In: Artificial Intelligence: New Research
Editors: R.B. Bernstein et al, pp. 299-321

ISBN 978-1-60456-282-8
© 2008 Nova Science Publishers, Inc.

Chapter 6

ANALOGY-MAKING IN SITUATION THEORY

Emre Şahin and Varol Akman[*]
Bilkent University
Department of Computer Engineering
06800 Bilkent, Ankara, Turkey

Abstract

Analogy-making is finding analogies between different situations. In this paper, we provide a new model of computational analogy-making which uses Situation Theory as its formal background. Situation Theory is a semantic and logical theory which provides a naturalistic way to represent relations in situations. The system described in this paper is aimed at solving analogy problems made by basic geometric figures in a chessboard-like environment.

1. Introduction

Despite its importance in understanding creativity and in general intelligence, Computer Science research on analogy is narrow and does not fully reflect the importance of this faculty of human mind.

As an example of the natural analogy making ability, we offer this excerpt [12] (p. 9):

> Consider the following discussion between a mother and her four-year-old son, Neil, who was considering the deep issue of what a bird might use for a chair. Neil suggested, reasonably enough it would seem, that a tree could be a bird's chair. A bird might sit on a tree branch. His mother said that was so and added that a bird could sit on its nest as well, which is also its house. The conversation went on to other topics. But several minutes later, the child had second thoughts about a tree is to a bird: "The tree is not the bird's chair – it's the bird's backyard!"

The following definition, which we regard as the most encompassing in the literature, is from Gentner [10] (p. 107):

[*]E-mail address: {iesahin, akman}@bilkent.edu.tr

> Analogies are partial similarities between different situations[1] that support further inferences. Specifically, analogy is a kind of similarity in which the same system of relations holds across different objects. Analogies thus capture parallels across different situations.

In this paper we introduce SITAR, a new model of analogy-making based on Situation Theory. We will concentrate on the modeling of analogy from a computational perspective. In the next section (section 2), we describe three models that represent the basic ideas available in the literature. Then, the microworld of SITAR is described (in section 3). The representation of analogy via situation-theoretic means and the architecture of SITAR are found in section 4.

2. Representatives from the Literature

Analogical Reasoning is one of the earliest projects in Artificial Intelligence. One of the pioneering works is the ANALOGY program of Evans. Structure Mapping Theory (SMT) and its implementation, the Structure Mapping Engine (SME) of Gentner and colleagues, and the Copycat of Mitchell and Hofstadter are two other influential works.

2.1. ANALOGY

This is a program to model analogy questions appearing in IQ tests. Despite the modest computational standards of 1960s, it represented an important attempt.

ANALOGY receives a set of 3+5 figures as input. The first three of these are labelled as A, B and C. There is a relation between A and B. The program selects one of the five candidate figures as having the same relation with C. The analogy is represented like $A : B :: C : X$ where X is the label of one of the five figures supplied to the program. An example can be seen in Figure 1.

In the example, the change between A and B is detected to be (i) disappearance of the dot in A and (ii) relocation of the rectangle outside of the triangle. Comparing all candidate figures with C, the program finds a similar change between C and 2, to wit (i) disappearance of the dot in C and (ii) relocation of the Z-shape outside of the "pie slice".

ANALOGY takes input figures as strings, denoting basic elements like dots, lines and curves symbolically. It first determines the transformation that took A to B. Then it compares rules that transform C to each candidate with the rules that transform A to B. It selects the most similar set of rules between C and the candidates.

2.2. Structure Mapping Theory and SME

Gentner established foundations of this theory in [9]. The theory assumes a predicate logic representation of the situations it tries to map, so it is the foremost representative of the symbolic approaches to analogical reasoning.

[1]The word "situation" in this definition is used in the usual sense. Later in this paper, we'll study situations in the situation-theoretic sense.

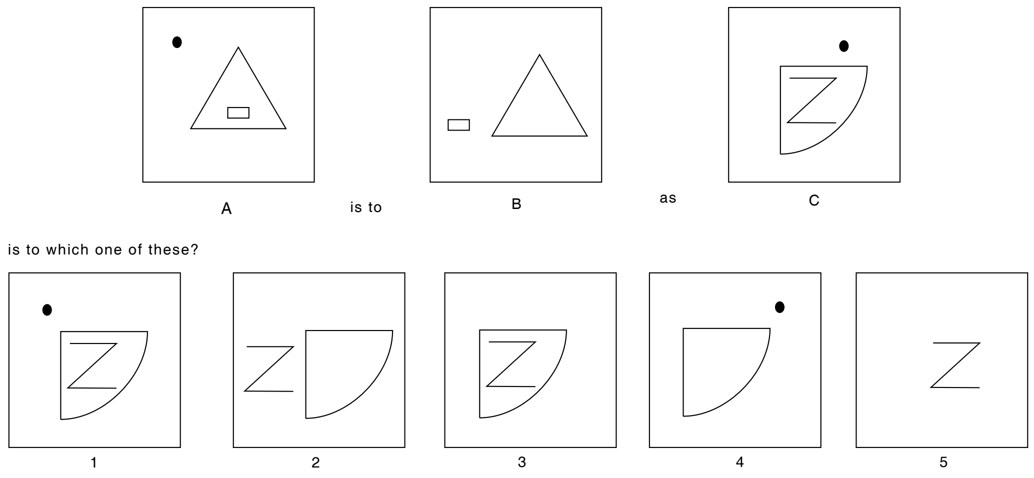

Figure 1. Evans' ANALOGY program (from [3]).

Gentner does not assume some domain dependent representation like ANALOGY. Two contributions of the Structure Mapping Theory are the following:

- The relation-matching principle: Analogies are mappings that are based upon (polyadic) relations instead of attributes.
- The systematicity principle: Mappings of coherent systems of relations must be preferred over individual relations.

Gentner identifies four types of mappings: Literal Similarity, Analogy, Abstraction and Anomaly. Except in Anomaly, in all cases there are relational mappings between the base and the target. In Literal Similarity, however, there are also many attribute mappings along with relation mappings and the "quality" of mapping is thus reduced by these attribute mappings. An example of Literal Similarity is "The K5 Solar System is like our solar system." For Analogy, in which attribute mappings are few, "The atom is like our solar system" is an example. The difference between Analogy and Abstraction lies in having few object attributes in the base and target domains. An example to the latter is: "The atom is a central force system."

The computational counterpart of the Structure Mapping Theory is the Structure Mapping Engine [8]. SME identifies three subprocesses of analogy-making. In *access,* a structure in long-term memory which is similar to the target is sought. In *mapping and inference*, a structure in the target which is analogous to the base is built. In *evaluation and use,* an evaluation regarding the quality of inferences found in the mapping process is made.

In SME, predicate calculus is used to represent the facts. There are individuals and constants, attributes and relations to describe the state of affairs, which are composed into description groups (called *dgroup* in the theory.)

SME consists of three parts. First it matches expressions and entities of the base and target. Then it builds a set of candidate inferences, which are hypothesized to be holding in the target domain. Finally a score indicating the quality of matching is determined by

```
(defDescription simple-water-flow
   entities (water beaker vial pipe)
   expressions (((flow beaker vial water pipe) :name wflow)
                ((pressure beaker) :name pressure-beaker)
                ((pressure vial) :name pressure-vial)
                ((greater pressure-beaker pressure-vial) :name >pressure)
                ((greater (diameter beaker) (diameter vial)) :name >diameter)
                ((cause >pressure wflow) :name cause-flow)
                (flat-top water)
                (liquid water)))
```

Figure 2. SME water flow example (from [8].)

```
(defDescription simple-heat-flow
   entities (coffee ice-cube bar heat)
   expressions (((flow coffee ice-cube heat bar) :name hflow)
                ((temperature coffee) :name temp-coffee)
                ((greater temp-coffee temp-ice-cube) :name >temperature)
                ((flat-top coffee)
                (liquid coffee)))
```

Figure 3. SME heat flow example (from [8]).

structural properties. Note that there is no knowledge other than the syntactic relationships of predicate calculus. The “analogy” is defined only in terms of these structural properties.

An example of SME may be given between the dgroups in Figures 2 and 3. First, SME builds all corresponding predicates with identical arity between Figures 2 and 3. In this case, `(temperature ???)` predicate of Figure 3 can match both `(diameter ???)` or `(pressure ???)` of Figure 2, since both are unary.

In the second step, a series of candidates for global mapping between the base and target dgroups are produced. In these mappings, there is a set of criteria which increases or decreases the plausibility of mappings. If the mapping between different dgroups can match identical predicates better, the mapping gains higher plausibility.

In the third step, the most plausible mapping between elements is selected. Factors to determine the plausibility of a mapping are also supplied to the program, considering whether the type of mapping is Literal Similarity, Analogy or Mere Appearance.

2.3. Copycat

For Copycat, the goal is to solve letter analogies like this one:

$$\begin{array}{ccc} abc & \rightarrow & abd \\ ijk & \rightarrow & ? \end{array}$$

There are several plausible answers here, as Hofstadter and Mitchell discuss in [11]. The most frequent answer is ijl (the relation between c and d is extended to k and l). Other answers are ijd (convert the last element to d) and ijk (convert all c’s to d.)

Another example is as follows:

$$\begin{aligned} abc &\rightarrow abd \\ ijjkkk &\rightarrow \ ? \end{aligned}$$

The most obvious solution is to map c to kkk. In this case, the analogous string is $ijjlll$. Another grouping may be the last letter of each group, and in this case, the resulting analogy is $ijjkkl$. A related point is to have different types of similarity between these letter groups. As abc are first three letters of the alphabet, and the number of letters in letter groups of $ijjkkk$ are also (1,2,3), one can build an analogy like $ijjkkkk$.

Copycat is able to solve problems such as these with a stochastic approach. For example, 1000 runs of the second example above gives 731 $ijjlll$, 165 $ijjkkl$, 53 $ikklll$, 33 $ijjkll$, 7 $ijjddd$, 7 $ijklll$, 3 $ijjkkd$ and 1 $ijjkkkk$ answers.

Copycat has three major components. *Slipnet* is the home of all predefined concepts like the successorship relation between the letters a and b. *Slipnet* contains only the types of concepts and not their instances. It is similar to the long-term memory (LTM) in other analogical reasoning systems.

The second component is *Workspace* where all perceptual activity of Copycat takes place. It resembles the short-term memory (STM) found in other systems. In *Workspace*, instances of letters with their relations retrieved from *Slipnet* live together. These elements are evolved by the third component of system, *Coderack*. In *Coderack*, many agents, each having a precise task wait for their turn to work on the elements in *Workspace*. The scheduling of these small agents (which are called *codelet*s) is determined stochastically, thus allowing different solutions to emerge.

3. The Microworld of SITAR

Throughout the history of scientific development, specific problems and solutions usually predate general approaches, and reversing this order, in general, does not result in good science. A well-known example is the Ptolemaic view of celestial objects, which defines Earth as the center of the universe and explains all phenomena, at least superficially, in its own terms. Since there was a partial explanation to the problems of celestial bodies, no further investigation was made to scrutinize the foundations of the Ptolemaic theory and thus, the theory (also due to particular social and political dynamics of the era) hindered the development of a better understanding of the universe until Copernicus and Kepler came along.

In our own work on analogical reasoning we proceed from the specific towards the general. This characteristically means that we try out our ideas first on the so-called microworlds. Another reason stemming from practical needs to use microworlds may be tied to the difficulty of analogy research in general. Theoretical computational limits, which are not investigated thoroughly in analogy research, must be taken into consideration. As the number of objects in a situation increases, the number of possible relations between objects increases exponentially, thus making large scale figures inappropriate subjects for analogy research, at least in light of current techniques.

The microworld of SITAR consists of basic geometric shapes which have the following attributes:

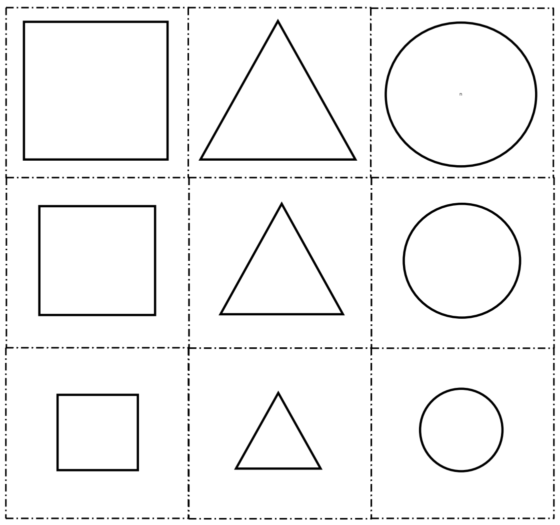
Figure 4. Elements in the SITAR domain.

Type The type of a geometric figure determines its shape. In the current version of the project, for the sake of compatibility with Tarski's World [1], we determined the object types as *cube*, *pyramid* and *dodecahedron*. Note that in the illustrations below, we'll use 2D figures instead of these and denote cubes with squares, pyramids with triangles and dodecahedra with circles.

Size Each object can have an associated size. These are the values, "large", "medium" or "small".

Location In the microworld, each object has its own location on a grid. The grid is an 8 by 8 chessboard.

There are also a number of possible relations between objects. We use those relations found in Tarski's World [1]:

SameRow(a, b**)** If objects a and b have the same y value in their location attribute, this relation holds.

SameColumn(a, b**)** If objects a and b have the same x value in their location attribute, this relation holds.

SameSize(a, b**)** If a and b have the same size, this relation holds.

SameShape(a, b**)** If a and b have the same shape (square, triangle, etc.) this relation holds.

Larger(a, b**)** Holds if a is larger than b.

Smaller(a, b**)** Holds if a is smaller than b.

Adjoins(a, b**)** a and b are located on adjacent (but not diagonally) squares.

LeftOf(a, b) a is located nearer to the left edge of the grid than b.

RightOf(a, b) a is located nearer to the right edge of the grid than b.

FrontOf(a, b) a is located nearer to the front of the grid than b.

BackOf(a, b) a is located nearer to the back of the grid than b.

Between(a, b, c) a, b and c are in the same row, column, or diagonal, and a is between b and c.

Despite its misleading austerity, the microworld of SITAR is a considerably rich domain for analogy problems. The following examples illustrate analogy problems that can be formulated within this domain. Note that in these examples we use a 3-by-3 grid as opposed to an 8-by-8.

Example 1 (Illustrating the role of attributes) This example is shown in Figure 5. In a, the types of three figures are *square,* and all objects in b are turned to *circle.* When one sees that there is a rule "squares $\rightsquigarrow$ circles" in the base situation, it is easy to select the corresponding situation of c as d. However, as in Copycat, there may be other options for the analogy illustrating the role of relations.

Example 2 (Illustrating the role of relations) In Figure 6, the analogy between a and b can be represented by the rule "objects in the same row $\rightsquigarrow$ objects in the same column" (or more precisely, "middle row" and "middle column"). The type attributes of objects play no role in the relation between a and b, so for the target situation d, a change in object types is not playing a role in the analogy.

Note that, in examples 1 and 2, the difference of a and b is applied to the similarity of a and c. In the first example, difference of a and b were object types, and similarity of a and c were also object types and application of dissimilarity of a and b to similarity of a and c was straightforward.

Adopting an idea found in [15], we can denote dissimilarity of two situations a and b by $Dis(a, b)$ and similarity of two situations a and c by $Sim(a, c)$. $Dis(a, b)$ is a relation of conjunction of relations that hold in b but not in a and $Sim(a, c)$ is a relation of conjunction of relations that hold in both a and c. Therefore, in both examples, $d = Sim(a, c) \cap Dis(a, b)$ However, as we can see in these examples, relations Sim and Dis do not always have a fixed order, namely, similarity or dissimilarity of relations cannot be expressed always in first order, second order, $\ldots$ terms. It is not straightforward to represent similarity and dissimilarity of situations with a set of relations, since similarity or dissimilarity may occur in higher order relations which require us to quantify over subsets, subsets of subsets, ..., etc.

The kernel of this problem is to build algorithms that can check the most appropriate levels for analogy in a given problem. For example, a figure that is a square composed of squares is analogous to a triangle composed of triangles. The first level in this analogy is the figures' composition from identical elements. However, the interesting analogy in

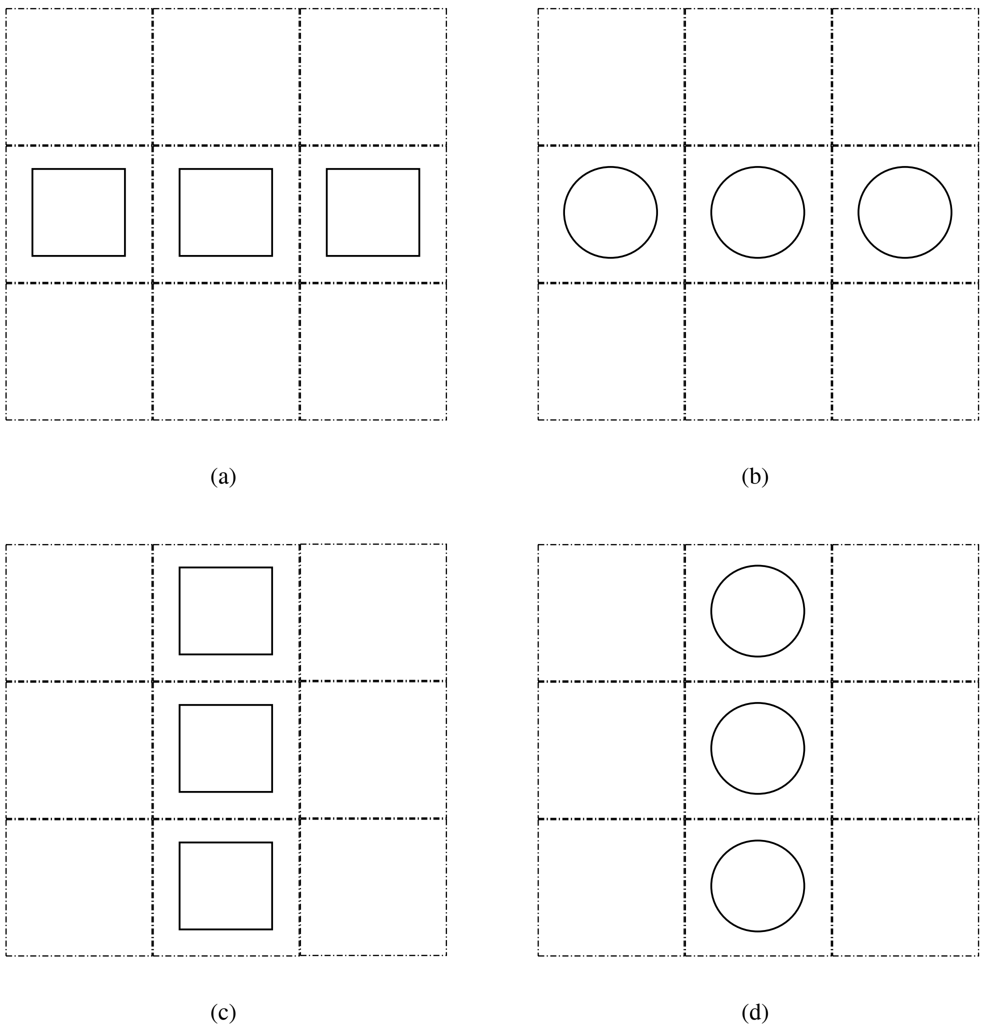

Figure 5. Analogous Situations in Example 1.

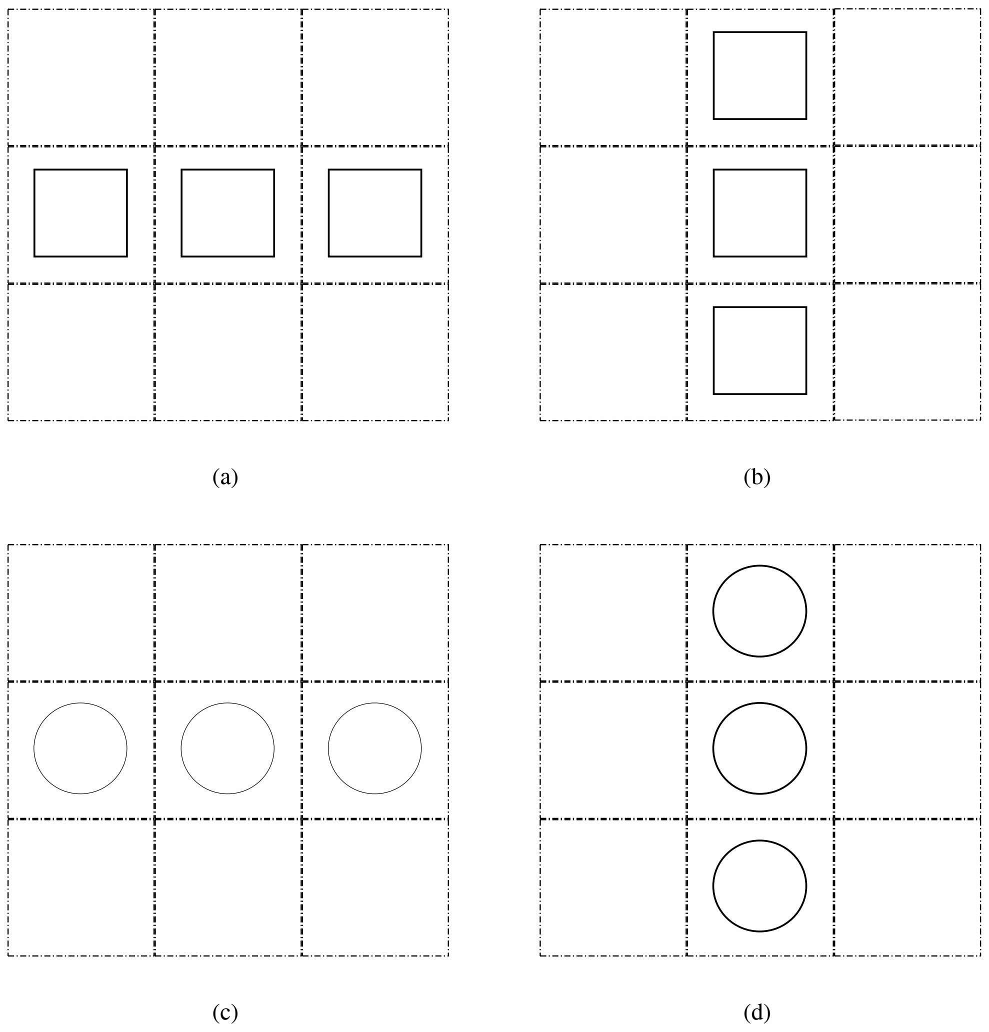

Figure 6. Analogous situations in Example 2.

these figures is not of this kind, but from a bird's eye perspective. If we ask what would be analogous to a figure consisting of a pentagon composed of pentagons, one reasonable answer might be a square composed of squares.

4. Architecture of SITAR

In [14], it is reported that there are six processes that every analogical reasoning model contains. The paper also states that there is currently no model that incorporates all of these, but different models focus on different processes. The processes are:

Representation-building A system is supposed to produce its own representation from a natural input. An example may be Copycat's input of letter strings. At this level, only the input elements but not their relations are fed to the system and the system is expected to produce a representation from this input.

Retrieval This is the retrieving of the base analogy from a set of different situations. It is best accomplished when shared objects, attributes and a theme exist between the base and target of an analogy.

Mapping In this procedure, elements of the base situation are mapped onto elements of the target situation. The mapping should preserve the semantics of particular elements and might suffer from a combinatorial explosion if the search is not constrained.

Transfer This process transfers the knowledge from the base domain to target domain for search. By way of transfer, new relations are sought in the target domain.

Evaluation This process checks whether the transferred knowledge is applicable to the target domain.

Learning This is the process which focuses on the incorporation of transferred knowledge to the target domain.

SITAR has an architecture which combines the first three processes above with a learning module. The evaluation is implicit in user response to the program and transfer of knowledge from a domain to another is not applicable, since there is a single domain at hand. In this manner SITAR is one of the few architectures that combine these processes.

Representation of a problem is one of the most important tasks in AI. Winston [16] identifies the following criteria for a good analogical representation:

It makes important facts explicit;

It suppresses irrelevant detail;

It is perspicuous;

It exposes constraint;

It can be computed from a natural input.

Due to the discrete nature of the microworld of SITAR, it is easier to represent situations in symbolic notation instead of images. For this particular domain, a representation scheme must also satisfy the following criteria:

- There must be a way to represent hierarchical situations. In pictures, there may be higher level pictures which have a stronger alignment for analogies. An example would be a square made up of circles. This situation must be represented both as a square and a set of circles.
- The representation must satisfy partial groupings. There may be sub-situations in a situation. Groupings must also be represented explicitly. Representation should be permissive enough to add further information for groupings.
- The representation must have some means to represent transformation of base to target situations. There must be a facility about information flow and mapping between the base and the target.

These considerations led us to Situation Theory as a representational framework. Since it allows situations to be contained in other situations, and situations are used as any other object in the language, first of these criteria is met. Second one is also met by this property of the theory. For the third one, constructs of the theory called *constraint*s can be used. They represent information flow between different situations, hence making Situation Theory one of the suitable candidates for the representation.

4.1. Situation Theory

Situation Theory is a pragmatic proposal, aiming to represent the world in a natural way. It began its development in early 1980s. Since then, many enhancements were developed. The account we give here is based on a recent article [5] of Devlin. For a rather detailed formal account of the theory, the reader is referred to [4] and for an historical exposition to [2].

Situation Theory is an information based theory. The initial motivation was to fill the gap between everyday understanding of the world and its formal representation in logic. As a beginning observation, Devlin [5] (p. 601) cites from Barwise and Perry's 1980 paper (*The Situation Underground*):

> The world consists not just of objects, or of objects, properties and relations, but of objects having properties and standing in relations to one another. And there are parts of the world, clearly recognized (although not precisely individuated) in common sense and human language. These parts of the world are called situations. Events and episodes are situations in time, scenes are visually perceived situations, changes are sequences of situations, and facts are situations enriched (or polluted) by language.

Information (and truth) are evaluated in Situation Theory always with respect to a situation.

A piece of information that can be supported by a situation is called an *infon*. An infon consists of a relation which has n places to be filled, and a value called *polarity* (either 1 or 0), denoting truth or falsity. If a situation s supports an infon σ, we write

$$s \models \sigma$$

where

$$\sigma = \ll relation, object_1, object_2, \ldots, object_n, polarity \gg$$

Infons may be combined via conjunction, disjunction and quantification. An example may be

$$s \quad \models \quad \ll taller, Adam, Eve, 1 \gg \wedge \ll smarter, Adam, Eve, 0 \gg$$

Situation Theory is strongly type-theoretic. Basic types in the theory include (but are not limited to):

- IND : The type of an object (individual)
- REL^n : The type of an n-ary relation
- SIT: The type of a situation
- INF: The type of an infon
- TYP: The type of a type
- PAR: The type of a parameter (described later)

Apart from these, there can be types for a given situation (or an infon). For example, a situation can be of type $meeting$, hence making it a meeting situation.

Situations and infons can have parameters to denote the generalizations. An agent must make generalizations and inference based on interrelated infons and situations, so that an adaptive behavior with respect to changing situations can arise.

Parameters are of a type, hence restricting the available individuals that can fill a parameter. A parameter of type SIT can be shown like $\dot{s}$, a parameter of type IND can be shown like $\dot{i}$. Within the infon,

$$\ll smarter, Adam, \dot{i}, 1 \gg$$

$\dot{i}$ denotes an individual whom Adam is smarter than. There are constructs called *anchor*s which instantiate parameters to specific objects. Anchors are functions which assign a member of a set to a parameter. There are also restricted parameters, combining a parameter with a condition and quantifying over a set with respect to a condition.

An important concept is *type abstraction* which is basically a way of inferring types through situations or objects. For example

$$[m | m \models \ll talking, \dot{a}, \dot{b}, 1 \gg \wedge \ll listening, \dot{b}, \dot{a}, 1 \gg]$$

defines a meeting situation of type m via a compound infon.

Constraints define a way to build causality relations between situations. For example, the sentence

Smoke means fire

describes a relationship between smoky situations and fiery situations. More generally the expression

$$S \Rightarrow S'$$

describes a causal relationship between the situation types S and S'. ($S \Rightarrow S'$ is read as "S involves S'")

Constraints link not situations, but types; hence they are the common means to represent any rule between situation types. Linking situation types, they can be instantiated for a particular situation when the antecedent occurs. Therefore, when an agent sees a smoky situation, she can infer that the situation is also a fiery situation. (Notice that her inference is defeasible.)

Following remarks, which will be of importance in this work, are from Devlin [5] (p. 610):

> It should be noted that the ontology of situation theory has no bottom layer; every individual or situation can be subdivided into constituents, if desired. This implies that it is possible to represent and analyze a domain at any degree of granularity, to move smoothly up and down the granularity scale during an analysis, and to "zoom" the granularity to investigate specific issues in an analysis, while keeping the remainder of the representation fixed.

4.2. Representation

Basic Objects Basic objects in SITAR are represented with infons depicting the relations they stand in. The existence of an object in a situation is stated by an infon in the form

$$\ll occupies, NAME, SHAPE, SIZE, LOCATION, 1 \gg$$

where $NAME$ is the name of the object, $SHAPE$ is whether it's a pyramid, cube or dodecahedron, $SIZE$ is whether it's small, medium or large, and $LOCATION$ is the location in the situation where the object is positioned.

For example a large cube at point (3,3) is represented by:

$$\ll occupies, p_1, cube, large, (3,3), 1 \gg$$

A situation s like in Figure 5(a) can be represented as:

$$\begin{array}{rcl} s & \models & \ll occupies, p_1, cube, medium, (1,2), 1 \gg \\ & \wedge & \ll occupies, p_2, cube, medium, (2,2), 1 \gg \\ & \wedge & \ll occupies, p_3, cube, medium, (3,2), 1 \gg \end{array}$$

4.2.1. Relations and Groupings

Relations in SITAR which are described in Section 3. can be inferred from attributes of objects. For instance, if there are two objects in a situation, one large and other small, then there are $smaller$ and $larger$ relations holding between these two objects.

Such relations are represented with infons in the form

$$\ll RELNAME, OBJECT1, OBJECT2, OBJECT3, 1 \gg$$

where $RELNAME$ is one of the relations and $OBJECT$s are the names of the objects standing in this relation. However, due to the arity of relations, there may be fewer objects standing in a relation, and only those are written.

For example if there are two objects in a situation like

$$\begin{aligned} s \;\models\; & \ll \mathit{occupies}, p_1, \mathit{pyramid}, \mathit{small}, (1,2), 1 \gg \\ \wedge\; & \ll \mathit{occupies}, p_2, \mathit{pyramid}, \mathit{medium}, (2,2), 1 \gg \end{aligned}$$

then the following relations can be inferred from these:

- $\ll sameRow, p_1, p_2, 1 \gg$
- $\ll sameShape, p_1, p_2, 1 \gg$
- $\ll smaller, p_1, p_2, 1 \gg$
- $\ll larger, p_2, p_1, 1 \gg$
- $\ll adjoins, p_1, p_2, 1 \gg$
- $\ll rightOf, p_2, p_1, 1 \gg$
- $\ll leftOf, p_1, p_2, 1 \gg$

Hence, the situation can now be written as

$$\begin{aligned} s \;\models\; & \ll \mathit{occupies}, p_1, \mathit{pyramid}, \mathit{small}, (1,2), 1 \gg \\ \wedge\; & \ll \mathit{occupies}, p_2, \mathit{pyramid}, \mathit{medium}, (2,2), 1 \gg \\ \wedge\; & \ll sameRow, p_1, p_2, 1 \gg \\ \wedge\; & \ll sameShape, p_1, p_2, 1 \gg \\ \wedge\; & \ll smaller, p_1, p_2, 1 \gg \\ \wedge\; & \ll larger, p_2, p_1, 1 \gg \\ \wedge\; & \ll adjoins, p_1, p_2, 1 \gg \\ \wedge\; & \ll rightOf, p_2, p_1, 1 \gg \\ \wedge\; & \ll leftOf, p_1, p_2, 1 \gg \end{aligned}$$

Along with these basic relations, analogy-making also depends on groupings of elements. Groupings are higher level structures in a situation. A grouping in SITAR is called a $gestalt$. A gestalt can also be considered as a (sub)situation, supporting a generalized infon that describes the interrelated position of its elements.

A situation can support an infon in the form

$$\ll gestalt, GESTALTTYPE, GESTALTNAME, 1 \gg$$

where $GESTALTTYPE$ shows one of the possible types for groups and $GESTALTNAME$ is a name for this. It can also have infons representing which objects are in a particular group by

$$\ll belongsGestalt, NAME, GESTALTNAME, 1 \gg$$

where $NAME$ is the name of an object and $GESTALTNAME$ is the name of the group it belongs to.

For example, in the preceding situation s there is a very primitive gestalt consisting of two pyramids. The higher level relation of this structure is a horizontal line segment.

$$\begin{array}{rcl} s & \models & \ll gestalt, horizontalLine, hl_1, 1 \gg \\ & \wedge & \ll belongsGestalt, p_1, hl_1, 1 \gg \\ & \wedge & \ll belongsGestalt, p_2, hl_1, 1 \gg \\ & \wedge & \ldots \end{array}$$

The type of a gestalt can be one of the following types:

Same Type All of a gestalt's elements have the same type.

Same Size All of a gestalt's elements have the same size.

Horizontal Line Gestalt's elements appear like a horizontal line.

Vertical Line Gestalt's elements appear like a vertical line.

Square Frame Gestalt's elements appear like a square, consisting of other objects. Inside of the square is blank.

Rectangle Frame Gestalt's elements appear like a rectangle, consisting of other objects. Inside of the rectangle is blank.

Square Full A full square, which lacks a blank in it.

Rectangle Full A full rectangle, which lacks a blank in it.

All-R R is a relation from Section 3. In a gestalt of this kind, all elements satisfy some relation (e.g. *frontOf*, *between*, *larger*).

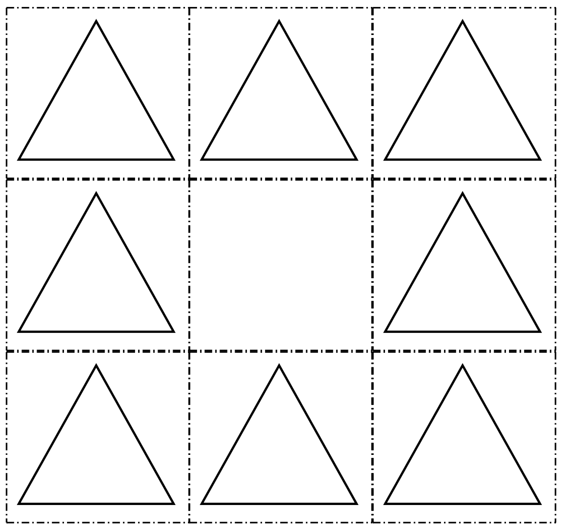

Figure 7. A square composed of pyramids.

4.2.2. Representing Complex Gestalts

In SITAR, a situation is not always a collection of basic objects and their properties. Instead, as we indicated about gestalts, a situation may cover only a part of these objects and may include more information than attributes and relations of basic objects. Another implication of this is to have situational relations among elements. In Situation Theory we can write infons which contain situations as parameters.

Yet another obvious implication of this is to have multiple situations associated with basic objects of a figure. These situations can be flat (not containing other situations) or hierarchical (containing other situations). Situations containing other situations are not allowed to make circular reference to themselves, thus permitting only a directed acyclic graph representation.

One can represent the situation in Figure 7 in a number of ways. One example is

$$\begin{array}{rl} s \models & \ll gestalt, verticalLine, vl_1, 1 \gg \\ \wedge & \ll belongsGestalt, p_1, vl_1, 1 \gg \\ \wedge & \ll belongsGestalt, p_2, vl_1, 1 \gg \\ \wedge & \ldots \\ \wedge & \ll gestalt, horizontalLine, hl_1, 1 \gg \\ \wedge & \ldots \end{array}$$

hence representing the situation as a union of two vertical lines and two horizontal lines. (In this case, pyramids in the corners will belong to both vertical and horizontal line gestalts.) Another way to look at things is like

$$\begin{array}{rl} s \models & \ll gestalt, squareFrame, sf_1, 1 \gg \\ \wedge & \ll belongsGestalt, p_1, sf_1, 1 \gg \end{array}$$

```
P3P3P3
P3__P3
P3P3P3
```

Figure 8. A square composed of pyramids, in input format.

$$\begin{array}{ll} \wedge & \ll belongsGestalt, p_2, sf_1, 1 \gg \\ \wedge & \ldots \end{array}$$

and representing the situation as a single square frame.

There are two options in this case. One is representing such gestalts as hierarchical gestalts, namely representing the square in Figure 7 as a combination of two horizontal and two vertical lines. The other is to have independent gestalts, a square formed by basic objects and a set of lines again formed by basic objects, but no relation between these gestalts whatsoever (except the shared basic objects).

The way selected in SITAR is the former, hence seeing more complex gestalts as combinations of simpler gestalts. A square frame is formed by two horizontal and two vertical lines, and for the cases when a situation supports more complex frames, like lines composed of squares in turn composed of basic objects, gestalts are always represented in a hierarchical manner.

4.3. Building a Representation

In human analogy-making, one of the most important phases is building an inner representation for the given problem. This phase of analogy-making is ignored by several projects in the literature, except those of Copycat and its sisters, and some others like Evans' ANALOGY. SITAR is also one of the projects that contain a representation building phase.

The input given by the user to the program is not a situation description. Instead it is more or less a "visual" representation of situations. From these situations the program creates its inner representations as infons and situations, and infers the necessary relations without the aid of the user.

The best way to illustrate the input format is through an example.

The example situation given in Figure 8 is the input format of the situation in Figure 7. In the input format, a basic object is denoted by a letter and a digit. The letter, being one of `C`, `P`, or `D`, shows the type of object, and the digit, being one of `1`, `2`, or `3`, shows the size of it. For blank locations __ (double underscores) are used.

Once the situations are fed to the program in this format, it produces its inner infon oriented format and builds gestalts and relations in a situation. However, since a situation can be represented in a large number of ways, where most of these are irrelevant for the problem, there must be a pruning process in representation building to prune irrelevant relations and gestalts to be built.

This brings us to the heart of the problem, as the number of possible relations in situations causes combinatorial explosion and finding the relevant relations needs a well adjusted architecture.

```
situation a:
________________
C2C2C2__________
________________
________________
________________
________________
________________
________________


situation b:
________________
D2D2D2__________
________________
________________
________________
________________
________________
________________


situation c:
_C2_____________
_C2_____________
_C2_____________
________________
________________
________________
________________
________________
```

Figure 9. The problem in Figure 5, in input format.

In order to solve this problem, Copycat and related projects (and also AMBR [13]) employ a "microagent" approach where stochastic processing of situations leads to a compromise between finding a good solution and filtering irrelevant relations. Unlike these, SITAR uses a monolithic approach where the processing of candidate relations are done by a single agent.

In order to tackle the problem, SITAR architecture gathers as much information from the context. To show this, we can use a problem just like in Figure 5. The input format description of that problem (in an 8-by-8 setting) is shown in Figure 9. In the figure, first 3 situations, namely a, b and c, are shown and the program is asked to find situation d that is analogous to c in terms of the analogy between a and b.

In this case, when the program is building representations, it must do this in a way to minimize irrelevant paths of comparison. In order to achieve this, however, the program should gather information not only from a single situation like b in Figure 9, but also from other situations in the problem case and previous problem cases. In other words, while finding gestalts and relations in a single situation, the program should check whether these gestalts and relations have counterparts in other situations. This way it can prevent irrelevant gestalts and relations that lead to combinatorial explosion.

However, this also means that representation building, retrieval and mapping phases of the program are to be merged. Since information from previous problems and other situations are required while deciding grouping, the architecture must be flexible to pipe this information from “higher” to “lower” levels. Other analogy-making projects like Copycat and Tabletop solve this with hand-coded likelihood information without a retrieval module. However, in our case the domain is large and we need to use previous cases as a launching pad for representation building and mapping.

In the next two sections we’ll first discuss how retrieval and mapping work. Then an engine that combines all these modules will be discussed.

4.4. Retrieval

Retrieval of previous cases in SITAR depends on indices created in the learning phase. These indices use relations and arguments of infons to index problem cases and bring other “possible” relations and gestalts for the problem at hand.

In the learning phase, where a problem and its solution are stored, indices of its “occupies” and “gestalt” relations, as well as relations that take gestalts as arguments are created and added to a central index. The central index includes all gestalt elements, basic objects, gestalt relations and mapping relations (in other words, almost all problem case information) and these are grouped together per case and per similarity.

Suppose we have, in our central index, the solution for the problem case in Figure 6. The retrieval process finds higher level properties like lines with the cues derived from lower level attributes like object type. In the problem case, situation (a) is a horizontal line composed of cubes, hence when a cube is met in a new situation, program must check for the availability of a horizontal line gestalt.

In order to do this, program must estimate the conditional probabilities of different types of gestalt with lower level relations and when a lower level relation is met, the program should consider the existence of a higher level relation. For example, intuitively we can propose that there is a direct relationship between *adjoins* relation and lines. Although not all relations can give clear cut cues for finding gestalts, at least there will be candidates to focus the search in a more meaningful fashion.

4.5. Mapping

Mapping works between situations of a problem case. The most studied aspect of analogy-making is mapping. For some (e.g. [9]), analogy-making is identical to mapping of relations between different situations.

The mapping process in SITAR aims to find the most relevant mappings between situations’ elements. In order to achieve this, the mapping process should go hand in hand with the representation building and retrieval processes.

Mapping establishes correspondences between groups, objects and relations of situations. There are three given situations, a, b and c and the “meaningful” correspondences are difference correspondences between a and b and similarity correspondences between a and c. For example, in Figure 6 mapping process should find that both a and c have horizontal lines, whereas a and b differ in type of line gestalt. The resulting situation d is obtained by applying difference of a and b to similarity of a and c.

The quality of mapping, as discussed in [9], depends on the systematicity principle. Systematicity principle states that robustness of a mapping established between two situations depends on the higher level relations mapped to each other. In SITAR, we don't have "higher level relations" as in SME. However, gestalt structures can be considered as higher level relations and any relation (like "adjoins" or "between") that uses these gestalt structures can be considered as higher level.

Additionally we use this heuristic: more encompassing relations are better in mapping. Unlike SME, SITAR tries to find a solution to a problem and looking from this perspective, it tries to "assign" meaning to all elements in a situation. (This is also the human reaction to such analogy questions. People try to establish correspondences that encompass *all* elements in a situation, rather then considering some of the elements as *noise*.)

These two ideas are the guiding principles in the mapping process. In the next section, we discuss how these three processes, representation building, retrieval and mapping, can be unified.

4.6. The Unified Engine of Representation, Retrieval and Mapping

In analogy-making literature the common solution to unify representation, retrieval and mapping processes is through micro agent based connectionist architectures. Both in Copycat [11] and AMBR [13], the solution is to divide the problem into smaller chunks and forward these to stochastic micro agents. These microagents then build a representation along with retrieving necessary information from the LTM and perform a mapping.

Although such a model may work for our problem as well, we don't see why such a diversion is necessary in the first place. Microagents in these projects are not really microagents working themselves; they are pieces of action waiting to be executed according to their priority. We think naming things as they are actually results in a simpler design compared to the alternative of delegating the problem to *pseudo*-microagents.

The logical foundation established by Situation Theory introduces a flexible architecture for the relations between elements and situations. SITAR uses "hypons" and "factons" to represent the information about situations and problem cases.

In the current architecture hypons are triples and factons are tuples and they differ in their actuality. Hypons are hypothetical information about situations and problem cases, and factons are factual.

Both hypons and factons employ an infon about the problem. This infon may be one of the simplest, as in "occupies" or a more complex one like "maps" (which is used to represent mapping relations between situations) or "gestalt" to denote the groups in situations. Any infon can be made into a hypon and a facton.

Along with infons, both hypons and factons have importance, which is a number. Importance is determined in run time and, once determined, is fixed. The "importance" of a hypon determines the likelihood that the engine will process it, and the importance of a facton shows the generality of it.

Additionally, hypons have an action part which is triggered when the hypon's infon is checked about correctness. This action may be retrieving some other situation from the case library, creating another hypon, etc. A hypon whose infon is checked for validity automatically creates a facton with an identical infon and importance.

All hypons and factons are stored in short term memory. Hypons wait their turn to be checked and factons are used to find the most important relations about situations and the problem. The case library, which is a repository for previous problem cases, contains factons about cases. The importance metric provides an aid in finding the most relevant relations about previous cases.

Supposing there are only "occupies" relations in situations, the engine works in the following way:

1. Build factons from infons with "occupies" relations. Set importance of these to 1.
2. Retrieve a set of previous cases from the case library using the factons at hand. Use their factons to create an initial set of hypons with importance 2.
3. In each step, select a hypon and check its infon's validity. If the infon is true, perform its action and create a facton from the hypon's infon. Do this until there arises a set of important factons which solve the mapping problem, or until a predetermined number of steps (in which case "no solution" will be reported).

This architecture loads the problem to the shoulders of hypons. Hypons should lead to relevant actions and these actions are supposed to search the solution space in a meaningful fashion. However, through pattern matching of infons and scanning through important information about the problem, we look forward to preventing the combinatorial explosion in solving analogy problems.

The last step we didn't mention above is the creation of situation d in problems. Once the mapping between a, b and c are found, creating d becomes the task of finding a situation which satisfies the criteria we mentioned in section 3.

4.7. Learning

After finding a solution, SITAR saves it to its case library for future reference. All of the case library is made up in this way gradually, using solutions.

What if no solution can be found? If no solution can be found or the solution of the program seems incorrect, the user is expected to supply a valid solution to the system and the system analyzes the solution and saves it like any other one. If the mapping could not be established in the given solution, the user is requested to articulate the mapping and the complete problem case in terms of factons and relations.

The learning phase in SITAR is thus integrated with the evaluation phase. Learning occurs in any case after evaluation of the solution and the program's ability to solve further problems is a function of its case library.

When a solved problem case is inserted into the case library, it is indexed by relations of infons, types of the gestalts, types of objects, size of objects. These indices are used to retrieve the necessary elements from the case library.

5. Summary

This work is an initial step towards having an analogical reasoning system built on Situation Theory. Our motivation stems from various projects aiming at solving this problem. Three representatives of these projects are Evans' ANALOGY, Hofstadter and Mitchell's Copycat and Gentner and colleagues' SME. First of these programs solves geometric analogy problems found in I.Q. tests. The second one produces letter analogies. The third one is a general analogy engine applicable to domains described in predicate logic.

Our model assumes a microworld composed as a chessboard having figures in places of chessmen. Figures can have various attributes and can stand in various relations and compose structures called gestalts. Gestalts are figure groups supporting a pattern.

We employed Situation Theory as a means to describe situations in this microworld. Situation Theory allows nesting of situations which we will use extensively in future studies. It can model the information flow between different situations and thus a more natural representation becomes possible.

Our system builds the representation of situations by itself. It produces the necessary relations and discovers most important aspects of a situation by combining representation building, retrieval and mapping. The evaluation of solutions is done by the user and a solution for the problem is stored for learning (either the program's successful solution or a solution supplied by the user).

As an initial step towards an understanding of Analogical Reasoning in Situation Theory, this work naturally has omissions. We look forward to reporting our further findings in our upcoming studies.

References

[1] Jon Barwise and John Etchemendy. *Language, Proof and Logic*. CSLI Publications. 2002

[2] Jon Barwise and John Perry. *Situations and Attitudes*. MIT Press. 1983

[3] Simon Ben-Avi. *T.G. Evans I.Q. test solver: ANALOGY*. `http://www.aaai.org/aitopics/html/analogy.html`

[4] Keith Devlin. *Logic and Information*. Cambridge University Press. 1991

[5] Keith Devlin. Situation theory and situation semantics. In: *Handbook of the History of Logic*. Dov Gabbay and John Woods (eds.). Volume 7. pp. 601-664. Elsevier. 2006

[6] Robert L. Goldstone. Douglas L. Medin and Dedre Gentner. Respects for similarity. *Psychological Review*. **100**(2):254-278. 1993

[7] Thomas G. Evans. A program for the solution of Geometric-Analogy Intelligence-Test Questions. In: *Semantic Information Processing*. Marvin Minsky (ed.). MIT Press. 1969

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

[9] Dedre Gentner. Structure Mapping: A theoretical framework for analogy. *Cognitive Science*. **7**:155-170. 1983

[10] Dedre Gentner. Analogy. In: *A Companion to Cognitive Science*. William Bechtel and George Graham (eds.) pp. 107-113. Blackwell. 1998

[11] Douglas Hofstadter and Melanie Mitchell. The Copycat Project: A model of mental Fluidity and Analogy-making. In: *Fluid Concepts and Creative Analogies*. Douglas Hofstadter (ed.). pp. 205-269. Basic Books. 1995

[12] Keith Holyoak and Paul Thagard. *Mental Leaps: Analogy in Creative Thought*. MIT Press. 1995

[13] Boicho Kokinov. A hybrid model of reasoning by analogy. In: *Analogical Connections, Advances in Connectionist and Neural Computation Theory*. Volume 2. Keith Holyoak and John Barnden (eds.). pp. 247-320. Ablex. 1994

[14] Boicho Kokinov and Robert M. French. Computational models of analogy-making. In: *Encyclopedia of Cognitive Science*. Volume 1. Lynn Nadel (ed.). pp. 113-118. Nature Publishing Group. 2003

[15] Amos Tversky and Itamar Gati. Studies of similarity. In: *Cognition and Categorization*. E. Rosch and B. B. Lloyd (eds.). pp. 79-98. Lawrence Erlbaum. 1978

[16] Patrick Henry Winston. Learning and reasoning by analogy. *Communications of the ACM*. **23**(12):689-702. 1980