Computational Models of Analogy-Making
An Overview Analysis of Computational Approaches to Analogical Reasoning

Minithesis

written by

Tarek R. Besold
(born 16 February 1985 in Bayreuth, Germany)

under the supervision of **Prof. Dr. Michiel van Lambalgen**, and submitted to the Board of Examiners in partial fulfillment of the requirements for the

Logic Year Certificate

at the *Universiteit van Amsterdam*.

Institute for Logic, Language and Computation

Abstract

We provide an overview of research on computational models of analogy-making. The survey ranges from a fairly basic introduction to the topic of computational analogy-making to reporting some quite recent advanced results concerning the study of these systems, their properties and particularities. When doing so, we mainly take a cognitive modeling and computer sciences inspired point of view, mostly discarding other possible criteria such as psychological or biological adequacy. We introduce different abstract types of computational models of analogy-making in terms of symbol-based models, connectionist models and hybrid models, before having a more detailed look at one or two characteristic examples for implemented systems of each category. Concludingly, after summarizing the characteristics of the studied systems in a crisp synopsis, we present some basic aspects of Heuristic-Driven Theory Projection, a mathematically sound framework for analogy-making currently under construction.

Chapter 1

Introduction

Analogy and analogical reasoning can be encountered everywhere, from studies in academic philosophy ([1]) to the real-world use of a formerly unknown operating system on a computer without prior instruction, or from formal enquiries into forms of reasoning ([2]) to shortening the description of a task in an instruction manual of a refrigerator (e.g., "*To set the high limit value HL, proceed analogously as described for LL.*", p.31, [3]). Still, for a long time analogy was merely considered a special case of reasoning, mostly to be found in application when encountering creative solutions or in arts, as for example in the case of poetic writing. The scene has changed dramatically during the last decades, and today it is undoubted that the ability to see two a priori distinct domains as similar based on their shared relational structure (i.e., analogy-making) is one of the basic elements of human cognition ([4]). Some prominent cognitive scientists nowadays even consider analogy the core of cognition itself ([5]). Key abilities within everyday life, as communication, social interaction, tool use and the handling of previously unseen situations crucially rely on the use of analogy-based strategies and procedures. As also pointed out in [4], relational matching is also the basis of perception, language, learning, memory and thinking (i.e., the constituent elements of most conceptions of cognition). Even more, it seems these capabilities are not exclusive to our species, but can also be accessed by other primates ([6]).

Since the advent of computer systems, researchers in cognitive science and artificial intelligence have been trying to create computational models of analogy-making. Their motivation for doing so is twofold: On the one hand, a widely applicable and working analogy-making system would form a great step towards the goal of creating general artificial intelligence, i.e., human-like reasoning and thinking capabilities within an artificial computational framework. Also, on the other hand, the performance of certain theoretical paradigms and theories on how analogy-making works can be tested by means of computational implementations, by this also allowing for deductive inferences back into the domain of cognitive science. The present essay wants to sketch an overview of this development, starting with a historical note on Evan's ANALOGY ([7]) as one of

the first systems, and afterwards presenting a representative choice of the most popular systems in the field (whilst staying far away from raising any claim to completeness).

The paper is structured as follows: Sect. 2 offers a crisp general introduction into the field of computational analogy-making systems, also providing a categorization of different paradigms of system architecture into symbolic, connectionist and hybrid. Sect. 3 gives an overview of the historical ANALOGY system, which can be considered one of the forefathers of modern models of analogical reasoning. The presentation continues with the Structure-Mapping Engine and MAC/FAC in Sect. 4, using these as modern examples of symbolic systems. The connectionist domain within the field is represented by LISA in Sect. 5, followed by a contrasting elaboration on the STAR system family. As hybrid systems, Copycat and AMBR have been chosen, serving as representatives of a larger family of systems within a concise presentation. These overviews are followed by a short summarizing synopsis in Sect. 7, before briefly addressing HDTP, a mathematical sound framework for analogy-making currently under development, in Sect. 8. Sect. 9 concludes the paper.

Chapter 2

A General Categorization of Computational Models of Analogy-Making

In this short introductory part to the overall field of analogical reasoning by means of computer systems, the presentation will mostly go along the lines of a similar undertaking in [8]. As lined out there, analogy-making can broadly be characterized as a mapping process between a source and a target domain (see also Fig. 2.1 for a schematic visualisation). Following a schematization proposed in [10], and adding an initial step introduced in [11], the following five abstract processes are arguably considered necessary for analogy-making:

- The initial representation-building.
- The recognition of a source, based on a target description.
- The elaboration of a possible mapping between both.
- The evaluation of the mapping, and the transfer of the corresponding information from source to target.
- The final consolidation of the outcome, i.e., a concluding learning process, possibly leading to a certain behaviour or application of the newly obtained insights.

Whilst expansions or slight modifications of this scheme have been proposed in the meantime (see, for example, [12]), at this point the just sketched schema serves as a general formal model of analogy-making, and as a basic listing of the steps a computer system has to perform when doing analogical reasoning (although, as pointed out in [11], the building of a representation step is absent from many computer-based models).

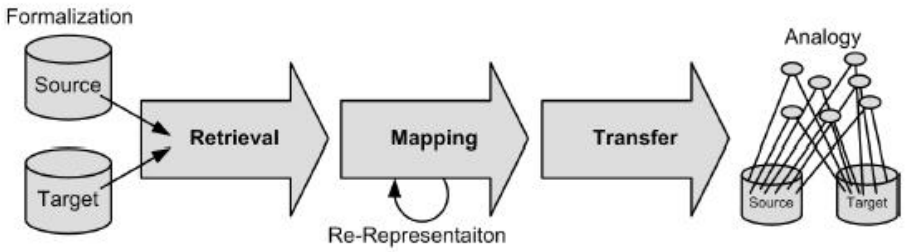

Figure 2.1: A schematic overview of the analogy-making process. ([9], p.252)

In the following subsections, we want to establish a categorization of analogy-making systems into three different classes, according to the respective underlying architectural paradigm (of course also other categorizations are possible, e.g., see Sect. 5 of [13] for a more pragmatic, application-based classificatoin).

The symbolic approach: Making things explicit

The symbolic approach in analogical reasoning systems broadly follows the symbolic paradigm known from the field of artificial intelligence, i.e., an account being based on the idea of manipulation of explicit symbols (named "*Good Old Fashioned AI*" or GOFAI by Haugeland in [14]). In the conception of symbolic models of analogy-making, usually the main focus lies on symbols and symbol-bound processing, as planning, searching and logical methods. One of the main advantages of this approach to constructing a model of analogical reasoning is the possibility of tracking and reconstructing the mechanisms at work, due to the explicit way of giving and storing information. Critics often doubt the cognitive adequacy of symbol-based systems, arguing that humans probably don't mentally do explicit symbol handling or logical inference. Also, these systems often show deficits in adaptivity and handling of unforeseen situations, due to the static (often rule-based) architecture of the system.

The connectionist approach: The power of many

The antithesis to the symbolic approach is the connectionist paradigm. In this setting, analogical reasoning is modeled as emerging from interconnected networks of more simple units, possibly down to the level of mere McCulloch-Pitts neurons within an artificial neural network ([15]). Characteristic element of such a framework is a graph-like model consisting of interconnected nodes, weights, and the propagation of activation patterns. Some of the advantages of this

construction paradigm are a more cognitively plausible architecture, as also human brains are made up of interconnected neurons (of course still huge gaps and deficits between the concepts remain), a good capability of modeling relational structures, and high adaptivity and learning abilities. Downsides are the loss of explicitly represented knowledge, a mostly only indirectly controllable system behaviour, and a general lack of a priori predictability or a posteriori explicability of the system ("*black box behaviour*").

The hybrid approach: Unifying philosophies

The third category of analogy-making systems are hybrid models, sharing features of both, symbolic systems and connectionist ones (where the latter are here ment to be understood in a rather broad reading, generally covering systems based on network architectures). The use of techniques from both subfields offers various advantages, as shortcomings of one approach may be remedied by some functionality of the other, and vice versa. Also, in our reading these systems provide the highest degree of cognitive adequacy, as simple rule or reflex triggered mechanisms may be modeled alongsides functionality emerging from a connectionist network. Unfortunately, some disadvantages of the inidividual paradigms, as e.g. the black box behaviour characteristic from the connectionist side, carry over, too. Still, the author of the present paper is convinced that on a longer perspective, only hybrid architectures might allow for a quasi-human like style of reasoning behaviour to be shown by an artificial computer system (if so at all).

Chapter 3

A historical account: ANALOGY

Although the first computer model of analogy-making probably was Reitman's Argus system ([16]), a program which solved fairly simple proportional analogies (but from the architectural point of view already included quite advanced principles such as conceptual networks and the automatic building of representations for the source and the target of the analogy process), the arguably most prominent model from the early days of computerized analogical reasoning and analogy-making is the ANALOGY system by Evans ([7]). ANALOGY also was a system designed for solving proportional analogies, in this case between simple geometrical figures (i.e., source and target domain are the same, being populated by geometrical figures). Due to its relative popoularity, in the following we will give a crisp overview of the main features and the operating principles of ANALOGY, serving as a historical introduction to the more recent models and systems studied in later sections of the present paper.

[7] provides a thorough introduction to ANALOGY (for an even more detailed, complete elaboration see [17]). Evan's model was designed to solve geometric-analogy problems, as can classically be found in intelligence tests (actually, Evans took his testing samples from the 1942 version of the "*Psychological Test for College Freshmen*", issued by the American Council on Education). An example for a typical task of this kind is given in Fig. 3.1, where the test subject has to decide which out of the figures in the second row systematically continues the series of objects in the first row (i.e., has to decide the question "*A is to B, as C is to X*", where X is to be taken from figures 1 to 5). In the given example, the desired answer would, e.g., be "*Element 4*", based on the relation established by the conservation of the outer circular shape, and the disappearing of the inner square, in analogy to the disappearing of the inner triangle when comparing element A to element B.

For solving this type of problem, ANALOGY followed a quite straightfor-

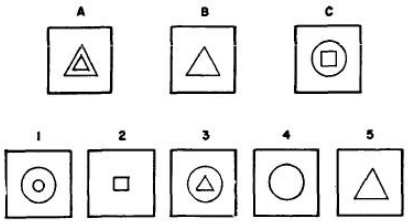

Figure 3.1: A typical analogy problem as addressed by ANALOGY: The three problem figures are to be found in the upper row, the possible solution figures in the lower one. ([7], p.327)

ward, yet remarkably elaborate approach. As an initial step, descriptions of the given figures have to be read as input. In [7], these descriptions are still hand-made by the operator of the system, using LISP as representation language. Still, according to Evans, also an automatic generation of the needed input would have been realizable without greater efforts. On an abstract level, the descriptions represent the respective figures using straight line segments and arcs of circles, up to an arbitrary precision (given no limitations on the representation length). Given these initial descriptions, ANALOGY starts a decomposition process, partitioning every problem figure into subfigures (in the following also referred to as "*objects*"). Evans points out that already a very simple subdivision approach, only separating a given figure into its connected subfigures, yielded quite satisfying results. However, a more sophisticated program was implemented, also allowing for the separation of overlapping objects. The intermediate state of such an implementation, reported in [7], was capable of finding all occurences of a simple closed figure in any other connected figure, being subsequently able to separate the connected figure into the simple closed subfigure, and the remainder of the initial connected figure.

Taking the output of the just described pre-processing steps, a routine calculating specified properties of the objects under consideration, and relations among those, is called (where ANALOGY allows for an easy exchange of the property set). A simple example for such a relation would, e.g., be that one object lies within another. If a relation is found, a corresponding statement is added to the output description of the respective object. This part of ANALOGY requires a quite broad variety of routines and mechanisms for analytic geometry operations, for example allowing for the handling of intersections of arcs of circles and straight line segments in every situation possible. Additionally, at this stage of the program, calculations concerning similarities are conducted. For each suitable pair of objects, all elements of a certain set of transformations mapping one object of the pair into the other are found. This set of transformations is constructed via composition from Euclidian similarity transformations (i.e., rotation and uniform scale alteration), together with hor-

izontal and vertical reflections. Given descriptions of any pair of line-drawings, the corresponding parameters of all members of the transformation set for a mapping of one drawing into the other are computed (using metrical tolerance as tunable parameter within the respective programs).

Summarizing the part of ANALOGY just sketched, one can interpret it (in accordance with [7], p.331) as a "*pattern-recognition program with built-in invariance under scale changes, rotations and certain types of reflections*", augmented with a module for the recognition and expatiation of relational structures. Its main functionality is the construction of a topological matching, based on metric comparisons between selected figurative elements.

The second part of the analogy-making system is given the similarity information for every required pair of objects within a problem figure, as well as between figures, and the decomposition information, together with the property and relational information, and subsequently yields the number of the solution figure as output, or answering that no solution could be found. The first step on the way to this final result is the generation of a single rule, or a set of alternate rules, transforming one figure into another (i.e., listing how objects have to be deleted, expanded or altered in properties and relations in order to allow for a mapping between figures). Then, the crucial step is the generalization of the just obtained rule(s), weakening each rule just as much as to maintain the mapping between the original figures, but also making the mapping of the third problem figure into (exactly) one of the possible answer figures possible. As this construction of a shared generalization may create a set of possible generalizations, being distinct in their discriminative power, a single strongest rule, according to some predefined criterion, has to be selected. Finally, the outcome of the application of this rule (if the result is unique) is considered the answer to the problem task. At this point it should be remarked that no semantical information concerning the properties and relations is used in the process, i.e., no interpretation of geometrical meaning is involved.

In the implementation of ANALOGY presented in [7], the number of elements which are altered, removed or added for the mapping between the two initial figures, has to be the same as the one taken into account when selecting the answer figure corresponding to the third problem figure. If a generalized rule does not comply with this criterion, it is discarded, and the process will continue with another generalized rule (if there are any, if not, the problem will be answered as unsolvable for the system). If the outcome is not unique, ANALOGY takes, for a given matching of the third problem figure to an answer figure, each rule between the two initial problem figures, and tries to adapt it to the new setting (i.e., comparing each rule between the initial two problem figures with each rule between the third problem figure and a possible solution figure, and finding more general candidate rules covering both of the original rules for each admissible comparison). This gives one or more new generalized rules fitting both, the mapping between the two initial figures, and the mapping between the third figure and the possible answer. Now, again the strongest (i.e., most specific) rule is selected, considering ties between rules as a reason for discarding the entire method as failed and trying another configuration (if

there is any left).

Cutting a longer story short, in accordance with a similar résumé in [10], it can be stated that ANALOGY elaborates analogies between two representations within one problem domain via a two-level mapping procedure. Direct object mappings can be encountered during the rule generation process, followed by object role mappings during the comparison part of the program. Constraints on these mapping processes are given in the requirement of shape equality (i.e., both objects have to be of the same geometric type) in the case of direct object mappings, and in that objects have to play an equivalent role (e.g., addition or omission) within rules in the object role mapping stage. Due to its construction principles (i.e., the restriction to object-level comparisons only), ANALOGY in some cases shows to be severely limited in that it cannot find solutions to analogy tasks which would require both, transformation of relations as well as of objects. Cases in which an above relation between the initial two problem figures should correspond to a below relation between the third problem figure and the answer, ANALOGY would drop these relations when generalizing, possibly not allowing for a correct choice of answer.

In [7], Evans also provides a very crude comparison between the performance of ANALOGY and human test subjects on his test sample, claiming that ANALOGY would be able to compete with 9th to 12th grade college-preparatory students on this kind of geometric-analogy task. Furthermore, he states that rather simple modifications to the first part of ANALOGY's program structure might improve the system's performance in a way to significantly outperform its human competitors. Unfortunately, to the best of our knowledge, no further performance evaluation, confirming or refuting these conjectures, seems to be available at present.

Chapter 4

Symbol-Based Modeling: The Structure-Mapping Engine and MAC/FAC

Departing from the baseline given by Evan's ANALOGY system (which, with its rule-oriented architecture, and explicit information storage and handling, can clearly be categorized as symbol-based), we now want to turn to a more recent symbolic approach to computational analogy-making, the Structure-Mapping Engine (SME) presented by Falkenhainer, Forbus and Gentner ([18]), together with a prominent model of similarity-based retrieval MAC/FAC ([19]), commonly used as front-end for the SME.

As pointed out in [11], an influential family of computational analogical reasoning systems has been based on the Structure Mapping Theory (SMT) as introduced by Gentner ([20]). SMT assigns a crucial role in the process of analogy-making to structural similarities between base and target domains: The theory emphasizes the dependence of rules exclusively on syntactic properties of the knowledge representation, and the possibility of distinguishing analogies from literal similarity statements or other, distinct types of comparison. The corresponding mapping principles can be characterized as a mapping of relations between objects from base to target domain, and a choice of mapped relations based on a certain systematicity, rooted in the existence of some designated higher-oder relations. [11] gives a short list of conjectures underlying the Structure-Mapping Engine implementation of SMT:

- The mapping part of the process is widely isolated and disjoint from other sub-mechanisms.
- Relational matches get assigned a higher priority than mere property-based matchings.

- In order to be interpreted as corresponding relations in the base and target domains, relations have to be identical.
- Relations which form arguments of higher-order relations, which in turn can also be used for a mapping, get assigned a higher priority than mere isolated relations.

In [21], Forbus gives a short sketch of how the SME works (for a more detailed account see Sect. 4.1). As input, the engine is given two structured propositional representations, with an internal predicate/argument structure, of the base and the target domain of the analogy. Possible elements of these representations can be unary predicates indicating features, relations expressing connections between domain entities, and higher-order relations expressing connections between relations. Given this initial information, SME searches for possible mappings, where each mapping contains a set of correspondences aligning entities in the base domain with entities in the target domain, and candidate inferences, representing statements about the base domain which are conjectured to carry over to the target domain due to the correspondences.

MAC/FAC was developed by Forbus and Gentner, later also together with Law, as a front-end to the SME. As explained in [8], MAC/FAC is a two-stage analogical retrieval engine, providing SME with the needed input representations (for a more detailed treatment of MAC/FAC and the underlying mechanisms, see Sect. 4.2). The underlying idea is the conjecture that episodes are encapsulated representations of past events, to be found in a dual encoding within long-term memory (see [11]): A full-fledged predicate calculus representation of properties and relations of entities within an episode, and a compressed vector representation indicating relative frequencies of predicates used in the detailed representation. The retrieval process subsequently performed by MAC/FAC can then be divided into two sub-parts, a superficial search for episodes sharing predicates with the target problem, and an in-depth search for the episode matching best with the target based on the predicate calculus representations of the highest ranked episodes from the step before. As noted by Kokinov and French, the two-staged approach also reflects two basic characteristics of SMT, the dominant role of superficial similarity, and the influence of structural similarity.

4.1 The SME framework: Emphasizing structural similarity

As Falkenhainer, Forbus and Gentner state already in the abstract of [18], the Structure-Mapping Engine has been developed to explore Gentner's SMT, providing an SMT-conform framework for the construction of matching algorithms for analogies (i.e., SME is not a single matching system, but wants to serve as a simulator for several possible matchers). The SME has soon become a highly influencial model in the field, paving the way for numerous more or less similar

successors and related conceptions, and to the best of our knowledge can today be regarded as one of the cornerstones within the field of computational models and implementations of analogy-making systems. Due to this vast success and importance, the following presentation will in parts be more detailed than most of the more schematic sketches of distinct systems and models in later sections. Also, by this more extensive insights on how the SMT as an influential and far-reaching theory from the more conceptual side of cognitive sciences can be implemented and tested in a computational system shall be provided.

In the reading of the concept of structure-mapping underlying the presentation in [18] (which has closely been followed as reference for the elaborations on the SME in this section), analogical processing can be subdivided into three consecutive steps: An access phase, during which, given a target situation, another description similar or analogous to the target is retrieved from long-term memory, subsequently constituting the base of the analogy; a mapping and inference phase, representing the construction of a mapping, which consists of correspondences between base and target and possibly also includes the candidate inferences (i.e., specifications of what additional base knowledge might be transferred to the target) admissible within the analogy; an evaluation and application phase, during which the quality of the match is estimated according to the three criteria structure (i.e., number of similarities and differences, degree of structural similarity, quantity and quality of knowledge provided via the candidate inferences), validity (i.e., checking inferences against world knowledge for meaningfulness and soundness, possibly performing inferential refinement steps), and (pragmatical) relevance of the analogy for the reasoner. The SME now focuses on the mapping phase, providing a structural and domain-independent evaluation of the match. Here, the structure-mapping notion of structural consistency is already initially included in the system, but can be expanded or replaced by implementing new match rules at any time.

Facts in SME are represented by means of a typed higher-order predicate calculus, whilst trying to keep the representation as general as possible, as to ensure domain-independence of the engine. The language is constructed of entities, predicates and description groups (dgroups), which in turn are characterized as follows:

- Entities are logical individuals, the domain objects and constants. Examples for typical entities would be physical objects, their material, aggregate phase or temperature. Primitive entities are the tokens or constants of a description, an option for establishing hierarchies of entity types is provided.

- Predicates are divided into three classes: Functions, attributes and relations. Functions serve for mappings between entities, or entities and constants. The structure-mapping framework allows for substitution of functions, acknowledging the possibility of indirect referral to entities. Attributes have a property describing function, and are by definition restricted to take only one argument. The case where there would be multiple arguments is solved via classification of the predicate as relation.

Whilst generally an attribute can be seen as logically equivalent to a function and a constant, under structure-mapping this is not the case. In the implemented style of analogy-making, attributes are discarded, as long as they do not form part of a higher-order structure of some kind. When performing a matching, attributes have to match identically, and it is assumed that a reasoner has a certain datum represented in only one of the two possible forms at a time. Relations, similar to attributes, range over truth values. They always have multiple arguments, where the arguments can take the form of other predicates, or entities. Also relations have to match identically in the structure-mapping framework. When declaring a predicate, after indicating the type, syntactic information concerning the predicate can be provided to the SME via explicitly declaring the predicate commutative or n-ary in the number of arguments.

- Dgroups are collections of primitive entities and expressions (i.e., predicate instances and compound terms) concerning those. The expressions and entities in a description group are collectively referred to as items.

In order to give an overview of the SME algorithm, vocabulary for addressing certain structural relations between items of the just defined type is needed. As these relations form directed acyclic graphs, each item can be seen as corresponding to a vertex in a graph. If item i has item j as an argument, we may introduce a directed edge from the node i to the node j, i.e., the offspring of an expression are its arguments. Primitive entities have no offspring, expressions naming entities by compound terms are handled equal to other items. An item k within the transitive closure of an item l is called a descendant of l, whilst l is said to be an ancestor of k. Items without ancestors are called roots, and $Reachable(x)$ denotes the transitive closure of the subgraph starting at x, with the depth of an item with respect to $Reachable(x)$ being equal to the minimal number of edges needed to reach the item departing from x.

Two more concepts play a crucial role in the SME implementation of the structure-mapping process, namely global mappings (gmaps) and match rules. A global mapping is a structurally consistent interpretation of the comparison between a base and a target, consisting of three parts: A set of pairwise matches between expressions and entities of the two respective dgroups, called correspondences; a set of new expressions which all are conjectured to hold in the target dgroup due to the comparison, called candidate inferences; a numerical estimate of match qualities exclusively based on the gmap's structural properties, called structural evaluation score (SES). The data for the estimation of the match qualities is provided by match rules, which specify which pairwise matches are possible, and provide local measures of quality for the SES computation. Here, great flexibility is introduced to the SME, as for building a new matcher, only the set of matching rules has to be modified accordingly.

Given these preliminaries, the SME algorithm can conceptually be divided into four steps:

1. The construction of local matches, finding all pairs of items in the base and

target domain which can possibly match, and creating a match hypothesis for each such pair (for representing the possibility that the local match is part of a global match).

2. The construction of gmaps, combining local matches into maximally consistent collections of correspondences.

3. The candidate inference construction, deriving the inferences corresponding to each gmap.

4. The match evaluation, attaching evidence to local match hypothesis, and consecutively assigning structural evaluation scores to each gmap.

Local match construction

The initial local match construction phase of SME serves for finding potential matches between items in the base and target domain. Allowable matches are specified by so called match constructor rules, roughly consisting of a base variable, a target variable, possibly an associated test form, a rule body and a trigger. Given that base variable and target variable can be bound to items from the respective dgoups, and (if present) the condition stated in the test form is fulfilled by the binding, the rule body will be executed. The trigger decides on the type of execution, if the trigger type is set to filter, the rule will be applied to each pair of items from base and target, creating a set of match hypotheses between individual expressions in base and target, respectively. If the trigger is set to intern, the rule will be applied to each newly created match hypothesis, binding variables to corresponding base and target items, and thus resulting in more matches suggested by the given match hypotheses. Having run the match constructor rules, a collection of match hypotheses has been obtained. In the following, if item $i \in base$ and $j \in target$ match, we will denote this by $Match(i, j)$. Again, for the structural properties of graphs of match hypotheses the graph-theory inspired vocabulary, introduced before for dgoups, can be used, as the collection of match hypotheses can also be viewed as a directed acyclic graph, having at least one root.

Gmap construction

The most complex of the four SME phases is the global map construction, where local match hypotheses are combined into collections of gmaps. Global maps here consist of maximal, structurally consistent collections of match hypotheses from the local match construction phase. To be structurally consistent, here a match hypothesis has to be one-to-one (i.e., the assignments of base to target items given by the totality of hypotheses in the collection have to be bijective), and it has to hold that if a certain hypothesis is in the collection, so are all match hypotheses which pair the arguments of this hypothesis' base and target items (support property). Maximality of a collection is reached if adding any

additional hypothesis would cause the collection to become structurally inconsistent. The gmaps are built in a two-step process, the computation of consistency relationships and the merging of the match hypotheses:

- In the computation of consistency relationships, for each match hypothesis the set of entailed entity mappings is found, together with a list indicating which of the remaining match hypotheses are locally conflicting with the hypothesis under consideration, and a similar list of match hypotheses which are structurally inconsistent with respect to the initial hypothesis.

  Recursively making use of the support property for structurally consistent collections of match hypotheses, each individual match hypothesis implies a certain set of entity correspondences, summarized within an entity map (emap). An emap thus expresses a match hypothesis between entities, and $Emaps(Match(i,j))$ denotes the emap implied by the match hypothesis $Match(i,j)$, i.e., denotes the union of the emaps supported by the descendants of $Match(i,j)$ (where also hypotheses involving functions are included). As to guarantee that all occuring mappings are one-to-one, it is helpful to associate with $Match(i,j)$ the set of competing match hypotheses, which would also provide alternate mappings for i and j. We call this set $Conflicting(Match(i,j))$ in the following, referring to its elements as conflicting match hypotheses of $Match(i,j)$ (where naturally no element of $Conflicting(Match(i,j))$ can be an element of the same gmap as $Match(i,j)$ is). Taking $Conflicting(Match(i,j))$ and $Emaps(Match(i,j))$, by means of recursion the set $NoGood(Match(i,j))$, i.e., the overall set of match hypotheses which may never be in the same gmap as $Match(i,j)$, can be constructed. If $Match_i$ is an emap, $NoGood(Match_i) = Conflicting(Match_i)$, whereas in any other case $NoGood(Match_i)$ is given by the union of $Conflicting(Match_i)$ with all the $NoGood()$ sets of descendants of $Match_i$.

  Having computed the $Conflicting()$ set for each match hypothesis, $Emaps()$ and $NoGood()$ can be computed for each emap, followed by the computation of $Emaps()$ and $NoGood()$ for all other hypotheses, based on a propagation of the $Emaps()$ data to the respective ancestors. When this process is completed, a search for internally inconsistent match hypotheses, which cannot be part of any valid gmap, is conducted. Here, a match hypothesis is called inconsistent, if the emaps entailed by one subgraph of its descendants are in conflict with the emaps of another subgraph.

- Gmap construction collects sets of consistent match hypotheses, following a top-down approach, starting at the roots. As inconsistencies propagate upwards in the graph, if a root is found to be consistent, the entire structure under it has to be consistent. As there are typically several consistent roots of the graph of match hypotheses, several initial gmaps have to be considered. As maximality was one of the characteristics of true gmaps, the initial gmaps have to be merged into larger, structurally consistent collections, resulting in maximal collections of match hypotheses.

For doing so, initial combinations are formed from the descendants of the highest-order structurally consistent match hypotheses, initial gmaps with overlapping base structure (subject to structural consistency) are merged, and complete gmaps are formed by merging the partial maps (subject to structural consistency), discarding all but the maximal results.

In order to form the initial combinations, interconnected and consistent structures have to be combined. Here, each consistent root, together with its descendants, forms an initial gmap. If a root is found to be inconsistent, each immediate descendant is considered a potential root, and the procedure is applied recursively, finally resulting in a set $Gmap_1$. Starting from an empty $Gmap_1$ set, for every root $Match(i, j)$, if it is found to be consistent, a gmap GM is created, with its set of elements being equal to $Reachable(Match(i, j))$. In case the root is found to be inconsistent, recursion on the offspring of $Match(i, j)$ is started. Having done so, for every gmap GM within $Gmap_1$, the set $NoGood(GM)$ is defined as the union of the $NoGood()$ sets of all matching hypotheses serving as a root in GM. Similarly, the set $Emaps(GM)$ is set to the union of the $Emaps()$ sets of all matching hypotheses serving as a root in GM. Concludingly, in order to obtain maximal sets of correspondences, mutually consistent gmaps (i.e., the elements of one gmap are not to be found in the $NoGood()$ set of the other gmap, and vice versa) have to be merged.

Now, $Gmap_1$ is revised, and a search for elements with identical roots in the base structure is conducted. If such elements with shared base structure are found, and their correspondences are not structurally inconsistent, there has to be some connecting structure in the base, not to be found in the target. Combining such elements, the resulting partial gmaps are collected in a set $Gmap_2$. Now, taking two elements from $Gmap_2$ with disjoint relational correspondences, each such pair can be merged, given that they sanction consistent sets of emaps. By doing so, finally all consistent combinations of gmaps from $Gmaps_2$ can be obtained by successive unions, discarding all but maximal combinations.

Candidate inference construction

Gmaps represent sets of correspondences which can be used for interpreting the match. For information to be carried over from the base to the target (i.e., for the essential function of analogy-making), two conditions have to be satisfied: It has to be consistent with the substitutions imposed by the gmap, and its subexpressions must at some point have shared parts with the base information belonging to the gmap (a property called structural grounding by Falkenhainer, Forbus and Gentner). Structures complying with these constraints are considered the candidate inferences for the respective gmap. These candidate inferences for a gmap GM are computed by examining each root in the base dgroup, and checking whether it is an ancestor of the root of a match hypothesis in the gmap. If this test is positive, any elements in the set of descendants of

the root from the dgroup, which are not to be found in the base items of GM, are included in the set of candidate inferences.

Match evaluation

In many cases, a single base and target pair might yield several gmaps, each corresponding to a distinct interpretation. As a means to select the most fitting interpretation as analogy, a structural evaluation score is introduced. This score is obtained in yet another two-phase process. A first stage assigns weights to individual match hypotheses, the second phase summarizes these scores into a single score for the gmap comprised of the hypotheses. SME uses a belief maintenance system (BMS, see for example [22]) for handling the numerical evidence, so that each match hypothesis and gmap gets assigned a BMS node to record the respective evidential information. The structural evaluation score of a gmap is computed by summation of the evidence of its match hypotheses. In turn, match evidence rules can add evidence to a match hypothesis in two ways, by installing relationships between different hypotheses and propagating a certain percentage of a hypothesis' degree of belief to the respective offspring, or by directly altering the local properties of a rule. Finally, having obtained the structural evaluation score, the interpretation with the highest score is selected as analogy.

Concerning the performance of the SME, [18] gives positive reports of applications in cognitive simulation studies and a machine learning project. Guiding SME with adequate analogy rules, the system has shown to replicate human performance in the cognitive simulation studies, and has also shown its usefulness in providing the means for constructing new qualitative explanatory theories in the framework of a machine learning project. Although a computational complexity analysis indicates that the worst-case performance of the SME algorithm lies within $O(N!)$, in practice the system exhibits surprisingly moderate requirements, with most steps being polynomial and bounded by an order of N^2. It is worth remarking that, contrary to many other cases, here the determinant characteristic for computational efficiency is not length, but the degree of structure of knowledge, where SME performs better with systematic, deeply nested descriptions. Concerning follow-up developments, a later version of the SME ([23]) included the use of pragmatics, together with re-representation techniques for matching related, but still distinct predicates.

4.2 The MAC/FAC system: Many are called, but few are chosen

Two years after the publication of the SME in [18], Gentner and Forbus presented MAC/FAC ([19]), a model of similarity-based retrieval, which can conceptually be seen as a front-end to the structure-mapping engine. MAC/FAC is an attempt at constructing a model producing similarity and analogy phenomena

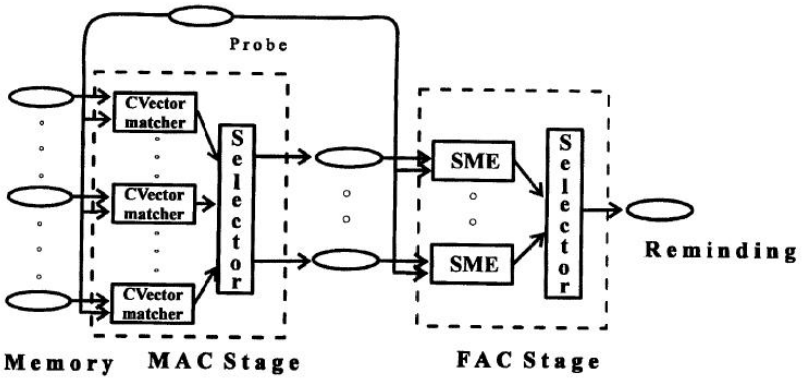

Figure 4.1: A schematic overview of the 1995 version of the MAC/FAC system, depicting the memory input, the MAC stage with parallel content vector matchers for processing the sample (here: "*probe*"), the FAC stage, and the final output. ([24], p.149)

in a statistically plausible manner: Whilst producing mainly literal-similarity (i.e., based on both structural and superficial commonalities) and superficial reminding, by times also a spontaneous analogy (i.e., structural similarity only) shall occur. The underlying theory assumes that similarity-based transfer can be decomposed into three subprocesses. Assuming that a target situation is currently present in a subject's working memory, a similar base situation in long-term memory (LTM) has to be found and accessed, subsequently a mapping from base to target has to be established, concluding with an evaluation of the mapping. As mechanism for the alignment and the import of inferences, SMT is the process model of choice, and (as already in the SME setting) structural consistency and one-to-one mapping govern the computation of inter-situation mappings.

The MAC/FAC model is (as already suggested by its naming) a two-stage process model (see also Fig. 4.1), where (MAC stage) a computationally efficient filter is used to select a subset of likely candidates within the large number of cases in memory (which is simply assumed to be a pool of descriptions), followed by (FAC stage) a more costly processing step. The inputs to MAC/FAC are the pool of memory entries and a sample description for which a match is to be found, the outputs take the form of a certain memory description and a comparison of the sample with the latter. Both, MAC and FAC, consist of a matcher and a selector, with the matcher being applied to every input description, and the selector using the matcher's evaluations when selecting which comparisons are to be returned as output of the respective stage.

The task of the MAC stage can be summarized as using a cheap matcher for estimating how FAC would rate certain comparisons, and by this narrow down

the number of candidates for the FAC stage to a processable number. An obvious estimate is the number of match hypotheses that FAC would generate when comparing a memory item to the sample, i.e., the numerosity of the comparison. If only few local matches are hypothesized, the best global interpretation cannot be large either. Unfortunately, the converse does not hold, as a large number of local matches does not automatically induce a large global interpretation, making numerosity not a perfect estimate. Numerosity is estimated in a quite elaborate manner (whilst earlier versions of MAC/FAC used a mere brute-force approach, see [25]): Denoting with P the set of functors (predicates, functions, connectives) within the descriptions that constitute memory items and samples, a content vector of a structured description now is an n-tuple of numbers, each entry corresponding to a single element within P. Given a description D, the number of occurrences of P in D is indicated by the value of the respective entry in the content vector. Clearly, concent vectors are easy to compute departing from structured representations (as computation boils down to simple counting the number of occurences of functors), and can be stored in an economical way.

Now, given pairs of memory entries and corresponding content vectors, whenever a sample is handed over to the MAC stage, its content vector is computed, and each item within the memory pool gets assigned a score equal to the dot-product of the item's content vector and the sample's content vector. The MAC selector then searches for the highest score, and hands the corresponding element, together with additional elements scoring over a certain threshold (90% of the maximal score in [19]), over to the FAC stage (which subsequently mitigates also one of the main weaknesses of the content vector approach to measuring similarity by adding inferential power via its structural matcher, making up for the total lack of correspondences and candidate inferences not provided by the content vectors).

MAC/FAC uses literal similarity computation in the matchers of the FAC stage, implemented by using the Structure-Mapping Engine in literal similarity mode (i.e., in difference to the analogy mode described earlier, both relational predicates and object descriptions are mapped, allowing for the computation of relational similarity, object similarity or overall similarity, see [24]). SME is used to calculate structural alignments of the sample with the items handed over from MAC (with MAC only considering functor overlap, a big part of MAC's output will be discarded during the structure-sensitive FAC stage, as pointed out in [26]). Thus, the FAC stage acts as a structural filter, taking as input the memory descriptions from MAC, does the alignment computation, and outputs a set of correspondences between structural descriptions, a numerical evaluation score of the overall match quality, and a set of candidate inferences representing the conjectures concerning the sample which are sanctioned by the comparison. Here, the FAC score for each pair is the structural evaluation score of the best-ranked mapping, and the returned output consists of the top-scoring match plus any others within a certain range from it (again, a score of 90% of the maximal score has been used in [19]).

A comparison of the performance of MAC/FAC on a similarity/analogy task

with the performance of humans undergoing the same experiment, shows that MAC/FAC's results match those of humans ([25]), also meeting the requirements concerning statistical distribution of the different types of similarities and analogies already mentioned initially. Sensitivity studies reported in [24] moreover strengthen the position that these encouraging results are not due to random or unintended phenomena (choice of algorithms, nontheoretically motivated parameters), but really are a consequence of the underlying theory and model.

Chapter 5

Connectionist Modeling: LISA and the STARs

A different, almost antithetical approach to constructing a computational model of analogy-making is taken by the designers of connectionist models of analogical reasoning. Although the ACME system by Holyoak and Thagard ([27]) probably was the first connectionist model in the field, relying on a localist constraint-satisfaction connectionist network, in the present paper we want to focus on two more recent examples: the LISA model by Hummel and Holyoak ([28]), and the STAR system family by Wilson, Halford et al. ([29], [30]). Where the SME was built in accordance with a symbolic architectural paradigm, LISA and the STARs are representants of the connectionist type of systems (where LISA constitutes a special case by itself, being a so called symbolic connectionist model, [31]). The first subsection of this part of the present paper will give a more in-depth overview of LISA, being followed by an introduction to STAR, also pointing at some of the main differences to LISA, in the second subsection. But before providing these more detailed reports on the aforementioned systems, again we want to give a short conceptual overview, based on [8] and [11].

For associating structures, LISA relies on partially distributed concept representations, selective activation and dynamic bindings, only binding node structures together which oscillate in synchrony, allowing both working memory (WM) and long-term memory to interact during retrieval and mapping ([8]). The idea underlying this approach is based on the introduction of an explicit time axis, making the time of activation an additional, independent model parameter, and providing a means for modeling patterns of activation which are oscilating over time ([11]). If patterns oscilate in synchrony, they are assumed to be bound together, if the oscilation is asynchronous, no binding is considered. Thus, a whole statement can be represented by periodic alternations of groups of statements. Representations in LISA's WM are distributed over a network of semantic primitives, whereas in its LTM a localist paradigm is followed, using

separate units representing episodes, propositions, components of propositions, predicates, arguments and bindings. Within these structures, retrieval is performed by spreading activation, and mapping is modeled as a process of learning new connections between active nodes. Functionally, the LISA model manages to integrate both processes, the retrieval of a base and the subsequent base to target mapping.

The STAR model family follows a conceptually different path. According to [11], STAR-1 ([29]) was the first distributed connectionist model in the field, being based on the tensor product connectionist modeling approach suggested by Smolensky (i.e., on the idea that pairs of values and variables can be represented through accumulation of activity in a collection of units, each of which computes the product of a feature of a variable and a feature of the corresponding value, see [32]). Here, complex propositions are represented by the tensor product of the vectors corresponding to its primitive parts, subsequently also allowing for the extraction of any primitive part by (possibly iterated) application of a generalized dot product operation. STAR-1's LTM is represented in form of a tensor, which is the sum of all tensor products representing individual statements. The application scenario of STAR-1 was the solving of proportional analogies. STAR-2 ([30]) is a more mature version of STAR-1, mapping complex analogies by sequentially concentrating on parts of the domain, i.e., simple propositions with at most four dimensions, and selecting a best map for the propositions' arguments by means of a parallel computation in a constraint satisfaction network (a method for coping with the – in the number of arguments of predicates exponential – blow-up of the needed number of units for tensor product representation).

5.1 The LISA model: Learning, Inference, Schemas and Analogy

In this subsection, we want to have a closer look at one of the in our oppinion most ambitious systems within the field of connectionist models of analogy-making, LISA ([28]). The presentation will stay close to Hummel and Holyoak's report in [31].

As mentioned before, LISA constitutes a special case within the class of connectionist models of analogy, being an attempt at building representations of a symbolic style in a neurally inspired network model architecture, earning it the name of a symbolic connectionist system. Hummel and Holyoak try to carry over some abilities of symbolic systems, namely the possibility to bind roles to their fillers dynamically, and to represent the obtained bindings independently of the roles and fillers (i.e., a process resulting in the binding of representations into propositional structures), to a neurally more plausible seeming neural network architecture, representing propositions as distributed patterns of activation. Thus, whilst aiming at preserving the generalization capacities of a connectionist model, the structure sensitivity of a symbolic system shall be

added ([28]). This shall be achieved by the already described synchrony of firing approach for binding representations of roles to representations of their fillers: A binding between two elements is represented by a synchronous firing pattern of the corresponding units, whilst an asynchronous pattern indicates an absence of any kind of connection. Thus, binding information can be represented explicitly and independently of the representation of the bound elements, and the model accords with the computational requirements for dynamic binding as to be found in a symbolic system. As pointed out in [28], dynamic binding still has its limitations: The domain of the dynamic binding process is naturally limited to the working memory, and unsuitable for use in LTM, where a static binding mechanism has to be used (for example LISA here applies a hierarchy of structure units). Also, dynamic binding does not allow for some kind of unlimited parallel processing over distributed representations, but introduces hard capacity limitations. In the case of LISA, roughly speaking, the number of distinct dynamic bindings that can be represented at a time is given by the maximal number of different asynchronous firing schemes between simultaneously active groups of units. A third constraint is given by the limitation of dynamic binding to at a time representing only one level of abstraction or hierarchy, as only one level of embedding can be represented simultaneously at the level of semantic primitives (i.e., higher-order propositions cannot be processed directly). LISA handles this last difficulty by means of serial processing for maintaining structure sensitivity during the task of mapping hierarchical structures.

Like already explained, the core of LISA's architecture is a subsystem performing dynamical binding of roles to fillers in WM, and enconding those bindings in LTM. In the WM part, case roles and objects are represented as distributed firing patterns on a set of semantic units, as schematically shown in Fig. 5.1. The subsequent encoding in LTM is conducted by a hierarchy of structure units, see Fig. 5.2 for a schematic depiction. The lowest level of the hierarchy is formed by predicate and object units, having bidirectional excitatory connections to the respective semantic units. Semantically related predicates and objects share units in corresponding roles, giving explicit evidence of the similarities in semantics. The next layer is constituted by subproposition units, binding roles to respective fillers, and thus constituting an intermediate level of proposition complexity. A subproposition again has bidirectional connections to the respective predicate or object units. Proposition units, representing entire propositions, composed of possibly several subpropositions, are at the top-level of the hierarchy, also having bidirectional connections to their constituent subproposition units. Propositional units may fulfill a dual role in hierarchical (higher-level) structures, showing different behaviour depending on whether they are at the top-most parent position, or if they are fulfilling a child function, e.g., serving as argument of another proposition. Structure units of any kind do not directly encode semantic content, but have a structuring and relating function, only being used for storing that content in LTM, and for interacting with the corresponding synchrony patterns on the semantic units.

The architecture is completed by so called mapping connections between structure units of the same type in different analogues. Proposition units in dif-

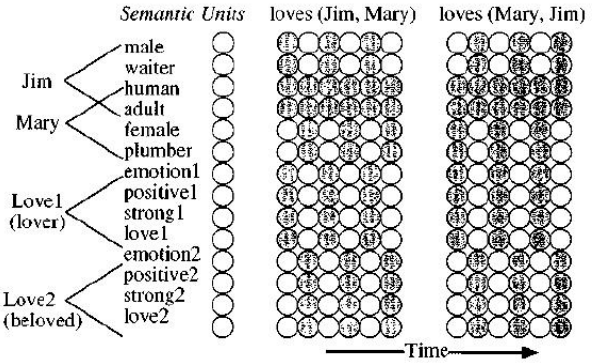

Figure 5.1: A schematic overview of an example for the representation of propositions in LISA's WM, encoding the proposition "*Jim loves Mary.*", as can be seen from the synchronous firing patterns between "*Jim*" and "*Love1 (lover)*", as well as between "*Mary*" and "*Lover2 (beloved)*". ([31], p.166)

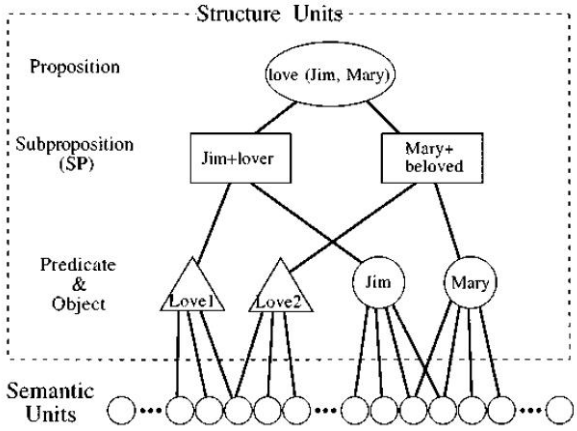

Figure 5.2: A schematic overview of an example for the representation of propositions in LISA's LTM, encoding the proposition "*Jim loves Mary.*". At the bottom, triangle-shaped predicate units and circular object units have bidirectional excitatory connections to semantic units, binding the corresponding semantics together in LTM. Role-filler bindings are encoded in rectangular subproposition units, having excitatory connections with the respective role and filler. At the top-level, the complete proposition is accumulated from role-filler bindings and encoded into the elliptic proposition unit, having excitatory connections with its constituent subpropositions. ([31], p.167)

ferent analogues share a mapping connection with each proposition unit in every other analog, and so do subproposition units with subproposition units. Finally, the same holds for objects and predicates. Mapping and retrieval is conducted over a bipartition of the set of analogues: Two disjoint sets are formed, one containing a driver analog, which controls the retrieval and mapping process, the other containing one or more recipient analogues. A form of guided pattern matching now conducts the mapping process within LISA. When a proposition unit in the driver analog becomes active, due to the cascade of excitatory connections interlinking the different levels of the structural hierarchy, for each role-argument binding a synchronized pattern of activation is created on the semantic units. As the semantic units are used by all analogues, the patterns generated by a proposition in one analog will, due to the bidirectional nature of the excitatory connections, probably activate one or more similar propositions in distinct analogues in LTM, which corresponds to analogical access of episodes, or in WM, constituting an analogical mapping. The difference between mapping and retrieval is to be found in the type of mapping connections. Whilst retrieval uses constant connections, mapping alters the weights on mapping connections, making them grow between the respective units whenever a simultaneous activation pattern is performed, thus allowing LISA to learn correspondences generated during retrieval. By repeated performance of this process, subsequent memory access and mapping gets more and more constrained, by the end of a simulation run allowing to find correspondences between structure units with large positive weights on the relating mapping connections. Noncorresponding units, to the contrary, will expose strongly negative weights.

We want to also report on an expansion to the original LISA framework and functionality, presented in [31], too. Adding unsupervised learning and, subsequently, intersection discovery features to LISA, the system additionally is able to conduct inference and schema induction. The latter is performed in the following way ([33]): As an example, consider the two situations "*Alfred is friends with Bertrand, Bertrand likes to smoke a pipe, and Alfred buys a pipe for Bertrand.*" and "*Martin is friends with Jeroen, Jeroen likes to smoke cigarettes.*". Now, during mapping, corresponding elements in the two situation analogues will be activated simultaneously, e.g., "*isFriends(Alfred, Bertrand)*" will activate "*isFriends(Martin, Jeroen)*", and in accordance with the overall principle, corresponding elements ("*Alfred*", "*Martin*") will fire in synchrony, noncorresponding elements ("*Alfred*", "*Jeroen*") will fire asynchronously. On the level of semantic units, "*Alfred*" will probably share features like "*male*" with "*Martin*", as well as "*Bertrand*" and "*Jeroen*" do, so a reasonable proposition to induce would, e.g., be "*isFriends(male, male)*". For making explicit the commonalities between corresponding elements, LISA uses some sort of intersection discovery. As activation is not only fed to semantic units by the driver but also by the recipient analog, any semantic unit that is common to driver and recipient will receive input from both at a time, and approximately become twice as active as a semantic unit which receives input only from one analog (as the activation of a semantic unit is a linear function of its inputs). This allows for locating common semantic units, which are then encoded into LTM

by means of the aforementioned unsupervised learning algorithm. For doing so, the algorithm makes use of unrecruited structure units, which are not yet used for representing the two situation analogues. These spare units are randomly connected to one another and to the semantic units, basically forming a third schema analog. Predicate and object units within this schema analog structure were assigend high thresholds, only allowing for activation by highly active semantic units, i.e., semantic units which in turn were activated by both, driver and recipient analog. These unrecruited schema units now learn in an unsupervised way to respond to the common elements of the known analogues, whilst at the same time unrecruited subproposition units learn to respond to certain conjunctions of predicate, object and possibly propositon units (in case of hierarchical propositions), and unrecruited proposition units learn to respond to specific subproposition unit combinations. In consequence, propositions describing the common elements of the driver and recipient analog are encoded into LTM as schema analog. But LISA is not only capable of intersection detection, but can also transfer additional knowledge in an analogical form from one situation to another, i.e., conduct analogical inference. Based on the analogical structure of both situations, going back to the known mappings, the slightly augmented version of LISA is able to draw the reasonable inference "*Martin buys cigarettes for Jeroen.*". For doing so, the same unsupervised learning algorithm is applied, implementing a copy with substitution and generation method ([34]). Now, when "*Alfred buys a pipe for Bertrand.*" enters WM in the driver, no match in the second situation can be found, giving LISA a cue to infer a new proposition in the recipient situation, stating that Martin will buy cigarettes for Jeroen. The main difference to the schema induction mechanism presented before thus is that the unrecruited units are not any more placed in a completely separate analog, but are located in the recipient analog itself, allowing for some filling in of additional structure ([33]).

Concludingly, we shall shortly come back to one of the major limitations of LISA and of the taken symbolic connectionist approach in general, reproducing results from [28]. As already stated, due to the finite number of activation patterns which can simultaneously be active, whilst still staying mutually asynchronous, LISA's processing capabilities at one point in time are limited. Thus, LISA requires systematical serial processing (for example driver propositions are activated serially during mapping) combined with limited parallel constraint satisfaction. Surprisingly, for a considerable number of suitably structured problems, the serial mapping mechanism returns results which are closeley similar to those that could be obtained by unlimited parallel constraint satisfaction (as applied in many other systems for computational analogy-making). But still, WM intensive or ill-structured analogies may reveal LISA's limits.

5.2 The STAR family: Structure and Tensors

After having seen the symbolic connectionist LISA, we now want to turn to the arguably first family of distributed connectionist models of analogy-making, the

STAR series by Wilson, Halford et al. ([29], [30]).

In [32], Smolensky introduced a new technique for the representation of value/variable bindings. As already sketched initially, his approach is based on the idea of accumulating activation in a group of units, each computing the product of a feature of a variable and a feature of its value, by this allowing for a completely distributed representation of bindings and symbolic structures. He shows that the tensor product representation forms a more general case of some well-known forms of representing structured data in a connectionist way (as, e.g., applied in NETtalk, [35], Derthick's μKLONE system, [36], or Rumelhart and McClelland's model for learning the formation of the past tense form of English verbs, [37]), exhibiting numerous pleasant properties, as the capability of performing recursive construction of complex representations from simpler ones, and independent generation and maintenance of multiple bindings in parallel. Smolensky's technique has also been applied by Wilson, Halford et al. when constructing STAR-1, replacing the synchronous oscillation model Hummel and Holyoak had decided for in LISA at the moment of representing the propositions that comprise the base and target of the analogy.

Characteristics of the STAR systems

We will now give an account of the common principles underlying the architecture of both, the STAR-1 and the STAR-2 system, mainly following the presentation in [30]. Propositions and relational instances are represented by means of assigning a vector to the relation symbol and each argument, whilst the binding is represented by computing the tensor product of these vectors. Thus, an n-ary relation is represented in a tensor product space of dimension $n+1$, with orthonormal vectors representing concepts in the factor spaces. Information retrieval is possible by means of a generalized dot-product operation (also called inner-prodcut operation): Given a representation of "*relation(X,Y)*", if "*relation(X,?)*" shall be answered, the dot-product of the tensor product representing "*relation(X,-)*" (i.e., leaving the second place empty) together with the tensor product of the original representation is computed. Several propositions can be superimposed on the same representation by adding the corresponding tensor products. This already allows for the solving of proportional analogies by the STAR-1 system, performed by superimposition of a fitting set of propositions on a neural net representation.

A tensor product representation T of a set of propositions is defined as the superimpositon of the propositions. Given an appropriate T, a proportional analogy of the type "*A:B::C:?*" can be solved by first using "A" and "B" as input, thus obtaining all the relation symbols of propositions which have "A" in the first argument position, and "B" in the second one, yielding a so called "*relation-symbol bundle*" ([30], p.137). This bundle consists of superpositions of relation symbols, i.e., a sum of vectors representing the respective relation symbols. Now, this bundle is used for the retrieval of the analog to "C" by computing the tensor product between the bundle and the vector representing "C", followed by a dot-product operation of the result with T. The ouput of

this procedure is another vector bundle, equal to a weighted sum of the vectors representing the possible solutions for the analogy. This second bundle now is used for computing the dot products with the vectors representing arguments to the analog relation ([29]). Whenever the inner product (i.e., the weight of the vector within the bundle) is nonzero, the argument is a possible solution. Furthermore, the larger the value of the inner product for an argument "D" is, the more relations are satisfied by the pair "$C{:}D$" in common with the pair "$A{:}B$", giving a possible measure of salience of the respective solution to the analogy task. Finally, the argument with the greatest value for the last dot product is chosen as answer. The functionality of STAR-1 is not limited to this type of retrieval, also other forms of analogy (amongst others: missing relation-symbol, retrieval of a base) can be solved.

As explained in more detail in [29], STAR-1's means of representation undergo certain limitations. A concept of n-ary dimensionality in representation corresponds to a tensor product of rank $n + 1$. This induces a blow-up of the computational demands of the representation when progressing to higher-order structures: Treating with n vectors of m elements each, the number of elements in the tensor product already equals m^n. This makes necessary the use of special techniques for treating complex structures (normally of complexity order 5 or higher). Two possible solutions for mitigating these difficulties are conceptual chunking and segmentation, the former one referring to a recoding of a concept into fewer dimensions (making some relations temporarily inaccessible), the latter one meaning a decomposition of tasks into smaller steps (similar to a serial processing strategy). At this point, capacities for processing hierarchically structured knowledge representations would be useful, which were therefore introduced in the construction of STAR-2, the direct descendant and successor to STAR-1.

The STAR-2 system

STAR-2 is capable of processing complex structures, given in a hierarchical fashion similar to the one depicted in Fig. 5.3. Arguments are either elements representing basic objects, or chunked propositions indicating more complex concepts. The hierarchical level of a proposition for obvious reasons is directly proportional to the amount of chunked structure which can be found in it (where the height of a higher-order proposition is equal to the height of the highest unchunked argument plus one). The limitation to mappings of order at most 4 has been preserved in STAR-2, whilst nonetheless being able to form mappings between domains which contain multiple propositions. Sequentially, corresponding pairs of propositions from base and target of the analogical process are selected by a constraint satisfaction network (CSN), and another CSN is applied for mapping relation symbols and arguments in each selected pair in parallel, followed by a new choice of base and target pair of propositions. By this, a form of segmentation is implemented, sequentially treating with propositions of acceptable dimensionality in order to process a mapping between higher-order concepts. The STAR-2 model can be broken down into

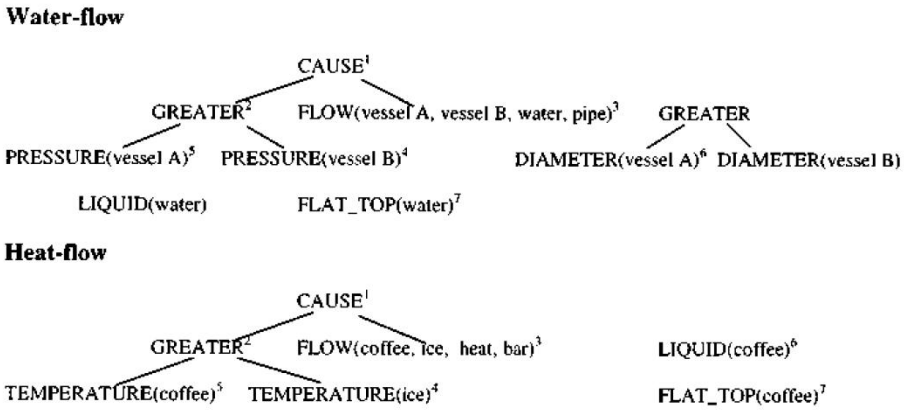

Figure 5.3: A schematic overview of an analogical mapping between relational structures representing water-flow and heat-flow, where the hierarchical structures represent the input in form of domains. The order of focus of the model is represented by the superscript numbers within the relational instances. ([30],p.140)

three main structures: A focus selection network (the first of the two aforementioned CSN), an argument-mapping network (the second CSN), and information storage structures, including a map-storing network (in form of a mapping tensor of rank 2). Given these ingredients, a run of STAR-2 can conceptually be sketched as follows:

1. Establishing information storage structures for the domains being mapped, specifying needed input information.

2. Establishing the focus selection network, applying the network in (heuristically) selecting a single base/target pair of propositions for an initial mapping.

3. Establishing the argument mapping network, forming mappings between relation symbols and arguments of the focus propositions, subsequently storing the mappings in the map-storing network.

4. Learning by modifying connectivity in the focus selection network, incorporating the newly obtained mappings from the previous step, in order to avoid the reselection of propositions. Selecting another pair of propositions for mapping.

5. Reestablishing the argument-mapping network for the current propositions, again followed by a storage process in the map-storing network.

6. Repetition of steps 4 and 5, until termination criterion is met.

The argument-mapping network is a CSN, consisting of up to five columns and rows of mapping nodes. Each of the rows here corresponds to the relation-symbol or an argument position of the base currently focused on (as the dimensionality is limited to a maximum of 4, five nodes are sufficient for representing every possible proposition), while each column corresponds to the respective elements of the target. Thus, each mapping node can be seen as representing a potential mapping between an element of the base and an element of the target. The focus selection network in turn is a two-layer CSN, built for the purpose of selecting the pair of propositions to be passed on to the argument-mapping network. Each layer is structurally similar to the just described design of the argument-mapping network, each row within a layer representing a base proposition, each column referring to a target one. Summarizing, each node represents a pair of chunked propositions from base and target. The first layer of the double-layer CSN is used for selecting propositions with similar features, as for example height in the hierarchy, number of arguments or corresponding relation-symbols and arguments, whilst the second layer selects a single winning pair of chunked propositions from the set provided by the first layer. Finally, the information storage structures are a number of tensor networks, storing different types of entity-related information (similarity between pairs of entities, salience of entites, etc.) and chunked proposition/unchunked proposition associations, supplemented by the already mentioned rank 2 tensor serving as a map-storing network, used for storing mappings between entities formed by the mapping network. As a criterion for triggering a test for termination of the above sketched procedure, three possible configurations are used: Either the selection of an already previously selected focus, or the selection of a focus inconsistent with a previous focus, or convergence of the argument-mapping network to a state not allowing for the interpretation of a set of mappings from the result. In any of these cases, the percentage of chunked propositions in the smaller domain which have formed a focus is computed, with successful termination of the algorithm if a value of 90% or higher is encountered. Alternatively, successful termination may also take place when all chunked propositions in the smaller domain have formed a focus.

Concerning the performance of STAR-2, Wilson, Halford et al. in [30] report several successful applications of the model, ranging from the Rutherford solar system analogy (setting the structure of the solar system in relation to the structure of a hydrogen atom), over a formal analogy concerning the properties of associativity, commutativity, and the existence of an identity element on numeric addition and set union (originally introduced by Holyoak and Thagard in [27]), to several examples known from psychological testings of human analogical reasoning. These tests also show that, as has already been predicted, STAR-2, due to the limitations in dimensionality, reaches its limitations when dealing with more complex analogies (which actually in many cases also constitute problems for human subjects).

Chapter 6

Hybrid Modeling: Copycat and AMBR

Having seen representants of two competing system design paradigms, SME and the MAC/FAC combination serving as examples for symbol-based analogy-making systems, and LISA and the STAR system family standing for connectionist models of computational analogical reasoning, we want to direct our attention to the third class of models, the hybrid systems. These systems unify elements of both competing design paradigms, trying to make additional use of the synergy effects and individual advantages of the distinct approaches when integrating both aforementioned modeling styles side by side into one system architecture. In the following, we will have a look at two prominent members of the group of hybrid analogy-making systems, Hofstadter's Copycat model ([38], [39]) and Kokinov's AMBR family ([40], [41]). In these hybrid models, high-level cognition results from a continual interaction of low-level, localist processing units, where on an intermediate level semantic knowledge is incorporated to achieve a context-sensitive computation of the similarity of elements from the base and target domain ([11]).

The Copycat architecture is built on a concept later named parallel terraced scan ([42]), a technique resembling pruning of search trees without totally abandoning less promissing paths. As explained in [11], Copycat was explicitly designed to perform the integration of top-down semantic information with the bottom-up emergence of connectionist processing. Three of its most important features ([8]) are the ability to build its own representation of base and target, together with the mapping between them, the use of simulated parallelism (allowing the representation building process and the mapping to run in parallel and interact), and its stochastic nature. Copycat is able to solve letter-string analogies ("*EFG is to MNO, as EFH is to ?*"), and its architecture ([11]) is based on a working memory, a simulated long-term memory realized as a semantic network, for defining concepts and concept relationships, and a pro-

cedural memory, serving as a store for nondeterministic computational agents influencing the structures in WM and interacting with LTM.

The origins of the AMBR system family date back to [43], where Kokinov presented the principles underlying the later implementations of his analogy-making system. As summarized in [11], AMBR-1 ([40]) is based on a general cognitive architecture whose LTM consists of micro-agents, each representing rather small bits of knowledge (more complex concepts are represented by coalitions of agents). Each micro-agent is hybrid by itself: Whilst the symbolic part encodes declarative or procedural knowledge, the connectionist part represents the relevance of this knowledge to the current context by means of its activation level. In turn, the symbolic processors operate at a speed proportional to their respective relevance, making the system behaviour context-sensitive. Both, AMBR-1 and its descendant AMBR-2 ([41]) implement interactive parallel processes of recollection, mapping and transfer emerging from the agents' collective behaviour. In AMBR-2, through activation of aspects of event information, general knowledge and knowledge of other episodes, together with the formation of a coherent representation corresponding to the target, recollection is reconstruction of the base episode in WM. This results in an analogy, but can also turn out to be a re-representation of illusory memory.

6.1 The Copycat model: Non-determinism and creativity

As stated in [39] (which will also serve as main source for our presentation of the basic principles underlying Copycat), Copycat aims at being a computer program for discovering insightful analogies, possibly operating in a psychologically realistic way on a modeled subcognitive, but superneural level. Copycat addresses the problem of solving analogies over strings of letters of the form "*EFG:MNO:::EFH:?*", with Hofstaedter et al. claiming that the overall goal is not only about analogical reasoning itself, but in a more general framework about simulating fluid concepts and cognitive fluidity. In Copycat, high-level cognition-like features emerge from the independent activity of numerous parallel processes, so called codelets. These codelets serve as basis for a nondeterministic process of creation of temporary perceptual constructs, possibly yielding a suitable answer to the analogy task at some point.

Copycat's architecture is tripartite, allowing for a subdivison of the system into three architectural main parts: the Slipnet, the Workspace, and the Coderack. The Slipnet constitutes a form of LTM, holding types of concepts (but no concrete instances), similar to the idea of Platonic concepts. The distance of concepts in the Slipnet gives indication of the probability of a slippage between concepts, with a high slip probability corresponding to a closeness of the concepts. The Workspace can be described as close in spirit to blackboard systems as known from the architecture of numerous multi-agent systems in artificial intelligence. There, the equivalent of perceptual activity takes place. The

Workspace contains instances of concepts represented in the Slipnet, combining them into temporary perceptual structures, making the Workspace Copycat's WM. The Coderack finally houses numerous agents, waiting to be invoked for carrying out tasks in the Workspace. Conceptually, the Coderack may be seen as close to an agenda, with the important difference that agents are selected stochastically, i.e., in a non-determinist manner.

The Slipnet

The Slipnet can conceptually be described as a network of interrelated concepts, each concept being represented by a node, and conceptual relationships serving as links between these nodes. These links are assigned numerical weights or lengths, indicating the conceptual distance between two nodes. The Slipnet dynamically responds to situations: Its nodes acquire varying levels of activation (roughly going in accordance with the relevance of the node to the situation at hand), spread activation to neighbours, and over time lose activation similar to a cooling down process. Additionally, the weights/lengths of links is adjusted dynamically, allowing an influence of the perception of the situation at hand on the probability of certain slippages between concepts (the shorter the link, the easier the slippage). Nodes in the Slipnet have an important static feature, referred to as conceptual depth. Each node gets assigned a numerical value, corresponding to the generality and abstractness of the concept the node represents (roughly speaking, the depth of a concept indicates how far the concept is from being directly perceivable in a situation). The architecture includes a drive to prefer deep aspects, once they have been perceived, in constructing an overall understanding of a situation. Also, conceptual depth is directly proportional to the resistance of a concept to slippage, making shallow concepts more likely candidates for slipping. Concerning the flow of activation within the Slipnet, each node spreads activation into its neighbourhood, where the intensity decreases with distance from the source, and the conceptual depth value of a node defines its cooling down rate, with deep concepts cooling down more slowly than shallow ones. Links in the Slipnet are grouped into link types, where all links within the same group get assigned the same label. Labels itself are concepts to be found in Slipnet, with each link constantly adjusting its length in accordance with the activation level of the label concept (high activation corresponding to short links, low activation causing long link length). Via its outgoing links, each node is placed in the center of a probabilistic cloud or halo, representing the likelihood of slippage into other nodes. In interpretation, neighbouring nodes can be seen as being part of a given concept probabilistically, i.e., as function of their proximity to the concept node, thus converting the representation of a concept within Slipnet from a pure, discrete node-based one into a more diffuse, halo-oriented conception. Still, the existence of an explicit core to each concept stays crucial, as slippability vitally relies on the (discrete) change from one core to another. Concludingly, it has to be noted that whereas the Slipnet's shape changes during and within a single run of Copycat, no new permanent concepts are created, and Slipnet is reset to its initial state after completion of a run, i.e.,

no permanent learning process takes place.

The Workspace

The Workspace strongly contrasts the topologically invariant Slipnet. At the start of a run of the Copycat system, the Workspace consists of a collection of unconnected raw data, representing the situation at hand. Initially, each item in the workspace, i.e., each letter token, only carries its alphabetic type, and, for the letters at the very edge of a string, the descriptor leftmost or rightmost is provided. This state changes over time, as agents (the aforementioned codelets) add various descriptions of features. Also, items are linked together by perceptual structures, which in turn are built from concepts in the Slipnet. The amount of attention objects in the Workspace receive from the agents is regulated by the object's salience, i.e., by a function of the set of descriptions the concept currently has. A two-factor model is applied: Objects with descriptions consisting of highly activated concepts in the Slipnet are perceived as important, whilst also weakly integrated objects, i.e., objects with no or only few connections to other objects in the Workspace, are taken care of. Salience is a dynamic quantity, depending on both just described factors, and thus simultaneously on the state of the Workspace and the state of the Slipnet. Processing is characterized by a stochastical selection of pairs of neighbouring objects (i.e., objects within a single letter-string), follwed by a scanning for similarities or relationships, with subsequent reification of the most promising ones as inter-object bonds in the Workspace. Bonds have dynamically varying strengths, reflecting activation and conceptual depth of the concept representing it in the Slipnet, together with the prevalence of similar bonds in the neighbourhood. Sets of objects in the Workspace, which exhibit the same bond type, are candidates for chunking into a higher-level object, called a group. Again, the more salient the component objects, and the higher the strength values of bonds, the higher the probability for reification. When a group has been established, the same procedure starts again, also assigning a description and a salience value to the group, and featuring all the characteristics just described for simple objects. This results in the creation of hierarchical perceptual structures, guided by biases coming from the Slipnet. Another process at work is the probabilistic selection of pairs of objects in different frameworks (i.e., originating from different letter strings), followed by another scan for similarity, leading to a process of bridging (i.e., reification in form of correspondences) between the most promising candidates. A bridge within two objects in the Workspace signalizes that the system considers two objects each other's counterparts, going back to being regarded as intrinsically similar objects, or to playing similar roles in the respective framework. Bridges obviously entail slippages, whose ease, amongst other things, is indicated by a strength value assigned to the bridge. Taking all this together, the final purpose of the bridge building process is the creation of a coherent mapping between two frameworks. During the course of a run of Copycat, the workspace evolves in complexity, and new structures have to be more and more consistent with preexistent structures. Being consistent here can

(roughly speaking) either mean, that two structures are instances of one and the same Slipnet concept, or that their Slipnet types are at least close to each other. Thus, the Workspace is building a more and more coherent vision, called a viewpoint (i.e., a set of mappings). Concerning the interaction between the Slipnet and the Workspace, any discovery taking place in the Workspace (e.g. the formation of a group of a certain type) sends activation to the corresponding concept in the Slipnet. So not only does the Slipnet influence the Workspace, but also the other way round, binding both parts of Copycat together in an operationally almost unseparable bundle.

The Coderack

The third part of the tripartite Copycat architecture is the Coderack, serving as a basis for the numerous codelets. The set of agents can be subdivided into two categories, scout codelets and effector codelets. Scout codelets serve to estimate the promise of potential actions, possibly creating one or more other scout or effector codelets, which follow up on the initial scouts' findings. Effector codelets are used for the creation or removal of a structure in the Workspace. After creation, each agent is placed in the Coderack, together with an urgency value, determining the probability of being chosen from the codelet pool for execution as next agent. This urgency value is given by a function of the estimated importance of the codelet's envisioned action, ranging back to biases from the current state of Slipnet and Workspace (i.e., a measure of how well the agent's action would fit into the Workspace's viewpoint). Also, a distinction can be drawn between bottom-up and top-down codelets: Bottom-up codelets perform an unfocused exploration, whilst top-down codelets search for specific patterns or phenomena. In accordance with these characteristics, bottom-up codelets are continuously active and added to the Coderack, top-down codelets on the other hand have to be triggered from the Slipnet (for example activated nodes may spawn top-down scout agents, subsequently scanning the Workspace for instances of the spawning concept). Each run of Copycat starts with a standard initial population of bottom-up codelets on the Coderack. At each time step, one agent is stochastically chosen for execution, and gets removed from the current Coderack population. In order to keep the population of the Coderack at a high level, new codelets can be generated in three ways: Bottom-up codelets are continually being generated, running agents can call further follow-up codelets into the Coderack, and active nodes in the Slipnet may add top-down codelets as already mentioned. This induces a dynamical adjustment of the Coderack population to the system's needs, influenced by the judgment of previously run agents, as well as by activation patterns in the Slipnet.

After completing the presentation of the main parts of Copycat, which also included several hints and comments on how these parts interact, we want to comment on some of the further particularities and characteristics of the model. We will start with some remarks concerning the already initially mentioned parallel terraced scan, followed by short elaborations on time-evolving biases

and the concept of temperature within the model, before concluding with a short summary of general trends during a run of Copycat:

- The parallel terraced scan: As a consequence of the intertwinedness of different conceptual pressures, Copycat seems to be constantly testing various possible pathways, some with higher speed, others with less emphasis. In the Workspace, only one viewpoint is taken at any time, but conceptually nearby variants of this actual viewpoint are constantly flickering, due to the probabilistic design of the Coderack. If any of these virtual viewpoints is considered sufficitenly promissing by some scout agents, they will trigger the creation of effector codelets, which in turn will follow up on the scouts path, possibly starting a switch of viewpoints. These scouting processes are structured in a terraced way, being carried out in a sequence of stages of rising depth, with each stage relying on the successful termination of its predecessor.

- The time-evolving biases: In Copycat's initial state, the Coderack only contains bottom-up similarity-scanning agents, used for discovering first information about the situation, and subsequently triggering the use of situation-specific codelets. Whilst these early agents are active, the Workspace slowly starts to get structured and populated by small groups, by this activating certain nodes in the Slipnet. These activations in turn initiate a generation process of top-down codelets spawned by the concepts in the Slipnet, gradually leading to a domination of the Coderack by top-down agents. Thus, at the beginning, the Slipnet is neutral, and all observations within the Workspace are local and not very deep. In the course of a run of Copycat, the Slipnet gives up this neutrality, and gets more and more biased towards certain themes (i.e., individual activated deep concepts or groups of those), which then guide overall processing.

- The temperature: During a run of Copycat, the system develops a more and more coherent viewpoint, focusing on certain themes. Thus, top-level decisions should not be made at will anymore. To cover this need, a temperature variable was introduced, monitoring the stage of processing, and mediating the focusing process. Temperature is controlled by the degree of perceived order in the Workspace, resulting in temperature as an inverse measure of the quality of structure. The temperature in turn controls the degree of randomness used in decision making: During high temperature phases, for example, the difference in salience between Workspace objects, guiding the attention of the codelets, will not be as important for the agents' decisions, as in a low temperature setting. As the system starts with high temperature, followed by a process of cooling down (which does not have to be monotonic), Copycat's behaviour changes from open-minded to resembling a conservative decision maker, slowly converting the non-deterministic parallel processing into a deterministic serial mode of operation. Concludingly, the final temperature of a run may serve as

indication of the quality of an answer, with a low temperature indicating an answer as good answer in Copycat's judgement.

- Overall trends: Typical runs of Copycat show a set of characteristic tendencies. In many cases, the initially activated concepts in the Slipnet are conceptually shallow, followed by an increasing depth of concepts during the run. Also, a tendency towards theme building can be found. The Workspace typically changes from unstructured to a state with much structure, as well as from a state showing many local, unrelated objects towards global, coherent structures. Processing moves over time from a parallel towards a serial, and from a non-deterministic towards a deterministic style, together with a change from bottom-up to top-down mode.

Copycat has been highly successful and far recognized as a model of computational analogy-making, due to both the new architectural paradigms, seeming to be in accordance with findings from cognitive science and psychology, and the performance of the system itself. Also, Copycat served as forefather for different more recent models, some of which directly build on the Copycat model and extend it (see, e.g., Metacat, [44], which incorporates some self-observation capabilities into the original Copycat framework), whereas others are different in some aspects (application, micro-domain, etc.).

6.2 The AMBR model: Associative Memory-Based Reasoning

As a second example for an up-and-running hybrid analogical reasoning system, we want to have a closer look at Kokinov's AMBR system family, formed by AMBR ([40]) and its direct descendant and successor AMBR-2 ([41]). Before directly investigating into some of the basic principles of AMBR-1 and AMBR-2, we shortly have to focus on the paradigm underlying the AMBR program family, the general-purpose cognitive architecture DUAL ([45], [46]).

The DUAL architecture

DUAL is an architecture aiming at offering a unified description of mental representation, memory structures, and processing mechanisms. Following the description in [41], DUAL tries to meet the design principles listed in Fig. 6.1 by means of a hybrid, multi-agent design, which supports dynamic emergent computation. DUAL is based on numerous micro-agents, with coalitions of these representing pieces of knowledge, and locally communicating agents carrying out computations. Here, the forming of coalitions amongst the micro-agents is done dynamically, initiated by context-dependent communication amongst the agents. Single agents are kept simple, representing a simple proposition, or an aspect of a concept or an object, with the total information concerning one particular object or concept being distributed over several micro-agents.

Methodological principles	• Integrating analogy-making with memory, perception, learning, reasoning, i.e., reintegrating human cognition • Integrating various subprocesses of analogy-making such as representation building, analogical reminding, mapping, transfer, evaluation, learning, i.e., reintegrating analogy • Grounding the model of analogy-making in general cognitive architecture
Design principles	• Dynamic context-sensitive emergent computation • Dynamic context-sensitive emergent representations • Integrating symbolic and connectionist processing by microlevel hybridization

Figure 6.1: An overview of the principles underlying DUAL and AMBR. ([41],p.88)

Also, the architecture is free of a central processor, but the micro-agents related to a proposition/object/concept do carry out the respective computation. The level of an agent's participation in these processes is graded, ranging from passive behaviour to very active involvement, with the level of participation reflecting the relevance of the agent's knowledge concerning the current task and context. The agent design is hybrid, with the connectionist part continuously calculating and updating an activation value for each micro-agent (implementing the aforementioned relevance estimation). The symbolic side is given by a symbolic processor, capable of performing simple symbolic operations, whilst being regulated in speed and performance by the activation level issued by the connectionist side. In turn, the evolution of the context, altered by symbolic operations, influences the agents' activation levels. Micro-agents communicate via message exchange with a small group of neighbours, allowing for computations to span over entire populations of agents. The message exchange is conducted via temporary or permanent weighted links, serving for spreading both, activation (the weight of the link measures the strength of the coupling) and messages. Overall, micro-agents are (permanently or temporarily) placed in big communities, which correspond to the system's LTM. Agents with an activation level over a certain threshold are considered as additionally forming the working-memory of the system, and thus are responsible for the outcome of current computations. Most links within these communities are stable, having been established over time, but can also be of temporary nature. Thus, with the possibility of creating both, temporary agents and temporary links, the network can adapt dynamically to tasks. Fig. 6.2 provides a summarizing overview of relevant terms and concepts within the DUAL framework. As a concluding incidental remark, it shall be emphasized that Kokinov (see, e.g., [41]) claims his DUAL architecture to be a concrete and fully implemented instantiation of Minsky's "*Society of Mind*" ([47]).

DUAL term	Meaning
Agent (or Microagent)	• The basic computational unit in DUAL
Hybridization	• Each agent has both symbolic and connectionist aspects
Communication	• Via preestablished long-term links or via temporary links created on the spot. Both activation and symbolic structures are exchanged over the links
Coalitions	• Distributed representation of concepts, episodes, and objects
Large communities	• Long-term memory
Motivational power	• Activation level as computed by the connectionist part of the agent; reflects the estimated relevance of the agent to the current context
Graded and variable participation	• Variable individual speed of symbolic processing of each agent determined by its motivational power

Figure 6.2: An overview of the basic terms within DUAL. ([41],p.93)

AMBR-1

As also explained in [48], the design of AMBR is based on some of the principles stated in Fig 6.1, namely integration, unification and context-sensitivity. Integration here means that AMBR has to be seen as integrated model based on a parallel emergent architecture, i.e., each computational process does not only produce an immediate result, but also influences other mechanisms. Unification is given in that AMBR shall be a general model of reasoning with emphasis on analogy-making, but also capable of doing deduction and induction (i.e., generalization), which may be interpreted as extreme, degenerate cases of analogy-making. Finally, with context-sensitivity Kokinov wants to refer to a form of reasoning which does not only take into account the particular task, and the content of the LTM, but is also influenced by external factors. According to the outline in [41], AMBR-1 integrated memory and mapping, as well as transfer and simulated analogy-making in a commonsense domain. Under experimental conditions, it was shown that AMBR-1's ability to remember previous situations (which allows for priming with foreknowledge) is crucial for solving some elaborate analogies. In AMBR-1, remindings are based on the spreading of activation within the connectionist part of the model. As sources of activation serve the perceived elements from the environment and the goals of the system, which trigger a complex emergent process (involving local marker-passing and structure-comparisson processes), constituting a mapping. For implementing this mapping, a constraint satisfaction network is used, which follows some arguably psychologically valid guidelines:

- The system does not construct all possible hypotheses, but only those for

which at least one agent finds a justification, based on semantic similarity or structural consistency (i.e., only hypotheses which seem relevant for the actual context).

- Both, mapping and memory processes, are conducted in parallel, and can possibly interact.
- Hypotheses are constructed dynamically, amongst others resulting in temporally staggered establishing of hypotheses.
- The CSN forms part of the overall network of micro-agents, allowing for an activation-mediated interaction between memory and mapping, fitting the particular context.
- The semantic similarity computation is conducted in a context-dependent, dynamical way, again based on interactions between the micro-agents, and thus subject to influence by the agents' activation levels.
- Finally, the structure correspondence process is not constrained by any kind of arity restriction, but AMBR-1 is able to map any kind of relations as soon as a semantic similarity has been detected (the interactive mechanism between agents disambiguates the correspondence between arguments of different propositions, based on the semantics represented in the network).

Still, AMBR-1 only met some of the aforementioned principles (Fig. 6.1), in that it was based on dynamic context-sensitive computation, but limited to static representation of episodes only. For each episode, there is a key agent in the system, pointing to all agents representing the various aspects, making the coalition centralized around a leading micro-agent. Due to this deficiency, AMBR-1 was refined into a second version of the system, AMBR-2.

AMBR-2

As described in [41], AMBR-2 is a direct descendant of AMBR-1, relying on emergent context-sensitive computation, but moreover being able to also represent episodes emergent and context-sensitive. Concepts and objects are still represented in the same way as in AMBR-1, using coalitions of micro-agents, centered around one leading agent, which normally gets associated the name of the concept or object. Episodes, to the contrary, normally are more complex and unnamed, leading to a decentralized representation in AMBR-2: No micro-agent in the respective coalition knows all the agents forming the group, making the coalitions in the case of episodes more dynamic and susceptible to influences from the context. This augmented decentralization comes at a quite high cost, as mapping and transfer turn out to be difficult to realize without complete lists of propositions on both sides. Difficulties arise, for example, when deciding what to map, after sufficient correspondences have been found, or what still

has to be transferred. Therefore, the version of AMBR-2 reported on in [41], implements mapping and memory, but excludes transfer.

As already stated before, the representation scheme underlying the AMBR systems is framelike, with slot fillers only being pointers or lists of pointers to other micro-agents. Thus, the actual fillers are represented by separate agents, converting even simple propositions into a representation over a small coalition of agents. On the level of interpretation, both the connectionist and the symbolic paradigm can be seen in this: On the one hand, this is a localist, symbolic form of representation, whilst on the other hand, a single representation is also distributed over the entire coalition, and other representations might overlap with it. Still, the representation stays centralized, as there is one leading micro-agent, knowing all other coalition members. Particular objects are also represented by a centralized coalition, with the important difference that the leader, which is standing for the object itself, does not know directly all other members of the coalition, representing the objects properties or relations to other objects and classes of objects. Also concepts are represented in the same decentralized, distributed way with only partial leader knowledge. This allows for pieces of generic knowledge to be moving through the coalition space without being listed explicitly in any coalition. A disparity can be found in the relation between concepts and corresponding instances. Whilst the leader of an object representing coalition normally has a pointer to the concept, the leader of the concept coalition will only rarely be connected to the object representation. Amongst others, this is an attempt at avoiding a fan-out effect when spreading activation from concepts to instances. Instead, concept to object links are only established in case of familiar or recently used objects, leading to an altering set of concept to object connections, finally influencing the reminding process in a way that seeing a certain object will not trigger all situations involving this particular object, but only some. Goals in AMBR-2 are specially tagged propositions, which, whenever being activated, are recognized as goals, and added to the system's goal list. The set of goal propositions changes over time, as new goals can be added by AMBR-2's reasoning mechanism, or older goals may be reactivated. Finally, the most important difference in representations between AMBR-1 and AMBR-2 is the decentralized, distributed representation of episodes conducted in AMBR-2. Episodes are represented in fairly big coalitions without leading micro-agents, i.e., without any agents having a (partial) list of members of the coalition. Instead, there is one designated agent representing the time and place location, to which all other micro-agents within the coalition point. Whilst there are no outgoing pointers from this agent to any members of the coalition, it still is the only evidence that all the micro-agents represent aspects of one certain event.

In AMBR-2, the connectionist mechanism of spreading activation across populations of micro-agents is the basic memory mechanism, influencing all processes mediated by the impact on agents' activation levels. As already sketched, the level of activation of an agent depends on a dynamic estimate of each agent's relevance to the current context. As the system's behaviour thus depends on the participation level of all agents, AMBR-2 can rightfully be called a system

based on context-guided emergent processing. Each agent's activation level and output activation are computed as a function of the agent's input, also taking into account spontaneous exponential decay of activation when lacking external support. Activation is obtained from input and goal nodes, where the input node forms an uplink to all agents corresponding to currently perceived elements of the environment, and the goal node serves as connection to the leading agents of currently active goal coalitions. Due to a quite low decay rate, residual activation stays present for some time, making an earlier memory state influence a later one, and allowing for priming effects.

We now want to focus on the mechanism of collective reasoning within the system. Mapping in AMBR-2 is performed by gradually altering a constraint satisfaction network, which is built incrementally and distributedly by micro-agents, operating on local information only. The purpose of the CSN then is to integrate the local results of the involved agents, and aggregate them into a globally consistent mapping at the coalition of hypotheses level. Structurally, the CSN is made up by a group of temporary hypothesis agents, together with temporary excitatory and inhibitory links within the group. Via numerous connections to the main network of permanent agents, hypotheses receive activation from the permanent micro-agents, as well as give activation feedback to those, making the CSN synchronise harmonically with the overall system. There are four main mechanisms involved in constructing and operating the CSN, as well as in promoting and selecting winning hypotheses:

- The marker-passing mechanism: Semantic similarity is computed by a marker-passing mechanism. Each micro-agent is capable of passing markers along to its neighboring superclass agent with a speed corresponding to the agent's current activation level. Markers are emitted by coalition leader agents (representing an object, a property or a relation), when they enter WM. On its way upwards through the superclass hierarchy, a marker signals the presence of an instance of a particular type. Thus, when markers originating from different instances intersect, at a certain level of abstraction these instances have to belong to the same class, creating the possibility to consider them similar in a certain way. The micro-agent which detects the marker intersection creates a new temporary agent representing the hypothesis that the instances actually might correspond, incorporating the assumed semantic similarity into the CSN in a context-sensitive way (as the speed of marker passing is proportional to agents' activation levels).

- The structure-correspondence mechanism: Structural consistency is ensured by the ability of hypothesis agents to create other hypothesis agents representing hypotheses consistent with the creating agent's actual hypothesis. Top-down hypothesis construction takes place when a correspondence hypothesis between two propositions is established, yielding hypotheses about the correspondence of the propositions parts, as well as excitatory links between them. Bottom-up hypothesis construction is performed when establishing a correspondence hypothesis between two con-

cept instances, resulting in the establishing of correspondences between the concepts. This in turn leads to a facilitated construction of additional hypotheses (or a strengthening of already existing ones) at the instance level of the same type. In its totality, this mechanism guarantees the emergence of global structural consistency in the winning hypotheses from the CSN.

- The CSN-consolidation mechanism: The local establishing of hypotheses by micro-agents can possibly lead to different justifications for establishing the same correspondence, which in turn would yield two distinct, competing hypothesis agents representing the same correspondence. AMBR-2 avoids this by a duplicate hypotheses merging mechanism, producing one hypothesis with possibly several justifications. Permanent agents keep track of the hypothesis micro-agents related to them, and newly established hypothesis agents are kept in a tentative state, issuing registration requests to all corresponding agents. Only after locally checking with corresponding secretaries whether the tentative agent may mature into a full-fledged hypothesis agent in its own right (i.e., whether a new hypothesis is represented, or only an already existing one would be duplicated), the hypothesis agent is established. Otherwise, its justification is added to the already existing hypothesis micro-agent, and the tentative one is discarded. Over time, this leads to a gradual construction of the CSN, based on local addition of agents and links.

- The winning hypotheses mechanism: There is continuous building and relaxation of the constraint satisfaction network ongoing in AMBR-2. For each object, relation or concept, a secretary holds a winning hypothesis at each time, allowing for parallel mapping, transfer and evaluation. Also, backward influences from the transfer and evaluation of hypotheses processes on the mapping are made possible, for example causing the abandonment of a winning hypothesis invalid in the target domain. As secretaries constantly have registered all hypotheses involving the current agent, it is possible to decide which of the hypothesis is the actually most promissing one. A candidate hypothesis is promoted into a winning hypothesis only after having maintained its status as actual candidate for a certain period of time, and only if the difference in activation to the next best competitor is sufficiently large. Once a winning hypothesis has been established, evaluation and transfer mechanisms may use it as starting point for further processing.

To conclude this closer look at AMBR-2, we shall have a quick glance at a particular phenomenon occuring during some runs of AMBR-2 (again see [41]): Blending of episodes. A typical pattern when running AMBR-2 is that at an early stage different agents belonging to distinct episodes are brought to the working memory from LTM, based on their semantic similarity to some target element. Only then, the dynamics of the gradually constructed CSN slowly drives the system into a state of consistent mapping between the target and

one specific source episode by influencing the activation pattern over the entire agent population. Still, in some rare cases blends of episodes occur, i.e., several sources are partially mapped to the target. Simulation experiments indicate that these blends may happen when no episode from LTM matches the target sufficiently well, or when the matching episode is for some reason (as, e.g., priming or context effects) superseded by another one. As mapping in AMBR-2 is done element by element, and the CSN only exerts mild pressure to keep the dominant source episode the same, under the aforementioned circumstances it can happen that one source maps to some fraction of the target, and another source maps to the rest.

Chapter 7

E Pluribus Unum: An Intermediate Synopsis

The main part of the present paper (Sect. 4 to Sect. 6) consists of more or less detailed presentations of several well-known computational analogy-making systems, also representing the three architectural paradigms within the field. In this section, we want to summarize some of the insights from the previous sections, and try to give a helpful synopsis and summary.

To start with, Table 7.1 gives a very high-level summarized overview of the main systems presented in this paper, making accessible the main features and functionalities.

Table 7.1: Overview of Computational Analogy-Making Systems

Model	**Paradigm**	**Characteristics/Main Features**
SME	symbolic	• Implementing the mapping phase of Gentner's Structure Mapping Theory. • Propositional representations as input (typed higher-order predicate calculus). • Searching for mappings (set of correspondences aligning entities in base and target) and candidate inferences (statements about the base carrying over to the target).

Table 7.1: Overview of Computational Analogy-Making Systems

Model	**Paradigm**	**Characteristics/Main Features**
MAC/FAC (+ SME)	symbolic	• Analogical retrieval engine providing input for SME. • Episodes as encapsulated representations of past events, dual encoding in LTM: Complete predicate calculus and compressed vector representation. • Two-stage retrieval process: High-level search for episodes sharing predicates with target (vector based), in-depth search for best match episode (predicate calculus based).
LISA	connectionist	• Symbolic connectionist model. • Integrating retrieval and mapping. • Interaction between LTM and WM during retrieval and mapping. • Distributed representation in WM, localist representation in LTM. • Associating structures based on synchrony of activation of corresponding groups of nodes in a connectionist network.

Table 7.1: Overview of Computational Analogy-Making Systems

Model	**Paradigm**	**Characteristics/Main Features**
STAR-2	connectionist	• Based on tensor product connectionist modeling. • LTM as aggregating tensor over tensor products representing statements. • Mapping complex analogies by sequentially concentrating on parts of the domain. • Best match selection via parallel computation in a constraint satisfaction network.
Copycat	hybrid	• Based on parallel terraced scan. • Building own representation of base and target, together with mapping between both. • Simulated parallelism between representation building and mapping. • Partially stochastic processes. • Slipnet, Workspace, Coderack.

Table 7.1: Overview of Computational Analogy-Making Systems

Model	**Paradigm**	**Characteristics/Main Features**
AMBR-2	hybrid	• Based on general-purpose cognitive architecture DUAL. • Using hybrid micro-agents. • Implementing interactive parallel processes of recollection, mapping and transfer. • Recollection as reconstruction of base in WM (possible occurence of illusory memory).

It should have become clear from the last sections that efforts in the field of computational analogical reasoning have given rise to several quite distinct paradigms of modeling and system architecture, leading to a wide variety of implementations and programs, which makes the creation of clear-cut categories, possibly allowing for the establishment of a systematisation (and the comparison of respective prototypical systems) on a not too abstract level (more specific than the threefold subdivision according to architectural paradigms we performed, but less detailed than a classification according to lines of development and system families, as the STAR or the AMBR clans) complicated or almost impossible to achieve. Also, such an undertaking would clearly go beyond the scope of the present report. Therefore, we will have to make do with stating a few additional observations concerning the systems we presented in the previous sections, pointing at some conceptual links, and providing some comparative information.

As already pointed out by Kokinov in [41], one commonality between some of the presented computational models of analogy-making, namely MAC/FAC, LISA and at least AMBR-1, is that they use a storehouse-like conception of memory, with their LTM storing a collection of static representations of past episodes. MAC/FAC clearly has a centralized collection of past memories, from which the retrieval process tries to select the best one, LISA's LTM (contrasting the distributed WM) is based on centralized localist representations of episodes, and AMBR-1 also relies on stable and complete representations.

[41] also provides some insights concerning the differences in conception between AMBR/DUAL and the Copycat architecture: Whilst the ideas of codelets and micro-agents are conceptually close to each other, Copycat maintains with its Workspace a separate storage area, creating a platform for copies of the codelets to run and construct representations. No similar construct is to be found in DUAL. Also, DUAL is determinist in nature, contrasting the (par-

tially) inherently stochastic nature of Copycat (which the latter actually shares with LISA, as also LISA may yield distinct outputs for multiple runs over the same input data, [28]). Where a certain dependence on probabilistic mechanisms is an internal part of the Copycat architecture, the variability of DUAL's behaviour is due to the permanent influences from the context.

Hofstadter and Mitchell, in [39], provide an overview comparison of Copycat with SME. Following their presentation, Copycat and SME agree in several crucial points, most importantly on a basic principle originating from Gentner's SMT: Systematicity, i.e., that the essence of a situation, and therefore the crucial part for mapping, is not only a collection of independent low-level facts, but rather a high-level coherent whole. But, as was to be expected, of course there are also fundamental points of disagreement, some of the most salient ones being SME's emphasis on syntactic connections between potential mapping partners, as opposed to Copycat's integration of semantic information into the process, the total absence of notions like conceptual similarity or slippage from Gentner's SMT (and thus also from SME), which are characteristic features of Copycat, and the need for a precise and unambiguous (predicate logic) representation of situations in SME (due to its emphasis on syntax), which is by far less present in Copycat (wich exhibits some real-time representational flexibility in processing).

LISA and STAR both address in their way the role of WM capacity in the process of analogical mapping, whilst SME forms numerous possible local matches between base and target elements (most of which are again discarded during the matching process) and is moreover theoretically capable of mapping analogies of arbitrary size ([28]).

In general, it should not be surprising to find quite different ambitions underlying the presented models, in our eyes ranging from SME and MAC/FAC on the one side, which – though being inspired by concepts from cognitive science and psychology – can mainly be interpreted as implementations of the Structure Mapping Theory, emphasizing the conceptual and algorithmic aspects and serving as a test bed, to Copycat at the other extreme, trying to step by step constitute a full-fledged model of mental fluidity, perception and analogical reasoning in a bigger framework. Both LISA and the STAR system family already show signs in the same direction, but stay way more moderate in their pretensions and just want to incorporate some features of human-like reasoning, as for example the aforementioned working memory limitations. Concludingly, AMBR in turn (as already indicated by its close connection to the general cognitive architecture DUAL, though still staying considerably more tempered in its claims) from our understanding has to be positioned somewhere between LISA/the STARs and Copycat.

Chapter 8

Work in Progress: Heuristic-Driven Theory Projection

In this penultimate section of the present study, we want to turn to a slightly different framework for analogical reasoning, which at the current moment is still in a mostly conceptual stadium, being subject to ongoing research, and for which no official, full-fledged implementation is yet available (although it has partially been implemented in a PROLOG program): Heuristic-Driven Theory Projection (HDTP).This framework has been conceived as a mathematically sound framework for analogy-making (see, e.g., [9]). In the following we will give a crisp outline of the basic principles and functionality of HDTP, before concludingly comparing the framework to some of the analogical reasoning systems from earlier sections.

As explained in [49], HDTP has been created for computing analogical relations and inferences for domains which are given in form of a many-sorted first-order logic representation. Base and target of the analogy-making process are defined in terms of axiomatisations, i.e., given by a finite set of formulae. From there, HDTP tries to align pairs of formulae from the two domains by means of anti-unification. Anti-unification is the dual to the more prominent unification problem, which has thoroughly been studied for example in logic programming and automatic theorem proving. The less popular undertaking of anti-unification, to the best of our knowledge firstly studied by Plotkin in [50], basically tries to solve the problem of generalizing terms in a meaningful way, yielding for each term an anti-instance, in which distinct subterms have been replaced by variables (which in turn would allow for a retrieval of the original terms by a substitution of the variables by appropriate subterms, see [9] for a more detailed description). Fig. 8.1 gives some examples for first-order anti-unification in a Plotkin-like style.

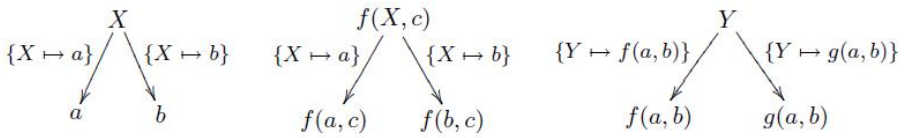

Figure 8.1: Some examples of first-order anti-unifications: Minuscules represent instances, capitals represent variables, the terms in braces indicate substitutions. ([9], p.253)

The goal of anti-unification is to find a most specific anti-unifier, i.e., the least general generalization of the involved terms. Plotkin has shown that for a proper definition of generalization, for a given pair of terms there always is a generalization, and that there is exactly one least general generalization (up to renaming of variables). HDTP now extends Plotkin's classical first-order anti-unification to a restricted form of higher-order anti-unification, as mere first-order structures have shown to be too weak for the purpose of analogy-making: Just think of structural commonalities which are embedded in different contexts, and therefore not accessible by first-order anti-unification only.

Restricted higher-order anti-unification as used in HDTP was introduced in [51]. In order to restrain generalizations from becoming arbitrarily complex, a new notion of substitution is introduced. First of all, classical first-order terms are extended by the introduction of variables which may take arguments (where classical first-order variables correspond to variables with arity 0), making a term either a first-order or a higher-order term. Now, anti-unification can be applied analogously to the original first-order case, yielding a generalization subsuming the specific terms. As already indicated by the naming, the class of substitutions which are applicable in HDTP is restricted to (compositions of) the following four cases: renamings, i.e., substitutions replacing one variable by another, fixations, i.e., substitutions replacing a variable by a function of the same argument structure, argument insertions, i.e. a substitution of the form $F(t_1, \ldots, t_n) \rightarrow F^*(t_1, \ldots, t_i, G(t_{i+1}, \ldots, t_{i+k}), t_{i+k+1}, \ldots, t_n)$ (where F is a variable of arity n, G a variable of arity k, and F^* a variable of arity $n-k+1$, and $0 \leq i \leq n$, as well as $k \leq n-i$), and permutations, i.e., rearrangements of a term's arguments. In [51], it is shown that this new form of (higher-order) substitution is a real extension of the first-order case, which has proven to be capable of detecting structural commonalities not accessible to first-order anti-unification. Unfortunately, the least general generalization loses its uniqueness (which in turn may be interpreted as corresponding to the multiple possibilities humans may find in drawing analogies between a base and a target domain).

Therefore, as explained in [9], HDTP introduces a way of ranking generalizations according to a complexity order on the complexity of generalization, which in turn is based on a complexity measure for substitutions. HDTP chooses the least complex generalization as preferred generalization.

From a practical point of view, it is also necessary to anti-unify not only terms, but formulae. Therefore, HDTP extends the notion of generalization also to formulae by basically treating formulae in clause form and terms alike (as positive literals are structurally equal to function expressions, and complex clauses in normal form may be treated component wise). Furthermore, analogies do in general not only rely on an isolated pair of formulae from base and target, but on two sets of formulae. Here, as also pointed out in [49], a heuristics is applied when iteratively selecting pairs of formulae to be generalized: Coherent mappings outmatch incoherent ones, i.e., mappings in which substitutions can be reused are preferred over isolated substitutions, as they are assumed to be better suited to induce the analogical relation. Once obtained, the generalized theory and the substitutions specify the analogical relation, and formulae of the base for which no correspondence in the target domain can be found may by means of the already established substitutions be transferred to the target, constituting a process of analogical transfer between the domains.

Based on more detailed elaborations in [9], we will now shortly compare HDTP to some of the other models of analogy-making discussed in the present paper. Whilst SME and HDTP may have equally expressive representation formalisms, the mapping HDTP performs may be interpreted as more powerful than SME's counterpart, as – contrary to SME – HDTP also accounts for some semantic information (as, e.g., from quantified variables in general laws). Also, in the case of HDTP, the mapping is less rigid, as HDTP aligns basically any entity, function or predicate (although it clearly prefers literally-matching alignments over non-literally ones, and equivalent structures to structural mismatches), whilst SME requires identical labels for the alignment of attributes and relations. A third main difference can be found in the way of establishing a mapping between domains: Here, SME constructs a mapping without generalizing across domains (a generalization is built only on demand), contrasting the explicit generalizations performed by HDTP. Concerning the hybrid systems Copycat and DUAL, the fundamentally different architectural paradigms and modeling approaches cause extensive differences. These become noticeable for example when considering DUAL's way of representing domains and establishing analogical relations, as compared to the respective mechanisms in HDTP. Also it seems unclear how generalization can be handled in DUAL. With respect to Copycat, one of the main architectural differences is the difference in the relation to the system's domain: Where Copycat's string domain is hard-wired into it, HDTP is an open domain system, only defining the representation format of its domains. Finally, whilst also HDTP claims a certain degree of cognitive plausibility, its ambitions in these matters seem considerably more moderate than it is the case especially with Copycat (where the designers in parts even claim a small degree of biological plausibility due to the emergence-based nature of some mechanisms).

Concerning application scenarios, HDTP has successfully been used in some proof-of-concept applications in the domain of modeling mathematical reasoning and conceptual blending in mathematics. A fairly crisp overview of these efforts can be found in [52], a more detailed report is also provided by [49].

Chapter 9

Conclusion

In this survey, we have looked at several different systems and models for computational analogical reasoning, covering a time span from the late 1980s to the present day. The selection we made of course stays far from being complete, but should rather be seen as a presentation of some representative systems, showing some of the most prominent features and functionalities analogy-making systems can have, whilst still leaving much unsaid. For more high-level overviews, which on the other hand may be more exhaustive in the number of presented models, we may point the reader to [8] and [11]. A perspective from the end of the 1980s, which we still consider very worth reading, can be found in [10]. Also, the in this study often cited book by Gentner, Holyoak and Kokinov ([53]) gives a fairly recent overview of research on the modeling of analogy-making, not only limiting its scope on the computational side, but amongst others also treating more with issues from cognitive sciences, developmental and neuropsychology, and cognitive linguistics. This book may be seen as a descendant of the proceedings of the 1998 workshop on "*Advances in Analogy Research*" ([54]), which also offer some interesting articles for the interested reader.

We finally want to emphasize our confidence in that understanding analogy-making will show to be crucial for getting a grasp on understanding and modeling human thinking, reasoning and cognition in general. Also we are confident that analogical reasoning will find numerous applications in artifical intelligence, but also in other disciplines as, e.g., economics and decision theory. Already by now, publications using analogical reasoning techniques and concepts from analogy research range from areas as diverse as the Semantic Web ([55]) to game theory ([56]), and still numerous issues, questions and possible applications remain to be explored in future research.

9.1 Acknowledgements

We want to thank Prof. Dr. M. van Lambalgen, chairholder of the Chair of Logic & Cognitive Science, at the Institute for Logic, Language and Computation

(ILLC) and the Department of Philosophy of the Universiteit van Amsterdam, for (amongst many other things) teaching an interesting course on "Rationality, Cognition and Reasoning", which in parts inspired this survey, as well as for his willingness to serve as a supervisor during the creation of the present overview study.

Bibliography

[1] James F. Ross. Analogy and the resolution of some cognitivity problems. *The Journal of Philosophy*, 67(20):725–746, Oct. 1970.

[2] A. Juthe. Argument by analogy. *Argumentation*, 19(1):1–27, 2005.

[3] Biometra biomedizinische Analytik GmbH, Göttingen, Germany. *Whatman Biometra: Refrigerated Circulator KH-4*, June 2006.

[4] A. Schwering, K.-U. Kühnberger, and B. Kokinov. Analogies: Integrating multiple cognitive abilities - guest editorial. *Journal of Cognitive Systems Research*, 10(3), 2009.

[5] D. Hofstadter. *The Analogical Mind: Perspectives from Cognitive Science*, chapter Epilogue: Analogy as the Core of Cognition, pages 499–538. MIT Press, Cambridge, MA, 2001.

[6] Keith J. Holyoak, Dedre Gentner, and Boicho N. Kokinov. *The Analogical Mind: Perspectives from Cognitive Science*, chapter Introduction: The Place of Analogy in Cognition, pages 1–19. MIT Press, Cambridge, MA, 2001.

[7] Thomas G. Evans. A heuristic program to solve geometric-analogy problems. In *Proceedings of the April 21-23, 1964, spring joint computer conference*, AFIPS '64 (Spring), pages 327–338, New York, NY, USA, 1964. ACM.

[8] Robert M. French. The computational modeling of analogy-making. *Trends in Cognitive Sciences*, 6(5):200 – 205, 2002.

[9] A. Schwering, U. Krumnack, K.-U. Kühnberger, and H. Gust. Syntactic principles of heuristic-driven theory projection. *Journal of Cognitive Systems Research*, 10(3):251–269, 2009.

[10] Rogers P. Hall. Computational approaches to analogical reasoning : A comparative analysis. *Artificial Intelligence*, 39(1):39 – 120, 1989.

[11] B. Kokinov and R. M. French. *Encyclopedia of Cognitive Science*, volume 1, chapter Computational Models of Analogy Making, pages 113–118. Nature Publishing Group, London, 2003.

[12] David J. Chalmers, Robert M. French, and Dougles R. Hofstadter. High-level perception, representation, and analogy: A critique of artificial intelligence methodology. *Journal of Experimental and Theoretical Artificial Intelligence*, 4:185–211, 1992.

[13] George Spanoudakis and Panos Constantopoulos. Elaborating analogies from conceptual models. *International Journal of Intelligent Systems*, 11:917–974, 1996.

[14] John Haugeland. *Artificial Intelligence: The Very Idea*. MIT Press, Cambridge, MA, 1985.

[15] Warren McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 52:99–115, 1990. 10.1007/BF02459570.

[16] Walter R. Reitman, Richard B. Grove, and Richard G. Shoup. Argus: An information-processing model of thinking. *Behavioral Science*, 9(3):270–281, 1964.

[17] Thomas G. Evans. *A Heuristic Program to Solve Geometric-Analogy Problems*. PhD thesis, MIT, 1963. An abridged version can be found in Marvin Minsky (ed.): "Semantic Information Processing", MIT Press, 1968, pp. 271–353.

[18] Brian Falkenhainer, Kenneth D. Forbus, and Dedre Gentner. The structure-mapping engine: Algorithm and examples. *Artificial Intelligence*, 41(1):1 – 63, 1989.

[19] Dedre Gentner and Kenneth D. Forbus. MAC/FAC: A Model of Similarity-based Retrieval. *Cognitive Science*, 19:141–205, 1991.

[20] Dedre Gentner. Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2):155–170, 1983.

[21] Kenneth D. Forbus. *The Analogical Mind: Perspectives from Cognitive Science*, chapter Exploring Analogy in the Large, pages 20–58. MIT Press, Cambridge, MA, 2001.

[22] Brian Falkenhainer. Towards a general-purpose belief maintenance system. In *Proceedings of the Uncertainty in Artificial Intelligence 2 Annual Conference on Uncertainty in Artificial Intelligence (UAI-86)*, Amsterdam, NL, 1986. Elsevier Science.

[23] B. Falkenhainer. Analogical interpretation in context. In *Proceedings of the tweleth annual Conference of the Cognitive Science Society*, pages 69–76, Hillsdale, NJ, 1990. Lawrence Erlbaum Associates.

[24] Kenneth D. Forbus, Dedre Gentner, and Keith Law. MAC/FAC: A model of similarity-based retrieval. *Cognitive Science*, 19(2):141 – 205, 1995.

[25] D. Gentner. Finding the needle: Accessing and reasoning from prior cases. In K. Hammond, editor, *Proceedings of the DARPA Workshop on Case-Based Reasoning*, volume 2, pages 137–143, San Mateo, CA, 1989. Morgan Kauffmann.

[26] K. Law, K. D. Forbus, and D. Gentner. Simulating Similarity-Based Retrieval: A Comparison of ARCS and MAC/FAC. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, pages 543–548, Hillsdale, NJ, 1994. Lawrence Erlbaum Associates.

[27] Keith J. Holyoak and Paul Thagard. Analogical mapping by constraint satisfaction. *Cognitive Science*, 13:295–355, 1989.

[28] John E. Hummel and K. Holyoak. Distributed representations of structure: A theory of analogical access and mapping. *Psychological Review*, 104(3):427–466, July 1997.

[29] Graeme S. Halford, William H. Wilson, Jian Guo, Ross W. Gayler, Janet Wiles, and J.E.M. Stewart. *Advances in Connectionist and Neural Computation Theory*, volume 2: Analogical Connections, chapter 7: Connectionist Implications for Processing Capacity Limitations in Analogies, pages 363–415. Ablex, Norwood, NJ, 1994.

[30] W. Wilson, G. Halford, B. Gray, and S. Phillips. *The Analogical Mind: Perspectives from Cognitive Science*, chapter The STAR-2 model for mapping hierarchically structured analogs, pages 125–159. MIT Press, Cambridge, MA, 2001.

[31] Keith J. Holyoak and John E. Hummel. *The Analogical Mind: Perspectives from Cognitive Science*, chapter Toward an Understanding of Analogy within a Biological Symbol System, pages 160–195. MIT Press, Cambridge, MA, 2001.

[32] P. Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46:159–216, November 1990.

[33] J. E. Hummel and K. J. Holyoak. LISA: A computational model of analogical inference and schema induction. In G. W. Cottrell, editor, *Proceedings of the Eighteenth Annual Conference of the Cognitive Science Society*, pages 352–357, Hillsdale, NJ, 1996. Erlbaum.

[34] K. J. Holyoak, L. R. Novick, and E. Melz. *Advances in connectionist and neural computation theory*, volume 2: Analogical connections, chapter Component processes in analogical transfer: Mapping, pattern completion, and adaption, pages 113–180. Ablex, 1994.

[35] T. J. Sejnowski and C. S. Rosenberg. Parallel networks that learn to pronounce English text. *Complex Systems*, 1:145–168, 1987.

[36] M. Derthick. A connectionist architecture for representing and reasoning about structured knowledge. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, pages 131–142, Seattle, WA, 1987. Routledge.

[37] D. E. Rumelhart and J. L. McClelland. *Parallel Distributed Processing: Explorations in the Microstrucutre of Cognition*, volume 2: Psychological and Biological Models, chapter On Learning the Past Tense of English Verbs, pages 216–271. MIT Press/Bradford Books, Cambridge, MA, 1986.

[38] Douglas R. Hofstadter. The Copycat Project: An Experiment in Nondeterminism and Creative Analogies. *Massachusetts Institute of Technology*, AIM-755, 1984.

[39] Douglas Hofstadter and Melanie Mitchell. *Advances in Connectionist and Neural Computation Theory*, volume 2: Analogical Connections, chapter The Copycat project: a model of mental fluidity and analogy-making, pages 31–112. Ablex, New York, NY, USA, 1994.

[40] B. Kokinov. *Advances in Connectionist and Neural Computation Theory*, volume 2: Analogical Connections, chapter A hybrid model of reasoning by analogy, pages 247–318. Ablex, Norwood, NJ, 1994.

[41] Boicho N. Kokinov and Alexander A. Petrov. *The Analogical Mind: Perspectives from Cognitive Science*, chapter Integrating Memory and Reasoning in Analogy Making: The AMBR Model, pages 59–124. MIT Press, Cambridge, MA, 2001.

[42] D. J. Hofstadter and J. Rehling. The parallel terraced scan: An optimization for an agent-oriented architecture. In *Proceedings of the 1997 IEEE International Conference on Intelligent Processing Systems (ICIPS'97)*, volume 1, pages 900–904, 1997.

[43] B. Kokinov. Associative memory-based reasoning: How to represent and retrieve cases. In T. O'Shea and V. Sgurev, editors, *Artificial Intelligence III: Methodology, systems, applications*, pages 51–58. Elsevier Science Publishers B.V. (North Holland), 1988.

[44] James B. Marshall and Douglas R. Hofstadter. Beyond copycat: Incorporating self-watching into a computer model of high-level perception and analogy-making. In *Online Proceedings of the 1996 Midwest Artificial Intelligence and Cognitive Science Conference*, 1996.

[45] Boicho N. Kokinov. The DUAL Cognitive Architecture: A Hybrid Multi-Agent Approach. In A. Cohn, editor, *Proceedings of the Eleventh European Conference on Artificial Intelligence*, pages 203–207, London, 1994. John Wiley & Sons, Ltd.

[46] Boicho Nikolov Kokinov. The Context-Sensitive Cognitive Architecture DUAL. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, Hillsdale, NJ, 1994. Erlbaum.

[47] Marvin Minsky. *The society of mind*. Simon & Schuster, Inc., New York, NY, USA, 1986.

[48] Boicho Kokinov. The Cognitive Model AMBR. Webpage, March 2011. Available online (last checked 09.03.2011), dating on 05.02.2003: http://nbu.bg/cogs/personal/kokinov/ambr_i.html.

[49] Markus Guhe, Alison Pease, Alan Smaill, Maricarmen Martinez, Martin Schmidt, Helmar Gust, Kai-Uwe Khnberger, and Ulf Krumnack. A computational account of conceptual blending in basic mathematics. *Cognitive Systems Research*, In Press, 2011.

[50] Gordon D. Plotkin. A note on inductive generalization. *Machine Intelligence*, 5:153–163, 1970.

[51] U. Krumnack, A. Schwering, H. Gust, and K.-U. Khnberger. Restricted higher-order anti-unification for analogy making. In *20th Australian joint conference on artificial intelligence (AI'07)*, volume 4830 of *Lecture Notes of Artificial Intelligence*, Gold Coast, Australia, 2007. Springer.

[52] Markus Guhe, Alison Pease, Alan Smaill, Martin Schmidt, Helmar Gust, Kai-Uwe Khnberger, and Ulf Krumnack. Mathematical reasoning with higher-order anti-unification. In S. Ohlsson and R. Catrambone, editors, *Proceedings of the 32nd Annual Conference of the Cognitive Science Society*, pages 1992–1997, Austin, TX, 2010. Cognitive Science Society.

[53] Dedre Gentner, Keith J. Holyoak, and Boicho N. Kokinov. *The Analogical Mind: Perspectives from Cognitive Science*. The MIT Press, March 2001.

[54] K. Holyoak, D. Gentner, and B. Kokinov, editors. *Advances in analogy research: Integration of theory and data from the cognitive, computational, and neural sciences*, Sofia, Bulgaria, 1998. NBU Press.

[55] Akshay Bhat. Analogy engines for the semantic web. In A. Bernstein, D. R. Karger, T. Heath, L. Feigenbaum, D. Maynard, E. Motta, and K. Thirunarayan, editors, *8th International Semantic Web Conference (ISWC2009)*, volume 5823 of *Lecture Notes in Computer Sciences*, Chantilly, VA, October 2009. Springer.

[56] Philippe Jehiel. Analogy-based expectation equilibrium. *Journal of Economic Theory*, 123(2):81 – 104, 2005.