractability in RD models.

Evans [16, p. 271] developed one of the first computer programs “*Analogy*” to solve simple geometrical analogy problems. This program represents geometrical objects symbolically. The problems were one-dimensional and hard-coded; and the solutions were chosen according to a topological and metric matching.

Following this research tradition Carpenter and colleagues [13] started to systematically analyze rules necessary to solve the Advanced Progressive Matrices test. They described similarity and analogy as invariants to morphisms respecting relations between symbolically represented knowledge. This led to the implementation of “FAIRAVEN” and “BETTERAVEN” by [13]. These two procedural models simulate the solution process of Raven’s *Advanced Progressive Matrices* to model the answer correctness of average and highly above average human adults. They identified a set of six general rules which are able to solve the Raven problems. Differences between the models depend on working memory limitations. They implemented these rules into the cognitive architecture 3-CAPS.

Further computational implementations of geometrical analogy building were based on the *Process of Structure-Mapping* as developed by Gentner and Markman [33]. A next level was reached when [34] implemented “CogSketch” a sketch understanding tool. A very general approach could subsequently be developed by [17] combining CogSketch [35] for visual analysis with the Structure Mapping Engine [36] to compute the analogies, to solve Evan’s problems using some general assumptions about human cognitive processes [17, p. 1194]. The new system used semantic and relational knowledge automatically generated by “CogSketch” to successfully solve sections B through E of Raven’s *Standard Progressive Matrices* [37]. Calculating the solutions to the set of problems used by [16]. Lovett and colleagues [17] state that they successfully simulate the answer correctness and reaction time of human adults. Problems that could not be solved were considered difficult for humans [37, p. 2765]. Recently, a program, which is able to solve 28 of the 36 SPM problems (Sets C to E) was developed by Cirillo and Ström [38, 39]. The program computed the solutions without considering the given possible answers [38, 39, p. 32]. However, their program does not solve arbitrary geometrical problems and they concentrated on solving the problems without (explicitly) relying on cognitive findings.

A previous study [40, 41] analyzed and classified geometrical analogy reasoning problems. Based on the types of functions necessary to solve these problems, a simple functional complexity measure is introduced, which for the simple problems is able to reliably capture human reasoning difficulty in Cattell’s Culture Fair Test and Evans’ Analogy problems, i.e., it is able to reproduce the difficulty ranking. A distinction between strong associations and weak associations can be made (cp. Fig 2). Strong association (red solid line) to identify transformation and weak association (depicted in red dashed) for object identification. The identified transformation on weakly associated stimulus is then applied to specify the attributes of the stimulus in the answer field.

Taken together, there are programs that try to solve the Progressive Matrices or similar problems optimally (which we call the classical AI-approach) [16, 38] and programs which solve them similarly to humans [37, 13]. None of these approaches have been applied to APM and SPM problems at the same time and none of these approaches uses working memory assumptions, makes response time predictions or can be generalized to arbitrary analogical problems (prob-

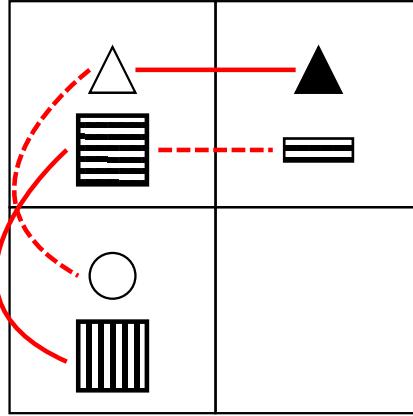


Fig. 2. A distinction between strong and weak associations can be made [40]. Strong Association (red solid) are used to identify transformation; Weak association (red dashed) to transfer transformations; identified transformations are applied on weakly associated stimuli to specify the attributes of the stimulus in the answer field

bly besides [37, 43, 44]). An excellent overview and classification of the existing computational models can be found in [43]. Correa and colleagues [44] took a new approach: Instead of applying proportions to feature-based descriptions of the pictures they applied them to a pixel level representation of the problems. This leads to a high prediction rate of the missing pattern without using any of the candidate solutions. Their approach can be seen as an explanation of reasoning based on visual representations.

Instead of translating the visual representation into an amodal propositional representation, as most of the previous approaches did, Kunda and colleagues [43, 45, 46] present a new approach using modal reasoning of visual representations. They apply affine and set transformations to these visual representations based on considerable evidence that humans apply such representations [43].

We decided to use the cognitive architecture ACT-R [42, 47] which is able to model both the formal and cognitive approach to overcome the gap between models that try to solve Raven's problems and those that try to solve them like humans do. Additionally, we aimed at a more generalized reasoner that is cognitively more adequate at the same time. ACT-R is a hybrid architecture and a production rule system (which is Turing complete [48]) consisting of a symbolic and a subsymbolic layer. A model, specified by production rules in the cognitive architecture, can predict error rates, response times and even the BOLD function [42]. Knowledge is represented by so-called chunks, which are n-tuples coding information about the objects and their relative positions in our model. Following [43]'s classification, ACT-R models would nearly always have an amodal propositional representation of information. ACT-R assumes a modular structure of the mind. Each module deals with specific kinds of information

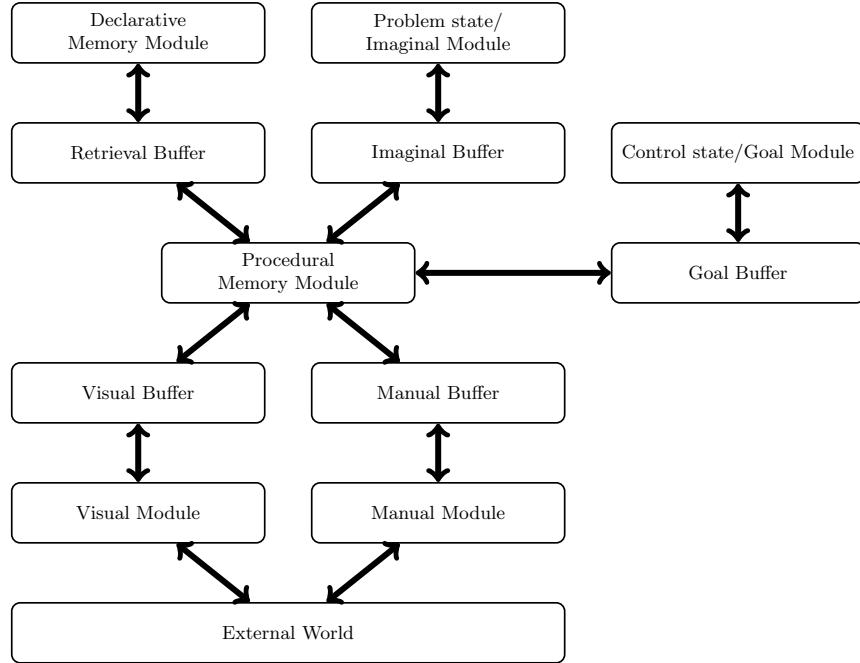


Fig. 3. A schematic representation of the modular structure of the cognitive architecture ACT-R [42]. ACT-R is a production rule system consisting of modules and buffers. The modules contain problem specific knowledge. Information between the modules is exchanged by using production rules.

and there are buffers attached to each module: the *visual module* deals with perceiving objects; the *goal module* is for general control issues; the *imaginal module* deals with problem specific representations; and the *declarative module* stores declarative knowledge. More information can be found in [42, 47]. Two further advantages are that chunk activation with a subsymbolic layer allows the model to remember the (correct) solution instead of just a random remembrance. The parameters allow for a better assignment of problem difficulty for humans. For instance, the *visual-num-finsts* parameter, which defines the number of objects the model can keep track of has to be higher for harder problems.

The remainder of the article is structured as follows: We will briefly present the rules, which are able to solve such problems, and our developed model implemented in the cognitive architecture ACT-R. This approach is then evaluated on Raven's SPM and APM. From this computational model we derive consequences for reasoning difficulty and the generation of functionally equivalent problems. An approach that has recently gained some attention (e.g., [49]). A brief summary concludes the article.

2 A Computational Model

2.1 Problem Attributes

A typical geometrical reasoning problem in IQ-tests, consists of cells containing geometrical objects. These objects can be classified by the following attributes: *shape* (e.g., triangle, circle etc.), *size* (e.g., represented numerically 1, 2, 3), *number of sides* (e.g., a triangle has 3 sides), *width, height* (for objects which are not of the same height and width, e.g., a rectangle), *color* (e.g., black, white etc.), *texture*, *line art* (e.g., solid, dotted), *rotation* (e.g., 0 degrees), *position* (inside a cell), and *quantity*. Consider the triangles in Figure 1a. The shape is *triangle*, size one, number of sides three, width one, height one, color *white*, texture *white*, line art *solid*, rotation zero, position two and quantity one.

To be able to compare the attribute values appearing in a row, the model stores the attribute name as well as the three appearing values in a chunk. If, for example, rectangles with a rotation of 0, 45, and 90 degrees are present in the first row (as in Figure 4b), the model creates a chunk of the type (*rotation*, 0, 45, 90) representing the change in the rectangles across the row.

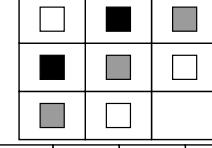
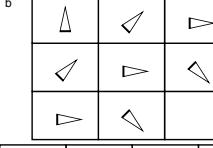
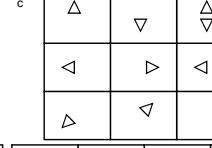
| | | | |
|---|---|--|--|
| a |  |  |  |
| 1 |     |     |     |
| 2 |     |     |     |
| 3 |     |     |     |
| 4 |     |     |     |
| 5 |     |     |     |
| 6 |     |     |     |
| 7 |     |     |     |
| 8 |     |     |     |

Fig. 4. Problems illustrating the rules from [13]. Problem 4a requires the rule *distribution of three values*. The correct solution is solution no. 4. In Problem 4b the rule *quantitative pairwise progression* is required. In this problem, the correct solution is no. 4. Problem 4c requires the rule *figure addition*. The correct solution is no. 8.

Some geometrical intelligence test problems have an increasing number of objects in a row, e.g., one triangle in the first cell and three in the second cell, in this case the model must conclude that it is likely that the next cell will contain 5 triangles. Consequently, we added different number series of two numbers each to the model's declarative memory. Each number series encodes if the series increases or decreases and the function. With the number series, the model is able to validate attribute values when applying *quantitative pairwise progression* (see below). So if a row contains objects with the sizes 1, 2 and 3, the model retrieves the number series 1, 2 and 2, 3, from declarative memory both of which contain the direction *increasing* and the factor +1.

The model also requires addition facts to be able to check the objects' quantity, if it assumes the rule *figure addition* (see below). If the cells in a row contain

two, one and three objects, the model retrieves the addition fact ($2 + 1 = 3$) to validate the rule. The objects are grouped according to their names appearing in the shape attribute which is in accordance to the *matching names heuristic* [13, p. 417]. This grouping of objects is given with the encoding of the problem, which means that each element contains a shape attribute that is identical in grouped elements.

2.2 The Model and the Applied Rules

Initially our ACT-R model randomly chooses an object in the first cell. Objects that share its form – and are thus grouped with it – are considered first. All other remaining objects are ignored. In the following, the associated object in the second cell is considered and then compared to the first object. If an attribute differs then the name of the attribute and the two existing values are stored. Then the third object is compared to the second object and its attribute value is also stored. Now the model assumes a rule using the stored values. In the following the second and third row are analyzed like the first row and, in addition, the attribute values are compared to the stored values to find possible discrepancies and another rule if required. In the third row, the missing attribute value is retrieved from declarative memory to construct the solution. If further attributes differ in the focused group of objects, the model repeats the procedure for each differing attribute and adapts the attribute value in the existing solution. In the following, possible further objects, ignored up to now, are analyzed as described. Finally, the model moves on to find the correct solution.

In the following we briefly introduce the rules we have implemented in our model to solve such geometrical problems. Five rules have been identified by Carpenter and colleagues [13, p. 408]; listed below, they are sufficient to solve almost all problems of the Progressive Matrices:

Rule 1: Constant in a row. The rule *constant in a row* is applied if all attributes of the objects are identical in a row. The objects can, of course, differ among the rows. The solution is identical to the objects in the last row. The model processes as follows: It checks if the objects differ in some attributes by means of a pairwise comparison of objects within each row.

Rule 2: Distribution of three values. The rule *distribution of three values* is applied, if an attribute of the objects in a row differs and the same three values of this attribute occur in each row. The order in which the objects' attribute values appear within a row is not relevant. In Problem 4a the attribute *color* differs, and its values occurring in each row are *black*, *grey* and *white*. The model stores the name of the differing attribute and the existing values in declarative memory. In the following rows the objects' attribute values are checked for concordance with the first row. The value that is missing in the third row is identified by comparing the stored values to the two values in the third row, and is then entered into the solution. In Problem 4a, a square with the missing color value *black* is created. Figure 5 shows the model's processing of a problem requiring *distribution of three values*.

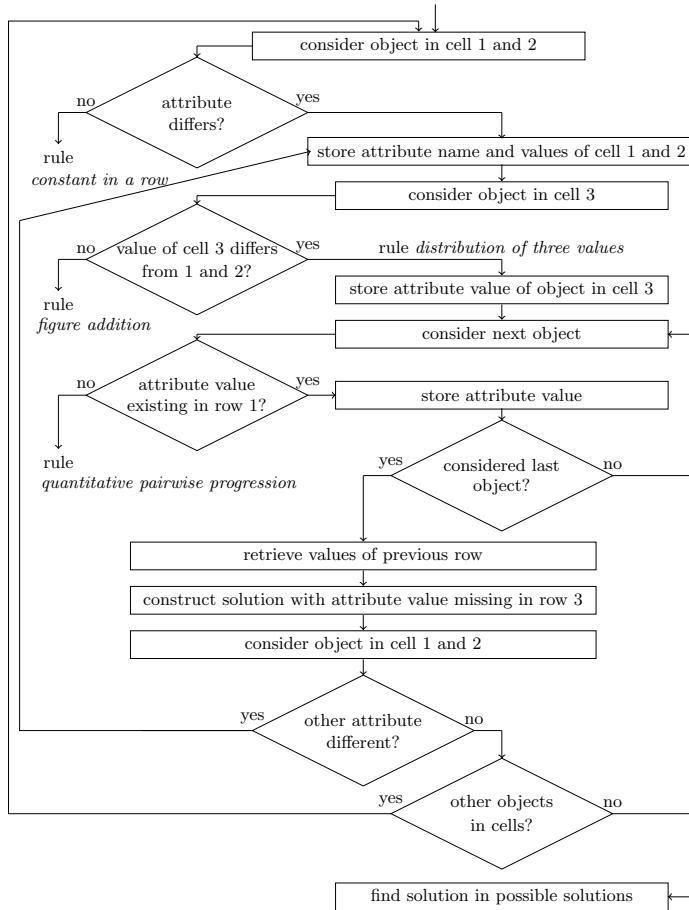


Fig. 5. Flow chart of the computational model processing a problem.

Rule 3: Quantitative pairwise progression. If the values of an attribute increase or decrease continuously in a row, *quantitative pairwise progression* can be applied. In Figure 4b the objects' rotation increases by 45 degrees in each cell. Once the model has associated the first and the second objects in a row, it retrieves a number series containing the changes of the attribute values. The information about the increase or decrease of the values and the functional change is then stored. Again, a matching number series is retrieved for the associated objects in the second and the third cells of this row. This procedure is repeated in every row. Problem 4b first requires a retrieval of a number series with the values 0 and 45 for the first two associated objects' rotation. Consequently, the direction is *increasing* and the function is +45 degrees.

Rule 4: Figure addition. If all objects of the third column appear in the first two columns in their respective rows, *figure addition* can be applied. An example is depicted in Figure 4c. If the considered objects are identical except for their quantities, the model adds the first two objects' quantities, resulting in the quantity of the last object. The model tries to retrieve an addition fact which contains the first two objects' quantities as addends and the quantity of the last object as sum. If additional object attributes differ, the model checks these attributes for each object, if it also appears in the third cell of the respective row; no addition facts are required. For each object in the last row, a separate solution is created. An example for this type of rule is given in Figure 4c.

Rule 5: Distribution of two values. For the model, there are two characteristics of this rule. On the one hand, two identical objects are present in each row, therefore one of the cells is empty. If in the last row a single object is present, the model constructs a solution identical to this object. If two objects are present, no solution is created. In Figure 1a, an example for this characteristic of the rule can be found. In each row there are two diamonds, triangles and squares. The model also assumes *distribution of two values*, if there are two equal and one differing attribute values in a row. This characteristic of the rule is required in Problem 6a. The model stores the existing attribute values, which are *black* twice and *white* once. In the second and third row the model checks whether the same values appear once or twice. To construct the solution, the missing value is identified by comparing the two values in the third row and the stored values. In the example the missing texture value is *black*.

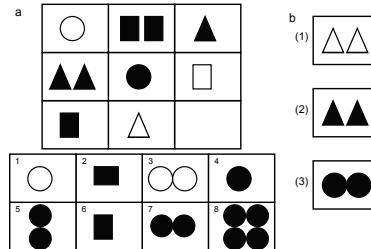


Fig. 6. The model processes Problem 6a several times. Analogously to the human reasoner, it considers one attribute at a time. Figure 6b depicts the solution created or adapted in each run. The model first considers the quantity and constructs a solution (1), adapts the texture value in this solution (2) and finally adapts the number of sides (3). There is no predefined order in which to consider the attributes. The correct solution is no. 7.

Note that the description of the rules is focused on rows. These rules can also be applied to columns. However, to solve the SPM and APM problems, the application to rows is sufficient. Hence, the application to columns is not implemented in the model.

Some problems contain ambiguous object arrangements which can be solved with different rules. For example, if one object is present in the first cell, two objects in the second cell and three objects in the third cell of a row, both the *quantitative pairwise progression* with increasing quantity and the *distribution of three values*, with the quantity values 1, 2 and 3, rules can be applied.

2.3 The Evaluation of the System

We briefly report our evaluation. The model was tested on different sets of problems which are functionally equivalent to Set II of the Advanced Progressive Matrices and Sets C through E of the Standard Progressive Matrices. The reason for using functionally equivalent problems lies in the restricted visual features of ACT-R.

Altogether the model solved 66 of the 72 tested problems. Since ACT-R is able to simulate and vary human-like features such as the ability to forget identified patterns, we eliminated these features, which allows a more deterministic processing of the model. This in turn can be used to test how many problems can be solved overall. The model solves 98.9% of the 66 problems correctly, since different production rules can still be in conflict and rarely lead to incorrect answers for some problems. The parameter values to achieve the described performance are the retrieval threshold of -1 and the *visual-num-finsts* parameter of 35. To achieve the mentioned result, the subsymbolic layer of ACT-R was enabled so that the model can remember the solution that was created later. Altogether the model is able to solve 90.7% of all 72 tested SPM and APM problems.

2.4 Consequences for the Analysis of the Reasoning Complexity

The complexity of each problem is composed of several criteria. One behavioral finding, which goes along with computational results, is that subjects' error rates of the subjects depend on the number of rules required [13, p. 411]. Hence, problems which are simple for subjects require fewer rules; but difficult problems require up to eight rules. Especially the number of rules directly required to create a solution influences the difficulty of a problem, because creating a single incorrect solution leads to a wrong answer. In addition, rules that can only be applied to the first and the second rows or do not lead to the creation of a solution (e.g., the rule *distribution of two values*) influence the problem difficulty. For example, previously created solutions can be forgotten while processing these rules.

Subjects apply these rules in a prioritized order. The description of the rules above follows this order, starting with the most simple rule *constant in a row* to the most difficult, *distribution of two values*. Carpenter and colleagues identified the principle of prioritized order from different aspects, such as simplicity of *constant in a row*, the preference of *figure addition* over *distribution of two values* and the preference of *quantitative pairwise progression* over *distribution of two values* [13, p. 421]. Because for our model the rule *distribution of three values* requires less usage of the declarative memory than *quantitative pairwise*

progression and is thus more simple to apply, our model deviates from Carpenter and colleagues' order, who preferred *quantitative pairwise progression* over *distribution of three values* [13, p. 421]. The deviant order also achieves a higher correlation with subjects' data.

The difficulty of a rule is composed of several factors. First, the more declarative retrievals a rule requires, the more errors can occur. For example, stored values cannot be remembered or incorrect values are retrieved. A higher number of objects covered by one rule causes the neglect of some of these objects, which can lead to the application of an incorrect rule. Ambiguous situations evoke incorrect rules, leading to discrepancies in the further processing. These ambiguous situations and consequent discrepancies also depend on the number of objects covered by one rule: Depending on the objects neglected, ambiguous situations are evoked.

How many rules are applied to a group of objects poses another criterion. The created solution has to be adapted for each additional rule applied to the same group of objects. Furthermore, the solution has to be retrieved, which can lead to errors.

Another aspect of reasoning difficulty is to associate the objects covered by one rule, if the cells of a problem contain more than one object. The association is clear in most of the problems, however, in some APM problems the association can be difficult. The model groups the objects according to the *matching names heuristic* [13, p. 417]. Hence, objects with the same name are grouped, all squares as well as all diamonds and all triangles in Problem 1a for example.

Identification and visual aspects, such as the perception of objects, poses another aspect of difficulty (e.g., is it a square or four single lines arranged to a square?). Moreover, object transformations among the cells can be ambiguous (e.g., considering two squares, one can be rotated by 90 degree, rotated by 180 degree, mirrored, or there can be no difference). The total number of objects in a problem yields another difficulty. A reasoner has to keep track of all the identified patterns. In problems containing many objects, this is an unusually great amount of information. In ACT-R, the number of objects that the model can keep track of is reflected by the *visual-num-finsts* parameter, which has to be set to 32, whereas the standard value is four. Storing the current x-coordinate and thus forcing the model to consider the objects in the order of an increasing x-coordinate can avoid this high parameter setting. The disadvantage of this solution is that it pretends a fixed order in which the model has to consider the objects. Taken together there are several aspects that can make such problems difficult: First, the application of an incorrect rule can be caused by ambiguous situations which call for several production rules. Second, retrieval errors can cause incorrect results or an already computed solution can be no longer retrievable. Third, visual difficulties can occur, such as the association of objects in different cells, the recognition of objects and the changes of objects among cells. Altogether, the reasoning difficulty our proposed ACT-R model considers the complexity of a problem by the following factors:

- the number of rules pertinent for the solution,

- the number of rules which do not directly influence the solution,
- the difficulty of the rules, consisting of
 - the number of retrievals from declarative memory,
 - the number of objects covered by this rule,
 - the number of ambiguous situations and differences or discrepancies;
- the number of rules for a group of objects and
- the total number of objects.

Additional differences that arise for humans are (which are not reflected in the model):

- difficulty identifying different objects as the same,
- ambiguous visual object changes, and
- the actual identification of objects.

3 Generating Functionally Equivalent Problems

Once we have a computational model and have identified the best fitting parameters, we can try to develop systematically functionally equivalent problems based on the structure of the problems. The number of objects of a functionally equivalent problem is identical to the number of objects of the original Progressive Matrices problem. Type and number of rules required to solve the task are also identical. The objects' appearance and the types of attributes varied are different than in the original problems. An example for two equivalent problems can be found in Figure 1. In this example, the original problem (depicted in Figure 1a) requires the rule *distribution of two values* applied to triangles, squares and diamonds with a particular position. The equivalent problem (depicted in Figure 1b) requires the same rule, applied to large, medium and small circles with a particular position. Also, instead of size or position, another attribute like rotation could be varied, since the model's performance is equal for each attribute. Figure 7 depicts three problems, that are all functionally equivalent. The required rules (*distribution of three values* twice and *constant in a row*) are identical. *Distribution of three values* is applied to the rotation of the rectangles in Problem 7a; to the rotation of the black triangles in Problem 7b and to the texture of the triangles on the left in Problem 7c. The same rule is also applied to the rotation of the lines in Problem 7a, to the white triangles' rotation in Problem 7b as well as to the rotation of the triangles on the right in Problem 7c. Note that the rotation values are not the same in each task. In addition, the rule *constant in a row* is applied to the lines' quantity in Problem 7a, to the white triangles' quantity in Problem 7b and to the texture of the triangles on the right in Problem 7c. The performance of the model is equal for all of these problems. Hence, performance is not influenced by attribute type or specific attribute values – as long as their relations are the same. Visual difficulties like the similarity of objects, were disregarded as ACT-R does not take into account the visual similarity of objects.

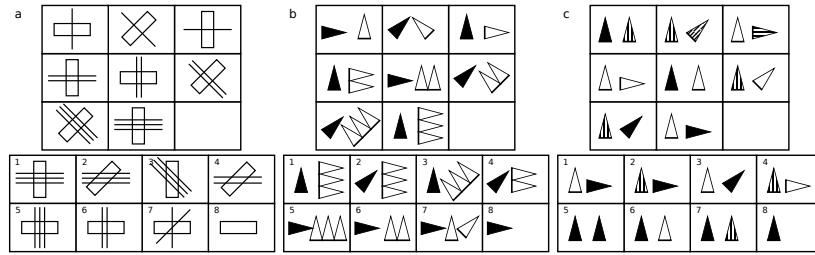


Fig. 7. Three functionally equivalent problems. The black triangles in Problem 7b refer to the rectangles in Problem 7a, and the white triangles to the lines. In Problem 7c, the triangles on the left refer to the rectangles in Problem 7a, and the triangles on the right refer to the lines. The correct solution is no. 5.

4 Evaluating Model Predictions Experimentally

We ran an experiment to have access to response times and individual error rates about the performance of subjects on the equivalent problems. The subjects were tested on both the APM and the equivalent problems.

4.1 Method

Participants. We tested 17 students (11f, $M = 25.4$, $SD = 8.89$). They received course credits for their participation.

Materials, Design & Procedure. The problems presented were the 36 Raven’s Advanced Progressive Matrices Set II and 36 problems that are functionally equivalent to an associated problem of the original APM. Participants had to work through the trial problems of the classical APM. These problems could be solved by completing a visual pattern and thus did not require one of the rules above.

The experiment was split into two sessions. In the first session, half of the subjects were presented with the original APM problems, while the other subjects were presented with the analogous APM problems; they are equivalent regarding the underlying mathematical functions, but controlled regarding their stimuli. In the second session, the subjects were presented with the test they did not process in the first session so that each participant worked on both the original APM and the analogous APM problems. In each session the subjects had a time limit of 40 minutes to solve the problems; all problems were available during the experiment, so that subjects were able to correct solutions or skip a problem for the moment. The response time for each equivalent problem was recorded, since the equivalent problems were presented digitally. The subjects first were given instructions and five test problems.

Table 1. Correlations of subjects' data in our experiment, the model and data collected by Heller and colleagues [50].

| | Mean error rate | Deviation | Pearson's Correlation with [50] | Pearson's Correlation with our model |
|-----------------|-----------------|-----------|---------------------------------|--------------------------------------|
| APM | 9.06 | 4.88 | $r = .85$ | $r = .739$ |
| Analog Problems | 4.94 | 2.47 | $r = .72$ | $r = .787$ |

4.2 Results

In the 36 APM problems the number of errors a subject made ranged from two to 20, with a mean of 9.06 and a deviation of 4.88. The results of our subjects and Heller and colleagues' subjects [50] correlate with $r = .85$.

The subjects' error rate ranged from two to 13 in the equivalent problems, with a mean of 4.94 and a standard deviation of 2.47. Heller and colleagues' data [50] correlated with our data with $r = .72$. An overview over the results and correlations can be found in Table 1. On average the analog problems were solved correctly slightly more often than the classical APM problems (85% or 30,06 problems vs. 76,5 % or 27,5 problems on average).

There is, as expected, no effect between first presenting the classical APM ($N = 12$: $M = 26, 58$) and then the analogous problems ($M = 28, 55$; $t(21) = -1, 216$, $p = n.s.$), or vice versa. The same holds for response times Raven ($M = 42178$ ms) vs. analogous first ($M = 42903$ ms; $t(21) = -0, 223$, $p = n.s.$). Taken together there is – as expected – no significant training effect.

The correlations for the accuracy are highest between the analogous APM problems and the ACT-R model ($r = .787$, $p < .001$; $df = 29$). For the APM problems and the ACT-R model it is ($r = .739$, $p < .001$; $df = 29$). A similar pattern holds for the response times between analogous APM and the ACT-R model ($r = .623$, $p < .001$; $df = 29$). The results indicate that the visual complexity involved in solving Raven's problems is smaller than the functional difficulty. In other words, the difficulty of the problems depend rather on the underlying mathematical function and not so much on the presentation of the stimuli.

5 Summary & Discussion

Identifying patterns in reasoning problems about geometrical structures still poses challenges for computational systems and there are only few approaches which are able to explain the differences in human processing. Previous approaches, most of them purely symbolic, concentrated on solving problems optimally. Clearly, formal complexity measures (Turing complexity), which are asymptotic in nature, cannot be used to classify the reasoning difficulty of such small problems. We developed a computational system in the cognitive architecture ACT-R to capture peculiarities of human processing from attention to memory allocation, and additionally to have a Turing complete production rule

system [48] extended with subsymbolic computations to perform the necessary computations.

Consequently, we used the ACT-R model as a computational model to determine reasoning difficulty. The structure of ACT-R consists of different modules and buffers in contrast to a Turing machine.

The differences of our ACT-R model to the very similar models ‘FAIRAVEN’ or “BETTERAVEN” can be summarized as follows: First, we have a general enough model for both APM and SPM problems while Carpenter and colleagues only dealt with the APM. Second, we can model psychological findings and, by switching parameters can use the model to solve the problems optimally while Carpenter’s model concentrates on the necessary rules. Third, we are able to compute exact error rates and response times while Carpenter’s model dealt only with general rules that can solve problems. Finally, our ACT-R model allows the classification of the number of visual elements that have to be compared simultaneously. Additionally, we tested which parameter settings are necessary to solve each problem, e.g., how many objects must be stored in visual memory to solve the problems. This is represented by the *visual-num-finsts* parameter. Overall, the model is able to solve 66 of the 72 problems.

In contrast to classical approaches, a cognitive model in ACT-R does not adhere to a complexity formula. As described above, ACT-R uses production rules and depending on the execution time of each production rule, the time it needs to store the information in the declarative memory or to manipulate the visual and other buffers allows a determination of error rates and latencies. Finally, the way ACT-R controls the attention shift explains why specific aspects are inspected and others are not. This is the central aspect, namely that besides the formal difficulty and the visual difficulty a working memory processing difficulty comes into play: The way information is processed in a modular structure must be considered as well and allows for a suitable comparison with human findings about difficulty from [50] and our own experiment. To put this in other words the cognitive bottlenecks of a cognitive architecture such as ACT-R, i.e., that any buffer can contain only one chunk or that only one production rule can fire at a time can contribute to the difficulty. Thus is the third component besides the comparison with formal (as analyzed by [16, 13] and others) and visual difficulty (as dissected by [25] and others). To put this in other words: The interplay between the modules – the architecture of information processing should be considered as well.

One limitation of our approach is that we do not take visual perception into account, a problem that is partially due to the restricted visual processing abilities of ACT-R. Furthermore our modeling is mostly symbolic (with subsymbolic aspects such as working memory activations. Integration of more perception based representations (e.g., [44, 43]) or “CogSketch” [34] could be interesting future directions to follow. In fact this would allow for a more complete cognitive model taking the visual perception into account and providing a connection between different representational levels especially combining the symbolic with the implementational level (see above).

Such a characterization of problems (as above) together with another computational model [40] has been used to generate functionally equivalent Raven like IQ-test problems. In contrast to classical problems, the generated equivalent problems only use simple well-known shapes, like circles, squares or triangles; Each object can be uniquely identified. This effect also decreases the visual or perceptual ambiguity in object recognition. In fact, in a second project, we generated different classes of problems. They can be found on a website.²

It is correct that the object recognition (i.e., what is part of an object or at least perceived as one and what is not) is already encoded. We believe that this is part of the perceptual process that is unfortunately underdeveloped in ACT-R. Regarding the visual perception ACT-R is actually more a theory of attention than of perception: Which features are attended when, which features are processed when and so on. And this is one part of visual complexity – namely that of visual attention. We conducted an empirical study to test these functionally equivalent problems for error rates and reaction times. The experiment above shows that using simpler equivalent problems leads to lower error rates than the original APM problems. Since the kind and number of rule applications are identical this impact can be at least partially traced back to the visual changes.

Although human intelligence has now been scientifically investigated for about 100 years, there are still open questions: For instance, how do we learn to recognize underlying mathematical functions? Is this easier if the functions are similar to already known functions? Is it possible to dissect the contributions of the computational aspects, the visual aspects, and the aspects due to the modular structure of the mind? Such questions will be investigated in future research.

Acknowledgements

MR's research was supported by the German Research Foundation (DFG) in the Transregional Collaborative Research Center, SFB/TR 8 within projects R8-[CSPACE] and by a grant from the DFG within the priority program "New Frameworks of Rationality" (SPP 1516). The authors are grateful to Stephanie Schwenke for proof reading and for various discussions above such topics with Philip Stahl.

References

1. Gardner, H.: *The mind's new science: A history of the cognitive revolution*. Basic Books (1987)
2. Simon, T., Binet, A.: Méthodes nouvelles pour le diagnostic du niveau intellectuel des anormaux. *L'année psychologique* **11**(1) (1904) 191–244
3. Wechsler, D.: *The Measurement of Adult Intelligence*. Williams & Wilkins, Baltimore (MD) (1939)

² www.iqcoach.de

4. Cattell, R.B.: Abilities: Their structure, growth, and action. Houghton Mifflin, New York (1971)
5. Raven, J.C.: Mental tests used in genetic studies: The performance of related individuals on tests mainly educative and mainly reproductive. Master's thesis, University of London (1936)
6. Newell, A., Simon, H.A.: Human problem solving. Prentice-Hall (1972)
7. Cattell, R.: Theory of fluid and crystallized intelligence: A critical experiment. *Journal of educational psychology* **54**(1) (1963) 1–22
8. Cattell, R.: Are IQ tests intelligent? *Psychology today* **1**(10) (1968) 56–62
9. Wechsler, D., Hardesty, A., Lauber, H., Bondy, C.: Die Messung der Intelligenz Erwachsener: Textband zum Hamburg-Wechsler-Intelligenztest für Erwachsene (Hawie). H. Huber (1961)
10. Cattell, K., Cattell, A.: Culture fair intelligence test. Institute for Personality and Ability Testing (1959)
11. Raven, J.C., Raven, J.C., Court, J.H.: Manual for Ravens Progressive Matrices and Vocabulary Scales. Oxford Psychologists Press, Oxford (2000)
12. Raven, J.C.: Advanced Progressive Matrices, Set II. H. K. Lewis, (Distributed in the United States by The Psychological Corporation, San Antonio, TX), London (1962)
13. Carpenter, P.A., Just, M.A., Shell, P.: What One Intelligence Test Measures: A Theoretical Account of the Processing in the Raven Progressive Matrices Test. *Psychological Review* **97**(3) (1990) 404–431
14. Snow, R.E., Kyllonen, P.C., Marshalek, B.: The topography of ability and learning correlations. *Advances in the Psychology of Human Intelligence* **2** (1984) 47–103
15. Heller, K.A., Kratzmeier, H., Lengfelder, A.: Matrizen-Test-Manual, Band 1. Ein Handbuch mit deutschen Normen zu den Standard Progressive Matrices von J. C. Raven. Beltz Test, Göttingen. (1998)
16. Evans, T.G.: A Program for the Solution of a Class of Geometric-Analogy Intelligence-Test Questions. In: Semantic Information Processing. Marvin L. Minsky (1968) 271–351
17. Lovett, A., Tomai, E., Forbus, K., Usher, J.: Solving Geometric Analogy Problems Through Two-Stage Analogical Mapping. *Cognitive Science* **33**(3) (2009) 1192–1231
18. Marr, D.: Vision: A Computational Investigation into the Human Representation and Processing of Visual Information. Freeman, New York (1982)
19. McClamrock, R.: Marr's three levels: A re-evaluation. *Minds and Machines* **1**(2) (1991) 185–196
20. Pylyshyn, Z.: Computation and cognition. MIT Press, Cambridge, MA (1984)
21. Embretson, S.: A cognitive design system approach to generating valid tests: Application to abstract reasoning. *Psychological Methods* **3**(3) (1998) 380–396
22. Green, K.E., Kluever, R.C.: Components of item difficulty of ravenís matrices. *The Journal of General Psychology* **119**(2) (1992) 189–199
23. Mulholland, T.M., Pellegrino, J.W., Glaser, R.: Components of geometric analogy solution. *Cognitive Psychology* **12** (1980) 252–284
24. Bethell-Fox, C.E., Lohman, D.F., Snow, R.E.: Adaptive reasoning: Componential and eye-movement analysis of geometric analogy performance. *Intelligence* **8** (1984) 205–238
25. Primi, R.: Complexity of geometric inductive reasoning tasks: Contribution to the understanding of fluid intelligence. *Intelligence* **30**(1) (2002) 41–70
26. Stone, B., Day, M.C.: A developmental study of the processes underlying solution of figural matrices. *Child Development* **52** (1981) 359–362

27. Vodegel Matzen, L., Van der Molen, M., Dudink, A.: Error analysis of raven test performance. *Personality and individual differences* **16**(3) (1994) 433–445
28. Chater, N., Hahn, U.: Representational distortion, similarity and the universal law of generalization. In: Proceedings of the similarity and categorization workshop 97, University of Edinburgh (1997) 31–36
29. Chater, N., Oaksford, M.: The rational analysis of mind and behavior. *Synthese* **122** (2000) 93–131
30. Li, M., Vitanyi, P.: An introduction to Kolmogorov Complexity and its Applications. Springer (1997)
31. Hahn, U., Chater, N., Richardson, L.B.: Similarity as transformation. *Cognition* **87**(1) (2003) 1–32
32. Müller, M., van Rooij, I., Wareham, T.: Similarity as tractable transformation. In Taatgen, N., van Rijn, H., eds.: Proceedings of the 31st Annual Meeting of the Cognitive Science Society. Cognitive Science Society (2009) 49–55
33. Gentner, D., Markman, A.B.: Structure mapping in analogy and simmilarity. *American Psychologist* **52**(1) (1997) 45–56
34. Forbus, K., Usher, J., Lovett, A., Lockwood, K., Wetzel, J.: CogSketch: Open-domain sketch understanding for cognitive science research and for education. In Alvarado, C., Cani, M.P., eds.: Proceedings of the Fifth Eurographics Workshop on Sketch-Based Interfaces and Modeling. (2008)
35. Forbus, K., Usher, J., Lovett, A., Lockwood, K., Wetzel, J.: CogSketch: Open-domain sketch understanding for cognitive science research and for education. In Alvarado, C., Cani, M.P., eds.: Proceedings of the Fifth Eurographics Workshop on Sketch-Based Interfaces and Modeling. (2008)
36. Falkenhainer, B., Forbus, K.D., Gentner, D.: The Structure-Mapping Engine: Algorithm and Examples. *Artificial Intelligence* **41** (1989) 1–63
37. Lovett, A., Forbus, K., Usher, J.: A Structure-Mapping Model of Raven’s Progressive Matrices. In Ohlsson, S., Catrambone, R., eds.: Proceedings of the 32nd Annual Conference of the Cognitive Science Society, Cognitive Science Society (2010) 2761–2766
38. Cirillo, S., Ström, V.: An Anthropomorphic Solver for Raven’s Progressive Matrices. Master’s thesis, Department of Applied InformationTechnology, Chalmers University, Goeteborg, Sweden (2010)
39. Strannegård, C., Cirillo, S., Ström, V.: An anthropomorphic method for progressive matrix problems. *Cognitive Systems Research* **22–23** (2013) 35–46
40. Stahl, P., Ragni, M.: Complexity in analogy tasks: An analysis and computational model. In Coelho, H., Studer, R., Wooldridge, M., eds.: ECAI. Volume 215 of Frontiers in Artificial Intelligence and Applications., IOS Press (2010) 1041–1042
41. Ragni, M., Stahl, P., Fangmeier, T.: Cognitive complexity in matrix reasoning tasks. In Kokinov, B., Karmiloff-Smith, A., Nersessian, N.J., eds.: European Perspectives on Cognitive Science, Sofia, New Bulgarian University Press (2011)
42. Anderson, J.R.: How can the human mind occur in the physical universe? Oxford University Press, New York (2007)
43. Kunda, M., McGregor, K., Goel, A.K.: A computational model for solving problems from the raven’s progressive matrices intelligence test using iconic visual representations. *Cognitive Systems Research* **22–23** (2013) 47–66
44. Correa, W.F., Prade, H., Richard, G.: When intelligence is just a matter of copying. In Raedt, L.D., Bessière, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F., Lucas, P.J.F., eds.: ECAI. Volume 242 of Frontiers in Artificial Intelligence and Applications., IOS Press (2012) 276–281

45. Kunda, M., McGreggor, K., Goel, A.K.: Reasoning on the raven's advanced progressive matrices test with iconic visual representations. In: 34th Annual Conference of the Cognitive Science Society, Portland, OR (2012) 1828–1833
46. Kunda, M., McGreggor, K., Goel, A.K.: Taking a look (literally!) at the raven's intelligence test: Two visual solution strategies. In: 32nd Annual Conference of the Cognitive Science Society, Portland, OR (2010) 1691–1696
47. Anderson, J.R., Bothell, D., Byrne, M.D., Douglass, S., Lebiere, C., Qin, Y.: An integrated theory of the mind. *Psychological Review* **111**(4) (2004) 1036–1060
48. Anderson, J.R.: *The Architecture of Cognition*. Harvard University Press, Cambridge, MA (1983)
49. Matzen, L.E., Benz, Z.O., Dixon, K.R., Posey, J., Kroger, J.K., Speed, A.E.: Recreating raven's: software for systematically generating large numbers of raven-like matrix problems with normed properties. *Behavior Research Methods* **42**(2) (2010) 525–541
50. Heller, K.A., Kratzmeier, H., Lengfelder, A.: *Matrizen-Test-Manual, Band 2. Ein Handbuch mit deutschen Normen zu den Advanced Progressive Matrices von J. C. Raven*. Beltz Test, Göttingen. (1998)

Perceptual Similarity and Analogy in Creativity and Cognitive Development

Georgi Stojanov¹ and Bipin Indurkhya²

¹The American University of Paris 147, rue de Grenelle, 75007, Paris, France
gstojanov@aup.edu

²Computer Science department, AGH University of Science and Technology, Cracow, Poland
bipin@agh.edu.pl

Abstract We argue for the position that analogy represents the core mechanism in human cognitive development rather than being a special cognitive skill among many. We review some developmental psychology results that support this claim. Analogy and metaphor, on the other hand, are seen as central for the creative process. Whereas mainstream research in artificial creativity and computational models of reasoning by analogy stresses the importance of matching the structure between the source and the target domains, we suggest that perceptual similarities play a much more important role. We provide some empirical data to support these claims and discuss their consequences.

Keywords: Creativity, Analogy, Perceptual Similarity, Cognitive Development

1 Introduction

Analogy, together with its close cousin metaphor, is considered, by some accounts, to be fundamental to thought itself [20, 45, 47, 85]. Similarities and analogies are also known to play a key role in creativity [36, 86, 87]. Here, although many studies of creativity emphasize that one of the underlying mechanisms is that of re-conceptualization [13, 52, and to some extent 29 through Karmiloff-Smith's notion of representational-redescription processes in human cognitive development] most artificial creativity systems focus on generating a product that is considered to be creative rather than focusing on the process by which the artifacts considered to be creative are generated and interpreted. For example, one hypothesis is that one of the key processes underlying creativity is that of re-conceptualizing a given object or situation [55, 88, 89]. To our knowledge, there have been only a few attempts to model this process of re-conceptualization and emerging properties [15, 16], [21] and [37]. Moreover, computational models of analogy largely consider it as a special cognitive skill or heuristic that is evoked in some situations on top of other cognitive processes.

The aim of this paper is to articulate a view that sees analogy as a fundamental mechanism of cognitive development, and examine the implications of this view for modeling creativity. The paper is organized as follows. We start in Sec.

2 with a discussion of analogy and creativity; in Sec. 3 we briefly review the computational approaches to modeling analogy and comment on their shortcomings. We continue in Sec. 4 by discussing the limitations of the existing artificial creativity systems. In Sec. 5, we present our framework by connecting analogy with Piaget's mechanisms of assimilation and accommodation. In Sec. 6, we discuss some implications of this framework for the role of similarity in analogy and creativity. Finally, in Sec. 7 we summarize our main conclusions.

2 **Analogy in Creativity**

Analogy has been recognized as a key mechanism of creativity [4], [13], [15, 16], [31], [36], [42], [90, 91]. However, one must distinguish between two modes of analogy. On one hand, analogy refers to ‘seeing one thing as another’, and on the other hand it refers to the process whereby the structure and the attributes of one object or situation (the source) are mapped to another object or situation (the target). This latter mechanism seems contrary to creativity according to many accounts [24], and so it needs a little elaboration.

Every conceptualization/categorization (of objects, situations, visual scenes) involves loss of some potential information: potential differences are ignored between two objects that are put in the same category, and potential similarities are ignored between two objects that are put in different categories. The concepts and categories, and their underlying cognitive structures that naturally evolve through a cognitive agent’s interaction with the environment, reflect the priorities of the agent. The information that is retained in the conventional conceptualization is the one that has been useful to the agent in its phylogenetic and ontogenetic past. So, as long as one stays in the familiar domain (in which the conventional conceptualizations are very useful), and the problem at hand does not require the potentially lost information, reasoning from conventional operations and conceptualizations may be very efficient. However, as soon as the problem does require new information, the existing conceptualization stops being useful, and a new conceptualization becomes necessary. In such situations, analogy, as it is traditionally construed, becomes a hindrance, because it reinforces the existing conceptualization; and metaphor becomes a very useful heuristic. (See also [13], [20], [23].)

If we follow this argument, analogy, in its traditional sense at least, which is based on structural similarities, turns out to be an anathema to creativity. The reason is that analogies are based on mapping the structure or attributes of the source, to the structure and attributes of the target. So an analogy, which is based on the existing conceptualization of the source, will retrieve sources that are similar to the structure, thereby further strengthening the existing conceptualization of the target. In some cases, this may be enough to solve the problem, e.g. by bringing to the forefront some of the less prominent attributes of the target. But if the problem could not be solved because of needing new information, then an analogy-based approach is not be very useful.

3 Computational Models of Analogy

Much of the research on computational modeling of analogy has worked with what might be characterized as *mapping-between-existing-representations* paradigm, where there are given representations of the source and the target, and various algorithms are applied for mapping parts of the source to parts of target. Models differ from one another with respect to whether mapping between relations is preferred over attributes; or whether an incremental or a distributed approach is applied to compute the mapping [11], [9], [17], [18, 19]. All these approaches have severe limitations in that they cannot model emergence of new structure, which is very crucial as far as creativity is concerned. (See [7] for a critical overview.) In what follows, we give a detailed description of one of the influential theories that tries to explain reasoning by analogy, and which has been computationally modeled as well.

We illustrate this theory, known as the Structure Mapping Theory (SMT), with an example taken from [9]. In Figure 1 (a) we have a situation that involves two containers filled with water to different heights, and connected with a pipe.

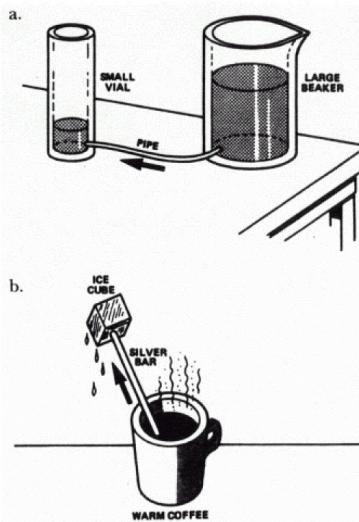


Figure 1 Two analogous situations

The higher water level in the large beaker leads to a pressure difference, which makes the water flow from the beaker into the small vial through the pipe. In Figure 1(b), on the other hand, we have a cup of warm coffee, and a silver bar, on one end of which is attached an ice cube, while the other end is submerged in hot coffee. Because of the difference in temperature, the heat flows from the coffee to the ice cube.

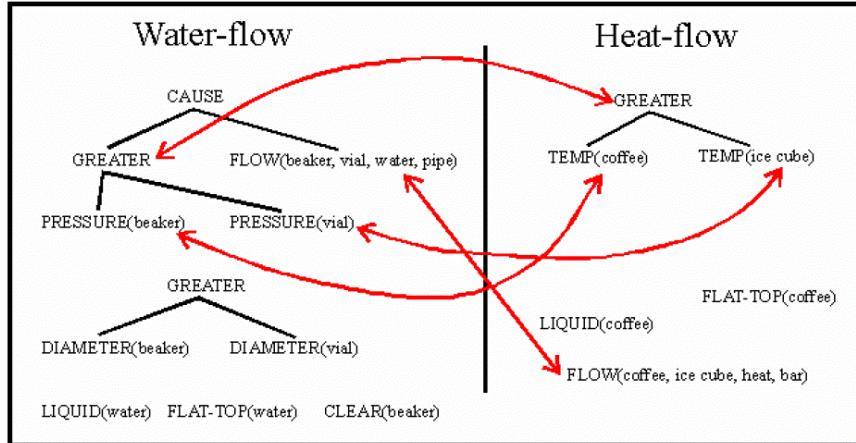


Fig. 2. Mapping between the representations of the two situations shown in Fig. 1 according to SMT (Adapted from Blank [61])

Now, according to SMT we have stored representations of these two situations from the real world, and they might look like the ones shown in Figure 2. On the left we have a representation of the WATER-FLOW situation and on the right-hand side a representation of the HEAT-FLOW situation. SMT predicts that structural mappings depicted in the figure is preferred and leads to the following analogies: pressure \leftrightarrow temperature, beaker \leftrightarrow coffee, vial \leftrightarrow ice, water \leftrightarrow heat, pipe \leftrightarrow silver bar, and, of course, flow \leftrightarrow flow. The *systematicity principle* of SMT claims that structural mappings that include higher-order relations, such as causality, are preferred. So, in this example, **greater (temp (coffee), temp (ice-cube))** is mapped onto **greater (pressure (beaker), pressure (vial))** rather than onto **greater(diameter (beaker), diameter (vial))**, the latter relation being an isolated one.

SMT takes an objectivist view since ‘reality’ is represented via some canonical symbolical representation (predicate calculus formulæ in the above example) to the cognitive agent, who is supposed to reason analogically. As such, finding an analogy is reduced to searching through all the possible mappings with the constraint expressed in the systematicity principle. It has been criticized for neglecting the actual phase of building relevant representations, i.e. the perception part in the process [7].

Though these models do capture certain aspect of creativity in noticing new connections between existing knowledge, and in importing novel hypotheses from the source to the target, they do not produce a Kuhnian paradigm shift. Moreover, using hand-coded representations that are context independent makes it impossible to account for emerging properties.

In this regard, models based on corpus-based analyses and distributed representations seem more promising [56, 59], but so far they are limited to linguistic metaphors.

In contrast, some other approaches have focused on the process of representation building itself, notable among them being the works of Hofstadter *et al.* [15], [34], O’Hara *et al.* [38, 39], Dastani *et. al.* [97, 98, 99] and, more recently, of Schwering *et al.* [92]. In this paradigm, the appropriate representations of the source and the target and the mapping between them evolve together by parallel processes that interact with each other.

In what follows, we will illustrate the idea of *fluid representations* by giving more details on Hofstadter *et al.*’s Copycat program [62, 63], [15]. It is designed to model analogies in a psychologically realistic way. Essentially, Copycat discovers proportional analogies in the domain of strings of English alphabet letters. Here is an example:

$$\begin{aligned} \textit{abc} \text{ is to } \textit{abd} \text{ as } \textit{ijk} \text{ is to what?} \\ \text{or} \\ \textit{abc} : \textit{abd} = \textit{ijk} : ? \end{aligned}$$

In this example, we can reason in the following way: We can see the first string (*abc*) as a group of three letters from which the second string (*abd*) is obtained by replacing the last letter from *abc* with the letter *d*. So if we *do the same thing* to the first string on the right side (*ijk*) we replace *i* with *d* and get the following analogy:

$$\textit{abc} : \textit{abd} = \textit{ijk} : \textit{id}$$

This surely is a legitimate answer, but humans usually reason the following way: they perceive the string *abc* as an ordered sequence of the English alphabet; the last letter in this sequence *c*, is then replaced with *d*, which actually is its successor. So, it is easy now to establish the correspondence between the first strings on the left- and the right-hand side (*abc* and *ijk*) and apply the rule “replace the last letter in the sequence with its successor” to *ijk* and get the more satisfying answer *ijl*.

To further illustrate the richness of this microdomain, let us take the following example:

$$\textit{abc} : \textit{abd} = \textit{iijjkk} : ?$$

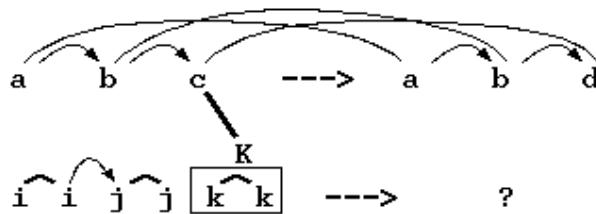
Here too, we perceive the sequentiality of the *abc* string and the “same” sequentiality of *iijjkk*, if we relate the letters *a*, *b*, and *c* to the *groups* of letters *ii*, *jj*, and *kk*. This naturally leads to the answer *iijjll* applying the rule “replace the rightmost *group* of letters with the successor group”. (See Figure 3). Sure enough, we can apply here too the rule “change rightmost letter to *d*” getting *iijjkd*, or the rule “replace the rightmost letter with its successor getting” *iijjkl* but these last solutions are hardly satisfying. Indeed, most subjects would judge the first solution (*iijjll*) as *deeper, more insightful* as it involves more of the features that were perceived (like the alphabetical order, the groups of letters of two, each group being mapped to the corresponding element from the first string, and so on). The point here is that we are able to see the letters in the strings as playing diverse roles: being successor to its neighbour, being equal to its neighbor (thus forming a group), being alphabetically first (or last), being first (or last) in the string... For instance, the letter *c* from the first string of the left-hand side is:

last letter of the string;
 alphabetic successor of its left neighbour;
 third letter of the alphabetic sequence;

...

We can use one (or more) of these descriptions in the process of mapping between the strings.

Fig. 3. Mapping individual letters to group of letters



Consider now this last example:

$$aabc : aabd = ijll : ?$$

Here, new elements are double “a” and “k”. Of course, we still perceive the alphabetical order in the strings, and answers such as *ijll*, or *ijkl* are still plausible. Yet, the fact that these answers don’t take into consideration the fact that the double “a” can be related to the double “k” gives the feeling that something is missing. So if we now map “aa” to “kk”, that is, the *first* group of two letters is mapped to the *last* group of two letters. This forces us to perceive the first term on the right hand side as an alphabetical sequence running in inverse order from right to left, and we map its last element “i” (which actually is the first letter of the string) to the last letter “c” from *aabc*. Thus, the rule for generating the fourth element of the analogy is “replace the last letter in the sequence string with its alphabetical predecessor”. The answer then will be **hjkk**. These processes of *slippage* of concepts (first – last, successor – predecessor) in the process of generating the rule for analogy, according to Hofstadter and his group, is essential for analogy making and other higher cognitive processes that require shifts in the view of how a situation is perceived.

We provide below more detail of the Copycat program, which was designed by Hofstadter and Mitchell.

Copycat has an emergent architecture and cannot be classified as a symbolic or connectionist system. It is inspired by the biological processes in the cell cytoplasm. There, many seemingly random activities take place in parallel, but the end products are complex molecules of diverse proteins. In Copycat, numerous simple agents called *codelets*, each specialized for some simple task, all work in parallel. For example, there is a codelet that notices that two neighbouring letters are the same, or

that one is the successor of the other. The result of their cooperation and competition (between conflicting hypotheses) leads to insightful description of the object.

There are three main components of the Copycat architecture: the *slipnet*, the *workspace* and the *coderack*.

Slipnet is a Platonic network of concepts. A portion of it is given in Figure 4. Concepts include the categories of all the letters from English alphabet, as well as more abstract concepts like ‘opposite’, ‘successor’, ‘predecessor’, ‘first (last) on the alphabet’, ‘first in the string’, ‘same’, and so on. They are connected with labeled links showing the relationship between the two concepts they connect, and the lengths of which capture the conceptual proximity between them. Labels are also nodes in the slipnet. The nodes have level of activation associated with them. When a node gets higher activation, the links to its neighbours shrink meaning that they come into closer proximity. For example if the node ‘successor’ get some activation, the node ‘predecessor’ comes in closer since the link ‘opposite’ which links them, gets shortened. Also, a degree of abstractness or depth is associated with each concept, which is fixed (e.g. the abstractness of the concept of ‘b’ is lower than that of the concept of ‘predecessor’).

Workspace is the second main component of Copycat, where the majority of activities take place. One can think of it as a blackboard, where initially the three terms of the proportional analogy are put (e.g. instances of the letters which are included $abc:abd=ijk:?$). Codelets start working on this raw data, and soon the alphabetical order is recognized, description of ‘b’ as being successors to their left neighbors is noticed, and so on. The respective concepts in the slipnet are activated, which activate their neighbors, and these activated neighbors cause respective codelets in the workspace to be introduced. This reflects the top down pressure in the architecture, and the new codelets try to find instances of the concepts from the slipnet that they represent. With time, more and more objects (descriptions) in the workspace come into existence, and all of them get some measure of salience, which increases with such factors as how active are the respective concepts in the slipnet, and how isolated is the object from the rest of the descriptions.

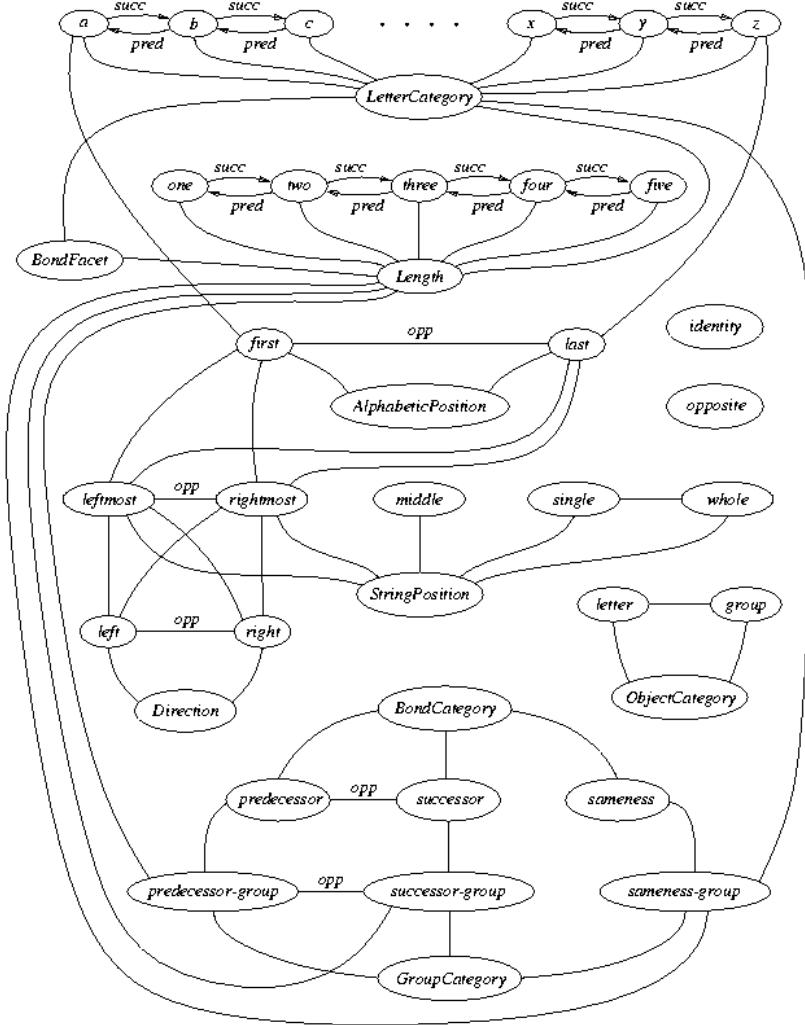


Fig. 4. A portion of the Slipnet concept network for Copycat (adapted from Jim Marshall's Sarag Lawrence College lecture slides)

Coderack, the last of the three main components of Copycat, represents a pool where codelets wait to be run. Each codelet is assigned a variable called urgency, which determines the probability for that particular codelet to be chosen to be run in the next moment. We can distinguish among top-down codelets, which are introduced as a consequence of the activation of some slipnet concepts, to look for objects in the workspace, and bottom-up codelets, which look at the strings of letters for interesting

properties. At the beginning, there are only three strings in the workspace, and few bottom-up concepts in the coderack.

We can see that the probabilistic nature of the Copycat architecture comes from the urgency assigned to the variables in the codelets in the coderack, and from the saliency level assigned to the objects in the workspace (the more salient an object is, the more attention it is likely to get). Finally, there is one global variable called *temperature* that reflects the variations of probabilities. For example, when the temperature is high, the distribution of the probabilities (which depend on the urgency level of the codelets) is more uniform, thereby making the whole system more random. Temperature reflects the coherency of the overall structures in the workspace, getting lower as more and more objects are incorporated in some bigger structure.

This non-determinism is reflected at the higher level because for the same input, the program can come up with different answers to the analogy puzzle. For example, for the problem $abc:abd=ijk:?$ out of 1000 runs, the answer ijl may appear 980 times, and the answer ijd 19 times, and ijj once.

To conclude the description of Copycat, we would like to stress again that this is one of the original computational analogy model, incorporating many features of human cognitive processes like their fluidity, non-determinism, the importance of non-determinism for creativity, and the like. A different approach to the similar problems in the microdomain of geometric figures is taken by O'Hara and his colleagues [38, 39]. One objection raised against these approaches is that they work only in microdomains, and not in ‘real-life’ situations like some other models such as SME or ACME. To such criticism, Hofstadter *et al.* give a two-pronged response. First, they argue that the letter string ‘toy-domain’ is sufficiently rich to incorporate many key aspects of analogy making. Secondly, systems like SME that claim to model ‘real-life’ domains (heat-flow, water-flow, say) actually merely manipulate meaningless tokens syntactically: though SME uses the token ‘water’, it does not really understand any semantic properties of water or of its related concepts.

The approach taken by Hofstadter *et al.* in the Copycat system resonates strongly with the *interactionist* approach to cognition. We briefly describe here one version of interactionism articulated in [20], as expressed in the following quote: “[O]ur concepts do not reflect some pre-existing structure in the environment, they *create* the structure. Yet, this conceptual organization cannot be arbitrary, and is somehow constrained by reality”. Thus, for somebody who is not familiar with the letter string domain, the string ‘abc’ would not have the property “three consecutive letters of the alphabet”. Our concepts are grounded in our interactions with the environment. Those interactions, on the other hand, are determined by our sensory-motor apparatus and the environmental constraints. During their life, cognitive agents build their conceptual networks, which then represent *filters* that guide their perceptions and actions.

The account of interaction laid out in [20] works as follows. On one side of interaction are the internal representations of the cognitive agent called *concept networks* (CN). CN are some symbolic systems having an operational structure that

corresponds to algebras. At the other end of interaction is the external reality, which is made available for conceptualization to the agent via the agent's *sensorimotor data set* (SDS). The ontology of the SDS is determined by the cognitive agent's perceptual and motor abilities. *Cognitive relations* (CRs), which connect concepts from CN with parts of the SDS, play a key role in this version of interactionism. As SDS is rooted in reality, CRs become the links between the internal concepts of the agent and reality. By forming a CR (by instantiating concepts from CN) the reality acquires an *experiential ontology* for the agent, which is referred to as the *environment*. A CN instantiated by a CR is called a *cognitive model*. The cognitive agent strives for *coherency* by trying to keep these two structures – the structure of the CN (determined by the agent) and the structure of the environment (determined by reality) – as close as possible, and ideally the same. This must be so if the agent wants to plan any action in the environment with some desired outcome. The coherency can be maintained by two mechanisms: a) *accommodation* – changing the structure of the CN while keeping the CR fixed; and b) *projection* – keeping the CN fixed but changing the CR so that the experiential ontology of the environment is changed in order to make its structure (which is determined by reality) coherent with the structure of the CN.

In order to illustrate this model, we present here an example taken from [20]. In Figure 5 we depict a simple agent called *Spinner* that inhabits a two-dimensional world, and has a simple perceptual and motor apparatus. It has an 'eye' consisting of five cells working in an on-off fashion. Any object in front of the eye forms an image by casting a shadow and turning off those cells that come under the shadow. The image Spinner 'sees' is represented with a sensory vector: e.g. 11111 would be the image of a horizontal line in front of its eye. Spinner can affect its environment by emitting a jet stream from any one of the cells of the eye, the intensity of which can take one of three possible values. The activation of the motor system is represented with the effector vector: e.g. 03000, which means that the second cell from the left emits a burst of intensity 3. All the possible sensory and effector vectors represent the sensory motor data set. Spinner inhabits a discrete environment, which consists of zones in the form of parallel strips (Figure 5 (b)). There is a line in each zone that can turn around its middle point and move vertically and horizontally within the zone. Spinner can be in only one zone in a given moment and can hop from a zone to its neighboring zone, which happens instantly as a discrete event.

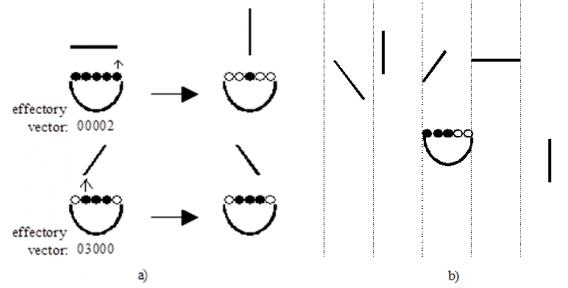


Fig. 5. a) Spinner's sensory and motor systems (see text for explanation); b) the ‘world’ of Spinner.

Spinner experiences its ‘world’ by interpreting its conceptual networks. An example of conceptual network is shown in Figure 6. One should note that in this CN Spinner can distinguish between objects that appear to be perceptually identical (e.g. lines with different orientations but yielding the same sensory vector 01110, Figure 5 (a)) by performing actions on them.

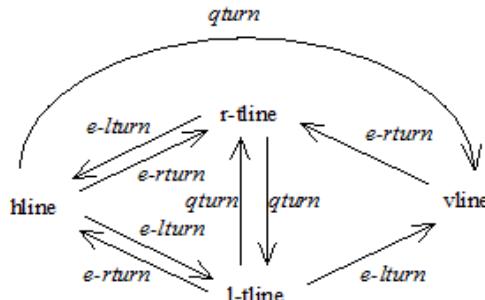


Fig. 6. An example of conceptual network for making lines turn. Nodes represent states of the sensory organ: hline (horizontal line, all cells off), vline (vertical line, one cell off), r-tline and l-tline (lines with different orientations, different number o of cells are off; note that different lines may result with a same image on the eye). Arcs represent actions of the effector system: $qturn(03000, 00030)$, $e-turn(02000, 00020)$.

Depending on the result of the actions, an object is categorized in one or another category. Another important thing is the grouping of different actions into the same class if their effect upon an object is the same. An interpretation of the concept network given in Figure 6 is given in Figure 7.

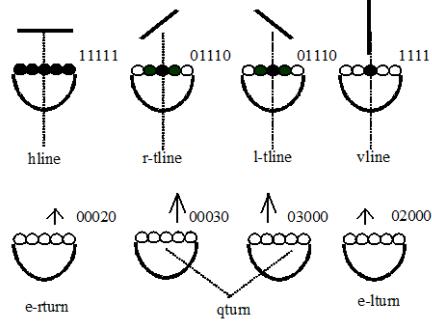


Fig. 7. An interpretation of the CN given in Fig 5; Symbols of the concept network correspond to the objects of the environment, and the operators to the actions that can change those objects.

Under this interpretation Spinner's world-view can be depicted as shown in Figure 8, and this interpretation creates Spinner's experiential ontology or the environment. In this case the structure of the environment is isomorphic to the structure of the conceptual network rendering the cognitive relation (or model) coherent. This means that Spinner can successfully predict the outcomes of its actions.

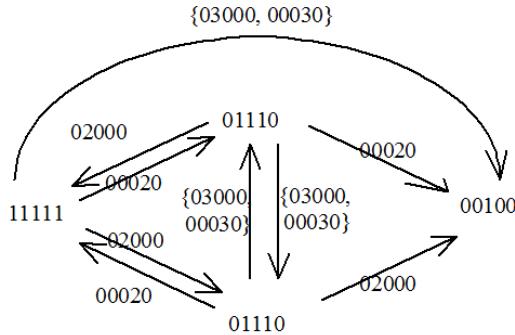


Fig. 8. The structure of the environment of Spinner as seen from the CN given in Figure 6 under the interpretation of Figure 7

If we now put Spinner in another world, where the length of the lines is only four, using the old model will show that it is not coherent any more. Spinner can try to make it coherent again by the process of projection: the structure of the conceptual network is kept invariant and the correspondence between the concepts and the new objects of the environment is varied until a coherent fit is reached. This regrouping of the objects of the environment according to an existing CN is analogous to what happens during metaphoric interpretation. We might imagine a cognitive agent with various CNs, each of which, when conventionally interpreted, is a model of a part of the environment. In Figure 9 we can see two such CNs with their respective

conventional interpretations. Whenever a cognitive relation of projection is instantiated, i.e. one part of the environment (the target) is interpreted with a different CN (the source) than the conventional one, and then a metaphor is formed. If the cognitive agent is aware of this novel interpretation, this process can lead to creation (or discovery) of new similarities between the source and the target.

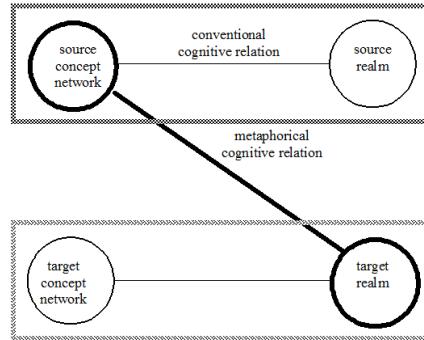


Fig. 9. Metaphor as projective cognitive relation: target realm is non-conventionally interpreted via the source CN.

This approach comes closer to being able to model creativity, for often creative insights emerge from applying a concept to an object (or a low-level representation of it) that is not habitually associated with it. In our earlier work, we have formalized this process [20], and have applied it to model creativity in legal reasoning [21], but clearly much more work remains to be done. Moreover, in real-life, a number of different cognitive processes may act in consort to generate a creative insight, modeling of which may require hybrid architectures [26], [34], [88], [93].

Artificial Creativity Systems

Research in creativity (usually conducted within psychology), has shown a bias towards exceptional individuals, which is known as the big-C creativity. Consequences of these views were felt in the research in Artificial Creativity (AC) and Computational models of Metaphor and Analogy (CMA), where the existence of creativity and analogy modules was often hypothesized. Virtually, all the attempts at modeling those modules aimed at big-C creativity as opposed to day-to-day creativity, as we explain below.

AC research is classified according to the type of artifacts the system is supposed to create. Some of the more popular research domains include: **storytelling** (e.g. [65-70]), **humor generation** [71-74], **creative music interpretation and composition** [75-77], and **visual arts** [78, 79]. In spite of the many dissonant voices when it comes to define what creativity is, there is pretty much a consensus about “what is a creative product?” Most researchers in AC accept that in order for an artifact to be deemed creative it should possess the qualities of **novelty** (being

non-obvious, surprising, unexpected) and **value** (the artifact should be interesting, aesthetically pleasing, useful in one way or another). The trickiest point, maybe predictably so, turned out to be implementing the *assessment of the value* of the artifacts generated by the AC systems. In some domains (e.g. proving a mathematical theorem, discovering a new law of physics) assessing novelty and value of the artifact can be fairly easy, although the problem here is to distinguish between *interesting* and *trivial* results. But once we get into the arts (e.g. novel writing, visual arts, humor), we move into a slippery territory at many levels. At best, an AC program has some heuristics for preferring some specific properties of the artifact that the designer of the program deems valuable. Afterwards, the designer usually applies another filter and selects among the generated artifacts according to their personal judgment, before sharing it with a wider audience (usually a conference or a journal paper and rarely in galleries or publishing houses). At this point, and without too much exaggeration, one can easily argue that the AC program was just a tool and that the creativity was exactly in the last step of **choosing** among the generated artifacts. Therefore, “where is creativity?” or “who is being creative?” are questions that naturally arise and hence, as the examples from the literature show, discussing these fundamental conceptual issues seems to be *de rigueur* in most of the AC papers. (See also [25].) This is in stark contrast with other domains in AI: for example, when designing a flight-scheduling program, we do not have problems ascribing *some* intelligence to a program that has created a good schedule. Such a program, once programmed, does everything autonomously and the output can be tested objectively..

Boden [80] has been very influential in setting the direction of artificial creativity research. She introduced the notions of P and H-creativity: P-creativity refers to ideas/objects/skills that are produced by an individual and that are new to that particular person; whereas, H-creativity refers to the production of ideas/objects/skills that have never before occurred in all human history. Much of the research in computational modeling of creativity so far has focused on designing programs that produce artifacts which would be perceived as creative by a wide audience (i.e. H-creativity). Boden also categorized creative processes into three types: a) exploratory, b) combinatorial c) transformational. However, her work had less impact on the actual implemented AC systems, notwithstanding an attempt by Wiggins [81] to formalize some aspects of Boden’s theory. It may be fair to claim that most of the existing AC systems can be categorized as exploratory and/or combinatory creative in a domain defined by the designer.

Let us now delve deeper into how designers of an AC system usually proceed. A creative storytelling system might start by looking into literary or narrative studies. One of the basic distinctions there is between *fabula* (what is told) and *discourse* (how it is told). The fabula is about the events in the story, actions, time, location and the like, whereas the discourse deals with perspective (first or third person), emphasis of particular aspect of an event or action or location, etc. To make the task manageable, designers usually restrict the domain and the context of the story: for example, an AC system designed by Klein *et al* [65] focuses on murder mystery stories that take place during a weekend party. There is a set of characters and events governed by probabilistic rules. These rules fire (probabilistically) to change of the state of the simulated world, which is represented as a semantic network. There are additional constraints for the sequence of events that are expected

to happen during a weekend party. As input, the system also gets the world description as well as the character traits of all the individuals. The victim and the murderer(s) are chosen based on these character traits. The narrative is generated in terms of state transitions in the simulated world. Here is an excerpt from a 2100-word story generated by *Novel Writer*:

"The day was Monday. The pleasant weather was sunny. Lady Buxley was in the park. James ran into Lady Buxley. James talked with Lady Buxley. Lady Buxley flirted with James. James invited Lady Buxley. James liked Lady Buxley. Lady Buxley liked James. Lady Buxley was with James in a hotel. James caressed Lady Buxley with passion. James was Lady Buxley's lover. Marion following saw the affair. Marion saw the affair. Marion was jealous."

Another storytelling program *Minstrel* by Scott R. Turner [82] aims to model the mental state of the author of the story. Turner approaches the process of creative storytelling as problem solving or, more precisely, multiple constraints satisfaction where the constraints are generated by the simulated author. Again, to make this process tractable, the domain of *Minstrel* is constrained to events at the court of King Arthur and involves relationships among ladies and knights. Here is an excerpt from one generated story (*Minstrel* generates about half-page long stories):

"It was the spring of 1089, and a knight named Lancelot returned to Camelot from elsewhere. Lancelot was hot tempered. Once, Lancelot lost a joust. Because he was hot tempered, Lancelot wanted to destroy his sword. Lancelot struck his sword. His sword was destroyed. One day, a Lady of the court named Andrea wanted to have some berries. Andrea went to the woods. Andrea had some berries because Andrea picked some berries. At the same time, Lancelot's horse moved Lancelot to the woods. This unexpectedly caused him to be near Andrea. Because Lancelot was near Andrea, Lancelot saw Andrea. Lancelot loved Andrea."

A more recent example of narrative generation AC system is *Makebelieve* [83]. This system does not use episodic memory (or other schema-like structures used by *Minstrel* to provide coherence) but instead uses a *ConceptNet*, which is structured like *WordNet*, and incorporates a rich CYC-like common-sense knowledge expressed in logic. *Makebelieve* expects the user to give the first phrase and then continues via associations, producing short stories that feel like the stream-of-consciousness literature:

"John became very lazy at work. John lost his job. John decided to get drunk. He started to commit crimes. John went to prison. He experienced bruises. John cried. He looked at himself differently."

Let us now consider the domain of creative music interpretation. Naturally enough, AC systems in this domain have to be able to read the musical notation, as well as have some way of representing and applying the notions of *timing* and *tempo changes*, *phrases*, *loudness variations*, as well as what musicians call *articulation* (transitions from one note or phrase to another). One such system is the YQX system [77] which uses a machine-learning algorithm based on a Bayesian model trained on a corpus of performances by expert human performers. It learns three numerical parameters that describe the expressiveness of the performance: timing, loudness variation (or the *dynamics*), and articulation. Thus, for the local score context, which

consists of parameters like pitch interval, rhythmical context, and other features stemming from musical theory like implication-realization analysis, the Bayesian network models the dependencies among the three numerical parameters of the expressiveness. For the final rendering of the performance, the learned model is combined with several other inputs: a reference tempo and loudness curve are established; the Bayesian model predicts the local timing, loudness, and the articulation of the notes from the score context; and eventually YQX incorporates effects like accents and fermatas where marked on the music score.

As one last example, this time coming from the domain of visual arts, consider Cohen's artificial painter AARON [79]. During a period spanning more than 40 years, Cohen has developed several different versions of AARON, and he uses the teacher-student metaphor to describe his relationship with the program. The AARON is a program that controls a robot that produces its paintings. Since the beginning of the 1970s, Cohen has been adding new components to AARON: from basic procedural skills for drawing different linear shapes (together with decision algorithms as to where on the canvas the drawing should continue, and when to stop the process) to more complex data structures storing declarative descriptions of the appearance of human figures and plants (e.g. natural body postures with rules about visibility), to a set of rules about perspective and coloring. There are also rules (including some randomness) about where to start the drawing. AARON starts always with the foreground, and moves then to the background. The current state of the drawing puts additional constraints on how to continue, and the program tends to fill up the whole of the available space. In addition, it is Cohen who chooses the most interesting drawings from among the outputs generated by AARON. For more on the philosophy behind AARON, interested readers can visit Cohen's web page <http://crca.ucsd.edu/~hcohen/> and McCorduck's 1991 book [84].

From this brief survey, we can make the following observations about the current status of AC systems:

- Virtually all of them focus on the product (a consequence of the *product-generating paradigm* in which they are working) rather than on the process. Thus, we may call this approach *top-down* or *product-first* approach.
- Most of them are given, in advance, *a detailed (hard-coded) description of the domain*. This can be: syntactic rules, narrative structure, and some semantics for artificial prose writers; musical notation and rules for artificial composers and creative interpreters; basic drawing primitives for artificial painters; basic mathematical operations, a lot of search heuristics with evaluation functions, and big knowledge/fact base for artificial scientists).
- All these AC systems appear to be *closed systems* in the sense that there is no way to appreciate, and build upon, the feedback from naïve (or not) observers.
- None of these AC systems are *socially embedded* except, in a certain sense, via their designers, who themselves receive the feedback from the audience and eventually make the necessary changes in their programs.
- Finally, virtually all of the researchers within AC looked for inspiration into the existing theories of the domain in which their systems are supposed to be creative: literary and narrative theory, music theory, visual arts, etc. This goes

counter to our intuitions and the empirical facts that many artists and scientists report that actually combining domains (in which they not need be widely recognized but simply familiar enough) has resulted in some of their most creative outputs.

On a more abstract level (and maybe at the risk of oversimplification), we could say that the majority of AC systems to date quite resemble the generic architecture of a GOFAI expert system from the '70s and '80s of the last century. The aim of such expert systems was to replace human experts in some particular narrow domain like: general MD practitioner who would come up with a diagnose given the symptoms of the patient, or an operating system administrator who would know how to fine-tune the parameters and optimize the functioning of a complex operating system. Just like the AC systems, they usually contained a huge knowledge base, many heuristics, and representative cases (in case-based reasoning). The knowledge representation was mainly symbolic, and some systems also included probabilistic reasoning.

Given the dominant approaches to computational models of analogy (i.e. focusing on relational matches between two hardcoded representations), it is probably not surprising that although (as mentioned in Section 2) analogy is often seen as a key factor to the creative process, we see only a few AC systems that use analogy [60], [94, 95].

Analogy in Cognitive Development

Reasoning by analogy is sometimes seen as a pinnacle of cognitive development. Goswami in [14], for instance, notes that in Piaget's account, analogical reasoning occurs only in adolescence during the formal operation stage. In contrast, Goswami goes on to review a number of research results that show that analogy is far more pervasive in cognitive development, and occurs much earlier, i.e. even in three- and four-year olds. Goswami argued that children in some of Piaget's experiments were not able to solve a relational analogy problem because they were not familiar with the causal relations among the objects (e.g. '*bicycle : handlebars :: ship : ?*', handlebars are used to guide the bicycle in the *same way* the *ship's wheel* is used to steer the ship). However, a close look at Piaget's numerous studies reveals that he has also noted the onset of analogical thinking manifested as a sensorimotor schema at a very young age:

"At 1;4(0) L. tried to get a watch chain out of a match-box when the box was not more than an eighth of an inch open. She gazed at the box with great attention, then opened and closed her mouth several times in succession, at first only slightly and then wider and wider. It was clear that the child, in her effort to picture to herself the means of enlarging the opening, was using as 'signifier' her own mouth, with the movements of which she was familiar tactually and kinesthetically as well as by analogy with the visual image of the mouths of others. It is possible that there may also have been an element of 'causality through imitation,' L. perhaps still trying, in spite of her age, to act on the box through her miming. But the essential thing for her, as the context of the behaviour clearly showed,

was to grasp the situation, and to picture it to herself actively in order to do so." ([44], p. 65; see also 39]

We included this long quote here for it illustrates that Piaget was fully aware of the key role played by analogy and its various manifestations in cognitive development as early as sixteen months. These differences (Goswami vs Piaget) may stem from Piaget's idiosyncratic approach to research: he was not trying to study particular modules or faculties (such as 'reasoning by analogy', for example) but as an acute observer he was trying to offer the best explanation that may account for certain types of behavior at certain age groups. We would like to offer the following speculation: if we were to ask Piaget about analogical reasoning and how/when it develops he might say that there is no particular age when children begin to reason by analogy. What happens is a gradual progression that starts from objects being understood only in terms of the sensorimotor schemas in which it is involved; an object 'is for something' and there is no independent representation of them. Their properties remain contextual (and not fixed) and hence reasoning about relations among objects will neither be stable nor be consistent, especially at an early age of 4 or 5. Piaget thus, we might assume, hesitated to call this reasoning by analogy, reserving the term for his formal-operations stage when abstract object and relation representations are fully developed and available for conscious manipulation. Later in the paper, we offer an alternative framework for interpretation of Piaget's theory by adopting a broader construal of what is meant by 'analogy'.

In the framework that we propose here, cognitive development is a series of small creative leaps where cognitive agent internalizes its interaction with the environment. Using the standard language of cognitive science or artificial intelligence, we can see these internal constructs as agent's representations of the environment, or, to be more accurate: representation of agent's embeddedness in that particular environment. Initially, these representations are entirely expressed in terms of the innate Piagetian sensory-motor schemas. That is, we can look at the innate schemas as the *source domain* for a metaphorical description of the agent environment (the unknown *target domain*). Through the processes of assimilation (current metaphors can explain new experiences) and accommodation (adaptation, or change in the structure of the source domain is needed in view of new experiences), as well as spontaneous reorganization of the internal schema space (for example, by finding similarities and connections among distant schema-subspaces) cognitive agents change themselves and their environments (physical, social, linguistic). (See also [20]). They need to master motor skills, language, social conventions and norms etc. This maturation process is comprised of many creative acts, driven by our genetic heritage as well as the micro and macro social context. The growth continues throughout the lifetime of the individual and, in some cases, particularly creative individuals may question some of the norms and conventions of their culture, and may impact significantly some particular established domain (arts, sciences, religion) or even create entirely new domains. The point here is that creativity is understood as a continuum and not as a binary ('yes' or 'no') attribute. It is also the driving force behind our cognitive development and it relies on more basic cognitive processes described above. An

initial outline of this approach can be found in [54] and its partial implementation in context of mobile robot learning can be found in [46].

If what we suggest is plausible, the dominant approaches in artificial creativity and computational modeling of analogical (metaphorical) reasoning will have to be re-evaluated and probably fundamentally changed.

Similarity and Analogy in Creativity

In cognitive development, it has been noted that younger children tend to focus on surface-level similarities, and only later they take into account relational and structural similarities. For example, Namy and Gentner [33] remark: “Children up to five years go for perceptually similar objects. Clearly, then, a large number of studies have converged to demonstrate that perceptual properties such as shape loom large in children’s responses on categorization tasks. This evidence suggests that children rely on shape or other salient perceptual features—perhaps even to an extent that seems detrimental to their acquisition of conceptually coherent object categories.” (p. 6)

Apart from Piaget’s theory of cognitive development which we mention above, other theories can be re-casted too as series of small P-creative leaps during which conventional conceptualization arise. Karmiloff-Smith [28, 29] proposes the *representational redescription model*, which comprises of an endogenously driven “process by which implicit information *in* the mind subsequently becomes explicit knowledge *to* the mind” [29 p. 18]. The process can be likened to Piaget’s stages from early sensorimotor schemas (which too are endogenously driven to be executed) to the final stage of formal operations where the agent deliberately/consciously manipulates different abstract schemas. Karmiloff-Smith’s developing agent goes through four phases, where the first one (**I**) is characterized by Implicit/behavioral knowledge/skill representations applied to certain task. These are rather detailed and task specific, and are not available for conscious manipulation by the subject. The next three phases E1, E2, E3 represent the emergence of more explicit, abstract, and finally (E3) verbalizable representations. These representations lose many specific details compared to the first (**I**) phase representations, but become more flexible/reusable, and declarative. As such, they also become available as a huge library of source analogies, which can be applied to different problems. (See [23] for a similar approach to modeling creativity in legal reasoning.)

Barsalou and Prinz’s [2] theory of mundane creativity in perceptual symbol systems [3] also comes close to the view we are outlining here. Their theory of cognitive development focuses on the formation of perceptual symbols, which originate from the perceptual input (across all modalities) during the agent’s sensorimotor interaction with the environment. By the processes of selective attention, the agent focuses on some aspects of the entire perceptual input, filtering out alternative potential aspects to a large extent. These perceptual aspects are transferred into the long-term memory, and in essence can be seen as concepts that can be recalled by similar perceptual input. In the language that we have adopted here, this would correspond to the emergence of conventional conceptualization. A

creative insight then would happen when a subject uses a non-conventional perceptual symbol or symbols to perceive the given object/situation/scene.

In creativity research, it has been widely recognized that similarities play a key role in the generation of new ideas [30], [53]. Although surface similarities are often found to influence memory access and recall [1], most of the research has focused on semantic aspects of the similarity, like structural alignment, for these are considered to be more helpful in problem solving and learning. In fact, surface similarities are often thought to be distracting [10]. A number of other creativity researchers, however, point out that focusing on structural similarities reinforce conventional way of viewing a given situation, and the crux of creativity lies in breaking the conventional structure and conceptualize the situation in a new way [6], [13], [44]. In conventional conceptualization (like in any generalization) some of the properties of the objects (i.e. superficial attributes) are lost. But maybe, in order to come up with a creative solution, we may need exactly those properties! Thus, focusing on surface/perceptual similarities can act as a cue to connect two (conventionally) unrelated objects in a new way.

There are many real-world cases where accidentally noticing surface similarities between two objects or situations suggested a novel idea that later led to a major innovation. One such example is provided by how Ignaz Semmelweis came up with the idea of the germ theory. (See, for instance, [33].) When Semmelweis was practicing (in the 1840s) at the Vienna General Hospital in Austria, there was no knowledge of bacteria and germs. Many women used to die during childbirth due to puerperal sepsis (childbed fever). Between the two maternity wards at this hospital, the death rate in one of them was more than six times higher than the other one. There were many speculative theories for the childbed fever, like foul air in the delivery wards, the presence of male doctors, which wounded the modesty of mothers, and so on. None of these explained the difference between the mortality rates between the two wards.

The insight came when one professor, who was helping a student through autopsy, received an accidental cut on his finger and died from the resulting infection. Semmelweis noticed that the symptoms were similar to the childbed fever victims. (These were all surface similarities, for there was no structural knowledge connecting the two cases.) A deeper investigation and some more research led him to formulate a theory of germs, according to which the germs from cadavers were the cause of the childbed fever, and the simple technique of washing hands in chlorinated water before handling the patient in the maternity ward brought down the fatality rate.

Some of our recent empirical studies further support this view. In one set of studies ([25], [36], [37], we have found that low-level perceptual similarities — that is, similarities with respect to texture, shape, color etc. determined algorithmically — facilitate creation of conceptual features and conceptual similarities. In another study [26], we focused on the creative process involved in connecting two pictures by painting another picture in the middle. This technique was involved in four *Infinite Landscape* workshops conducted by a visual artist at Art Museums in Japan and Europe 2007-11. Based on the artist's verbal recollection of the ideas that occurred to him as he drew each of the connecting pictures, we identified the micro-processes and

cognitive mechanisms underlying these ideas. We found that *surface features*, *contrast*, and *meaning deconstruction* play major roles in the generation of new ideas.

We can relate this idea with the role of surface-level and deep similarities proposed by Hofstadter and Sander [99]. They argue that when an agent is an unfamiliar domain, like a novice, then they necessarily rely on surface similarities, as that is all they have to go by. So, a novice physics student may categorize all the problems involving pulleys in one category, all problems involving springs in one category, and so on, which is based on surface features. However, as they acquire more knowledge, and understand the principles of physics like the conservation of energy, their classification starts to use these structural principles. When they become thoroughly familiar with the domain — become an expert — then these structural features, referred to as the *essence* by Hofstadter and Sander, are automatically recognized, and surface features fade away.

However, as far as creativity is concerned, this *essence* needs to be radically altered, as discussed in Sec. 2 above. For this, any analogy with respect to the deep structural similarity becomes a hindrance for creativity, and the recommended approach is to go back to being a novice; or, in other words, start focusing on surface similarities again. Indeed, creativity researchers propose precisely such techniques: for example, Rodari's [49] "see the object or situation as if for the first time"; and Gordon *et al.*'s [13] "make the familiar strange".

It is interesting to point out that this aspect of surface similarities to create new insights is one of the advantages claimed for the case-based reasoning approach. For instance, Riesbeck and Schank ([48], pp. 9-14) compare and contrast three modes of reasoning: 1) reasoning with ossified cases (rules or abstract principles), 2) reasoning with paradigmatic cases (cases with a given interpretation), and 3) reasoning with stories (cases with many possible interpretations and capable of re-interpretations). They argue that it is the third mode of reasoning that displays the most flexibility and power of having a knowledge base containing cases. But in reasoning with stories, one essentially relies on surface similarities, and constructs alternate structures on the fly as needed. Indeed, it has also been noted that surface similarities play a key role in memory access and recall [1].

Thus, we can conclude that the traditional models of analogical reasoning, which prefer relations over attribute mapping, may be useful when we have to solve a novel problem in a domain with high structural similarities to some familiar domain. However, the solutions that may result from this process are rarely deemed to be creative for they only reinforce traditional conceptualizations of both domains. On the other hand, we may have no or little knowledge of the deep structure of the target domain (for example, in early cognitive development). In these cases, perceptual similarities may lead to novel conceptualizations (of both source and target domains) and highly creative solutions or products.

Conclusions

We have presented a view here where analogy represents a core process in human cognitive development. We have argued that creativity in human agents represents a continuum: from everyday/mundane/P-creativity to the big-C creativity. Accepting the view that analogy is crucial for creativity, we attempted to make the case that superficial (attribute) similarities may actually lead to more original solutions or products. Structural analogies only reinforce the conventional conceptualization, which may be a hindrance in case the problem at hand requires information that is not normally a part of this conceptualization. We have re-casted Piaget's theory of cognitive development by describing assimilation and accommodation as progressive reasoning by analogy starting from early analogizing in terms of sensory motor schemas, to analogies in mature cognitive agents who have developed object representations. Within this framework for creativity, we gave a critical overview of today's artificial creativity models, and provided some empirical support for the claim that surface-level or perceptual similarities may play a more central role in creativity than has been supposed so far.

REFERENCES

1. Barnden, J.A., & Holyoak, K.J. (eds.): *Analogy, Metaphor, and Reminding*, Intellect Books (1994)
2. Barsalou, L.W. & J.J. Prinz, Mundane creativity in perceptual symbol systems, In T.B. Ward, S.M. Smith, & J. Vaid (eds.), *Creative thought: An investigation of conceptual structures and processes* (pp. 267-307). Washington, DC: American Psychological Association (1997)
3. Barsalou, L.W.: Grounding symbolic operations in the brain's modal systems, In G.R. Semin & E.R. Smith (eds.), *Embodied grounding: Social, cognitive, affective, and neuroscientific approaches* (pp. 9-42). New York: Cambridge University Press (2008)
4. Bonnardel, N.: Towards Understanding and Supporting Creativity in Design: Analogies in a Constrained Cognitive Environment, *Knowledge-Based Systems* 13, pp. 505–513 (2000)
5. Bruno, S.: *Raisonnement par analogie et conceptualisation*, Thèse de doctorat, Université Paris 5 - René Descartes (2000)
6. de Bono, E.: *New Think: The Use of Lateral Thinking in the Generation of New Ideas*. Basic Books, New York. (1975)
7. Chalmers, D.J. French, R.M. & Hofstadter, D.R.: High-level perception, representation, and analogy: A critique of artificial intelligence methodology, *Journal of Experimental & Theoretical Artificial Intelligence*, 4(3), 185–211 (1992)
8. Dunbar, K.: How scientists think: On-line creativity and conceptual change in science, In T.B. Ward, S.M. Smith and J. Vaid (eds.), *Creative thought: An investigation of conceptual structures and processes*. Washington, DC: American Psychological Association (1997)
9. Falkenhainer, B., Forbus, K.D., Gentner, D. The Structure-Mapping Engine: Algorithm and Examples, *Artificial Intelligence* 41: 1–63. (1989)

10. Faries, J.M., & Schlossberg, K.R., The effect of similarity on memory for prior problems, Proceedings of the 16th annual conference of the Cognitive Science Society, 278 – 282. 1994.
11. Gentner, D.: Structure-mapping: A theoretical framework for analogy, *Cognitive Science* 7:155-170 (1983)
12. Gentner, D., Brem, S., Ferguson, R. W., Markman, A.B., Levidow, B.B., Wolff, P., and Forbus, K.: Analogical reasoning and conceptual change: A case study of Johannes Kepler, *The Journal of the Learning Sciences* 6(1):3-40 (1997)
13. Gordon, W.J.J.: *Synectics: The Development of Creative Capacity*. Harper & Row, New York (1961)
14. Goswami, U.: Analogical reasoning in children in J. Campione, K. Metz, A. Sullivan Palincsar (eds.), *Children's Learning in Laboratory and Classroom Contexts Essays in Honor of Ann Brown*, Routledge (2007)
15. Hofstadter, D.: Fluid concepts and creative analogies: Computer models of the fundamental mechanisms of thought. New York: Basic Books. (1995)
16. Hofstadter, D.: *Analogy as the Core of Cognition*, in D. Gentner, K. Holyoak, and B. Kokinov (eds.) *The Analogical Mind: Perspectives from Cognitive Science*, Cambridge, MA: The MIT Press/Bradford Book, pp. 499–538 (2001)
17. K. Holyoak, P. Thagard, “Analogical Mapping by Constraint Evaluation, *Cognitive Science*, 13, 295-355. 1989.
18. J. E. Hummel & K. J. Holyoak, “Distributed representations of structure: A theory of analogical access and mapping , *Psychological Review*, 104, 427-466. 1997.
19. J.E. Hummel & K.J. Holyoak, A symbolic-connectionist theory of relational inference and generalization, *Psychological Review*, 110, 220-264. 2003.
20. B. Indurkhy, *Metaphor and Cognition*. Kluwer Academic Publishers, Dordrecht, The Netherlands. 1992.
21. B. Indurkhy, On modeling creativity in legal reasoning. *Proceedings of the Sixth International Conference on AI and Law*, 180–189. ACM, New York. 1997.
22. B. Indurkhy, “Emergent representations, interaction theory, and the cognitive force of metaphor, *New Ideas in Psychology*, 24(2), 133–162. 2006.
23. B. Indurkhy, “Rationality and reasoning with metaphors. *New Ideas in Psychology* 25 (1), 16–36. 2007.
24. B. Indurkhy, “On the role of metaphor in creative cognition, *Proceedings of the International Conference on Computational Creativity: ICCC-X*, Lisbon, Portugal (2010)
25. B. Indurkhy, “Whence is Creativity? *Proceedings of the International Conference on Computational Creativity: ICCC-2012*, Dublin, Ireland, 62-66 (2012)
26. B. Indurkhy, K. Kattalay, A. Ojha, P. Tandon, Experiments with a creativity-support system based on perceptual similarity. In H. Fujita and I. Zualkernan (eds.) *New Trends in Software Methodologies, Tools and Techniques*, Amsterdam: IOS Press (2008)
27. B. Indurkhy, S. Ogawa, “An Empirical Study on the Mechanisms of Creativity in Visual Arts, Proc. of the 34th Annual Conf. of the Cog. Sci. Society: CogSci 2012, Sapporo, Japan (2012)
28. A. Karmiloff-Smith, “From metaprocess to conscious access: Evidence from children’s metalinguistic and repair data, *Cognition*, 23, 95–147 (1986)
29. Karmiloff-Smith, A., *Beyond modularity: A developmental perspective on cognitive science*. MIT Press. (1992)
30. M.T. Keane, “Incremental analogising: Theory and model In K.J. Gilhooly, M.T. Keane, R. Logie & G. Erdos (Eds). *Lines of Thinking: Reflections on the Psychology of Thought*. Vol. 1. Chichester: John Wiley (1990)

31. A. Koestler, *The Act of Creation*. Hutchinsons, London (1964)
32. Kokinov, B., Holyoak, K. & Gentner, D. (Eds.), *New Frontiers in Analogy Research*, Sofia: New Bulgarian University Press (2009)
33. Levitt, S.D. & Dubner S.J.: *Superfreakonomics*. William Morrow (2009)
34. M. Mitchell, *Analogy-Making as Perception*. ISBN 0-262-13289-3, 1993.
35. L.L. Namy, D. Gentner, "Making a Silk Purse Out of Two Sow's Ears: Young Children's Use of Comparison in Category Learning, *Journal of Experimental Psychology*, 13(1), 5-15 (2002)
36. N.J. Nersessian, S. Chandrasekharan, Hybrid Analogies in Conceptual Innovation in Science, *Cognitive Systems Research* 10 (3), 178–188 (2009)
37. S. Ogawa, B. Indurkhy, A. Byrski, "A meme-based architecture for modeling creativity, Proc. of the Int. Conf. on Computational Creativity, Dublin, Ireland (2012)
38. S. O'Hara and B. Indurkhy (1994) Incorporating (Re)-Interpretation in Case-Based Reasoning, in S. Wess, K.-D. Althoff & M.M. Richter (eds.) *Topics in Case-Based Reasoning: LNCS*, vol. 837, pp. 246-260, Springer-Verlag, Berlin, Germany (1994)
39. O'Hara, S., Indurkhy, B.: Adaptation and Re-Description in the Context of Geometric Proportional Analogies, *AAAI-95 Fall Symposium Series: Adaptation of Knowledge for Reuse*, 80-86 (1995)
40. Ojha, A., Indurkhy, B.: Perceptual vs. conceptual similarities and creation of new features in visual metaphor, in B. Kokinov, K. Holyoak & D. Gentner (eds.) *New Frontiers in Analogy Research*, Sofia: New Bulgarian University Press (2009)
41. Ojha, A., Indurkhy, B.: On the role of perceptual features in metaphor, in E. Gola & F. Ervas (eds.) *Metaphor and Communication*. To appear.
42. Okada, T., Yokochi, S., Ishibashi, K., Ueda, K.: Analogical Modification in the Creation of Contemporary Art, *Cognitive Systems Research* 10 (3), 189–203 (2009)
43. Piaget, J.: *The Origins of Intelligence in the Child* (M. Cook, Trans.) Penguin, New York. (Original work published 1936) (1977)
44. Piaget, J.: *Play, Dreams and Imitation in Childhood* (C. Gattegno & F.M. Hodgson, Trans.) W.W. Norton, New York. (Original work published 1945) (1962)
45. Polya, G.: *How to Solve It*, (2nd ed.), Princeton University Press, Princeton (1957)
46. Poprcova, V., Stojanov, G., Kulakov, A.: Inductive Logic Programming (ILP) and Reasoning by Analogy in Context of Embodied Robot Learning, in *Proceedings of IJATS*, pp. 64-73 (2010)
47. Richards, I.A.: *The philosophy of rhetoric*. Oxford University Press, Oxford (1936)
48. Riesbeck C.K. & Schank R.C.: *Inside case-based reasoning*. Hillsdale (New Jersey): Lawrence Erlbaum & Associates (1989)
49. Rodari, G.: *The Grammar of Fantasy* (J. Zipes, Trans.) New York: Teachers & Writers Collaborative (1996)
50. Sangoi, M.: Interview with Bipin Indurkhy, *Humana.Mente Journal of Philosophical Studies* 23, 197–216 (2012)
51. Sawyer, K.: *Explaining Creativity*. Oxford (UK): Oxford University Press (2006)
52. Schön, D. A.: *Displacement of Concepts*. New York: Humanities Press (1963)
53. Shapira, O. & Liberman, N.: An Easy Way to Increase Creativity, *Scientific American* (Mind Matters), July 21 (2009)
54. Stojanov, G.: Embodiment as Metaphor: Metaphorizing-in The Environment in C. Nehaniv (ed.) *LNAI*, vol. 1562, pp. 88-101, Springer-Verlag (1999)

55. Stojanov, G.: Computational Models of Creativity: Taking Early Cognitive Development Seriously, to appear in Indyrkhy, B. and Manjalay, J. (eds.) *Cognition, Experience and Creativity*, Orient Blackswan Pvt. Ltd. (2012)
56. R. Sun, "A microfeature based approach towards metaphor interpretation, Proceedings of the 14th IJCAI, 424–429. Morgan Kaufman, San Francisco. 1995.
57. Veale, T. "Re-Representation and Creative Analogy: A Lexico-Semantic Perspective, *New Generation Computing*, 24 (3), 223-240. 2006.
58. T. Veale, Y. Hao, "A Fluid Knowledge Representation for Understanding and Generating Creative Metaphors, *Proceedings of COLING 2008*, 1–5. Manchester. 2008.
59. T. B. Ward, "Analyses, In M.A. Runco & S.R. Pritzker (Eds.) *Encyclopedia of Creativity*, (2nd ed.), New York: Academic Press. 2011.
60. J. Zhu, S. Ontanon, "Towards Analogy-Based Story Generation, in the Proceedings of ICCC-X, Lisbon, 2010.
61. D. Blank, "Implicit Analogy-Making: A Connectionist Exploration, 7. MAICS, Bloomington, US, 1996.
62. D. R. Hofstadter, THE COPYCAT PROJECT: An Experiment in Nondeterminism and Creative Analogies, AI Memo 755, MIT, The Artificial Intelligence Laboratory, 1984.
63. D. R. Hofstadter and M. Mitchell, An Overview of the Copycat Project , Technical Report CRCC-52-1991, Center for Research on Concepts and Cognition, Indiana University, Bloomington, 1991.
64. Goldschmidt G.: Visual analogy - a strategy for design reasoning and learning, in Eastman, C., Newsletter, W. & McCracken, M. (eds.), *Design Knowing and Learning: Cognition in Design Education*, New York: Elsevier, 199-219 (2001)
65. Klein, S., Aeschliman, J. F., Balsiger, D., Converse, S. L., Court, C., Foster, M., Lao, R., Oakley, J. D., and Smith, J.: Automatic Novel Writing: A Status Report, Technical Report 186, Computer Science Department, The University of Wisconsin, Madison, Wisconsin, 1973.
66. Dehn, N.: "Story Generation after Tale-Spin, In Proceedings of the Seventh International Joint Conference on Artificial Intelligence, 16–18. Los Altos, CA: William Kaufmann, Inc., 1981.
67. Lebowitz, M.: Story-Telling as Planning and Learning, In Proceedings of the Tenth International Joint Conference on Artificial Intelligence, Volume 1. Los Altos, CA: William Kaufmann, Inc., 1983.
68. Turner, S. R., Minstrel: A Computer Model of Creativity and Storytelling. Ph.D. dissertation, University of California at Los Angeles, Los Angeles, CA, 1993.
69. Pérez y Pérez, R.: MEXICA: A Computer Model of Creativity in Writing, Ph.D. dissertation, The University of Sussex, Falmer, UK, 1999.
70. Theune, M., Faas, E., Nijholt, A., and Heylen, D.: The Virtual Storyteller: Story Creation by Intelligent Agents, In Proceedings of the Technologies for Interactive Digital Storytelling and Entertainment (TIDSE) Conference, 204–215. Berlin: Springer, 2003.
71. Stock, O., and Strapparava, C.: The Act of Creating Humorous Acronyms, *Applied Artificial Intelligence* 19(2):137–151. 2005.
72. Manurung, R., Ritchie, G., Pain, H., Waller, A., O'Mara, D., and Black, R.: The Construction of a Pun Generator for Language Skills Development, *Applied Artificial Intelligence* 22(9): 841–869. 2008.
73. Tinholt, H. W., and Nijholt, A.: Computational Humour: Utilizing Cross-Reference Ambiguity for Conversational Jokes, In 7th International Workshop on Fuzzy Logic and Applications (WILF 2007), Camogli (Genova), Italy, Vol. 4578 of Lecture Notes in

Artificial Intelligence, ed. F. Masulli, S. Mitra, and G. Pasi, 477–483. Berlin: Springer Verlag, 2007.

74. Binsted, K., and Ritchie, G.: An Implemented Model of Punning Riddles, In Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94). Menlo Park, CA: AAAI Press, 1994.
75. Cope, D., Experiments in Musical Intelligence (EMI): Non-Linear Linguistic-based Composition. Interface Vol. 18: 117-139. 1989.
76. Grindlay, G.: and Helmbold, D., Modeling, Analyzing, and Synthesizing Expressive Performance with Graphical Models, Machine Learning 65(2–3), 2006.
77. Widmer, G., Flossmann, S., Grachten, M., YQX Plays Chopin, AI Magazine 30(3), 2009.
78. Colton, S.: Creativity versus the Perception of Creativity in Computational Systems Proceedings of the AAAI Spring Symposium on Creative Systems (2008)
79. Cohen, H.: On Modeling of Creative Behavior, Rand Corporation Report Series, P-6681, (1981)
80. Boden, M.: The creative mind, London: Weidenfeld and Nicholson (1990)
81. Wiggins, G.A.: A preliminary framework for description, analysis and comparison of creative systems, Knowledge Based Systems, 19 (2006) 449–458, Elsevier (2006)
82. Turner, S. A.: The Creative Process: A computer model of storytelling and creativity, Lawrence Erlbaum Associates, (1994)
83. Hugo Liu, and Push Singh.: Makebelieve: Using Commonsense Knowledge to Generate Stories. In Proceedings of the Eighteenth National Conference on Artificial Intelligence, 957-958, Edmonton, Alberta, Canada. AAAI Press (2002)
84. McCorduck, P.: Aaron's code: Meta-art, artificial intelligence, and the work of Harold Cohen, New York: Freeman. 1991.
85. Mark T. Keane.: Analogical Problem Solving. John Wiley & Sons, Inc., New York, NY, USA. 1988.
86. Keith J. Holyoak and Paul Thagard. Mental Leaps: Analogy in Creative Thought. The MIT Press (1995)
87. Ashok K. Goel. Design, Analogy, and Creativity. IEEE Expert: Intelligent Systems and Their Applications 12, 3 (May 1997), 62-70 (1997)
88. Kokinov, B., Petrov, A. Integration of 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. (2001)
89. Davies, J. & Goel, A.: Visual Re-Representation in Creative Analogies. Open AI Journal, 2:11-20 (2008)
90. Bhatta, S. & Goel, A.: An Analogical Theory of Creativity in Design. In Proc. 2nd International Conference on Case-Based Reasoning, Lecture Notes in Computer Science - 1266, pp. 565-574 (1997)
91. Swaroop s. Vattam, Michael e. Helms, and Ashok k. Goel. A content account of creative analogies in biologically inspired design. Artif. Intell. Eng. Des. Anal. Manuf. 24, 4 (November 2010), 467-481 (2010)
92. Schwering, A., Krünnack, K., K'uhnberger, K.-U., Gust, H.: Using Gestalt Principles to Compute Analogies of Geometric Figures. In 19th Meeting of the Cognitive Science Society (2007)
93. Keane, M.T., Analogical Problem Solving. (Cognitive Science Series).: Ellis Horwood. (1988)
94. Bhatta, S. & Goel, A.: A functional theory of design patterns. In Proceedings of the International Joint Conference on Artificial Intelligence, Nagoya, Japan, pp. 294–300 (1997)

95. J Davies, AK Goel, NJ Nersessian. A computational model of visual analogies in design, *Cognitive Systems Research* 10 (3), 204-215 (2009)
96. Dastani: Languages of Perception, M.: PhD Thesis, Universiteit van Amsterdam, Amsterdam (1998)
97. Dastani, M., and B. Indurkhy, B.: Modeling Context Effect in Perceptual Domains, in *Modeling and Using Context: Proceedings of the Third International Conference on Modeling and Using Context: CONTEXT 2001*, University of Dundee, Dundee, Scotland, 129-142. (2001)
98. Dastani, M., Indurkhy, B., and Scha, R.: An Algebraic Approach to Modeling Analogical Projection in Pattern Perception, *Journal of Experimental and Theoretical Artificial Intelligence* 15(4), 489-511 (2003)
99. Hofstadter, D., Sander., E.: Surfaces and Essences: Analogy as the Fuel and Fire of Thinking, Basic Books, New York, New York (2013)

Studies in Computational Intelligence Series - Springer Verlag

Analogical reasoning is known as a powerful mode for drawing plausible conclusions and solving problems. It has been the topic of a huge number of works by philosophers, anthropologists, linguists, psychologists, and computer scientists. As such, it has been early studied in artificial intelligence, with a particular renewal of interest in the last decade. The present volume provides a structured view of current research trends on computational approaches to analogical reasoning. It starts with an overview of analogical reasoning with an extensive bibliography. The 14 collected contributions cover a large scope of issues. First, the use of analogical proportions and analogies is explained and discussed in various natural language processing problems, as well as in automated deduction. Then, different formal frameworks for handling analogies are presented, dealing with case-based reasoning, heuristic-driven theory projection, commonsense reasoning about incomplete rule bases, logical proportions induced by similarity and dissimilarity indicators, and analogical proportions in lattice structures. Lastly, the volume reports case studies and discussions about the use of similarity judgments and the process of analogy making, at work in IQ tests, creativity or other cognitive tasks. This volume gathers fully revised and expanded versions of papers presented at an international workshop as well as invited contributions. All chapters have benefited of a thorough peer review process.