New Bulgarian University

Central and Eastern European Center for Cognitive Science

A DYNAMIC EMERGENT COMPUTATIONAL MODEL OF ANALOGY-MAKING BASED ON DECENTRALIZED REPRESENTATIONS

by

Alexander Alexandrov Petrov

A thesis submitted in partial fulfillment of the
requirements for the degree of

Ph.D. in Cognitive Science

Supervisors: Assist. Prof. Boicho Kokinov

Professor Encho Gerganov

Sofia, July, 1998

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION

1.1. Motivation

The key intuition underlying the research presented in this dissertation is that the mechanisms giving rise to human analogy-making are central to cognition. Analogy is not just a specific technique for problem solving and argumentation occasionally called upon when the more reliable methods such as deduction and proof do not work. As we view it, analogy is a manifestation of the fundamental cognitive ability to relate new information to old knowledge and to flexibly manipulate both until they fit into a harmonious whole. As such, it highlights a number of issues that are absolutely central to cognition in general — organization of memory, manipulation of complex structured representations, dynamic relevance, flexible allocation of resources, perception and categor-ization, generalization, learning, etc. Research on analogy, therefore, transcends the boundaries of the specific phenomenon and goes deeply into the core of intelligence.

The main instrument for the research presented in this thesis in the methodology of cognitive modeling. The aim is to analyze analogy-making in computational terms and to construct a working artifact in the form of a computer program. The behavior of the model is then compared to empirical data collected by psychological experimentation. The criterion for success is whether the model contributes to our theoretical understanding of the hidden mechanisms of human cognition.

This thesis describes a computational cognitive model called AMBR (*Associative Memory-Based Reasoning*). It provides a detailed account of its mechanisms and demonstrates its operation by reporting the results of numerous simulation experiments performed with a computer implementation of the model. Throughout the thesis, an attempt has been made to formulate the implications of AMBR for our understanding of human cognition as well as to compare it to other models presented in the literature.

The research reported here is part of a larger research project launched by Boicho Kokinov approximately ten years ago (Kokinov, 1988, 1990, 1994a; Kokinov, Nikolov, & Petrov, 1996; Kokinov & Hadjiilieva, 1997; Kokinov, 1998). The long-term goal of the project is to give a unified account of deductive, inductive, and analogical reasoning by realizing them with the same set of mech-anisms.

As analogy is in a sense representative of cognition in general, a model of analogy-making should be based on a full-fledged cognitive architecture. We do not expect that a small 'analogy machine' based on a few simple assumptions could explain such complex phenomenon. Neither do we expect that this could be done by some all-encompassing 'magic formula'. Instead, we conceptualize analogy as an emergent product of the collective effort of many interdependent mechanisms. The claim is that these same mechanisms can be used for other cognitive tasks too. As a consequence, modeling analogy-making requires a solution to a number of issues about knowledge representation, organization of memory, allocation of computational resources, perception, etc.

AMBR is based on a cognitive architecture that is a first step towards this very distant goal. The architecture DUAL (Kokinov, 1994a,b,c) provides a framework for building dynamic emergent computational models of cognitive phenomena. AMBR is one such model.

1.2. Main Ideas of DUAL and AMBR

1.2.1. Overview of DUAL

DUAL is a general-purpose cognitive architecture that comprises a unified description of mental representation, memory structures, and processing mechanisms. All these aspects of the architecture are organized around the principles of hybridity, emergent computation, dynamics, and context sensitivity.

DUAL is hybrid — it consists of complementary aspects. Moreover, it is hybrid in two ways. On one hand, it hinges upon the symbolic/connectionist distinction and the integration between the two. On the other, there is the declarative/procedural distinction and integration thereof. DUAL is also emergent, dynamic, and context-sensitive. All processing and knowledge representation in the architecture is carried out by a cohort of small entities called *DUAL agents*. There is no central executive that controls the whole system, allocates resources, resolves conflicts, etc. Instead, there are small-scale DUAL agents and local interactions between them. The global behavior of the system emerges from the self-organizing pattern of these interactions. An important feature of DUAL's operation is that it is constantly changing in response to influences from the environment. This is possible due to the emergent nature of the processing and the lack of rigid centrally imposed algorithm.

In a little more detail, each DUAL agent is a hybrid entity serving both representational and processing purposes. Each agent is relatively simple and has access only to local information, interacting with a few neighboring agents. It has a *micro-frame* storing declarative and procedural knowledge. Its *symbolic processor* can perform simple manipulation on symbols (discrete compositional entities) and to pass them to other agents. The complementary aspect of the processor is engaged in spreading *activation* (continuous additive quantity) between agents. Thus they can also be conceptualized as nodes in a network.

The *speed* of the symbolic processing performed by a given DUAL agent depends on its *activation level*. Active agents work rapidly, less active agents work slowly, and inactive agents do not work at all. In this way, each agent contributes to the overall computation in the system to a different extent. As activation levels change continuously, the speed of the symbolic processing changes accordingly. This is a key factor for the dynamic emergent computation which is characteristic of DUAL.

The *long-term memory* of the architecture consists of the total population of all permanent DUAL agents. The active subset of them plus some temporary agents constitute the *working memory* of the system. The contents of the working memory changes dynamically, reflecting changes in the environment and the internal course of computation. This is another factor for flexibility and context-sensibility.

1.2.2. Main Ideas of AMBR

AMBR is a dynamic emergent model built on the basis of DUAL. In its general form it is conceived as an integrated model of deductive, inductive, and analogical reasoning (Kokinov, 1988). All three kinds of reasoning are viewed as slightly different versions of a single uniform reasoning process. The overall approach is that reasoning establishes correspondences between two problems, schemes, situations, etc. and transfers some elements from one to the other, with due modification. The model explains deduction, induction, and analogy in terms of the relationships between the two descriptions that happen to be put in correspondence in each particular case. In this way, analogy can be viewed as the most general one, with deduction and generalization at the two extremities — where the *source* and the *target* are related in a special way, one of them being a specific instance of the other.

The research reported in this thesis concentrates on analogy-making. Therefore, AMBR is presented and discussed here as a model of analogy-making regardless of the fact that some of the considerations may have broader scope.

The models of analogy-making typically decompose it into separate 'stages' or 'phases'. For example, one possible decomposition includes: (*i*) representation of the target problem, (*i*) retrieval of a source analog from memory, (*ii*) mapping the two descriptions, (*iv*) transfer from the source to the target, (*v*) evaluation of the analogical inferences, and (*vi*) learning and generalization. Some researchers (e.g. Gentner, 1989) argue that the stages of analogy-making are relatively independent and thus are susceptible to piecemeal exploration. Others (e.g. Chalmers, French, & Hofstadter, 1992) oppose this view claiming that the process of analogy-making is inseparable in principle due to the high degree of interdependence among its components.

AMBR agrees with the second position. In this model the components of analogy-making are conceptualized as *subprocesses* that overlap in time and influence each other. The long-term goal of the AMBR project is to develop an *integrated* model of all these subprocesses on the uniform foundation of DUAL. At the time being, however, only two of them are implemented in detail. The version of AMBR that is reported in this thesis is an integrated model of analogical access and mapping. These two subprocesses and the computational mechanisms that implement them in the model are discussed in detail. Special emphasis is put on the ways that they can interact and on the dynamic emergent nature of the computations.

Another feature of AMBR that is central to this thesis is that the model uses *decentralized representations of situations*[1]. Each DUAL agent is relatively simple and cannot represent much. Therefore, a whole *coalition* of agents is needed for the representation of each episode, schema, or even proposition. AMBR coalitions are emergent and have fuzzy boundaries. The members of a given coalition can come in or out of it dynamically and to participate in it with varying intensity. There is no centralized data structure enumerating all agents belonging to a coalition. This allows for greater flexibility and integration of the various subprocesses of analogy-making. In particular, the mapping process can begin before the whole coalition is accessed from memory. The correspondences established by the active elements of a situation can then influence the activation of their coalition partners. As a consequence, the episodes that better map to the target tend to be preferentially accessed. This organization has a number of advantages which are discussed in the thesis.

The current version of the model relies on six computational mechanisms to carry out the tasks within its scope. These are: *(i) spreading activation, (ii) marker passing, (iii) constraint satisfaction, (iv) structure correspondence, (v) rating, and (vi) skolemization*. Each of them serves a concrete function in the model. Thus, spreading activation defines the working memory of the system, provides dynamic estimates of the relevance of each item, serves as a power supply for the symbolic processing, and underlies the relaxation of the network constructed by the constraint satisfaction mechanism. The marker passing is used for assessing semantic similarity, inheritance of properties, and carries out various information needed by other mechanisms. It also provides justifications for some of the hypotheses used by the constraint satisfaction mechanism. The latter underlies the process of mapping two structured descriptions and is a major instrument for achieving global consistency of the local activities in the model. The structure correspondence mechanism provides additional justifications for new hypotheses and dynamically modifies the topology of the constraint satisfaction network. The rating mechanism is responsible for promoting winner correspondences and for elimination of losers. Finally, skolemization uses general semantic knowledge to augment the description of a situation upon necessity.

[1] This term should not be confused with the *distributed representations* in neural networks.

1.3. The AMBR Family

As a matter of fact, *AMBR* is not the name of a single model but a generic name of a whole succession of models. Each of them builds upon the previous one and takes a few steps further in the long-term project. The major milestones along this road are the following:

Kokinov (1988) puts forth the conjecture that deduction, induction, and analogy can be conceptualized as different manifestations of a uniform reasoning process and gives it the name *Associative Memory-Based Reasoning.* An associative mechanism using spreading activation is proposed for the purposes of memory retrieval and estimation of relevance. The knowledge representation scheme is detailed in (Kokinov, 1989).

Kokinov (1994a) presents a much more elaborated version of AMBR. It will be denoted *AMBR1* when reference to the particular version is important. The constraint satisfaction mechanism is adopted for the purposes of the mapping process. The constraint satisfaction network (CSN) is constructed dynamically by the joint operation of the marker passing and structure correspondence mechanisms. The CSN is integrated with the main network of the model, which allows for interactions between the different subprocesses in analogy-making. AMBR1 uses centralized representation of situations — there is a frame containing a slot for each situation element.

Kokinov (1994a,b,c) also pulls out the architecture DUAL as something different from the specific model AMBR. The main architectural principles of DUAL are established: multi-agent approach, lack of central executive, hybridization at the micro-level, dynamic emergent computation, context sensitivity, etc. There is a computer implementation of the architecture and the model. It is used for simulation experiments.

In a M.Sc. thesis supervised by Boicho Kokinov, Petrov (1997) develops a detailed specification of DUAL and resolves some ambiguities of the original proposal. An exact and general mechanism for determining symbolic processor's speed on the basis of the activation level is specified. The connectionist aspect is identified as an energy supplier for the symbolic one. The notion of coalitions and the meso-level of description are explicated. A new portable implementation of the architecture is developed in Common Lisp with CLOS.

There are improvements of AMBR too. This version of the model (Petrov, 1997) will be denoted *AMBR2*. It introduces decentralized representations of episodes and designs the machinery for maintaining them. In particular, there are *secretaries* that register the hypotheses for each element and assist the construction of the constraint satisfaction network. The activation function of AMBR1 is changed with a better one. The model is fully implemented using the new implementation of DUAL. The knowledge base is expanded considerably and more extensive simulation experiments are performed.

The goals of the research reported in this thesis are to extend the model further. The new version will be denoted *AMBR3*. The process of mapping is to be completed until winner correspondences are identified, thus setting the stage for the transfer process. New mechanisms for using semantic knowledge for augmenting the description of episodes are to be designed and added to the model. The existing mechanisms and the computer implementation are to be improved and extended. The knowledge base is to be enlarged considerably, both by elaborating the existing descriptions and by adding new concepts and episodes. This larger knowledge base is to be used for new simulation experiments.

1.4. Outline of the Thesis

The structure and contents of this dissertation are summarized as follows:

Chapter II — *Background* — reviews some empirical data about analogy-making reported in the literature. It also presents briefly a selection of models and discusses their strengths and weaknesses.

Chapter III — *AMBR in Broad Strokes* — presents a concise and relatively self-contained description of the cognitive architecture DUAL and the model AMBR(3).

Chapter IV — *Knowledge Representation* — describes the knowledge representation scheme in detail. It contrasts the advantages and disadvantages of centralized and decentralized representation of situations. It also introduces the domain used for the simulation experiments.

Chapter V — *AMBR Mechanisms at Work* — provides a rigorous and systematic description of current AMBR mechanisms. The operation of the model is illustrated on a concrete example by showing how the mechanisms apply to a particular target problem. The chapter contains diagrams and transcripts from actual program runs.

Chapter VI — *Simulation Experiments* — reports results of simulation experiments involving ten target problems and more than 1200 runs of the program. These data are used to compare qualitatively the performance of AMBR with the regularities observed in human analogy-making.

Chapter VII — *Evolving AMBR: AMBR4 ?* — discusses the limitations of the current version and suggests ways in which the model could be extended in the future. In particular, it gives some ideas about modeling the subprocess of analogical transfer. On the other hand, it introduces a research project aimed at adding perceptual capabilities to DUAL and AMBR. It also presents the TEXTSCREEN micro-domain that can be used as a testbed for this project.

Chapter VIII — *Conclusion* — concludes this dissertation with a summary of its main points and a discussion of the contributions of this project.

Appendix A provides a sample of full-fledged agent definitions.

Appendix B gives simplified propositional representations of all episodes used in the experiments.

CHAPTER II

BACKGROUND

2.1. The Phenomenon of Analogy

Analogy has been the focus of much cognitive research. (For reviews see Gentner, 1989; Goshwami, 1992; Holyoak & Thagard, 1995; Keane, 1988). Still, there is no universally accepted definition. Michalski (1989) explained analogy as a superposition of induction and deduction. On the contrary, Holyoak and his collaborators (Gick & Holyoak, 1983; Holyoak & Thagard, 1995; Hummel & Holyoak, 1996) considered schema induction as a consequence of a successful analogy. There are, however, some ideas that have received widespread support. The following excerpt from Gentner (1989, p. 201) provides a starting point:

> *[A]nalogy is a mapping of knowledge from one domain (the base) into another (the target), which conveys that a system of relations that holds among the base objects also holds among the target objects. Thus, an analogy is a way of focusing on relational commonalties independently of the objects in which those relations are embedded.*

The importance of *structure*, or system of relations, has been demonstrated in many studies (Gentner & Landers, 1985; Gentner & Toupin, 1986; Clement & Gentner, 1991). Objects from the two situations are seen as counterparts when they fulfill similar roles in the respective relational structure. The degree of this structural overlap or *quasihomomorphism* (Holland et al., 1986; Holyoak & Thagard, 1989) determines to a large extent the soundness of an analogy. Central to the mapping process is the *principle of systematicity*: People prefer to map connected systems of relations governed by higher-order relations with inferential import, rather than isolated predicates (Gentner, 1983, 1989).

Thus, analogy making involves a *mapping process* that aligns structured descriptions of the two episodes and establishes a set of correspondences. There does not need to be any resemblance between individual elements of the two descriptions. Various theorists have suggested, however, and empirical evidence confirms, that object and predicate *semantic similarity* influence the mapping process, with high similarity leading to greater ease of mapping (Gentner & Toupin, 1986; Holyoak & Koh, 1987; Ross, 1987). This is especially clearly seen when objects and roles are 'cross mapped' (Gentner & Toupin, 1986; Ross, 1987, 1989).

Semantic similarity is much more important for *analog access* — the process of finding and accessing a suitable analog from long-term memory.

There is considerable evidence that this process relies more on semantic commonalties and less on structural commonalties than does mapping. For instance, people often fail to access potentially useful analogs if they have too little semantic overlap with the target problem (Gick & Holyoak, 1980, 1983; Ross, 1989). Spontaneous analogies from remote domains seem especially difficult (Seifert, McKoon, Abelson, & Ratcliff, 1986; Keane, 1987).

Holyoak and Thagard (1989, 1995) have proposed the *multiconstraint theory* in an attempt to summarize these experimental findings. According to this theory, analogy-making is governed by a combination of the following three constraints: (*i*) *structural consistency* — the pressure to identify and use an isomorphism between the descriptions of the two situations, (*ii*) *semantic similarity* — the pressure to map elements with some prior semantic similarity (e.g. joint membership in a semantic category), (*iii*) *pragmatic centrality* — the pressure to give preference to elements that are deemed especially important to goal attainment, and to try to maintain correspondences that can be presumed on the basis of prior knowledge. All three constraints are conceptualized as 'soft' — they do not operate as unviolable rules but rather as competing pressures (Hofstadter, 1984).

According to the multiconstraint theory, all three constraints play a role throughout the whole process of analogy making (Thagard, Holyoak, Nelson, & Gochfeld, 1990; Holyoak & Thagard, 1995). However, the constraints affect the different subprocesses to a different degree. Thus, semantic similarity seems to dominate the process of analog access but the other two constraints also play a role. Structural consistency exerts its major impact in the mapping process. Later stages of analogy-making are very sensitive to pragmatic pressures. More specifically, they are very important during *analogical inference* (or *transfer*) and *evaluation* — the processes of augmenting the target description and verifying the consistency of the inferences.

There are a number of other factors that also influence the course of analogy-making. Thus Keane (1994) has demonstrated an *order effect* on analogical mapping. Specifically, mapping is faster and more accurate when the order of presenting the target elements to the subject encourages a correct initial correspondence, which can then constrain subsequent mappings. Kokinov (1990, 1994a) provides evidence for *priming effects* on analogy-making (and problem solving in general). In these experiments, exposure and work on selected problems affected the performance on a later problem. The magnitude of this effect decreased with time. Other data by the same author and his collaborators (Kokinov & Yoveva, 1996; Kokinov, Hadjiilieva, & Yoveva, 1997) are indicative for *context effects* on problem solving.

All these empirical findings must be taken into account when building and evaluating cognitive models of analogy-making. The following section presents a brief overview of some of these models.

2.2. Models of Analogy-Making

Analogy-making is a very complex phenomenon and it is very difficult to embrace all of it at once. As a consequence, most models in the field could be characterized by the ancient maxim 'Divide and conquer!' That is, analogy-making is usually conceived of with reference to separate *stages* or *phases*. While this thesis advocates a different and more interactionist approach, this conceptualization is necessary for expository purposes. Thus, one possible division includes:

Perception (representation building) of the *target* problem;

Retrieval of an appropriate *analog* (or *base*) from long-term memory;

Mapping the base onto the target to find corresponding elements;

Transfer of knowledge from the base to the target.

Evaluation of the imported knowledge within the target framework;

Learning and generalizing the new experience for use in the future.

These stages are supposed to be relatively independent from one another and thus susceptible to piecemeal exploration. Different researchers focused their attention on different aspects of analogy making, each building a model that highlights some issues at the expense of others.

In contrast, the AMBR project advocates the strategy of integration. This does not mean, however, that we overlook the honored 'Divide and conquer!' On the contrary, we think it has given rise to quite a lot of knowledge which could (and should) serve as a springboard for any further research. Out of the many models reported in the literature (Anderson & Thompson, 1989; Carbonell, 1983; Evans, 1968; Hall, 1989; Holland, Holyoak, Nisbett, & Thagard, 1986; Kedar-Cabelli, 1988; Kolodner, 1993; Veloso, 1994), the following subsections discuss those which have directly influenced our work.

2.2.1. SME and MAC/FAC

The *Structure Mapping Engine* (Falkenhainer, Forbus & Gentner, 1986; Forbus & Oblinger, 1990; Forbus, Ferguson, & Gentner, 1994) is a computer implementation of Dedre Gentner's Structure Mapping Theory (1983). It is designed as a domain-independent analogical matcher. It takes two inputs: a base description and a target description. Both are in predicate calculus. The model does not use any semantic knowledge. It relies on purely syntactic operations to produce a set of correspondences. The underlying intuition is that syntax can capture meaning. Thus, entities (individual objects and constants) are mapped onto other entities, functions are mapped onto other functions, attributes (one-place predicates) are ignored, and relations are carried across. Special priority is given to higher order relations, thus conveying the systematicity principle postulated by the theory.

A serious limitation of the model is that it depends on *identicality* of predicate names. In the widespread version of the program, the matching algorithm requires that the predicates at the top of the relational structure are the same. These *local matches* are then recursively expanded to subordinate levels of the structure. Thus, the identicality restriction applies most forcefully precisely at the level which is most important according to the theory — the high order relations.

The shortcomings of the identicality restriction became apparent when SME was used as a building block for bigger systems (Falkenhainer, 1988, 1990a). Subsequent versions of the Mapping Engine (Falkenhainer, 1990a) have relaxed this restriction by applying *minimal ascension* through an `is-a` hierarchy and/or using *role* information (i.e. the dependencies which a given element satisfies). In our view, these are important improvements of the model. Still, most applications of SME reported in the literature (e.g. Forbus et al., 1994) use the 'default' rigid identicality.

More generally, the weakness of SME is that it relies very heavily on the *form* of the represented knowledge. For instance, the model differentiates strongly between *attributes* and *relations* — the former are ignored while the latter are the cornerstone of mapping. Yet, the only difference is that attributes are predicates with one argument while relations have two or more. Logically, each attribute can easily be transformed into an equivalent relation and vice versa, e.g. `hot(X) <--> temperature-of(X,high)`. The model thus depends on a putative re-representational module to accommodate this difficulties. However, there are no guidelines about how such module could be built and what criteria separate the facts that must be represented as attributes from those represented as relations. In practice, it is the human programmer who represents the situations in such a way that they be handled by the model.

Despite its limitations, the Structure Mapping Theory and SME are very influential pioneering work and their importance cannot be questioned. Gentner (1983) was the first to advocate that analogy depends on *structure* in a period when all kinds of similarities were explained by feature overlap (Tversky, 1977). Nowadays the importance of structure and systematicity is taken for granted. In general, the essence of a situation — the part that should be mapped — is a high-level coherent whole, not a collection of isolated low-level similarities.

Moreover, SME is one of the few analogy models that have been successfully used as building blocks for bigger systems (e.g. Falkenhainer, 1990b).

SME is a key component of the MAC/FAC model of similarity-based retrieval (Forbus, Gentner, & Law, 1994). The model explains retrieval in terms of a two-stage process. During the first stage (MAC), a cheap filter is used to weed out the majority of episodes in the long-term memory. This filter is based on dot products over *content vectors* — flat enumerations of the functors participating in the respective episode description. The second stage (FAC) then takes the output of MAC and subjects the candidates to more expensive processing. It uses SME (working in *literal similarity mode*) to assess the structural overlap between the candidate and the *probe*. Selection in both cases is based on comparing numerical scores against predefined thresholds.

This computational scheme has a number of engineering advantages. From a psychological point of view, however, it is questionable whether the cognitive system uses two different representations for each memory item. Moreover, each of the two representations seems too rigid and static. As argued later in this thesis (e.g. section 4.5), such centralized representations of situations cannot explain well the flexibility of analog access. It is not clear how MAC/FAC could account for the context and priming effects in analogy-making (Kokinov, 1994a).

2.2.2. IAM

The *Incremental Analogy Machine* (Keane & Brayshaw, 1988; Keane, Ledgeway, & Duff, 1994) is another model of the mapping stage in analogy-making. It starts by identifying a *seed group* in the base situation and picks up a *seed element* from that group. Like SME, IAM also relies on syntactic criteria for choosing the seeds. More concretely, the seed group is the group of predicates having the most higher order connectivity between its elements. The seed element is sought among the relations that take multiple arguments.

The main idea of the model is to establish a *seed match* relating the seed element to some target element and then use this match to incrementally grow a whole set of coherent matches. The *match rules* that carry out this task are sensitive to the structural, semantic, and pragmatic constraints on analogical mapping. The seed is used for disambiguation of ambivalent cases. All decisions are made sequentially, which requires backtracking when a commitment is inappropriate.

The backtracking algorithm allows IAM to work in limited working memory and produces order effects. Both properties are psychologically desirable (Keane et al, 1994). On the other hand, backtracking amounts to exhaustive search which casts doubts on IAM's abilities to scale up. It seems to us that mapping should be done as a combination of sequential and parallel processes. Most of the criticisms to SME apply to this model too.

2.2.3. ACME and ARCS

The *Analogical Constraint Mapping Engine* (Holyoak & Thagard, 1989) is another influential pioneering model. It has introduced the notion of *constraint satisfaction* to the analogy literature. The model uses a massively parallel connectionist algorithm to find a globally consistent set of correspondences between the two descriptions being mapped. The main idea is to build a *constraint satisfaction network* (CSN) with nodes representing hypothetical correspondences and positive and negative links enforcing the constraints. After a relaxation process, the network settles in a state representing a (potentially suboptimal) solution to the constraint satisfaction problem.

Like the models discussed above, ACME starts with a propositional description of the two situations. It then ‘translates’ these representations in connectionist terms using a centralized symbolic algorithm. There is no genuine interaction between the symbolic and connectionist components. Therefore, ACME can be considered as a precursor of hybrid models but in itself it does not constitute such a model.

A weakness of the model is that it constructs too many hypotheses — all elements from the target are paired with all elements from the base, with the restriction that objects must map to objects, one-place predicates to one-place predicates, etc. Most of these hypotheses are completely implausible and have to be suppressed later. In addition, the size of the resulting network is too demanding for the working memory of the system (Keane et al., 1994; Kokinov, 1994a; Hofstadter, 1995; Hummel & Holyoak, 1997).

ACME has other limitations that are discussed at various places in this thesis. Still, the idea of constraint satisfaction has been adopted in AMBR and is the foundation of one of its main mechanisms. The model has certainly influenced our work. There are many differences between the two models, however, as presented in detail in section 5.4.1.

A complementary model — ARCS (*Analog Retrieval by Constraint Satisfaction*) — applies the constraint satisfaction idea to the task of analog retrieval (Thagard, Holyoak, Nelson, & Gochfeld, 1990). The model first scans the whole episodic memory and looks for episodes having element(s) similar to some target element(s). It constructs a node for each tentative correspondence between a source episode and the target. More nodes hypothesize correspondences between individual propositions. ARCS then sets appropriate excitatory and inhibitory links and relies on the relaxation procedure to decide which analog best satisfies the constraints.

The model uses a semantic knowledge base for estimating the degree of semantic similarity between various entities. These estimates, however, are static. To illustrate, synonyms always count for 0.6, superordinates for 0.3, etc. As Kokinov (1992b, 1994a) has argued, however, this approach fails to reflect the dynamic and context-sensitive nature of human similarity judgements.

ARCS is broadly similar to MAC/FAC in that it uses a semantically based preliminary screening to identify candidate analogs and then applies the mapping machinery (although running in economical mode) to do more careful analysis. In effect, both models put a limited matcher inside the retrieval module. This creates redundancy when the retrieved episode is passed to the main mapping machine. We argue that there are better ways for integrating the two subprocesses in analogy-making.

2.2.4. LISA

Hummel & Holyoak (1997) propose an integrated model of analogical access and mapping called LISA (*Learning and Inference with Schemas and Analogies*). This is a structure-sensitive connectionist model and as such combines the advantages of the symbolic and subsymbolic approaches to cognitive modeling. The model represents propositions as distributed patterns of activation over units representing semantic primitives. The distributed representation brings flexibility and generalization capabilities. LISA uses *dynamic binding* to combine these representations into propositional structures. Thus it achieves the structure sensitivity that is crucial for analogy-making. The cost for this is that LISA must operate within inherent capacity limits given by the size of the *phase set* required for the dynamic binding. The authors of the model argue that similar limitations arise in human reasoning.

A key innovation is that LISA treats analogical mapping as a form of learning. The model establishes correspondences by gradually learning weights of the *mapping connections* between various elements. This allows the model to arrive at globally consistent mappings without the need of massively parallel constraint satisfaction. Moreover, analog access and mapping are integrated — they are treated as processes of guided pattern classification.

Due to these powerful and flexible mechanisms, LISA has been able to simulate various empirical phenomena with considerable success (Hummel & Holyoak, 1997). It advances the research on analogy in many ways. Still, the model is not without its problems.

One open question involves the size of the descriptions that the model can handle. Due to the distributed representations, quite complex machinery is required to maintain even a simple proposition. Things become even more complicated with hierachical structures. Although the representational scheme can in principle support descriptions of arbitrary complexity, most of the experiments reported so far deal with extremely simple cases — usually only one or two non-hierarchical propositions per episode. Given that the model depends on learning to accrue coherence by means of cycling through the propositions in the *driver*, it could face difficulties with bigger analogs. More concretely, when the length of the driver set increases there is a risk that later propositions undo the connections learned by earlier ones before the set is over and the initial propositions are reinforced.

Another shortcoming is that LISA uses what we call *centralized representations of situations*. Each situation could be in one of three modes (driver, recipient, or dormant) and all elements are simultaneously flipped from one mode into the other. This implies that LISA, like ARCS and MAC/FAC, treats the episodes in the long-term memory as units —they are either retrieved wholesale or not at all. As argued later in the thesis (subsection 4.5.1.) this approach has certain disadvantages.

2.2.5. Copycat and Tabletop

All models cited so far start from a hand-coded representation of the target problem 'implanted' into their working memory. In other words, they by-pass the task of building an appropriate representation of the target situation. There are strong arguments, however, that this latter perceptual aspect is crucial to analogy making (Chalmers et al., 1992). Without it, a large and important part of the over-all task is done by the coder of representations (i.e. the human programmer) instead of the model. All models discussed so far suffer from this limitation. The current version of AMBR makes no exception.

The intimate interplay between perception and analogy-making is the defining feature of the work of Douglas Hofstadter, Melanie Mitchell, and Robert French (Hofstadter, 1984, 1995; Mitchell, 1993; French, 1995). Their models — Copycat and Tabletop — constitute an important bridge over the gap that separated research on analogy-making from that on perception. Both models build their own descriptions of the problems they work with. For Copycat, the problems involve letter strings in a micro-domain; Tabletop deals with arrange-ments of objects on a table. The perceptual activity goes in parallel with the process of building correspondences between different elements of the situation. Thus the two processes can influence each other.

Fundamental to Copycat and Tabletop is the notion of *statistical emergence*: the program’s macroscopic behavior emerges from the interaction of a large number of low-level activities in which probabilistic decisions are made. There is no central executive that controls the operation the system. Instead, all processing is done by small entities called *codelets* that create, mediate, and respond to various *pressures*. This provides for great flexibility. The model presented here shares many of these ideas, although in a different form.

Both Copycat and Tabletop lack any episodic memory and do not address the problem of accessing a source analog from a large pool of past episodes. Thus, they leave an indispensable component of analogy-making out of their scope, just as models such as LISA and AMBR do with perception.

CHAPTER III

AMBR IN BROAD STROKES

The aim of this chapter is to present a concise and relatively self-contained description of the model AMBR and the architecture DUAL.

It is impossible to speak about AMBR without mastering DUAL terminology presented briefly below. DUAL is a general cognitive architecture which is the foundation of the model. It was proposed by Kokinov (1994a,b,c). By far, the most detailed description of the architecture can be found in (Petrov, 1997).

3.1. DUAL Cognitive Architecture

3.1.1. Main Ideas of DUAL

DUAL is a general-purpose cognitive architecture that comprises a unified description of mental representation, memory structures, and processing mechanisms. All these aspects of the architecture are organized around a small set of principles:

- **Hybridity.** DUAL is hybrid — it consists of complementary aspects. Moreover, it is hybrid in two ways. On one hand, it hinges upon the symbolic/connectionist distinction and the integration between the two. On the other, there is the declarative/procedural distinction and integration thereof. The four aspects derived from these two pairs are merged together coexist at every level of granularity in the architecture.

- **Emergent computation.** All processing and knowledge representation in the architecture is carried out by a cohort of small entities called *DUAL agents*. There is no central executive that controls the whole system, allocates resources, resolves conflicts, etc. Instead, there are small-scale DUAL agents and local interactions between them. The global behavior of the system emerges from the self-organizing pattern of these interactions.

- **Dynamics and context-sensitivity.** An important feature of DUAL's operation is that it is constantly changing in response to influences from the environment. This is possible due to the emergent nature of the processing and the lack of rigid centrally imposed algorithm.

3.1.2. Basic Terms and Levels of Description

The basic structural and functional unit of DUAL is the *DUAL agent*. Due to its importance, the DUAL agent has synonymous names: *micro-agent* or simply *agent*. Other names like *node* and *unit* are used to bring connotations from other theories, notably semantic networks and connectionism. It is important to note that throughout this thesis all the aforementioned terms refer to the same concept: the DUAL agent.

DUAL agents are the smallest building blocks of DUAL. Strictly speaking, in the architecture there is nothing but agents of various kinds. They interact with one another and thus combine into larger complexes. The *interactions* between agents are very important in DUAL because they keep the architecture together. They are often reified and called *links*, especially in contexts where the agents are called *nodes*.

A major architectural principle of DUAL is that larger structures emerge from the interaction of smaller ones. Thus, one can consider building blocks of increasing size. DUAL agents are at the beginning of this succession, followed by *coalitions*, and *formations*. There is no sharp boundary between the latter terms. As a rule of thumb, a coalition consists of a relatively small number (e.g. less than 20) of interconnected DUAL agents while formations are much bigger.

DUAL-based models are complex systems and must be analyzed at different levels of granularity. It is useful to distinguish the following three levels:

The microlevel (agent level) deals with DUAL agents. Relevant topics here include the internal structure of a agent, its information-processing abilities, the differences among agents of different types, etc.

The mesolevel (coalition level) deals with *coalitions* of DUAL agents. A coalition is a set of agents and a pattern of interactions among them. Coalitions have two very important properties: they are *emergent* and *dynamic*. Thus, the mesolevel deals with the interactions between the DUAL agents, the emergence of non-local phenomena out of local activities, the dynamics of the organization of DUAL agents into coalitions, etc.

The macrolevel (system level) deals with *formations* of DUAL agents and with whole *models*. Formations consist of big populations of agents and define the macroscopic structure of DUAL models. It is at this level where psychological concepts like *working memory*, *mapping*, and *analogy* start to play the lead. They help describe the overall behavior of DUAL-based models and to compare them with other cognitive models and with humans.

These three levels are not independent. In fact, it is impossible to tell them apart. To illustrate, any analysis of coalitions crucially depends on the properties of their individual members. Conversely, a large part of the description of a DUAL agent is devoted to its interactions with other agents. Changes made at one level propagate to neighboring levels, recursively. For expositional convenience, however, each level is discussed in a separate subsection below.

3.1.3. DUAL at the Microlevel

At this lowest level of granularity, the entity of main interest is the *DUAL agent*: its internal organization and operation, as well as its interactions with peers. Micro-agents are very important in DUAL because everything in the architecture ultimately boils down to them and their interactions. They are the 'building blocks' that compose larger structures — coalitions, formations, and systems.

A very fundamental property of DUAL agents is that they are *hybrid* entities. They bring together ideas that are usually considered in opposition. In DUAL, opposites are not treated as irreconcilable antagonists but rather as complementary aspects of a harmonious whole.

Moreover, DUAL agents are hybrid in two ways. On one hand, they have both connectionist and symbolic aspects; on the other, they serve both as representational and processing units. These two dimensions are orthogonal and thus form the four aspects shown in Table 3.1.

	Representation	**Processing**
Connectionist aspect	activation level	spreading activation
Symbolic aspect	symbolic structures	symbol manipulation

Table 3.1. Different aspects of Dual agents. (Compare with table 3.2.)

From the connectionist perspective, each DUAL agent is a unit in a neural network. It has an *activation level* attached to it and continuously spreads activation to other agents. From the perspective of the classical symbolic approach to cognitive modeling, DUAL agents are *symbols* — they stand for something else. Concretely, they represent various concepts, objects, relations, etc. In addition to this representational aspect there is a procedural one: agents manipulate on symbols. They can receive symbols from other agents, store them in local memories, transform them (thus producing new symbols) and so on.

DUAL agents interact intensively with one another. These interactions are very important because they are the fabric combining agents into larger complexes. DUAL interactions are relatively simple — they always involve only two micro-agents. One of them takes the initiative and either *reads* or *sends* some information to the other. Combined with the connectionist/symbolic distinction, this makes the four aspects summarized in Table 3.2.

	Type *read*	Type *send*
Connectionist aspect	activation level	spreading activation
Symbolic aspect	symbolic structures	symbolic exchange

Table 3.2. Different aspects of Dual interactions. (Compare with table 3.1.)

As mentioned earlier, it is often convenient to speak of *links* instead of interactions. In particular, we can speak of the attributes of a link, notably its *weight* and *label*. We can also discuss different types of links, draw diagrams with circles and arrows, etc. For instance, the phrase 'a population of interacting DUAL agents' translates into 'a network of interconnected nodes.' Throughout this thesis, both phrases mean the same thing.

3.1.3.1. Microframes

Each DUAL agent is a micro-frame. More precisely, it is the symbolic, representational aspect of a DUAL agent that is a microlevel frame. It has *slots* which in turn may have *facets*. Slots and facets are placeholders — they are filled up with *fillers*. Many fillers are references to other micro-frames and thus link the given DUAL agent to its peers. Consider the example on Figure 3.1.3.1. It shows the agent representing the concept `cup`. This frame has five slots, one of which has two facets.

```
cup
  :type     :concept
  :subc     (liquid-holder 1.0)
  :instance ((cup-1 0.3) (cup-5 0.2))
  :a-link   (saucer 0.5)
  :slot1
    :type    :relation
    :c-coref (cup-md-china 0.5)
```

Figure 31.3.1. An example of a micro-frame. See text for details.

There are two major kinds of slots: *general slots* and *frame-specific slots* (or *G-slots* and *S-slots* for short). The former have predefined semantics that is invariant for all micro-frames. There are different kinds of general slots depending on their *label*. For example, the slot `type` is filled by a *tag* denoting the type of the agent. The slot labeled `subc` denotes that the concept (or *class*) represented by this frame is a subclass of another concept. The slot `instance` is filled by (a list of) references to specific instances of the concept, etc. Note that each individual reference has a *weight*.

In contrast to general slots, *frame-specific slots* does not have invariant semantics. Thus, `slot1` in `frame1` may mean something very different from `slot1` in `frame2`. Frame-specific slots also have labels but these are only void identifiers serving to distinguish one anonymous slot from the other. S-slots (and only they) have *facets*. Facets can be conceived of as slots within slots. The same set of labels applies to both G-slots and facets.

3.1.3.2. Connectionist processing

DUAL employs a dual representation scheme. Facts are represented symbolically by micro-frames, while their *relevance* to the particular context is represented by connectionist means. Each DUAL agent (and hence each micro-frame) has an *activation level* attached to it. There is an automatic process of *spreading activation* that continuously restructures the knowledge base, making some nodes more accessible and others completely inaccessible. Thus, each DUAL agent can be viewed as a node in a connectionist network. It has an *input zone*, *activation function* and *output function* (Rumelhart & McClelland, 1986).

The output of a micro-agent influences the input zones of the agents that are interacting with it. The former acts as a sender in the interactions and the latter— as receivers. Using the node-and-link terminology, we can say that the node sends activation to its *neighbors* via links. The phrase 'there is a link from agent X to agent Y' means that agent X has a slot (or facet) filled up by a reference to Y. Each link has a weight that controls what portion of the sender's output is allotted to the particular receiver. Weights are usually normalized so that the sum of the weights of all outgoing links equals one.

The connectionist aspect of DUAL agents influences the symbolic one by determining the agent's *availability*. The notion of availability contributes very much to the hybrid nature of DUAL agents — it merges all four aspects from Table 3.1. Like the agent itself availability has declarative and procedural aspects:

Visibility. A DUAL system may consist of thousands of agents, each of which contains some particular small piece of knowledge. At any given moment, however, only a small fraction of this large knowledge base is visible. The symbolic processes that take place in the architecture can operate only on visible declarative elements. In addition, more active (and hence more visible) data elements are more attractive to the procedural machinery and thus are more likely to be taken into consideration.

Speed. The availability of a DUAL agent determines not only the visibility of its declarative aspect but also the speed of its procedural aspect. Very active agents work rapidly and thus determine system's overall line of computation, low-active ones work slowly, striving for more power, and inactive ones do not work at all. As the pattern of activation over the network of agents changes, the speed of individual processors changes accordingly, making the computation performed by DUAL-based models dynamic and context-dependent.

3.1.3.3. Symbolic processing

A great deal of the information processing in the architecture is symbol manipulation — deterministic construction, transformation, storage, and exchange of symbolic structures. We use the general term *symbolic processing* to refer to these activities. They are carried out by the *symbolic processors* of DUAL agents. Each agent has such processor. It also has *local memory* to support the processor's work. Part of the local memory is *permanent*; the rest is *volatile memory*. The former keeps the micro-frame with its slots, facets, and fillers. The latter consists of an *input zone* and a *buffer*. Thus, a typical symbolic transaction involves receiving a symbolic structure in the input zone, comparing it with old symbols stored in the buffer, and sending it with due modifications to some of the agents referenced in the micro-frame.

The *speed* of the symbolic processor depends on the connectionist activation level of the respective DUAL agent. The exact rule for determining the speed is based on an energetic analogy that is not discussed here. The interested reader is referred to section 3.2.5.3. of (Petrov, 1997). The main idea is that each symbolic operation requires the symbolic processor to do certain amount of *work* to carry it out. Doing work requires *energy* which is supplied to the symbolic processor by the connectionist aspect of the agent. The speed of the computation depends on the *power* (i.e. on the rate of energy supply and consumption) which in turn is linearly related to the activation level.

3.1.4. DUAL at the Mesolevel

DUAL agents are simple, they cannot do much in isolation. Therefore, they depend on one another and form coalitions. A *coalition* is a set of agents and a pattern of interactions among them. It is the entity of main interest at DUAL's mesolevel.

Coalitions have three very important properties: they are *decentralized, emergent,* and *dynamic*. None of these properties is present at the level of individual DUAL agents (the micro-level). There are 'tight' coalitions and 'loose' coalitions depending on the intensity of the interactions among their members. Tight coalitions are characterized by heavily weighted links and by intensive exchange of symbolic structures within the coalition. By contrast, loose coalitions are characterized by relatively weak links, often temporary ones, and by little or no symbolic interchange. There is a whole range between these two extremes. Moreover, coalitions do not have clear-cut boundaries. An agent can be involved in many of them at once, and to a different extent. Coalitions can 'recruit' new members, either permanently or temporarily. They may share members and thus 'flow' gradually from one into another.

Recall that DUAL agents can be seen as representational units — each of them stands for some single entity. By extension, coalitions of agents represent composite entities like propositions and situations. In the knowledge representation scheme adopted in DUAL even a simple proposition is represented by a number of agents. In such cases we say that there is a *meso-frame* that consists of several *micro-frames*.

Meso-frames can be quite complex, much more complex than any of the participating micro-agents). In this way, the expressive power of DUAL's representation scheme is not limited by the restriction that each agent can have only a few slots. Coalitions are limited only by the connectionist mechanism that controls the activation level of their individual members and hence indirectly restricts the number of agents that can be active at a time.

The connectionist mechanism is responsible also for determining which parts of a meso-frame are *relevant*. It is possible, especially in loose coalitions, that only part of their members are active enough to pass the threshold. Thus, only part of the declarative knowledge stored in the meso-frame will be visible. In other circumstances, another part of the knowledge will be brought to the fore. This makes DUAL meso-frames dynamic and context-dependent.

From a processing point of view, coalitions are important in DUAL because it is at their level where non-local computation emerges. Each individual DUAL agent contributes somehow to the collective performance by doing its small and local-specific job. Each agent runs at its own speed and in parallel with other agents. To succeed in its task, the agent usually depends on other members of its coalition. It cooperates with them and competes with the agents from other coalitions. The net result of all these activities is that the coalition as a whole accomplishes some computation that is beyond the reach of any individual agent. This accomplishment has resulted from an *emergent* process — it is not carried out by any centralized processor following a rigid routine.

It is important to note that the interaction pattern among the participants in a coalition changes dynamically over time. New agents join in, others stay back, fall out and so on. In the node-and-link terminology, the *topology* of the network changes via dynamic addition and/or removal of nodes and links. This *computational dynamics* plays a key role in the overall flexible and context-sensitive behavior of DUAL-based models (Kokinov et al, 1996).

3.1.5. DUAL at the Macrolevel

To summarize our presentation so far, at DUAL's microlevel we speak in terms of *DUAL agents*, at the mesolevel — of *coalitions*. Now, at the highest level of granularity we speak of *DUAL formations* and *systems*. A DUAL formation consists of a big population of agents — in the order of hundreds or thousands in number. A DUAL system consists of all agents that are present at a given instant of time, regardless of whether they are active or inactive, permanent or temporary, etc.

Most of the agents and, therefore, most of the knowledge and processing in the architecture reside in the *DUAL network*. Most agents in this network are permanent but additional temporary ones may be created during the computation and added to the total pool. Similarly, most links are permanent but additional temporary ones may be established. Thus, the topology of the network is relatively stable but not absolutely frozen.

The collection of all permanent nodes and links in the DUAL network comprise the *long-term memory* (LTM) of the architecture. It contains the system's knowledge (both declarative and procedural) about the world. LTM is very big — even for simple domains and situations one needs hundreds or thousands of agents.

In any given moment, however, only a small portion of this large formation is actually needed. DUAL provides special mechanisms, the most important of which is spreading activation, for effectively determining which agents (and coalitions) are *relevant* to the particular task and context. Recall that each agent has an activation level that is the system's estimate of its relevance. So, by definition the *working memory* (WM) of the architecture consists of the set of all agents whose activation level exceeds a certain threshold.

The working memory is the locus of almost all processing in DUAL and, therefore, we will consider it in more detail. An agent can enter WM in two ways: permanent agents enter it whenever they become active enough to pass the threshold; temporary agents must be explicitly created and linked to the network by a specialized *node constructor*. Agents stay in the working memory as long as their activation level is maintained above the threshold. When a permanent agent 'drops out' of WM, it returns back to *dormancy* and could enter WM again later. Temporary agents, however, have no second chance.

To sum up, the contents of the working memory may be expressed by the following formula:

WM = active portion of LTM + temporary agents .

The activation in the network originates from two special agents. The so-called *input node* models the influence of the environment[2]. The *goal node* is, in a very rudimentary sense, the medium of the ‘intentions’ of the system. The human user of the system attaches some agents to these nodes, thus allowing for the spread of activation to the DUAL network. The activation then propagates via the links and brings some agents from LTM to WM. There is a *decay* process which limits the total amount of activation and hence the size of the working memory.

[2] In future models it will be replaced by a whole formation — the *visual array*.

3.2. Associative Memory-Based Reasoning

This section begins the presentation of AMBR — a cognitive model built on the basis of the DUAL architecture. 'AMBR' is an acronym for 'Associative Memory-Based Reasoning' (Kokinov 1988, 1990, 1994a, 1997) and has been conceived as a model with very broad scope. Much of the work on it is still in progress. The current version of the model is numbered AMBR3. Previous versions were AMBR1 (Kokinov 1994a) and AMBR2 (Petrov, 1997). We fully recognize the fact that the model as it currently stands and is reported here is incomplete. Here and now AMBR3 is an integrated model of analogical access and mapping. We view this version only as an intermediate stage of a bigger project.

3.2.1. Main Ideas of AMBR

Since its initial conception (Kokinov, 1988) the AMBR model has advocated a set of ideas about human reasoning in general and analogy-making in particular. They have been distilled by Kokinov (1997) into the following three principles:

- **Integration.** The reasoning process cannot be partitioned into a sequence of independent stages performed by specialized module-like components. Rather, there are *subprocesses* that run together and each of them is potentially influenced by the rest. Each computational mechanism is responsible not only to produce its immediate result but also to create appropriate guiding pressures for other mechanisms. That is why AMBR is designed as an integrated model based on a parallel emergent architecture.

- **Unification.** Analogy is not a specific mode of reasoning. Rather, deduction, induction (generalization), and analogy are slightly different versions of the same uniform reasoning process. The same computational mechanisms are used in all cases — there is some sort of perceptual processing that builds internal representation of the problem being solved, there is some (sub)process that accesses relevant information from long-term memory, there is some (sub)process that tries to map the new problem to previous knowledge, etc. Deduction, induction, and analogy all fit into the same framework, the differences being in the outcome of the processing but not in the processing itself. Thus the term *deduction* applies to cases when the new problem happens to match with a general old schema, *induction* goes the other way around, and *analogy* applies when the two situations are at approximately equal level of abstraction. Conceptualized in this way deduction and induction are just two extremal (and hence very important) points on the analogy continuum. Therefore AMBR is designed as a general model of reasoning with emphasis on analogy-making.

- **Context-sensitivity.** Human reasoning is context-sensitive. Its outcome depends not only on the task and long-term memory knowledge but also on the environmental setting, recent activities of the reasoner, etc. AMBR is designed with the explicit aim to reflect this context-sensitivity of human thinking.

This thesis focuses on the first point from this list — integration. Deduction, induction, context and priming effects are treated elsewhere (e.g. Kokinov, 1990, 1992, 1994a; Kokinov & Yoveva, 1996). Hence it is warranted to explicate the principle of integration of sub processes in more detail.

As discussed in Chapter II, theories of analogy-making frequently partition the process into a sequence of stages (e.g. Gentner, 1989). The computational models that stem from these theories typically involve separate 'engines', each of which works on its own and dovetails with the next. The output from the retrieval module is fed into the mapping module, whose output in turn is fed to the transfer module, etc. This 'pipeline paradigm' is illustrated in Figure 3.2.1.1. Each module influences the next only via the data structures it passes to it. Occasionally a module could detect a failure and loop back to some earlier module.

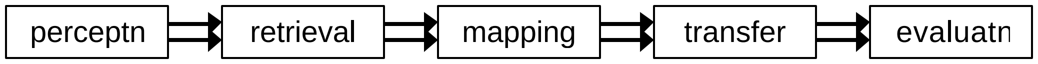


Figure 3.2.1.1. Schematic description of the 'pipeline paradigm' of analogy-making. The whole process is broken into a sequence of independent stages. They can interact only through the data structures (not shown in the figure) that each of them feeds to the next.

A problem with this approach is that it depends on the tacit assumption that all these stages are separable. While this assumption definitely merits careful consideration, it is questioned by a number of researchers (Chalmers, French & Hofstadter, 1992; Kokinov, 1994a; Hummel & Holyoak, 1997). AMBR follows a different track. It adopts an interactionist approach and treats analog access, mapping, transfer, etc. as parallel *subprocesses* rather than serial stages. These subprocesses are still ordered in time as suggested by the pipeline approach — for instance the mapping begins after the retrieval has begun. However, there is no requirement that a stage must end before the next one could begin. On the contrary, subprocesses overlap considerably and can interact. This leads to the cascade illustrated in Figure 3.2.1.2.

The interactionist approach seems problematic at first sight because each stage (or subprocess for that matter) depends on the result of the previous one. Indeed, how could the target problem be mapped to the source when it has not yet been even retrieved from memory?! It seems a logical necessity that the mapping comes after the retrieval. Similarly, the perceptual stage should come first, the transfer should follow the mapping, and so on.

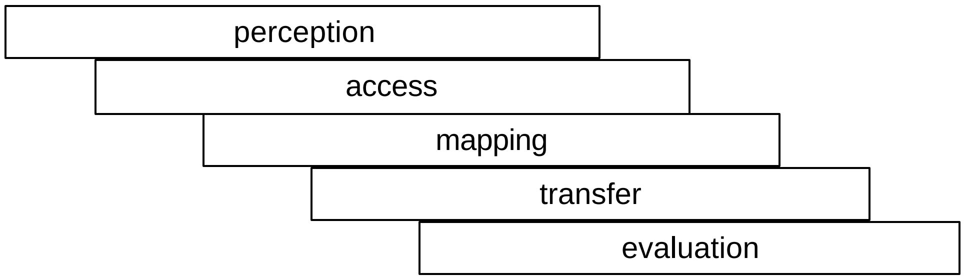


Figure 3.2.1.2. Schematic description of the interactionist paradigm of analogy-making. There are subprocesses that overlap in time and can influence each other. Compare with Figure 3.2.1.1.

AMBR overcomes this difficulty by representing information in smaller chunks. The model does not represent episodes as big units that are either manipulated wholesale or not at all[3]. Instead, it represents them as coalitions of small elements susceptible of piecemeal manipulation. This allows each sub-process to begin as soon as the previous one has produced some partial results.

More concretely, as soon as the perceptual mechanisms have built internal representations of a few elements of the target problem, the access subprocess starts looking in the long-term memory for information that relates to these new elements. The concepts, propositions, episodes, etc. that are accessed in this way can now influence the perception of the target. In addition, they trigger the mapping subprocess which starts constructing the first tentative correspondences. If a promising candidate correspondence emerges, it could influence both perception and access. Gradually, all subprocesses are at work and more and more is perceived, accessed, mapped, transferred, and so forth.

This is the upward motion of the 'wave' of the reasoning process. Sooner or later the wave goes down. A stable representation of the target problem has been built and the perceptual mechanisms go off stage. A source episode wins the competition with alternative episodes from memory and the access sub-process diminish. One by one, all subprocesses terminate roughly in the order they have started. In this way there is something that could be characterized roughly as a sequence of stages. However, the boundaries between the AMBR 'stages' are fuzzy and each one could in principle interact with everyone else.

Before closing this section we must make the following disclaimer. The current version of the model implements only two of the subprocesses drawn in Figure 3.2.1.2. — access and mapping. Thus AMBR3 in effect depends on the same assumption that was criticized above. It artificially separates these two subprocesses from the rest. We admit this is a major flaw of the current version. We hope, however, that the model is open-ended enough so that the missing components could be added without forcing radical changes in the existing ones. Chapter VII contains some preliminary efforts in this direction. Until then, the exposition concentrates on what is actually implemented and running.

[3] Therefore we prefer the term *analog access* to *retrieval*.

3.2.2. AMBR Protagonists: Concepts, Instances, and Hypotheses

As any model based on the DUAL architecture, AMBR consists of nothing but agents of various kinds. They represent the knowledge and do all information processing in the model. Therefore the natural way to begin the presentation of AMBR is to introduce the various types of agents used by it.

Each AMBR agent is a DUAL agent and as such has a micro-frame (see section 3.1.3.1). The micro-frame is a bundle of labeled slots one of which serves to designate the type of the agent. The label of this slot is `type` and it is filled by a list of tags such as `:concept`, `:instance`, `:hypothesis`, `:temporary`, etc. These tags are used in conjunction with one another to account for the variety of agents employed by the model. For example, the `type` slot of some agent can be filled by the list `(:temporary :instance :relation)` thus stating that the agent in question is a temporary agent representing an instance of some relation.

There are rules that restrict the combinations among different `type` tags. For instance all agents of `type :hypothesis` are also `:temporary`. Therefore, despite the big number of possible type combinations there are only three major types of AMBR agents: *concept-agent*, *instance-agent*, and *hypothesis-agent*. These major types have subdivisions as illustrated in Figure 3.2.2.1.

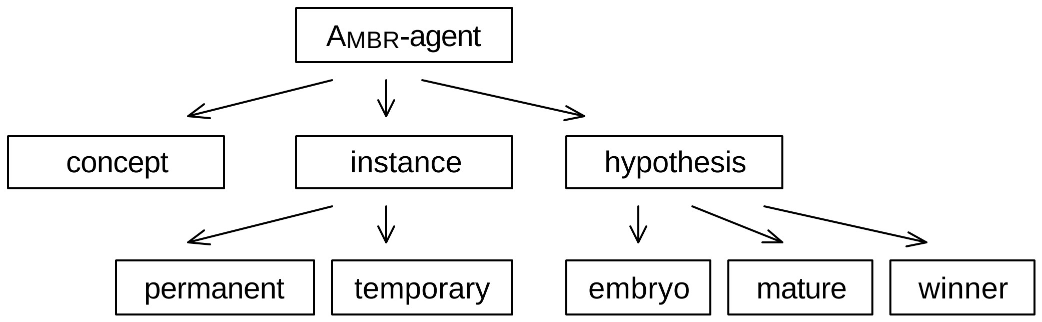


Figure 3.2.2.1. Main types of AMBR2 agents.

Concept-agents (or *concepts* for short) represent classes of entities. The taxonomy of classes is represented by `subc` and `superc` links between the concepts. Some concepts are classes of *objects* such as `teapot` and `liquid-holder` while others represent relations such as `temperature-of` and `cause`. A concept agent may also have references to some of its instances, to be associatively related (via `a-link`) to other concepts, etc. All concepts are permanent agents and form the backbone of AMBR's semantic memory.

Instance-agents (or *instances* for short) represent individual instances. Each instance agent has an `inst-of` slot filled by a reference to the concept agent representing the class of the instance (Figure 3.2.2.2). There are a several other slots with appropriate labels that relate the instance to other instances, concepts, or hypotheses. These links, like the taxonomy-oriented links mentioned above, are used by the mechanisms of the model for various purposes. Concept and instance-agents are sometimes collectively called *entity-agents*.

```
liquid-holder:
  :type     (:concept :object)
  :subc     container
  :superc   (teapot bottle cup)
  :a-link   liquid

teapot:
  :type     (:concept :object)
  :subc     liquid-holder
  :instance (teapot-1 tpot-73)
  :hypoth   teapot<->bottle

teapot-1:
  :type     (:instance :object)
  :inst-of  teapot
  :situation  sit-ABC
  :hypoth   (teapot-1<->bottle-3
             teapot-1<->bottle-4)
```

a) **b)**

Figure 3.2.2.2. Example of concept-agents, instance-agents, and some of the links between them. Each micro-frame has additional slots (not shown in the figure). All connectionist aspects are omitted. The corresponding node-and-link diagram is shown to the right.

Concepts and instances alike are characterized by one more tag in their `type` list — `:object`, `:relation`, or `:situation`. These tags are mutually exclusive. An `:object` tag means that the micro-frame represents some object or a class of objects. All agents in Figure 3.2.2.2. belong to this category. In contrast, the `:relation` tag is used to designate micro-frames that represent some relation. Such micro-frames usually have S-slots (see subsection 3.1.3.1) that represent the arguments of the relation. The AMBR representation scheme allows to represent both *specific propositions* such as `made-of(teapot-1, metal-1)` and *general propositions* such as `made-of(teapot,metal)`. The details of the knowledge representation scheme are given in the next chapter.

Situation-agents (or *situations* for short) are a special kind of instance-agents. They are distinguished by the tag `:situation` in their `type` slots. Contrary to the name of the tag, such agents do *not* represent whole situations. Rather, they represent the spatio-temporal contiguity of a coalition of instances. Most instance agents are *affiliated* to some situation. The medium of this affiliation is a slot labeled `:situation` filled by a reference to the respective situation-agent. In the example above, the agent `teapot-1` is affiliated to `sit-`

ABC. The other elements of this situation (both objects and propositions) will have the same reference in their respective slots. Thus the situation-agent that they all refer to represents the fact that all these instances have been perceived or inferred or remembered on the same occasion. On the other hand, there need not be any links from the situation agent to its elements. This is very important for the *decentralized representation* of situations used in AMBR. It is a whole coalition of instance-agents that represent a particular problem, scene, episode, etc. Each participant is linked to only a few other elements and no one 'knows' the situation as a whole.

The mechanisms for analogy-making try to establish *correspondences* between instances from different situations, between their respective concepts, and so on. These correspondences are represented in the model by correspondence-agents (not shown in Figure 3.2.2.1), the most important type of which are the so-called *hypothesis-agents* (or *hypotheses* for short). Each hypothesis represents a tentative correspondence between two entities based on one or more *justifications*. The justification of a hypothesis is the reason for its creation and maintenance by the system. In AMBR each hypothesis must have a justification. (This is one big difference between AMBR and ACME.) The justification is either semantic or structural, represented by a concept or hypothesis agent respectively.

The hypothesis-agents are organized in a *constraint satisfaction network (CSN)*. Coherent hypotheses are connected with excitatory links while contradictory ones inhibit each other. This is the main instrument for achieving global consistency based on local computations. This approach follows the ACME model of Holyoak and Thagard (1989) but there are important differences (discussed later in this thesis). Hypothesis agents have a special activation function (cf. section 3.1.3.2) that gives them competitive power in the CSN.

Hypothesis agents are constructed on the initiative of their justification. In the beginning of their life cycle they are created as *embryo hypotheses*. Those embryos that do not coincide with an existing hypothesis establish themselves and become *mature hypotheses*. They compete with the other hypotheses in the CSN and become either *winner* or *loser hypotheses*.

The presentation of the last few pages emphasized mostly on the symbolic declarative aspect of AMBR agents. Like all agents in the DUAL architecture, however, they are hybrid entities and have connectionist and procedural aspects as well (see Table 3.1). Different types of agents have different procedural knowledge and thus participate in the various computational mechanisms in the model.

3.2.3. AMBR Mechanisms

This subsection outlines the six basic mechanisms used in AMBR3: spreading activation, marker passing, constraint satisfaction, structure correspondence, rating, and skolemization. The presentation is intended to give a broad and relatively self-contained overview of these mechanisms and to show

how they fit together. Chapter V provides a rigorous and much more detailed coverage.

3.2.3.1. Spreading activation

As stated earlier, each AMBR agent has a connectionist aspect and acts as a unit in a neural network (subsection 3.1.3.2). It receives activation from the agents that interact with it, transforms this connectionist input according to its activation function, and in turn outputs activation to other agents along weighted links. Thus there is a pattern of activation over the whole population (or network) of agents. This activation originates from some special agents (see section 3.1.5) and then propagates the network. There is a decay factor and various thresholds that restrict the spread of activation.

This mechanism is of paramount importance in AMBR. It provides a dynamic estimate of the relevance of each individual agent. These estimates are then used by other mechanisms for various purposes. It defines the working memory of the model by bringing some agents above the threshold while keeping irrelevant ones away. This is the foundation of access subprocess in analogy-making. Spreading activation also underlies the relaxation of the constraint satisfaction network.

Activation plays another very important role in AMBR (and DUAL in general). It is the energy supply for the symbolic aspect. More active agents work faster and are more visible to other agents (section 3.1.3.3). Thus changes in the pattern of activation affect everything else in the model. This makes it dynamic, emergent, and context-sensitive.

3.2.3.2. Marker passing

Marker passing (MP) is the symbolic counterpart of the spreading activation. It has been developed within the semantic network tradition (Quillian, 1966; Fahlman, 1979; Charniak, 1983; Hendler, 1988, 1989). In its most basic form it is a tool for answering the question, “Given two nodes in the network, is there a path between them?”. The idea behind the marker passing is simple: the two *nodes of origin* are marked, they mark their neighbors, which in turn mark their neighbors and so forth.

AMBR markers originate in instance-agents and are then passed by concept-agents ‘upward’ in the class hierarchy. That is, markers can go only through links labeled `:inst-of` and `:subc`. For example, a marker can originate from `teapot-1` and then pass through `teapot`, `liquid-holder`, `container`, `artifact`, etc. Another marker starting from `bottle-7` could go through `bottle` and meet the first one in the concept-agent `liquid-holder`. The latter will detect this *marker intersection* and create a hypothesis that `teapot-1` corresponds to `bottle-7`. The concept node becomes the justification of the new hypothesis. In this way, the marker passing gives rise to semantically grounded hypotheses and triggers the constraint satisfaction mechanism.

The markers accumulate in the local buffers (see section 3.1.3.3) of concept-agents and provide a record of all instances of the particular class that are active at the moment. This information is then used by other mechanisms for various purposes.

3.2.3.3. Constraint satisfaction

The marker-passing and structure-correspondence mechanisms create hypotheses on the basis of local information only. The constraint satisfaction mechanism is responsible to achieve consistency at the level of whole coalitions. To that end, AMBR builds a *constraint satisfaction network (CSN)* with appropriate links between hypotheses. The pattern of activation in the CSN then gradually reaches a stable state in which a set of hypotheses emerge as winners while all others are suppressed.

In contrast with ACME (Holyoak & Thagard, 1989), the constraint satisfaction network in AMBR is tightly interconnected with the main network. This allows seamless integration with other mechanisms in the model. For example, suppose a particular hypothesis wins the competition and becomes highly active. Part of this activation spreads to the concept-agents involved in it. When the concepts become more active they process markers faster, which will tend to generate more hypotheses of the same kind. If the hypothesis is about instances, it will activate them and they in turn will support the other instances of the same coalition, etc.

Another important property of the constraint satisfaction network in AMBR is that it is built in a decentralized and incremental fashion. Individual hypotheses come one by one in the order of their creation. (Which, by the way, reflects the system's current estimates of the relevance of the elements involved.) This poses the question of how to avoid duplication of hypotheses and to establish the links needed for the relaxation algorithm. This is the responsibility of the hypotheses themselves aided by the so called *secretaries*.

Each instance- or concept-agent has a secretary associated with it. The secretary is not a separate agent; it is part of the entity-agent itself. The job of the secretary is to keep track of the hypotheses involving the agent in question. It records them in the `:hypoth` slot of the agent (cf. Figure 3.2.2.2) and handles *hypothesis registration requests*.

Whenever an embryo hypothesis is born it contacts the secretaries of its two elements and requests registration. The secretaries receive these requests, consult their records, and send *secretary answers* to the hypothesis. There are several kinds of answers but basically they all belong to one of the following two major types. If the new hypothesis is a duplicate of an existing one, it is advised to *resign* in its favor. The resigning hypothesis hands over its justification to the favorite and then fizzles out. In this way many hypotheses in the CSN have several justifications even though each of them is born with only one. The links to and from justifications are excitatory and connect the CSN with the main network.

The second major type of secretary answer is *establish*. It is sent to hypotheses that represent some novel correspondence. When the embryo hypothesis receives such answer it becomes mature and enters the competition with other mature hypotheses. The answer contains a list of the alternative hypotheses registered at the secretary. They are the rivals of the new one and it creates symmetrical inhibitory links with them. In this way each mature hypothesis becomes incorporated in the network. When it achieves this status it starts generating its own 'child' hypotheses via the structure correspondence mechanism.

3.2.3.4. Structure correspondence

The structure correspondence (SC) mechanism generates new hypotheses on the basis of existing ones. It is also responsible for the excitatory links between coherent hypotheses. Either way, it fosters the systematicity of the mapping that emerges out of the constraint satisfaction network (Gentner, 1983).

There are several types of structure correspondence in AMBR: bottom-up, top-down, weak, etc. They are explained in detail in Chapter V. This subsection only conveys the general idea by providing selected examples.

Suppose there is a mature hypothesis involving two instance agents, e.g. `teapot-1<->bottle-3`. The bottom-up SC will create a new embryo hypothesis at the level of concepts. Namely: `teapot<->bottle`. If the instances are affiliated to situations, the structure correspondence mechanism will construct an embryo hypothesis about them too, e.g. `sit-ABC<->sit-XYZ`. These new hypotheses are likely to coincide with ones created earlier by some other agent. In these cases the secretaries of, e.g., `teapot` and `sit-XYZ` will detect the duplication and the redundant hypotheses will be forced to resign in favor of the older ones. Still, excitatory links between `teapot-1<->bottle-3` and the respective concept- and situation-level hypotheses will be established. This creates the pressure that instances of the same concept and/or the same situation are mapped consistently to instances of the other concept/ situation and vice versa.

The top-down SC applies when there is a mature hypothesis involving propositions. For instance, suppose that the agent `made-of-1` represents the proposition that `teapot-1` is made of `metal-1`. Suppose further that `made-of-3` states that `bottle-3` is made of `glass-3`. Then the hypothesis `made-of-1<->made-of-3` will generate the hypotheses `teapot-1<->bottle-3` and `metal-1<->glass-3`. (It will also generate bottom-up hypotheses like `made-of<->made-of`, etc.)

The hypothesis `teapot-1<->bottle-3`, however, is probably constructed already by the marker passing mechanism (because both are liquid holders). The secretaries will then do their job and the SC-generated embryo will resign in favor of the MP-generated mature hypothesis. In the end the latter will have two justifications: semantic and structural. This gives it better competitive power in the CSN.

3.2.3.5. Rating and promotion

Another responsibility of the secretary is to rate the relative success of each hypothesis on its secretary list. It checks at regular intervals who is the current *leader* among the hypotheses. That is, which one has the greatest activation level. The secretary maintains *ratings* for each hypothesis. Ratings are numerical values indicating how long the particular hypothesis has led the competition. When a hypothesis maintains a leading status long enough, it is *promoted* into *winner*.

Thus, the rating mechanism promotes current leaders into final winners. This is done by a form of competitive learning algorithm. The secretary performs *rating surveys* at regular intervals. Each survey detects the leader and increases its rating at the expense of the ratings of its competitors. The magnitude of the change is proportional to the margin between the activation levels of the leader and the second best hypothesis. When a particular rating reaches some critical level, the rating mechanism triggers the *promotion mechanism* for the respective hypothesis.

In addition to promoting winners the rating mechanism also eliminates *losers*. When a particular rating drops too low and the activation level of the respective hypothesis is also low, the hypothesis is sent a *fizzle message* that causes it to die. Non-leader hypotheses that maintain a reasonably high activation level are kept as plausible alternatives to the leader. In this way the constraint satisfaction network is trimmed of very implausible hypotheses without ruling out any possibility a priory. This adds another dimension to the dynamics of the CSN — its topology changes both by adding and removing nodes and links.

Still another function of the rating mechanism is to trigger the *skolemization mechanism* upon necessity.

3.2.3.6. Skolemization

AMBR skolemization is a technique for augmenting the description of some particular episode on the basis of general semantic information. This is an advanced topic that is discussed in detail in Chapter V. This subsection provides an example that conveys the overall idea.

Suppose that the target situation contains a teapot and its material is explicitly represented: `teapot-1` is made of `metal-1`. Suppose further that `teapot-1` is mapped to `bottle-3` belonging to some other situation. The description of the latter, however, lacks explicit proposition about the material of `bottle-3`. Thus there is no counterpart of the target proposition `made-of(teapot-1, metal-1)`.

The semantic memory, however, contains a *general proposition* that bottles are (usually) made of glass. These general proposition is represented by an instance of the relation `made-of`. This instance is not affiliated to any situation (cf. section 3.2.2) and one of its arguments is a concept-agent. For example, it might be of the form `made-of(bottle, prototype-glass)`. This proposition is handled by AMBR mechanisms in the usual way — it emits a marker, that marker intersects in the concept-agent `made-of` with the marker emitted by the specific proposition in the target, the marker intersection gives rise to a hypothesis, etc. Suppose that this *general hypothesis* wins the competition in the constraint satisfaction network (for lack of a better alternative).

The rating mechanism detects that the leading hypothesis involves a general proposition and triggers skolemization. The latter will construct a *skolem proposition* that concretizes the general proposition. In the example above, the mechanism will create *skolem instances* of the concepts `made-of` and `glass`. No instance of `bottle` is needed because the recipient situation already has one as indicated by the marker from `bottle-3` stored in the local buffer of `bottle`. The final outcome of the skolemization is that the material of `bottle-3` is taken by default to be `sk-glass-3`, where `sk-glass-3` is a skolem instance of the concept `glass`. This new agent affiliates to the situation containing `bottle-3`. It then emits a marker, which will intersect in the concept `material` with the marker originating from `metal-1`. This will create the semantically-grounded hypothesis `metal-1<->sk-glass-3` which enters the competition with high chances of success as `teapot-1` is already mapped to `bottle-3`.

3.2.4. Overview of a Run

This final section pulls everything together and shows how the computational mechanisms described above can be applied to the task of analogy-making.

In the present version of AMBR, the work on a problem begins with a hand-coded representation of the target situation. Some of the agents that participate in the (decentralized) description of this situation are attached to the special nodes that are sources of activation in the model. The goal element(s) are attached to the goal node; some of the other elements are attached to the input node, thus mimicking the perceptual mechanism. The input list can also include elements that do not belong to the target situation, thus modeling the external context. It is possible that target elements are presented to the system not simultaneously but incrementally, giving rise to various order effects.

Once the target elements are connected to the source nodes, the associative mechanism begins to operate. The activation spreads through the long-term memory and brings relevant conceptual and episodic information to working memory. Shortly after, the marker-passing mechanism joins in, as instance-agents emit markers upon entering the WM. The markers begin propagating the active portion of the network.

Marker intersections provoke the construction of hypothesis-agents, thus triggering the constraint-satisfaction mechanism. After consulting the secretaries, the hypotheses initiate the structure-correspondence mechanism.

The secretaries register more and more hypotheses and rate their relative success.

Gradually, a number of agents enter the working memory. The activation does not spread unrestricted, however, and the intensity of memory access declines as the decay of activation prevents the nodes that are too far away from passing the threshold. Usually, two or three situations are retrieved in full and a few others only partially. These are the candidates for base analogs. In addition, the relevant concept-agents are also active and ready to guide the mapping.

The associative mechanism never stops completely because agents occasionally get in or fall out of the working memory. Moreover, the associative mechanism is responsible for controlling the speed of the symbolic aspect as well as for settling the constraint satisfaction network.

Meanwhile, the marker-passing mechanism has generated several hypotheses. In turn, they have created additional hypotheses via the structure-correspondence mechanism. The CSN has thus become fairly elaborate and winning correspondences begin to emerge. The hypotheses standing for such correspondences are promoted to winners. This makes them even more active and provides strong support for the respective entities in the main network. In this way, the base situation that best matches the target is fully and unambiguously accessed. All its elements enter working memory. The skolemization mechanism adds even more elements if such are needed to better match the target.

Sooner or later all secretaries of the target promote their winners. The mapping constructed by the model can be read from the set of winner hypotheses. (In fact, the system maintains a 'working answer' throughout the whole run. It is often unnecessary to wait for the end.) The mechanisms for transfer should have been triggered at that time. They are not yet implemented in the current version of the model, however.

It should be emphasized that everything described so far happens as a result of a dynamic emergent process. There is no central executive that controls the operation of the system. Instead, a multitude of micro-agents interact with their immediate neighbors and their local activities give rise to macroscopic phenomena that an external observer could interpret as analog access, mapping, etc.

CHAPTER IV

KNOWLEDGE REPRESENTATION

This chapter is devoted to knowledge representation in AMBR. It shows how the DUAL representation scheme (Kokinov, 1988, 1994a; Petrov 1997) is actually put to work in the model.

4.1. Domain

This section introduces the domain used for the simulation experiments reported in this thesis. It should be noted that AMBR mechanisms do not depend on this particular domain. It is simply a convenient testbed for the model.

The domain involves simple everyday tasks in a kitchen. It deals with concepts such as `water`, `high-temperature`, `baking-dish`, and `food`. Typical situations include heating and cooling liquids, boiling eggs, etc. For example, the knowledge base contains the following episode:

> *There is a teapot and some water in it. The teapot is made of metal and its color is black. There is also a hot-plate. The teapot is on the plate. The temperature of the plate is high.*
>
> *The goal is that the temperature of the water is high.*
>
> *The outcome of this arrangement is that the temperature of the teapot is high because it is on the hot plate. In turn, this causes the temperature of the water to be high.*

Equipped with episodes of this kind, AMBR is then presented with situations in which some of the objects necessary for achieving the goal are missing. For instance, the goal is to heat some milk in a teapot but no heating source is mentioned. Another kind of problem is to give all the objects in place and then ask what will happen, etc.

We admit that these problems are very modest by human standards. AMBR does not attempt to solve the radiation problem or to understand the Rutherford atom. Indeed, its abilities are even more modest than suggested by the description above. Despite the appearance, AMBR has no idea about what ‘real world’ water actually looks like. It has so little ‘knowledge’ that in fact it works in a micro-domain and this should be taken into account when evaluating its performance (Chalmers, French & Hofstadter, 1992). We argue, however, that reliance on such micro-domains is methodologically correct and even unavoidable for the current state of the art.

4.2. Desiderata

As simple as it is, AMBR's domain reveals a number of requirements that the knowledge representation scheme must meet to allow successful problem solving. We believe that the same requirements hold for any domain and become increasingly important for more complex ones.

4.2.1. Rich Descriptions of Episodes

The episodes stored in the long-term memory should be described in enough detail. This is not crucial for the mapping process but is absolutely necessary for transfer and evaluation. In particular, the causal structure should be quite elaborated. In the example above, the hot plate is important for heating the water but the color of the teapot is not. Without enough causal information the model could assume that in order to heat milk it should put it in a black teapot.

There is an additional complication — the rich representation of the source analog hinders its mapping to the target problem. The description of the latter is normally quite incomplete and, therefore, there are many elements in the source that do not have any counterpart in the target. Paradoxically, if there are two potential source analogs in the LTM, the one with sketchier description will map better to the target even though it may well be less useful for solving the problem. In the extreme case, a source analog that has absolutely nothing more than the target will achieve perfect match but zero utility.

One way around this obstacle is to partition the source descriptions into initial conditions, goals, solutions, etc. The target problem could then be mapped selectively to the appropriate sections of the base (e.g. Holyoak & Thagard, 1989). This approach, however, seems too rigid as it precludes any possibility of mapping elements from different compartments. Human problem solving does not observe such boundaries. For instance, the goal of one situation could map to some unintended side effect in another.

4.2.2. Semantic Knowledge

Most analogy models either do not use semantic information at all (Falkenhainer, Forbus & Gentner, 1986; Keane & Brayshaw, 1988) or use it solely for estimating semantic similarity (Holyoak & Thagard, 1989; Kokinov, 1994a; Hummel & Holyoak, 1997). It is clear, however, that human problem solving recruits much more semantic knowledge than that[4]. Even in our tiny domain the general fact that plates are heat sources and as such are used to heat things is of obvious importance when asking how to heat water. Still, such knowledge goes unused if the model deals exclusively with finding correspondences between two episodes.

[4] Moreover, research on memory suggests that remembering old episodes is often a matter of reconstruction rather than rote retrieval.

One challenge for the research community is to design mechanisms for using semantic knowledge in analogy-making. The skolemization mechanism proposed here is a first step in this direction. Note that one of the implications of this approach is that it blurs the boundary between deductive and analogical reasoning (which, according to our views, is quite fuzzy anyway).

4.2.3. Flexibility and Re-Representation

A third desideratum closely related to the first two is that episode representations should be flexible. Facts that follow from general rules need not be stored explicitly in each episode. They may be omitted from the representation and added again later upon necessity. The model should be able to switch between alternative representations such as `hot(X)` vs. `temperature-of(X,high-T)` or `left-of(X,Y)` vs. `right-of(Y,X)`.

There is a tacit assumption in analogy research about the asymmetry between source and target situations. It is quite often taken for granted that the target must conform to the source while the latter remains static. In our view the process of analogy-making consists of bringing both situations closer to each other by modifying either one when appropriate. In this way one and the same base episode can map to various targets and each mapping entails reconceptualization of the base.

4.3. Representation of Concepts and Instances

With full awareness that AMBR agents are nothing but ungrounded symbols (Harnad, 1990), we follow the common AI practice to use mnemonic names like `milk`, `taste-of`, and `cause`. Those names are irrelevant for the model itself; the program would work just as well (or as bad) had the agents been named `ag001`, `ag002`, etc.[5]

As introduced in Chapter III, AMBR uses *concept-agents* to represent classes of entities in the micro-domain and *instance-agents* to represent individual instances. The taxonomy of classes is represented by `subc` and `superc` links between concept-agents. Each class may be linked to zero, one, or more super- or sub-classes, different links possibly having different weights. Similar links — `inst-of` and `instance` — relate instance-agents to concept-agents. Figure 4.3.1. illustrates.

Some instance-agents are temporary. They does not belong to the long-term memory of the system. They are constructed by some inference or (putative) perceptual mechanism and 'live' as long as they stay in the working memory (section 3.1.5). In the current version of AMBR, temporary instance-agents are used to represent the target situation and for Skolem instances. In contrast, permanent instance- agents are used for all LTM episodes. Concept-agents are always permanent.

5 Indeed, the first version of AMBR (Kokinov 1994a) used such void names. It was very instructive from a philosophical point of view as it laid bare how little 'knowledge' the program actually had. It was not very practical, however, because it hindered enormously the process of developing, tuning, and documenting the model.

```
liquid-holder:
  :type     (:concept :object)
  :subc     (container 1.0)
  :superc   ((teapot 0.3)
             (bottle 0.3)
             (cup    0.2) )
  :a-link   (liquid 0.5)
teapot:
  :type     (:concept :object)
  :subc     ((liquid-holder 0.8)
             (kitchen-equipment 0.2) )
  :instance ((teapot-1 0.2)
             (tpot-73  0.1) )
teapot-1:
  :type     (:instance :object)
  :inst-of  (teapot 1.0)
agent007:
  :type     (:temporary :instance :object)
  :inst-of  (teapot 1.0)
```

Figure 4.3.1. Example of concept-agents, instance-agents, and some of the links between them. Each micro-frame has additional slots (not shown in the figure). Note that each reference has a weight used for spreading activation. Compare with Fig. 3.2.2.2.

'Top-down' links from concepts to instances deserve special attention. These `instance` links play a key role for analog access in AMBR. As discussed in earlier publications (Petrov, 1997, section 4.1.3), however, it is both psychologically implausible and computationally disadvantageous to maintain links to *all* instances of a given concept. Instead, there are such links to only *some* of them. This 'privileged set' varies as a function of time (though much more slowly compared to other events in the model). Thus, at any given moment each concept supports only a few of the vast number of instances potentially available in the episodic memory.

The exact mechanisms for this are open for discussion and are not implemented in the current version of AMBR. The main idea is to give priority to recently used instances, prototypes, or other salient agents without excluding anyone *a priory*. At present, there is an implemented tool for generating (static) variants of the knowledge base. The simulation experiments reported in this thesis are based on hundreds of such variants of the same 'core' knowledge base. The set of `instance` links of each concept is generated by random sampling. The instance agents have unequal odds of including in the sample thus approximating the mechanism suggested above.

Some instance-agents are distinguished by the tag `:prototype` in their `type` slots. These *prototype instances* are used as arguments in the so-called general propositions (see below).

4.4. Representation of Propositions

Individual AMBR agents are small and their micro-frames cannot represent much. Therefore, even relatively simple units of the representation such as propositions need be represented by a *coalition* of agents (section 3.1.4.). In the case of propositions, such coalitions are small and very tight.

```
color-of
  :type (:concept :relation)
  :subc physprop-rel
  :slot1
    :subc     (physprop-rel . :slot1)
    :c-coref  object
  :slot2
    :subc     (physprop-rel . :slot2)
    :c-coref  color
color-of-1
  :type (:instance :relation)
  :inst-of   color-of
  :slot1
    :inst-of  (color-of . :slot1)
    :c-coref  teapot-1
  :slot2
    :inst-of  (color-of . :slot2)
    :c-coref  green-1
teapot-1
  :type (:instance :object)
  :inst-of  teapot
  :c-coref  (color-of-1 . :slot1)
green-1
  :type (:instance :object)
  :inst-of  green
  :c-coref  (color-of-1 . :slot2)
```

Figure 4.4.1. A coalition of four micro-frames representing the proposition `color-of-1(teapot-1, green-1)`. The corresponding node-and-link diagram is shown to the right. All connectionist aspects are omitted.

There is an agent that represents the *head* of the proposition. In Figure 4.4.1., this is the micro-agent `color-of-1`. It has the tags `:instance` and `:relation` in its `type` slot and is an instance of the concept `color-of`. The arguments of the relation are represented by S-slots in the heading micro-frame. Each S-slot has several facets (see subsection 3.1.3.1).

The arguments (or roles) of the relation are bound to the actual entities involved in the particular instance of that relation by *conceptual coreferences* (or `c-coref`'s for short). In Figure 4.4.1., the first S-slot of the micro-frame `color-of-1` has a facet labeled `c-coref` and this facet is filled by a reference to the agent `teapot-1`. In a nutshell, the existence of `c-coref` links between two micro-frames (or their slots) mean that the two frames represent two complementary aspects of the same entity. In our example, these links represent the fact that `teapot-1` and the first argument of `color-of-1` are one and the same thing. Similarly, the second argument of the relation is bound to the particular shade of green that happens to be the color of `teapot-1`.

Note that S-slot labels (`slot1`, `slot2`, etc.) in any proposition are absolutely arbitrary and by no means serve to define the arguments within the relation. In the example above, it is *not* crucial that `slot1` is the object and `slot2` the color. The slots in the instance-agent `color-of-1` could just as well be labeled `slot5` and `slot6` (or even `slot2` for the object and `slot1` for the color). Moreover, two instances of the same relation could use entirely different labels. Each S-slot has a `inst-of` or `subc` facet that points to the corresponding slot in the parent concept. This gives distinct advantages over a positional notation (in which interpretation of arguments depends on their order in the proposition). AMBR propositions effectively have a *set* of arguments, not an ordered tuple. Thus it is possible that two slots in a 'child' inherit from the same slot in the 'parent', or that some parent slot is left unused, etc. As we shall see, this provides for great flexibility in analogical mapping. It is possible to map propositions with different number of arguments, to map two arguments from one proposition to a single argument in another, etc.

The proposition illustrated in Figure 4.4.1. is a *specific proposition* — it relates two specific instance agents. Such propositions typically encode episodic information. In addition to them, AMBR's knowledge base contains *general propositions* encoding semantic information. Their arguments are concepts or prototype instances. For example, the proposition shown in Figure 4.4.2. represents the general fact that each snowdrop is white. The skolemization mechanism uses such general propositions to create specific *Skolem propositions* about particular exemplars of the general class (see section 5.7).

```
each-snowdrop-is-white
  :type(:instance :relation)
  :inst-of   color-of
  :slot1
    :SUBC     (color-of . :slot1)
    :c-coref  snowdrop
  :slot2
    :INST-OF  (color-of . :slot2)
    :c-coref  prototypical-snowdrop-white
snowdrop
  :type(:CONCEPT :object)
  :subc flower
  :c-coref  (each-snowdrop-is-white . :slot1)
prototypical-snowdrop-white
  :type(:PROTOTYPE :instance :object)
  :inst-of  white
  :c-coref  (each-snowdrop-is-white . :slot2)
```

Figure 4.4.2. Example of a general proposition. Note the use of tags in the `type` slots and `subc` vs. `inst-of` facets. Compare with the specific proposition shown in Fig. 4.4.1

4.5. Representation of Situations

This subsection compares two alternative strategies for representing situations (problems, episodes) for the purposes of analogy-making. It considers their advantages and disadvantages and presents the approach adopted in AMBR3.

4.5.1. Centralized Representation: Pros and Cons

We speak of *centralized representation* of a situation when there is an explicit data structure enumerating all elements belonging to it. The data structure may be a list, frame, or something else. The criterion is whether the system has a means to go through all members of the situation and only those members.

Centralized representations simplify the mechanisms of the model. Each situation has distinct identity. It can be operated as a unit. It can be put in explicit competition with other situations. It can be checked for members with a given property, etc.

These computational advantages explain the widespread use of centralized representations in analogy models. Thus, the MAC/FAC model (Forbus, Gentner & Law, 1994) maintains two data structures for each episode (*memory item* in original terms). *Content vectors* are used for cheap preliminary screening based on dot products. The SME analogical matcher (Falkenhainer, Forbus & Gentner, 1986) then takes s*tructured descriptions* to produce a numerical score for each item that has passed the first stage. ACME (Holyoak & Thagard, 1989) and ARCS rely on predicate calculus descriptions to construct hypotheses for a constraint satisfaction network. The Incremental Analogy Machine (Keane & Brayshaw, 1988) starts with a predicate calculus description and looks for the group of predicates that have the most higher-order connectivity between its elements. It then picks up a *seed* from this *seed group* and goes to the description of the other situation searching for a *seed match*, etc.

LISA (Hummel & Holyoak, 1997) is a very interesting case. It employs distributed representation of concepts, localist representation of propositions (*P* and *SP units* in LISA terms), and centralized representation of situations (or *analogs*). Each situation can be in one of three modes: *driver*, *recipient*, or *dormant*. The propositions in the driver are selected to become active in the *phase set* one at a time according to a fixed schedule specified by the human user. Recipient and dormant propositions respond to the patterns generated by the driver. Only recipient units, however, participate in analogical mapping. Thus in order to enter the mapping, an analog from LTM must first switch from dormant to recipient mode. This transition occurs in a stop-and-go fashion — all members of the situation are simultaneously flipped from one mode into the other. In the current implementation of the model this is done by the human user (Hummel, personal communication, January 1998).

The first version of AMBR (Kokinov, 1994a) also used centralized representations of situations. There was a micro-frame standing for each situation as a whole. This micro-frame was called *head* and brought together all agents that built up the representation of the situation. There was one S-slot for each element — object or relation. The head was linked to *all* elements and some elements were linked back to the head, thus creating a network like the one schematized in Figure 4.5.1. In addition to the 'vertical' links between the head and its elements, there were many 'horizontal' links between the elements themselves (not shown in the figure).

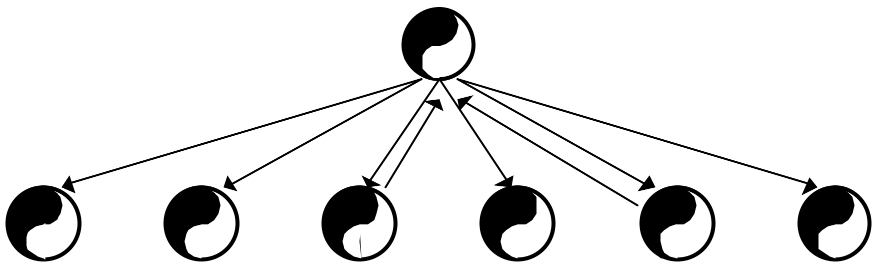


Figure 4.5.1. Schematic outline of centralized representation of a situation as used in the first version of AMBR (Kokinov, 1994a). There is one *head* connected to all elements of the situation. Cf. figure 4.5.2.

This representational decision provided ready solutions to many issues faced by the model. It was clear who was 'responsible' for the situation. To begin working on a problem, for example, it was sufficient to put the head on the goal list. To decide which base analog 'won', it was sufficient to compare the activation levels of the heads. The task of mapping one problem to another was reduced to a task of establishing slot-to-slot correspondences between two micro-frames. After the correspondences had been found, it was clear which elements of the source were unmapped and thus were potential candidates for transfer, etc.

However, each of these advantages can be viewed as a disadvantage in the same time. From a psychological point of view, it is controversial whether each episode in the LTM has such distinct and clear-cut identity. To illustrate, it is comfortable to suppose that *Hamlet* and *Westside Story* are salient and well-defined chunks for many people. It is acceptable to suppose that the radiation problem (Dunker, 1945) is a sufficiently self-contained chunk for some psychologists and a few of their subjects (Gick & Holyoak, 1983). The problems used to test AMBR, however, deal with mundane episodes such as boiling a pot of water. Most of the situations fall into this final category and it is far from clear whether they are represented in such neat and orderly manner. This conjecture contradicts with the numerous cases of omission, intrusion, blending, interference, etc. in human memory recall.

Second, centralized representations tend to be too static and inflexible as it is difficult to add or remove elements dynamically. They are also 'flat' in the sense that all members participate on an equal footing. Special measures (e.g. differentiated link weights) are needed to make some elements more salient or pragmatically more important than others. Even in these cases, however, an item is either *always* in the situation or not at all. With bigger situations (cf. section 4.2.1) this could lead to a mild version of the frame problem (McCarthy & Hayes, 1969).

In addition to these considerations which in our view should be taken into account by all cognitive models, there are other problems with centralized representations that are particular to AMBR. The slots in the heading micro-frame become too many. Even the simple situations used in the simulation experiments so far require at least a dozen S-slots in the head. For realistic situations this number would be in the order of one hundred. When the number of slots is that big, however, the frame problem appears again — it is necessary to specify which of the many elements are relevant to the task at hand. It also violates the architectural requirement that DUAL agents should have only a few slots. Worst of all, the fan-out effect makes the connectionist mechanism very inefficient. Even when the head is very active it fails to activate its children because the weight of each individual link is very small (due to normalization). When (and if) this finally happens, there comes another problem — the coalition becomes so stable that it never leaves the working memory because the reverberation is stronger than the decay.

In response to these problems, the newer versions of AMBR (starting with AMBR2 (Petrov, 1997)) have abandoned the centralized representation used by their predecessor. The shift to decentralized representations poses problems in its own right but also offers a number of substantial improvements.

4.5.2. Decentralized Representation: Pros and Cons

We speak of *decentralized representation* of a situation when there is no explicit data structure enumerating all elements belonging to it. This term should not be confused with the *distributed* representations prevalent in connectionist research. It is possible (like in AMBR) to have localist representation of individual elements and decentralized representation of situations.

The main idea of decentralized representations is to represent the situation as a coalition of micro-frames without designating any of them as a center (Figure 4.5.2). It is possible, though not required, that some (salient) coalitions have a head, but even in these cases the head is *primus inter parens.* It is not special in any way and do not have access to all elements of the situation.

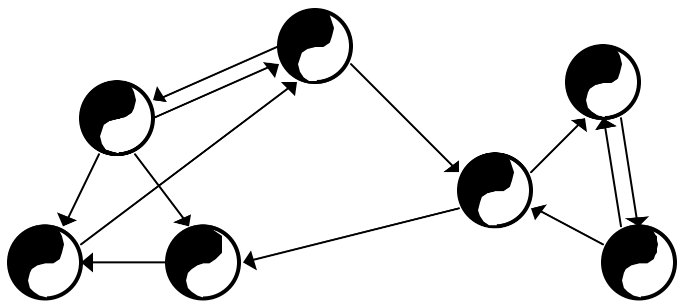


Figure 4.5.2. Schematic outline of decentralized representation of a situation. There are many interconnected agents, none of which is in privileged position with respect to the others. Compare with figure 4.5.1.

With decentralized representations, the principal unit of analysis is the *coalition* (or *meso-frame*, see subsection 3.1.4). This is an emergent entity which allows for great flexibility. It is easy to add new elements as they need not be 'registered' anywhere. Thus, it would be easier to design a perceptual mechanism that incrementally builds such representations. As there are no fixed and predefined representation rules, each particular situation can be described in a way that is most suitable for it. Each micro-frame (including the head, if any) can have only a few slots and yet it is possible to represent big situations.

Decentralized representations can be rich and detailed enough to support analogical transfer and evaluation. Thus they meet the criterion presented in section 4.2.1. In the same time, they can map successfully to impoverished and incomplete targets. This can be achieved when the mechanisms for access and mapping cooperate in the following way: The target problem acts as a driver and activates selected elements of several situations in the long-term memory. The full description of each of these potential source analogs can be very rich. At first, however, only a small fraction of the coalition members enter the working memory. These are the elements that are semantically similar to the target and their immediate entourage. Thus the working memory contains descriptions of comparable complexity — the impoverished target and two or three *partially activated* sources. This commensurability is favorable for the mapping mechanisms and they start building correspondences. If a source analog matches the target well, its elements receive additional support and become more active. In turn, this gives them resources to bring more coalition partners into the working memory. The analog that has emerged as winner unfolds its rich representation. Now the working memory contains an impoverished target and an elaborate source. The task for the mapping mechanisms thus becomes more difficult but they are aided by the initial correspondences that have had time to stabilize. When most (but not necessarily all) target elements have found their counterparts the transfer and evaluation subprocesses could begin. They can rely on the rich representation of the source to create plausible inferences in the target.

Of course, the advantages of decentralized representation come with a price: situations no longer have guaranteed and easily available identity. This is good from psychological point of view, as it offers possibilities for modeling complex analogies, blends, etc. From computational point of view, however, decentralization of representations increases the complexity of the mechanisms that operate on them. Classical top-down algorithms must give way to a decentralized and emergent mode of processing. The individual elements have to take the initiative and do the job themselves instead of being passive data manipulated from outside. This poses difficult issues about synchronization, coherence, conflict resolution, resource allocation, etc.

4.5.3. AMBR Situations in Detail

We close this chapter with a brief description of the concrete representational scheme used in the current version of AMBR.

As stated already, AMBR situations are represented as coalitions of agents. All situation elements are instance-agents, permanent or temporary (see subsection 3.2.2). Most of them represent the objects and propositions in the situation. There are, however, a few agents that stand for different *states* within the situation. States loosely bind several elements together and are useful for explicating the causal structure of the situation. For instance, there usually are an initial state, goal state, and end state. They are distinguished by tags in their `modality` slots — `:init`, `:goal`, or `:result`. The initial state is often divided into overlapping substates.

States are instance-agents with S-slots that point to some of the members of the state. Thus they resemble propositions with arguments (cf. Figure 4.4.1). Not all agents that could be considered members of a particular state need be explicitly mentioned, however. An element could be listed in zero, one, or more states, including states of different types (e.g. `:init` and `:goal`). Each element may have one or more tags related to states.

In turn, states are themselves arguments to relations such as `cause`, `follows`, and `prevents`. If we turn back to the example from section 4.1., the situation presented there could have an initial state that lists the three objects involved: `teapot-1`, `water-1`, and `hot-plate-1`. Another state-like agent combines the propositions that the teapot is on the plate and the plate is hot. This state is a left-hand argument of a causal relation stating that under these circumstances the teapot is also hot, etc.

Each situation in AMBR3 has a *situation-agent* associated with it. This is the most centralized aspect of current representations. Still, the situation agent is not equivalent to the *head* from previous versions. It bears very little representational content — it only materializes the spatio-temporal contiguity of the elements of the scene or episode. Situation agents are ordinary instance-agents. The sole peculiarity is the tag `:situation` in their `type` slot.

Each individual member has a `situation` slot filled by a reference to the situation agent. We say that the instance is *affiliated* to the situation. The situation agent itself, however, has at most associative links to a few members[6]. It is not possible to reach the members if given only the situation agent. It is possible, though, to determine whether two instances belong to the same situation. To that end, however, each element should enter the working memory on its own and 'claim membership'. This arrangement resembles the relationship between instance-agents and concepts.

[6] These `a-links` are used in DUAL for spreading activation only. They are ignored by the symbolic aspect.

CHAPTER V

AMBR MECHANISMS AT WORK

5.1. Sample Problem

This chapter presents a detailed description of AMBR mechanisms on the basis of a concrete example. The problem situation serving as basis for the presentation is taken from the domain outlined in section 4.1. It is about cooling milk in a teapot:

Target problem (situation **CM1**[7]): *There is a teapot and some milk in it. The teapot is made of metal.*

The goal is that the temperature of the milk is low.

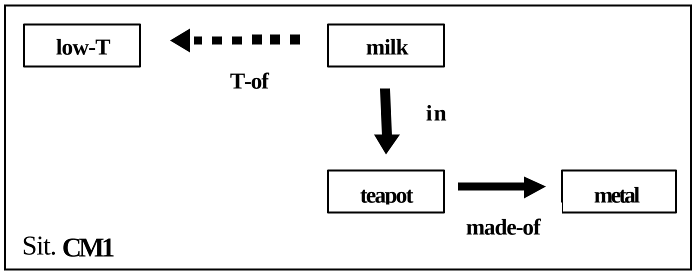


Figure 5.1.1. Schematic representation of the target situation described in the text. Objects are shown as boxes and propositions as arrows. Not all elements of the actual representation are present in the figure

This situation is represented in the current knowledge base by eleven instance-agents. Seven of them are illustrated in Figure 5.1.1. The representation also contains agents for the init and goal states, etc. Note that no cooling object (such as a refrigerator) is included in the original description of the problem.

In one of the many runs performed with the model, this problem happened to match to a long-term memory episode related to heating food in an oven (Figure 5.1.2). This particular run will serve as an illustration of the various mechanisms of the model.

[7] The current version of the knowledge base uses trigrams like **CM1** and **FDO** as situation names. **CM1** stands for 'cooling milk, variant 1' and **FDO** for 'food on a dish in an oven'. These trigrams appear in the transcripts listed in this chapter.

Base situation FDO: *There is a baking dish and some food on it. The shape of the dish is rectangular. There is also an oven. The dish is in the oven. The temperature of the oven is high.*

The goal is that the temperature of the food is high.

The outcome is that the temperature of the food is high. The fact that it is on the dish and the dish is in the oven entails that the food itself is in the oven. In turn, this causes the food to be hot, as the oven is hot.

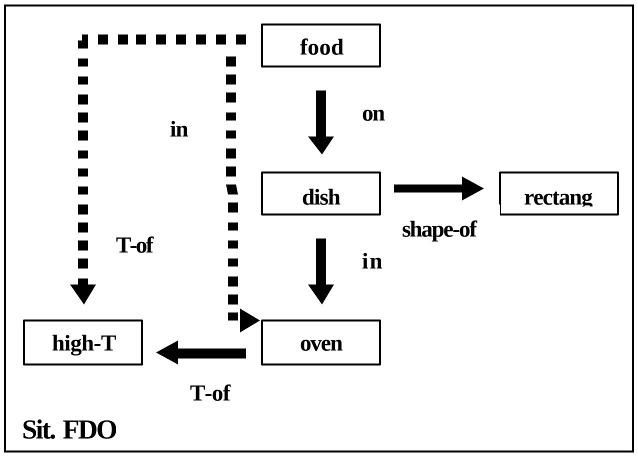


Figure 5.1.2. Schematic representation of the target situation described in the text. Objects are shown as boxes and propositions as arrows. The propositions explicating the causal structure of the situation are not shown.

Note that this description is much more elaborated than that of the target. It contains 21 agents (not all shown in the figure). As discussed in section 4.2.1., it is typical that the source analog is much richer than the target. In particular, there are many agents representing the causal links. For example, it is represented that the propositions `on(food, dish)` and `in(dish, oven)` taken together are the cause for `in(food, oven)`.

5.2. Spreading Activation

5.2.1. Purpose

As stated earlier the spreading activation mechanism is of paramount importance in AMBR (and DUAL in general). It is responsible for computing dynamic estimates of the *relevance* of each particular memory item. It defines the *working memory* of the system by bringing some agents above a threshold while keeping irrelevant ones away. The formula from subsection 3.1.5. is so important that merits replication here:

WM = active portion of LTM + temporary agents

Given that all information processing occurs in the working memory, spreading activation defines which agents take part in each particular computation. Moreover, it serves as energy supply to the symbolic aspect and thus determines the *speed* of each symbolic processor (subsection 3.1.3.3).

At a more global level, spreading activation is the basis of the *access subprocess* in analogy-making. It is responsible for accessing concepts and instances (and hence situations) that are relevant to the target. It also assures the relaxation of the constraint satisfaction network, which in turn is a key factor for the *mapping subprocess*. Various *context* and *priming effects* are also directly expressible in terms of that mechanism (Kokinov, 1988 1994a, 1995).

5.2.2. Spreading Activation in AMBR

The connectionist aspect is a general architectural feature of DUAL (subsection 3.1.3.2). This section is devoted to the particular design used in AMBR.

AMBR uses a modified version of the Grossberg activation function (Grossberg, 1978; Holyoak & Thagard, 1989). The function is chosen to meet the following requirements of the model:

- Time is continuous. (Or, the length of one elementary connectionist cycle is negligibly small with respect to the macroscopic time scale.)
- The activation level of any agent is bounded between zero and some fixed maximal value *M*.
- All links in the long-term memory are excitatory[8].
- There is spontaneous decay that forces each node to loose activation according to an exponential law in the absence of external support.
- There is a threshold *q* that clips small activation levels to zero.

If we neglect the threshold for the moment, the activation level *a* of any single node in the AMBR network is governed by the following differential equation:

$$\left|\begin{array}{l} a(t_0) = a_0 \\ \dfrac{da}{dt} = F(a, n) = -d.a(t) + E.n(t).[M-a(t)] \end{array}\right. ,$$

where $a = a(t)$ is the activation level as a function of time, $n = n(t)$ is the net input to the node, $M = const$ is the maximal activation value, and *d* and *E* are parameters that control the rate of decay and excitation, respectively. See (Petrov, 1997) for a mathematical analysis of this equation and for the discrete approximation used in the implementation.

8 Therefore, it could also be said that AMBR uses a modified version of the function proposed by McClelland & Rumelhart (1981). The two functions are equivalent for non-negative inputs. They differ, however, for the hypothesis agents in the constraint satisfaction network.

The function described above is the basis of the activation function of concept and instance agents in AMBR. (Hypothesis agents have a more complicated activation function described later.) There is one more complication, however — there is a working memory threshold. Whenever the activation drops below some predefined minimal level q, it is instantaneously brought to zero (and the agent is forced out of the working memory). Conversely, when the activation level of some node is zero and the magnitude of the net input n is bigger than some critical value n_θ, the activation level of the node jumps instantaneously to the threshold level q and then proceeds in the usual manner. The critical value n_θ is defined as the minimal support that an agent must receive from outside in order to resist the spontaneous decay and maintain activation at least at the threshold.

5.2.3. Example

This subsection shows how these abstract formulas apply to the problem of cooling milk in a teapot. The processing starts with the attachment of some agents to the special activation sources in the model — the *goal* and *input nodes* (subsection 3.1.5). In the particular case, the human user of the system links the agents `T-of-CM1` and `low-T-CM1` to the goal node. A few of the other agents comprising the description of the problem (e.g. `teapot-CM1` and `metal-CM1`) are attached to the input. These agents rapidly become very active and bring all their coalition partners to the WM. Thus the target problem is presented to the system. (The external context could also be represented on the input node (Kokinov, 1994a). This is not done in the example discussed here.)

Each instance agent from the target sends activation to its respective concept agent in the LTM. This allows them to enter the working memory and in turn activate related concepts and instances[9]. Transcript 5.2.3.1. illustrates this process. It is an excerpt from an actual AMBR run and tracks (roughly) the activation flow originating at `milk-CM1` and `tpot-CM1`.

```
T=0.04, adding milk to WM.
T=0.04, adding teapot to WM.
T=0.22, adding cook-vessel to WM.
T=0.22, adding liquid-holder to WM.
T=0.24, adding beverage to WM.
T=0.24, adding dairy-product to WM.
T=0.34, adding cheese to WM.
T=0.40, adding container to WM.
T=0.76, adding food-holder to WM.
T=0.98, adding plate to WM.
T=1.04, adding drinkable-liquid to WM.
T=1.06, adding liquid to WM.
T=1.08, adding food to WM.
T=1.42, adding cup to WM.
T=1.64, adding cow to WM.
T=1.84, adding baking-dish to WM.
```

9 The example discussed here starts with zero activation of all long-term memory agents. This, however, need not be the case. Kokinov (1994a) has modeled priming effects by starting from some residual activation pattern.

```
T=2.20, adding saucepan to WM.
T=2.50, adding bottle to WM.
T=2.68, adding non-drinkable-liquid to WM.
T=4.70, adding glass to WM.
T=5.80, adding soft-drink to WM.
T=5.80, adding alcoholic-drink to WM.
T=6.15, adding pan to WM.
T=7.85, adding water to WM.
T=8.25, adding ice to WM.
...
```

Transcript 5.2.3.1. Excerpt of an AMBR session showing the process of bringing concept-agents to the working memory. See text for details.

As evident from the transcript, activation propagates 'upward' in the class hierarchy, e.g. `milk-CM1 -> milk -> dairy-product -> food`. It also spreads 'horizontally' to concepts at the same level of abstraction, e.g. `milk -> cheese` (directly or via `dairy-product`). Some concepts that are associatively related to the active ones are also brought to the WM, e.g. `cow`. Sooner or later, however, the spread of activation is limited by the decay factor and new concept agents cannot pass the threshold. The number of active concept agents stabilizes, though individual agents occasionally get in or out the WM.

Recall from section 4.3. that there are top-down `instance` links from the concept agents to some of their instances in the LTM. These links mediate activation from the semantic to the episodic memory and initiate the access of source analogs. Transcript 5.2.3.2. shows the instances that happened to be activated by the concepts of the previous transcript.

```
T=0.34, adding milk-MTF to WM.
T=0.42, adding tpot-WTP to WM.
T=1.28, adding food-SFF to WM.
T=1.40, adding cup-IHC to WM.
T=1.80, adding dish-FDO to WM.
T=2.34, adding tpot-ERW to WM.
T=2.86, adding food-FDO to WM.
T=7.80, adding water-WTP to WM.
...
```

Transcript 5.2.3.2. Excerpt of an AMBR session showing the process of bringing instance-agents to the working memory. Compare with Transcript 5.2.3.1.

Note that initially there are isolated instances from disparate situations. The reason for their early inclusion in the working memory is their semantic similarity to the elements of the driver. As these instances participate in coalitions, however, they bring their partners to the WM too. Thus, retrieval of episodes is a bottom-up process in AMBR. Note that there is no need for any centralized data structure. This is in contrast to other models (e.g. Thagard et al., 1990; Forbus et al., 1994a) which treat analog retrieval as an explicit competition at the level of whole situations.

The spreading activation mechanism is influenced by the other mechanisms in the model. These influences are mediated by changes in the topology of the network. Many new agents and new links are added by various mechanisms. This greatly affects the flow of activation and contributes to the dynamic and emergent nature of AMBR computation. To illustrate, consider the activation history of one particular instance agent (`food-FDO`) shown in Figure 5.2.3.

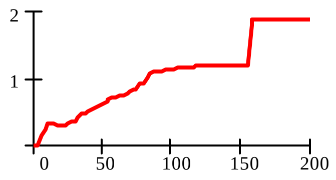


Figure 5.2.3. Activation history of the instance agent `food-FDO`. Time varies across the X-axis, activation level across the Y-axis. See text for details.

As we shall see later, `food-FDO` maps to the target instance of milk in our example. Thus the plot shows the gradual increase of the activation of a 'successful' agent. Note in particular the sharp bend at time 160. It is due to an influence by the rating mechanism (section 5.6). At that moment, the rating mechanism makes a commitment that `milk-CM1` corresponds to `food-FDO` and creates a temporary link between the two. In this way the highly active target element (attached to the input node) gives additional strong support to its counterpart.

5.2.4. A Prediction of the Model

In the example from the previous section all target elements are attached to the input and goal nodes simultaneously. This, however, need not be the case. On the contrary, it is more reasonable to expect that the elements are attached sequentially, in the order they become available to the system.

For example, suppose a student reads the verbal description of some problem from a textbook. The text is read sequentially and the internalized representation of this text would tend to be constructed sequentially too. In the AMBR model, this process could be crudely approximated by attaching the temporary agents that represent the target sequentially to the input node. In a more elaborated model, these elements will be constructed by the perceptual mechanisms. Similar considerations hold for the order of attachment to the goal node.

When some target elements are attached earlier than others, they will activate their respective concept agents earlier. This entails that the pattern of activation in the whole network shifts towards the association field of the early target elements. They have advantage over the elements that are attached to the source nodes later. Moreover, earlier elements tend to establish hypotheses earlier, which in turn reinforces their advantage. The net result of this process is that the order of presentation will have an effect on the processes of analog access and mapping. This is one prediction of AMBR that could be tested experimentally.

The order effect predicted here is similar to the one demonstrated by Keane, Ledgeway, and Duff (1994). There is a difference, however. The attribute-matching task used in their experiment did not involve access of a source analog from memory. The subjects were given two lists of propositions and asked to find a correspondence between them. The results of the experiment suggested that the order of propositions in the list affect subjects' latency to find the correct mapping. We predict that similar order effects could be demonstrated with respect to the process of analog access as well. Specifically, the order will affect the frequency of accessing episodes from memory. Episodes containing elements which are semantically similar to a given target element will be retrieved more frequently when this target element is presented earlier to the subjects. Section 6.4. presents a simulation experiment with AMBR that demonstrates such effect. A psychological experiment addressing the same topic is currently being prepared.

5.3. Marker Passing

As introduced in section 3.2.3.2., the marker passing (MP) mechanism is a tool for answering the question, "Given two nodes in the network, is there a path between them?". It is the symbolic counterpart of the spreading activation. Markers are emitted by certain *nodes of origin* and then propagate the network looking for a *marker intersection*. Figure 5.3.1. illustrates the variant of this general mechanism that is used in AMBR.

Each instance-agent in AMBR emits a marker when entering the WM. It sends it to its parent concept via the `inst-of` link. The concept agent stores the marker in its local buffer and in turn passes it further to its superordinate concept(s) via the `subc` link(s). Thus markers propagate 'upward' in the class hierarchy. Therefore, the presence of a marker in some concept indicates that the instance of origin belongs (directly or by inheritance) to that concept. For example, `drinkable-liquid` at Figure 5.3.1. collects markers from three instance-agents — `milk-CM1`, `milk-MTF`, and `water-WTP`. This information can then be used for inheritance of properties, for skolemization purposes, etc.

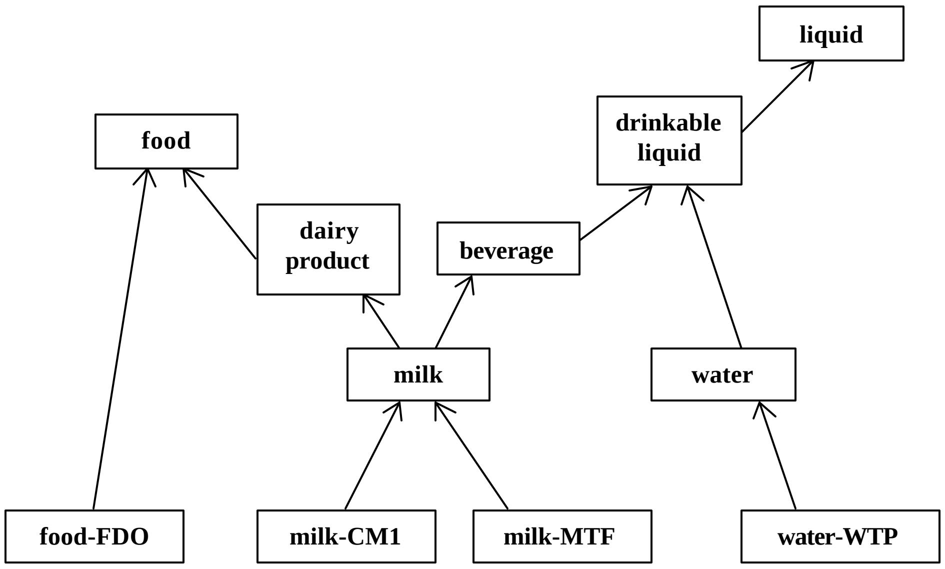


Figure 5.3.1. Illustration of the marker passing mechanism. The boxes at the bottom represent instance agents from different situations. All others stand for concept agents. See text for details.

The culmination of the marker passing mechanism, however, happens when two complementary[10] markers meet at some concept agent. This intersection indicates that the two instances are *semantically similar* as they belong to the same (super)class. The activation level of the intersection node can be used as a numerical estimate of the degree of similarity in the particular context (Kokinov, 1992b, 1994c).

Marker intersections have another important function in AMBR. They trigger the construction of semantically grounded hypotheses and thus initiate the constraint satisfaction mechanism. More concretely, when a concept agent detects an intersection it formulates a *node construction request* describing the new hypothesis-agent that is to be made. It then sends the request to one of the specialized node constructors which are the only agents in the architecture capable of making a new agent. The node constructor carries out the request and constructs a temporary agent of the prescribed kind. In the particular case, it will be an embryo hypothesis (cf. section 5.4) involving the two marker origins. The concept agent that has detected the intersection becomes the justification of the new hypothesis. In the example above, three such hypotheses are created: `milk-CM1<-->milk-MTF` justified by `milk`, `milk-CM1<-->water-WTP` justified by `drinkable-liquid`, and `milk-CM1<-->food-FDO` justified by food.

One of the biggest issues in marker-passing systems is the *attenuation* of the marking. Without such attenuation there would be too many marker intersections, most of which are useless and overwhelm the few useful ones. Different

10 Complementary markers have different origins and complementary *colors*.

systems use different attenuation strategies (see Hendler (1988) for an overview). Thus Quillian (1966) limits the number of links that a marker can traverse. Charniak (1983) checks the outbranching factor and prevents ‘promiscuous’ nodes from sending markers. Hendler (1988, 1989) uses an energy-like quantity called *zorch*, etc. In AMBR there is no need for a specialized mechanism for attenuation of markers as it follows naturally from the architectural principles of DUAL and the design of AMBR. Specifically, the spread of markers in the model is restricted by the following factors:

- Markers originate only from instance-agents. The concepts do not create new markers; they only pass the existing ones.
- Makers propagate only along links with certain labels (`inst-of`, `subc`, and `c-coref`).
- When there is a marker intersection the markers stop and do not propagate further.
- Only active agents can receive and send markers. Thus the spread of markers is limited by the boundaries of the working memory as determined by the spreading activation mechanism.
- Marker passing, as any other symbolic activity in the architecture, takes time and thus depends on the speed of the symbolic processor of the agent receiving and handling the marker. As a consequence, markers move very slowly in the ‘peripheral’ regions of the working memory where activation levels are low.
- Reporting marker intersection depends on a limited resource. There are only a few node constructors in the architecture and each concept agent must recruit one in order to create new hypotheses. When all constructors are busy the agent must wait until some of them becomes available. Thus there is an implicit competition in which the more active agents have advantage.

The net result of all these factors is that marker intersections are reported in a temporal order reflecting their potential usefulness for the particular task in the particular context. It is important to stress that this *global* marker passing is a dynamic emergent process. A whole coalition of DUAL agents is needed to cooperatively produce the final result. Each individual agent can do *local* MP only — instances know to create markers and concepts know to store and compare them. The overall result, however, is determined by a multitude of factors each of which has relatively minor impact on its own. Moreover, the relative strength of these factors vary dynamically in response to various external or internal events. Therefore, it is impossible to predict what marker intersections will happen in the particular case, when they will construct hypotheses, etc. Yet the process exhibits certain emergent regularities: more active (i.e. more relevant) areas of the network report more and faster marker intersections.

5.4. Constraint Satisfaction

5.4.1. Main Points

The multiconstraint theory (Holyoak & Thagard 1989, 1995) treats analogy-making in the light of three constraints: structural, semantic, and pragmatic. AMBR adopts this general idea. Like ACME, it uses a parallel connectionist algorithm for solving the constraint-satisfaction problem. This does not mean, however, that AMBR is a simple replication of ACME. There are a number of differences:

1. The constraint satisfaction network (CSN) is constructed incrementally by an emergent process. Hypotheses are created locally and are incorporated dynamically and asynchronously.

2. The CSN is integrated with the main network. This eliminates the need for special nodes mediating the semantic and pragmatic influences. This is responsibility of instance and concept agents themselves. Moreover, the hypotheses in the CSN send activation back to the agents in the main network. As discussed later, this is very important for the integration of analogical access and mapping.

3. Instead of covering all possible element pairs, AMBR builds only justified hypotheses. In addition to being much more economical, this eliminates the need for centralized representation of situations.

4. Several source situations compete in the CSN simultaneously thus allowing for complex analogies and blends when appropriate. Each situation is only partially accessed, however, and participates with its active elements.

5. It is possible to map relations with different number of arguments, to map two arguments from one side to a single one from the other, etc.

6. The network is not waited to settle in order to read out the 'solution' from the activation pattern. Instead, the CSN is in constant relaxation as the topology of the network changes. There is a rating mechanism that promotes winners and eliminates losers dynamically.

7. Each hypothesis agent undergoes an elaborate life cycle. The CSN involves hypotheses of different kinds.

8. There are hypotheses involving general propositions from the semantic memory (subsection 4.4).

The first two points are by far the most important. Previous constraint satisfaction models, and in particular ACME (Holyoak & Thagard, 1989) and ARCS (Thagard et al., 1990), work in successive stages. First, a source analog is retrieved from long-term memory or supplied manually by the experimenter. Second, the constraint satisfaction network is constructed by a sequential symbolic process. Finally, the CSN is waited to settle, thus identifying a coherent set of correspondences. This three-step process is illustrated in Figure 5.4.1.1. The stages are carried out by different and independent mechanisms.

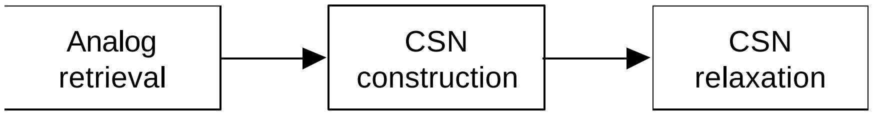


Figure 5.4.1.1. Constraint satisfaction as a three-stage process. The stages come one after the other and cannot interact. Compare with Figure 3.2.1.1.

In contrast, AMBR (Kokinov, 1994a) views constraint satisfaction as a single integrated process that has three interacting subprocesses. They all run together, each one influencing the rest (Figure 5.4.1.2). This is very much in the overall spirit of the model — compare with Figure 3.2.1.2. The whole computation is performed in an integrated fashion: the same representational structures and computational mechanisms are used for all three subprocesses.

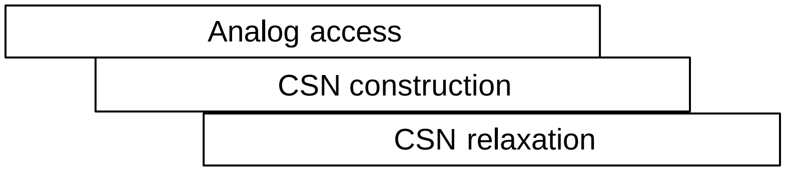


Figure 5.4.1.2. Constraint satisfaction as a set of interacting subprocesses. Compare with Figure 3.2.1.2.

This computational scheme has several important advantages:

- It allows for integration of the more global processes of access and mapping in analogy-making.
- The subprocess that builds the CSN can be guided by the associative mechanism to avoid blind construction of implausible hypotheses. In this way, AMBR builds only a small fraction of the hypotheses generated by ACME. This decreases the working-memory demands — a weakness of ACME that has been criticized by many researchers including its authors themselves (Keane et al., 1994; Kokinov, 1994a; French, 1995; Hummel & Holyoak, 1997).
- In the same time, AMBR retains the flexibility implied by the all-encompassing network used in ACME. AMBR does not construct *all* hypotheses, it constructs only *relevant* ones. And since relevance is dynamically determined, no possibilities are ignored a priory. This benefit is a direct consequence of the dynamic emergent computation that underlies AMBR's constraint satisfaction.

5.4.2. Hypothesis Agents

This subsection is devoted to the main actors in the constraint satisfaction network. From declarative point of view, hypothesis-agents carry four main pieces of information, each stored in a specific slot. The first two slots contain the two entities being mapped. They are called *hypothesis elements*. The hypothesis agent as a whole represents the hypothesis that the first element (from the *driver* situation) corresponds to the second element (from a *recipient* situation). There are also hypotheses involving concept agents.

The third slot of a hypothesis-agent contains its *justification(s)*. The justification of a hypothesis is the reason for which it has been created and is being maintained by the system. For example, one possible justification of the hypothesis that `milk-CM1` corresponds to `water-WTP` is that both are drinkable liquids.

There are two kinds of justifications: semantic and structural. A hypothesis has semantic justification when its two elements are semantically similar. Such justifications are established by the marker-passing mechanism. In most cases the two elements belong to close or even identical classes. On some occasions, however, AMBR can construct hypotheses between almost any two entities. This happens when the domains of the two situations being mapped are very remote and hence the markers can intersect only at some very abstract node such as `object`, `relation`, etc. In this way for example, `tumor` could be mapped to `fortress`. Such occasions are rare — usually the markers intersect earlier.

The second kind of justifications are the structural ones. A given hypothesis can have such justification when there is another hypothesis which interlocks with the first. For example, the hypothesis that two relations correspond justifies the hypotheses that the arguments of these relations also correspond. Structural justifications are established by the structure correspondence mechanism (section 5.5.).

Semantic justifications are always represented by concept-agents; structural justifications — by hypothesis-agents. It is possible (and frequent) that a hypothesis has several justifications. For instance, the hypothesis `milk-CM1<-->water-WTP` could be justified by `drinkable-liquid` (semantic) and by `temperature-of-CM1<-->temperature-of-WTP` (structural). In AMBR this particular hypothesis will be represented as schematized in Figure 5.4.2.1.

The fourth piece of information maintained by each hypothesis agent is a reference to the situation agent of the *driver situation*. In AMBR it is theoretically possible that two or more target problems are attached simultaneously to the goal node. Each of them initiates its own set of hypotheses. The fourth slot prevents mixing hypotheses from different sets. It is also useful for the other mechanisms of the model.

Figure 4.3.3.3. shows only the symbolic aspect of the hypothesis. In addition, there is a connectionist aspect (as always in DUAL). All references to other agents are also by which the hypothesis participates in the process of spreading activation. It supports its elements and in turn is supported by them. There are also excitatory links to the justification(s) and the driver situation.

```
milk-CM1<-->water-WTP:
  :type     (:mature :hypothesis :temporary)
  :t-link   ((milk-CM1<-->food-FDO -0.3)
             (milk<==>water        +0.3)
             (sit-CM1<==>sit-WTP   +0.3) )
  :slot1
    :c-coref  milk-CM1
  :slot2
    :c-coref  water-WTP
  :slot3
    :c-coref ((drinkable-liquid
              (temperature-of-CM1<-->T-of-WTP) )
  :slot4
    :c-coref sit-CM1
```

Figure 5.4.2.1. Example of a hypothesis-agent. It represents the hypothesis that `milk-CM1` corresponds to `water-WTP`. There are two justifications for this correspondence. Its driver situation is `sit-CM1`. Not all link weights are shown in the figure.

Finally, there are temporary links (`t-links`) that connect the hypothesis with other hypotheses. These links may be excitatory (for coherent hypotheses) or inhibitory (for conflicting hypotheses). They are invisible to the symbolic aspect of the architecture but are very important for the relaxation of the constraint satisfaction network.

Temporary links with negative weights deserve special comment. They embody the *one-to-one constraint* in analogical mapping. This constraint pushes the CSN towards a solution in which an element X from situation 1 is mapped to at most one element from situation 2. There is a strong pressure that the same element X should not be mapped to two or more elements, e.g., Y and Z. Thus, the hypotheses `X<-->Y` and `X<-->Z` are contradictory and should be connected with inhibitory links.

A problem arises at this point. The constraint-satisfaction network in AMBR is built by an emergent process. There is no central executive that goes through all hypotheses, identifies conflicting ones and puts inhibitory links between them. Rather, hypotheses are constructed one by one and the creator of each hypothesis has local information only. Under such circumstances, how does the hypothesis `X<-->Y` 'know' that there is a rival hypothesis (e.g. `X<-->Z`) to compete with?

The answer to this question is: The hypothesis will 'ask' the *secretary* of X.

5.4.3. Secretaries

Each entity-agent has a secretary associated with it. The secretary is *not* a separate agent; it is part of the entity-agent itself. The term *secretary* is used conventionally to refer to that particular part of a concept or instance agent that keeps track of the correspondences involving the agent.

The job of a secretary is twofold: it handles *hypothesis-registration requests* and carries out the rating mechanism. To that end, each secretary (i.e. instance or concept agent) is equipped with a slot and a few symbolic routines. The slot is labeled `hypoth` and is filled with references to all hypothesis-agents having the entity-agent as element. The same references are used as links that transmit activation from the agent (e.g. `milk-CM1`) to its hypotheses (e.g. `milk-CM1<-->water-WTP` and `milk-CM1<-->food-FDO`).

One of the first things that a hypothesis agent does after its creation is to send *hypothesis-registration requests* to the respective secretaries. Hypothesis-registration requests (or HR-requests for short) are symbolic structures notifying the secretary about the new hypothesis. Each of the two secretaries receives a request and sends a *secretary answer* back to the hypothesis. There are several kinds of answers but basically all of them could be aggregated into the following two major types:

- **'Resign'** — this answer means that the new hypothesis agent represents a tentative correspondence that already is represented by another hypothesis-agent. In other words, the new hypothesis is a duplicate of an older one. Such duplicate hypotheses are created because there usually are several justifications for a given correspondence. For example, the marker-passing mechanism could construct the hypothesis `milk-CM1<-->water-WTP` on the grounds that both are drinkable liquids. Later on, the structure correspondence mechanism could independently construct the same hypothesis on the grounds that `milk-CM1` and `water-WTP` are corresponding arguments in corresponding relations. This second hypothesis is conceptually identical with the first but will be represented by a different agent. Let us suppose (as is actually implemented in the program) that the name of the second hypothesis agent is `milk-CM1<-1->water-WTP`. When it tries to register at the secretary of `milk-CM1`, the latter will reply with an answer of type 'Resign'.
- **'Establish'** — this answer means that the hypothesis agent represents a novel hypothesis that does not coincide with any existing one. In the example above, the first hypothesis (`milk-CM1<-->water-WTP`) would receive such answer to its HR-request.

Secretary answers carry more information than the simple resign/establish distinction. Answers of type 'Resign' carry a reference to the *favorite* — the hypothesis in favor of whom to resign. Answers of type 'Establish' carry a (possibly empty) list of references to rival hypotheses.

5.4.4. Life Cycle of Hypothesis-Agents

Hypothesis-agents analyze the answers from the secretaries and act according to their directives. Due to the possibility of answers of type 'Resign', a new hypothesis is not guaranteed from the beginning that it has raison d'etre. It may be a duplicate of an existing hypothesis. If it manages to establish itself, there comes another struggle — it tries to win the competition with rival hypotheses.

AMBR distinguishes three main types of hypothesis-agents: *embryos, mature*, and *winner* hypotheses. They are marked by a tag in the `type` slot. More importantly, they differ in their activation functions and the repertoire available to their symbolic processors.

Each hypothesis-agent starts its life cycle as an embryo. Later on, it either *resigns* in favor of some other hypothesis or *establishes* and becomes mature. Mature hypotheses have the chance of being *promoted* to winner status (or demoted to loser status). In more detail, the life cycle is the following:

The main rule for hypothesis construction in AMBR is that each hypothesis must have a justification. As stated earlier, there are two possibilities for construction of a hypothesis-agent: by the marker passing and by the structure correspondence mechanism. Either way, the new embryo hypothesis is created (by a node constructor) and begins its life cycle. It sends hypothesis registration requests to the secretaries of its two elements and waits for the answers. Usually, the two answers are the same — either both are 'Establish' or both are 'Resign'. The embryo takes corresponding actions respectively. Sometimes the secretaries disagree in their answers. This is possible due to the asynchronous and parallel nature of DUAL interactions. Embryo hypotheses are equipped with procedural knowledge for resolving the ambiguities.

When it turns out that the new embryo hypothesis is a duplicate of an existing hypothesis (called *favorite*), the former resigns in favor of the latter. The resigning hypothesis hands over to the favorite all its declarative knowledge and in particular its justification. Having done that, it fizzles out. In the end, there is one hypothesis agent with two justifications instead of two separate hypotheses with one justification each. This is the mechanism that allows for multiple justifications of AMBR hypotheses despite that initially each has only one.

If the analysis of secretary information reveals that the embryo hypothesis represents a novel correspondence between two elements, the embryo establishes itself and becomes a mature hypothesis. From now on, its main goals are to win the competition with alternative hypotheses and to sprout out children.

The first goal is pursued by creating inhibitory links with the rivals. (The hypothesis receives a list of its rivals as an 'enclosure' to the secretary answers.) For fair play, the new agent sends its reference to all competing hypothesis, prompting them to establish symmetric inhibitory links.

The third phase of the life cycle of hypothesis agents begins when (and if) the hypothesis receives a *promotion incentive* from an *authorized secretary*. This topic is discussed in section 5.6.

5.4.5. The Constraint-Satisfaction Network

The mechanisms described so far gradually build many hypothesis-agents and establish connections between them. In this way, a constraint satisfaction network emerges. The CSN is a formation of agents that cooperatively solve a constraint satisfaction problem. It is integrated with the main network. Thus, they become complementary parts of the big population of agents that comprise the model as a whole.

The CSN involves the following kinds of links:

1. LTM—>CSN: Links from instance and concept agents to the respective hypothesis agents. These links are excitatory and are stored in `hypoth` slots of entity agents.

2. LTM—>CSN: Links from concept agents (e.g.`drinkable-liquid`) to the hypotheses justified by them (if any). These links are excitatory and are stored in `t-link` slots of concept agents.

3. CSN—>LTM: Links from hypotheses to their elements, semantic justifications, and driver situations. These links are excitatory and are stored in S-slots of hypothesis agents.

4. CSN—>CSN: Links from a hypothesis to its structure correspondence children (if any). These links are excitatory and are stored in `t-link` slots.

5. CSN—>CSN: Links from a hypothesis to its structural justifications (if any). These links are excitatory and are stored in S-slots.

6. CSN—>CSN: Links between competing hypotheses. These links are symmetric, have negative weights, and are stored in `t-link` slots of hypothesis agents.

The constraint satisfaction network thus embodies the three constraints posited by the multiconstraint theory. The structural constraint is manifested in categories 4, 5, and 6 above. The semantic constraint appears in category 2, and the pragmatic one — in categories 1 and 2. Note that besides the links discussed here, AMBR has additional mechanisms for enforcing the constraints.

The links from the CSN to the rest of the network (category 3) deserve special attention. Through these links, the constraint satisfaction mechanism influences the pattern of activation in the main network and hence everything in the architecture. This fact has important implications for the integration between analogical access and mapping.

Hypothesis activation function. Hypothesis-agents are special in that they receive not only excitatory but also inhibitory input from their neighbors. They have two separate input zones — *enet* and *inet*. The two connectionist inputs are combined with the current activation level of the agent to determine the change of activation. The change of activation is governed by a continuous modification of Grossberg's activation rule. (Compare with the equation from subsection 5.2.2.)

In the original version of Grossberg's function (Grossberg, 1978), the activation can take both positive and negative values. The specification of DUAL, however, postulates that all agents in the architecture must have non-negative activation functions. Therefore, AMBR uses a linear transformation of hypothesis' activation. The neutral point of the function (i.e. the resting level for zero input) is placed above the working memory threshold. The most negative branch of the function is truncated by the threshold. In this way the most pronounced losers are eliminated automatically — they simply fall out the WM and die (recall that all hypotheses are temporary agents). The upper part of the negative branch, however, is situated above the threshold. Thus the hypotheses that are judged implausible but not absolutely weird have a chance to survive.

Hypothesis output function. Hypothesis-agents are also characterized by a specific output function. Moreover, it is different for embryo hypotheses and mature hypotheses. Embryo hypotheses do not influence their neighbors at all. (In other words, their output function is the constant zero.) The reason for this decision is that the embryos are do not really participate in the CSN yet. If they maturate, however, their output function changes. Mature hypotheses have a threshold output function so that only hypotheses whose activation is above the neutral level can influence their neighbors.

5.4.6. Example

As an example of the mechanisms discussed so far, and in preparation for the structure correspondence mechanism that comes next, this section provides a transcript showing the construction of one particular embryo hypothesis — `milk-CM1<-->milk-MTF`. The example illustrates construction of a hypothesis by the marker-passing mechanism, followed by secretary inquiries. The concept `milk` detects a marker intersection at time 1.72 and sends a node construction request to the special agent `*NC6*`. It constructs an embryo hypothesis at time 2.84. After registering at its two secretaries the hypothesis maturates at time 4.98.

```
T=0.16, #<MRK MILK-CM1> received in MILK.
T=1.64, #<MRK MILK-MTF> received in MILK.
T=1.72, #<MRK MILK-CM1> and #<MRK MILK-MTF> intersected at MILK.
T=2.12, #<NCR MILK> received in *NC6*.
T=2.84, creating a new agent: MILK-CM1<-->MILK-MTF
T=3.72, #<HR MILK-CM1<-->MILK-MTF> received in MILK-CM1.
T=3.82, #<SA+ nil> received in MILK-CM1<-->MILK-MTF.
T=3.84, #<HR MILK-CM1<-->MILK-MTF> received in MILK-MTF.
T=4.96, #<SA+ nil> received in MILK-CM1<-->MILK-MTF.
T=4.98, establishing hypothesis MILK-CM1<-->MILK-MTF.
```

Transcript 5.4.6. Excerpt of an AMBR session showing the construction and establishment of a hypothesis. `#<MRK xxx>` is a marker originated at the instance agent with the given name, `#<NCR xxx>` is a node construction request, `#<HR xxx>` is a hypothesis registration request, and `#<SA+ nil>` is a secretary answer of type 'Establish'.

5.5. Structure Correspondence

The structure-correspondence (SC) mechanism generates new hypotheses on the basis of existing hypotheses. It is carried out by mature hypothesis-agents. Their symbolic processors are equipped with routines specialized for the task.

There are two major types of SC, conventionally termed *bottom-up SC* and *top-down SC*. Both come in *strong* and *weak* variants.

5.5.1. Bottom-up Structure Correspondence

Bottom-up SC takes place when there is a hypothesis involving two instance-agents. More precisely, it happens when there is a mature hypothesis whose elements have the tag `:instance` in their `type` slots. Under these circumstances, the symbolic processor of the hypothesis tracks the `inst-of` links of the two instances and retrieves their respective concepts. For example, if the two instances are `milk-CM1` and `water-WTP`, the concept agents will be `milk` and `water`. Then, the original hypothesis initiates a process for constructing a supplementary hypothesis stating a parallel correspondence between the two concepts. The new embryo is constructed in the usual way — by formulating and sending a node construction request. The original hypothesis becomes the justification of the new one.

It frequently happens that the new hypothesis is not really new — the same concepts have been already put into correspondence by an earlier invocation of the structure-correspondence mechanism. For example, the hypothesis `milk-CM1<-->water-WTP` generates the concept-level hypothesis `milk<-->water`. After a while, another hypothesis, e.g. `milk-CM1<-->water-FDO` constructs another instantiation of the same concept-level hypothesis. In such cases, the duplication is detected by the secretaries and the second hypothesis resigns in favor of the first. Eventually, `milk<-->water` will have two justifications and there will be appropriate excitatory links. The net result of this process is that the overall degree of connectivity in the CSN is enhanced.

The mechanism of bottom-up SC creates a pressure that correspondences at the instance level should be coherent with correspondences at the concept level. Stated differently, the mapping of two instances facilitates mapping of the classes to which they belong and vice versa.

The bottom-up SC also creates hypotheses involving the *situation-agents* to which the instances are affiliated. Recall that AMBR maps the target situation to several different bases simultaneously. The bottom-up SC creates hypotheses of the form `sit-CM1<-->sit-WTP` and `sit-CM1<-->sit-FDO`. The existence of such hypotheses in the CSN creates a pressure that situations are mapped as units. Blends are possible but they happen only when truly warranted. Normally the model tries to keep the mapping within the scope of two situations only: the target and a single base.

5.5.2. Top-down Structure Correspondence

Top-down SC is present in one form or another in all models of analogy-making. It captures an important aspect of the structural constraint as posited by Gentner (1983) and Holyoak & Thagard (1989): When two propositions correspond, it is highly desirable that their respective arguments also correspond.

The difficulties begin with the disambiguation of the phrase 'respective arguments' above. Some models (e.g. Falkenhainer et al., 1986) walk around this difficulty by assuming that the enumeration of the arguments in a proposition can be meaningfully transferred to another proposition. From our point of view, this approach seems too conservative and psychologically implausible. In contrast, Holyoak & Thagard (1989) follow an approach that seems too liberal—they consider all possible argument pairs.

Thanks to the elaborated knowledge representation scheme adopted in DUAL (Kokinov, 1988, 1992), AMBR does not have great difficulties with this problem. Each argument is represented by a separate S-slot with many facets. One of these facets points to the respective slot in the parent concept as discussed in section 4.4. This greatly facilitates the structure correspondence mechanism and relieves the model of implausible assumptions. Moreover, supports mapping propositions with different number of arguments (Kokinov, 1994a; Hummel & Holyoak, 1997).

The details of the top-down structure correspondence in AMBR are the following: The symbolic processor of each mature hypothesis checks whether the two elements are propositions. The criterion is whether they contain the tags `:instance` and `:relation` among the fillers of their `type` slots. If this is the case, the symbolic processor attempts to determine the slot-to-slot correspondences. To do this, it needs the so called *pivot concept*.

The pivot concept is a concept which is a common superclass of both relations. For example, if the propositions are instances of the relations `in` and `on`, the pivot concept could be `in-touch-with`, `asymmetric-binary-relation`, or something else depending on the particular problem and context.

The pivot concept is often identified by the marker passing mechanism. When such information is available, the symbolic processor of the 'proposition' hypothesis generates the appropriate 'argument' hypotheses. When the information is not available, the symbolic processor checks for the obvious (and frequent) case when both propositions are instances of the same relation. In other words, it checks whether the `inst-of` slot of the two instances point to the same concept agent and uses the latter as a pivot concept. Otherwise, it gives up and stops, hoping the MP mechanism will provide the missing information later.

5.5.3. Weak Structure Correspondence

In many cases it is bad to allow the SC mechanism create new hypothesis but it is desirable to make it create additional justification links between existing hypotheses. This is the purpose of the weak SC.

For example, suppose that two situations — **CM1** and **WTP** — are being mapped. As discussed in subsection 4.5.3. these situations may involve *states*. Suppose that `initst-CM1` and `initst-WTP` are two such states. The marker passing mechanism detects they are instances of the same concept (namely `init-state`) and creates the hypothesis `initst-CM1<-->initst-WTP`. Finally, suppose it establishes and becomes mature. Now the question is, "Should this hypothesis perform top-down structure correspondence?"

Each state has several S-slots pointing to the elements of the respective situation and the initial relations between them. Thus, the two states resemble propositions of type `and` and, therefore, one wishes to generate SC-motivated hypotheses about the arguments of these `and`-like propositions. Applying the usual (i.e. *strong)* structure correspondence mechanism indiscriminately, however, would lead to proliferation of useless hypotheses such as `milk-CM1<-->high-temp-WTP`. To avoid this, states (and all agents having the tag `:situation` in general) are exempted from the strong top-down structure correspondence — the hypotheses involving such agents never generate any new hypotheses.

On the other hand, they could establish new justification links. To see why, consider the hypothesis `milk-CM1<-->water-WTP`. It has a justification that has nothing to do with the membership of `milk-CM1` in `initst-CM1`. Still this hypothesis is consistent with `initst-CM1<-> initst-WTP` and it is desirable to establish excitatory links between the two. Such link would improve the connectivity of the constraint satisfaction network and strengthen the structural constraint on mapping.

The main procedure for weak top-down structure correspondence is the following: Retrieve all S-slots of the two states and construct all possible pairings among them. Do *not* issue node construction requests, however. Instead, check the `hypoth` slot (see subsection 5.4.3.) of the secretaries of each pair and look for an old hypothesis representing the same correspondence. If such hypothesis is indeed registered at the secretaries, establish excitatory links to it. If there is no such hypothesis, however, then simply ignore the pair.

The weak SC has a bottom-up variant too. It is the reverse of the ordinary top-down SC. That is, instead of descending from propositions to arguments, it tries to ascend from arguments to propositions. To illustrate, suppose `milk-CM1` is an argument in the proposition `in-CM1` and `water-WTP` in the proposition `in-WTP`. Suppose further that there is a mature hypothesis `milk-CM1<-->water-WTP`. Then this hypothesis will try to establish a link to the hypothesis involving `in-CM1` and `in-WTP` provided such hypothesis is registered at the respective secretaries.

5.6. Rating and Promotion

The mechanisms presented so far are concerned primarily with generating hypotheses and establishing links between them. The final goal of these efforts, however, is to arrive to a set of correspondences. To that end, the model must make commitments at some point. This is the main objective of the mechanisms discussed in this section.

5.6.1. Rating Mechanism

5.6.1.1. Motivation

Each hypothesis on the secretary list of an entity agent represents a possible correspondence between the entity and someone 'at the other side'. The one-to-one constraint on mapping imposes that each element from the one domain should map to one element from the other. There is ambiguity, however, because each element typically has several hypotheses registered at its secretary. The relaxation of the constraint satisfaction network resolves these ambiguities using the inhibitory links between the incompatible hypotheses.

A straightforward approach for determining the final set of correspondences is to wait until the CSN settles and then pick up the hypotheses with maximal activation levels. Thus ACME (Holyoak & Thagard, 1989) waits until all nodes in the network reach asymptote. There are, however, two drawbacks of this approach: (*i*) the decision to stop must be taken centrally and (*ii*) any post-mapping processing can begin only after the mapping stage is over.

The rating mechanism avoids these limitations by *promoting* hypotheses during the run. This allows for smooth integration between mapping and post-mapping processing. In particular, the processes of transfer (inference) and evaluation could begin before the CSN has settled completely.

5.6.1.2. Main ideas

Let us introduce the following terminology: a (current) *leader* is the hypothesis with the highest activation level in its set at the moment; a (final) *winner* is the hypothesis that has been explicitly *promoted* and has transformed itself into a winner-hypothesis agent (cf. section 5.4.4).

The main purpose of the rating mechanism is to monitor the hypotheses and send *promotion incentives* to those of them that emerge as stable and unambiguous leaders. It serves two collateral purposes as well: to eliminate hypotheses that are obvious *losers* and to trigger the skolemization mechanism.

The rating mechanism is carried out by the (secretaries of) instance agents[11]. Not all instances, however, are *authorized* to do so. Promoting winners is an important commitment that should be done carefully and by an 'impartial judge'. Therefore, only the secretaries of the driver situation are authorized to promote winners. (In the current version of AMBR, the driver situation is always the target. Future versions will switch the source analog as driver for the purposes of the transfer process (Hummel & Holyoak, 1997). They will probably grant some limited rating authority to recipient secretaries as well.)

Whenever an instance agent receives a hypothesis registration request (subsection 5.4.3), it checks if it is authorized to do ratings. The criterion is whether its respective situation agent has the tag `:driver` in its `modality` slot. If authorized, the secretary creates a data structure called a *rating table* and stores it in its buffer. (Recall from section 3.1.3 that each DUAL agent has some limited local memory.) The rating table keeps *individual ratings* for all mature hypotheses on the secretary list. Individual ratings are numerical quantities that characterize the relative success of the respective hypothesis.

The secretary periodically performs *rating surveys* to adjust the ratings. Each survey determines the current leader and increases its individual rating a little. The ratings of all other hypotheses are decreased. The magnitude of the change is proportional to the margin between the activation levels of the leader and its closest competitor. (If there is only one hypothesis, its activation is compared against the neutral level.) Thus, each rating value indicates how long the respective hypothesis has been a leader, how recently, and how strongly so.

Each new hypothesis starts at some intermediate rating level and then goes up or down depending on its relative standing in the total pool of competing hypotheses. If the rating reaches some *critical winner rating*, the hypothesis is considered for promotion. (It is not automatically promoted, however, as discussed below.) On the other hand, if the rating drops below some *critical loser rating*, the hypothesis is considered for elimination.

As a consequence of this computational scheme, hypotheses that are clear and unambiguous leaders rapidly reach promotion. On the other hand, when there are two or more competitors of equal strength or when there is a change in the leadership, the secretary refrains from making premature commitments. The decision is deferred until other secretaries announce their winners and change the balance in the CSN.

5.6.1.2. Promotions and ballotages

When the individual rating of some hypothesis reaches the critical winner level, it becomes a candidate for promotion. As this criterion alone is not always reliable, however, the secretary undertakes some additional last-minute checks to determine whether the candidate really merits promotion or not. If it does, the

[11] Concept agents do not rate their hypotheses in the current version of AMBR. The so called *promotion propagation* mechanism will extend this functionality. This mechanism, however, is in experimental stage and is not reported in this Thesis.

secretary sends it a *promotion incentive*. When the candidate is judged inappropriate, though, the secretary announces a *ballotage* which means that the rating procedure should be repeated.

The rating mechanism is based on local information only — the activation levels of the hypotheses registered at a single secretary. Hence, it sometimes favors hypotheses that are inconsistent with the global mapping as determined by the CSN as a whole. The relaxation algorithm almost always succeeds to produce a consistent set of leaders for all secretaries. This, however, often takes time, especially when there are strong local anomalies that must be overcome.

Consider the following example. The problem presented in section 5.1. involves a teapot: `tpot-CM1`. It so happens in one particular run that the target situation maps to a base with a dish (namely `dish-FDO`) instead of a teapot. As other bases also compete in the CSN, there are alternative correspondences for the target teapot. One of them — `tpot-WTP` — proves to be an especially strong competitor. In addition to its greater semantic similarity, it is also supported by the fact that there is an explicit proposition about its material. A similar proposition participates in the target description too. This leads to a triad of mutually supporting hypotheses: `tpot-CM1<-->tpot-WTP`, `made-of-CM1<--> made-of-WTP`, and `metal-CM1<-->metal-WTP`. It is difficult for `dish-FDO`, whose material is not explicated, to beat this cluster alone.

Still, strong factors elsewhere in the CSN (other propositions, causal structure, etc.) dictate that the target as a whole maps better to situation **FDO**, not to situation **WTP**. The secretary of `tpot-CM1` does not know this, though. On its local list it sees the hypothesis `tpot-CM1<-->tpot-WTP` that has come first and remained on top ever since. It consistently dominates the surveys and its individual rating reaches the critical level. If it is promoted, however, it (and its `made-of` entourage) would be an odd man out among all other winners from situation **FDO**.

One way to prevent blendings of this kind is to set a high critical level for promotions. This will give time to the constraint satisfaction network to settle globally and straighten up local inconsistencies. This approach, however, effectively entails that all promotions occur after the mapping process is over. Subsequent processes of transfer, evaluation, etc. must take place when the CSN is frozen. This brings the model back to the pipeline paradigm that is antithetical to the spirit of AMBR (cf. section 3.2.1).

The current version of the model adopts a different strategy. The authorized secretary takes a step out of the local neighborhood. Before issuing a promotion incentive, it checks whether the candidate hypothesis is consistent with the leader at the level of situations. In the example, the secretary of `tpot-CM1` reads the `situation` slot of the other element — `tpot-WTP`. Thus it learns that the latter is affiliated to `sit-WTP`. The secretary then contacts its own situation-agent (namely `sit-CM1`) and asks for the leader among the hypotheses at that level. It turns out that the leader there is the hypothesis `sit-CM1<-->sit-`

`FDO` which is incompatible with `sit-WTP`. Therefore the secretary does not promote the candidate hypothesis.

Instead, `tpot-CM1` announces a ballotage and undertakes measures to weaken the unwanted hypothesis. It sends a message to the agent `sit-CM1<-->sit-FDO` to create an inhibitory link to `tpot-CM1<--> tpot-WTP`. This speeds up the relaxation of the CSN. It also sets the individual rating of the unwanted hypothesis back to the initial level. The rating of the second best hypothesis is also modified if it is below the initial level. Finally, the secretary triggers the skolemization mechanism when appropriate (see section 5.7). After all these emergency measures, it restarts the rating surveys.

The previous paragraphs may create the impression that AMBR treats blends as something that must be avoided at all cost. This is not the case. Blends do happen in human analogy-making (e.g. Turner & Fauconnier, 1995) and should be accounted by cognitive models. Such blends, however, happen on quite special circumstances and involve bigger and more complex episodes. The ballotage discussed here is designed to prevent blends with few isolated intruders into an otherwise homogeneous mapping. This aspect may be improved in future versions of the model. Even the current version could in principle produce heterogeneous mappings when there is a change of the leading hypothesis at the level of situation-agents.

5.6.1.3. Elimination of losers

In addition to promotion winners, the rating mechanism is also useful for weeding out *loser* hypotheses. Recall that the ratings of all hypotheses except the leader are decreased on each rating survey. When a rating drops below a critical threshold, the respective hypothesis is considered for elimination. If its activation level is also low, the hypothesis receives a *fizzle message*. Those hypothesis whose activation levels are only moderately low, however, are retained as potentially useful.

The elimination of losers adds another dimension of the dynamics of the constraint satisfaction network. It both grows and shrinks. New hypotheses are added by various justifications. In the same time, losers hypothesis die out. As a consequence, the size of the CSN varies dynamically, growing rapidly at first and then shrinking back to retain only the most promising hypotheses. Usually, each promotion is followed by a number of eliminations. At the end of the run, each secretary list contains one winner and one or two (or zero) ‘reserve’ hypotheses.

When the target elements are presented incrementally to the system (e.g. by some perceptual mechanism), the ‘wavefront’ of the CSN follows suite. In this way the model seems to be able to handle situations that are much bigger than the ones used in current simulation experiments. The size of the CSN need never get very big. This has important consequences with respect to working memory limitations (Keane et al., 1994; Hummel & Holyoak, 1997). It also relates to the discussion of blending above — the target could match one base in the beginning, form some stable correspondences, and then shift to another base that better fits the target elements that have appeared in the interim.

5.6.2. Promotion mechanism

This section describes the events triggered by reception of a promotion incentive in a hypothesis agent. This incentive marks the beginning of the third phase of the life cycle of the hypothesis (cf. subsection 5.4.4.). The mature hypothesis transforms into a winner. In the current version of the model this change involves nothing but removing the `:mature` tag from its `type` slot and adding `:winner` in its place. More radical restructuring is also possible (e.g. modifying the activation function, decay rate, efficiency coefficient, etc.).

When the due restructuring is complete (and presumably it takes quite a lot of time), the new winner sends *metamorphosis notifications* to its two secretaries to inform them about the change. These notifications make them even more severe to the losers in their `hypoth` slots. Only a few of the strongest (in terms of activation level and/or ratings) alternatives are rescued. These survivors are marked by a tag `:loser` in their `type` slots. This tag is useful for detecting unmapped elements as a prerequisite for transfer.

Moreover, each instance agent creates a temporary excitatory link to its counterpart as designated by the winner hypothesis. For example, the metamorphosis notifications from `tpot-CM1<-->dish-FDO` cause each instance to create a link to the other. In particular, this creates a shortcut route for draining activation from the target, thus bringing more elements of the source situation to the working memory (cf. Figure 5.2.3).

5.7. Skolemization

5.7.1. Motivation

Most analogy models either do not use semantic information at all (Falkenhainer, Forbus & Gentner, 1986; Keane & Brayshaw, 1988) or use it solely for estimating semantic similarity (Holyoak & Thagard, 1989; Kokinov, 1994a; Hummel & Holyoak, 1997). It is clear, however, that human analogy-making recruits much more semantic knowledge than that. There are at least two ways in which the general knowledge about some domain is used when making spontaneous analogies.

First, semantic knowledge may be used for reconstruction and elaboration of source analog(s). Research on autobiographical memory provides evidence that recollection of past episodes involves much reconstruction in addition to rote retrieval (Barclay, 1986). It is reasonable to expect that the same is true for recollection of past problems and their solutions, examples from textbooks, etc. This, however, is largely ignored by current models of analog retrieval.

On the other hand, semantic knowledge may be used for elaboration of the target problem. It can even provide pieces of the solution. For example, the general fact that plates are heat sources and as such are used to heat things is of obvious importance when asking how to heat water. Still, such knowledge goes unused if the model deals exclusively with finding correspondences between two episodes.

5.7.2. Main Ideas

AMBR skolemization constructs specific propositions on the basis of general propositions. It is a mechanism for elaborating the description of a situation using general knowledge about its elements.

Recall from section 4.4. that a *general proposition* is a proposition involving a general class instead of individual instance. If we ignore the details of the representation scheme (cf. Figure 4.4.2), general propositions are most easily recognized by the fact that at least one of their arguments is a concept agent. Thus, `made-of(teapot, metal)` is a general proposition as opposed to the specific `made-of(teapot-MTF, metal-MTF)`. Typically only one of the arguments of the general proposition is a concept; the other arguments are *prototype instances.* This creates asymmetry that often better capture the semantics. To illustrate, the proposition `made-of(teapot, metal)` could be read in two ways: 'each teapot is made of metal' and 'each metal is the material of some teapot'. In contrast, the proposition `made-of(teapot, prototypical-teapot-metal)` allows only the first interpretation.

One way or another, a general proposition represents a fact about some class of objects. The target problem and the episodes in the long-term memory, however, involve specific instances. The purpose of skolemization is to bring the general fact to the level of specific instances. This is done by constructing a new *Skolem* proposition that parallels the general one but in which each concept or prototype argument is replaced by a specific instance.

A question arises at this point, "Where do these specific instances come from?" The AMBR answer is that they are either supplied by the marker passing mechanism or created from scratch. The first choice is preferred whenever possible, falling back to the second only in the absence of appropriate markers.

For example, suppose the skolemization mechanism works on the general proposition `made-of(teapot, metal)`. In order to specialize it, it needs instances of the classes `teapot` and `metal`. Looking for such instances, it checks the buffers of these concept agents for markers. Each marker originates from some instance agent and propagates upward in the class hierarchy (see section 5.3). Therefore, the origins of all markers arrived at a concept agent are instances of this concept. Suppose the buffer of `teapot` contains a marker from `teapot-MTF`. Thus, `teapot-MTF` could be used as a specialization of the first argument of the general proposition. The same check is done for the second argument. For the sake of the example, suppose that the buffer of `metal` contains no markers. Therefore the skolemization mechanism creates a new instance of this class. Such instances are called *Skolem instances*. Let the name[12] of the new agent is `*metal-1`.

[12] By convention, the names of all agents created by the skolemization mechanism begin with an asterisk.

After the skolemization mechanism finds an instance argument for each slot of the general proposition, it is ready to construct the *Skolem proposition*. The last ingredient is the *head* of the proposition. It is made after the template provided by the head of the general proposition. (Note that the latter is an instance agent belonging to some relational class, see section 4.4.) In the example above, suppose the new agent is called `*made-of-1`. It is an instance of the relation `made-of` and its two S-slots are filled by `teapot-CM1` and `*metal-1`, respectively.

The final result of the whole process is that there is a proposition explicating the material of `teapot-MTF`. Like all teapots, it is made of metal.

Note that the general proposition may involve a concept higher in the class hierarchy. To extend our example, saucepans, pans, and baking dishes are made of metal too. A single general proposition can cover them all: `made-of(cook-vessel, metal)`.

5.7.3. Triggering Skolemiz ation

Most of the work related to skolemization is carried out of the symbolic processor of a *general hypothesis*. This section describes how such hypotheses are created and prompted to perform skolemization.

A general hypothesis is a hypothesis involving a general proposition. It is created by the marker passing mechanism in the usual way. That is, the head of the general proposition (which is an instance agent) emits a marker when entering the WM. This marker propagates in the usual way and can intersect with other markers. As discussed in section 5.3, when two complementary markers intersect they give rise to a semantically justified hypothesis. Complementarity rules in this case specify that the other marker must originate from some driver element. Hence, the new hypothesis involves a proposition from the driver situation on one hand, and the general proposition on the other.

Note that even though the semantic memory can potentially have thousands of general propositions, only a small fraction of them (if any) are used in each particular task. These 'privileged' propositions are determined by the driver. The elements of the driver transmit the activation necessary for bringing the general propositions (like any agents) to the working memory. Their markers are prerequisite for the creation of general hypotheses (like any hypotheses).

Consider an example: The target problem **CM1** (see section 5.1) contains the proposition `made-of(tpot-CM1, metal-CM1)`. The head of this proposition is the instance agent `made-of-CM1`. In the run that serves as an illustration throughout this chapter, the latter agent happens to form the following general hypotheses: `made-of-CM1<--> ckves-made-of-metal` and `made-of-CM1<-->bottle-md-glass`. Each of them involves a general proposition and could be skolemized.

The actual skolemization process begins when the general hypothesis receives a *skolemization incentive* from an authorized secretary. The rating mechanism is responsible for determining which hypotheses receive such incentives, if any. General hypotheses register at the secretaries and participate in rating surveys in the usual way. If such hypothesis is the leader in its set, its rating goes up. When it reaches some critical level, the hypothesis receives a skolem incentive.

General hypotheses are quite weak compared to hypotheses involving specific propositions. It is, therefore, quite rare that a general hypothesis wins the rating. This is good because affiliated propositions should be preferred to Skolem propositions anyway. In the example above, suppose that `tpot-CM1` maps to some other teapot whose material is explicitly represented too. Under these circumstances the driver proposition `made-of-CM1` naturally maps to the respective recipient proposition and there is no need for skolemization. And so it happens — the specific hypothesis wins the rating and the general hypothesis never receives any skolem incentive.

Skolem incentives are also sent during ballotages (see sub section 5.6.1.2).

5.8. Putting It All Together

This section closes the description of AMBR by completing the example introduced in section 5.1. It shows how the mechanisms of the model work together.

After the target problem **CM1** is presented to the system, activation spreads in the network and brings relevant concepts and instances to the working memory (see section 5.2). Two base situations are activated most and become the major competitors to map to the target. These are the situations **FDO** and **WTP**. Of the two, **FDO** turns out to be the final winner. Figure 5.8.1. plots the *retrieval indices* of the two situations. The retrieval index is computed as the mean activation level of all agents affiliated to the respective situation-agent. It is, therefore, an aggregate numerical measure of the overall accessibility of each episode. Note that these indices are neither computed nor used by the model. They are instruments for monitoring the emergent behavior of the system from the point of view of an external observer.

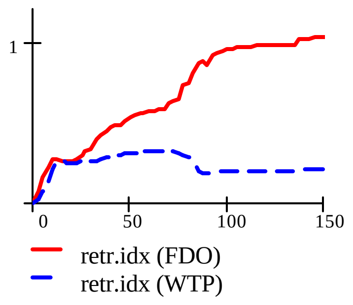


Figure 5.8.1. Retrieval indices for two competing coalitions: **FDO** (solid line) and **WTP** (dashed line). Time varies across the X-axis, retrieval indices across the Y-axis. See text for details.

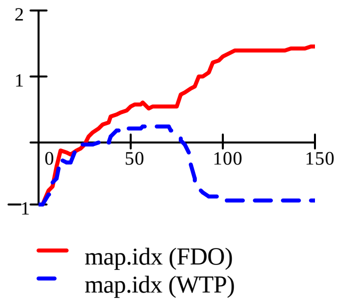


Figure 5.8.2. Mapping indices for two competing coalitions: **FDO** (solid line) and **WTP** (dashed line). Time varies across the X-axis, mapping indices across the Y-axis. See text for details.

Figure 5.8.1. shows that early during the run the two rival coalitions are equally active. Later on, however, **FDO** continues to grow while **WTP** flats out and then even goes down. This difference is due to the influence of the mapping process as discussed below.

Note that the winner coalition gets strength *gradually*. In other words, the base episode **FDO** is not retrieved in an all-or-nothing fashion. Instead, agents enter the working memory one by one. This is characteristic of the decentralized representation of situations discussed in Chapter IV. Transcript 5.8.1. lists the exact moments in which individual elements pass the working memory threshold. As evident from the transcript (and from the step-like increase of the retrieval index in Figure 5.8.1.), the description of the episode is retrieved from the long-term memory in three parts — roughly at times 8, 30, and 68.

The first group of agents enters the WM until time 8.20. It consists of the elements that are closest to the description of the target problem (cf. Figures 5.1.1. and 5.1.2. in section 5.1). The causal structure of the base episode is not unfolded yet. It is not present in the target either. Hence, the working memory now contains two descriptions of comparable complexity. This is favorable for the mapping process (cf. section 4.5.2).

```
T=0.40, adding t-of-FDO-o to WM.
T=0.42, adding in-FDO-do to WM.
T=0.78, adding t-of-FDO-f to WM.
T=0.80, adding oven-FDO to WM.
T=0.84, adding high-t-FDO to WM.
T=1.78, adding sit-FDO to WM.
T=1.80, adding dish-FDO to WM.
T=2.68, adding initst-FDO-1 to WM.
T=2.86, adding food-FDO to WM.
T=3.40, adding on-FDO to WM.
T=6.60, adding goalst-FDO to WM.
T=8.20, adding in-FDO-fo to WM.
T=25.30, adding interst-FDO to WM.
T=29.70, adding to-reach-FDO to WM.
T=29.80, adding cause-FDO-t to WM.
T=31.10, adding follows-FDO to WM.
T=31.20, adding endst-FDO to WM.
T=68.00, adding cause-FDO-i to WM.
T=68.10, adding initst-FDO-2 to WM.
```

Transcript 5.8.1. Transcript showing the moments in which various members of situation **FDO** enter the working memory.

Meanwhile, the marker passing and structure correspondence mechanisms generate a number of hypotheses. They register at their respective secretaries and incorporate into the constraint satisfaction network. The competition in the CSN can be monitored with the aid of the *mapping indices* plotted in Figure 5.8.2. The mapping index is an aggregate numerical measure of the strength of the hypotheses between two situations. Like the retrieval index, it is not used by the model itself.

At time 25 the hypotheses involving **FDO** elements start to dominate the CSN. The additional activation that they send to the main network allows a second group of agents to enter the working memory. These are the agents that explicate the causal structure of situation **FDO**. As this episode emerges as the likely winner, it is getting ready for the processes of transfer and evaluation.

There are more obstacles to be overcome, however. The leading set of correspondences includes an unwanted element — `tpot-WTP` and its supporting proposition `made-of-WTP` and `metal-WTP`. It manages to beat the agents from the leader situation because the latter does not have an explicit proposition about the material of its `dish-FDO`. As discussed earlier (subsection 5.6.1.2) the rating mechanism detects the blend and announces a ballotage. It also triggers the skolemization mechanism.

The semantic memory contains a general proposition that all cook vessels are made of metal. As the baking dish from situation **FDO** belongs to that category, the skolemization mechanism generates a Skolem proposition stating that it is made of metal too. This happens between time 80 and 87.60. Transcript 5.8.2. illustrates this and shows the *Skolem messages* that are exchanged during this process. Skolemization results in adding two new agents to the recipient situation.

```
T=80.00, #<SkI MADE-OF-CM1> received in MADE-OF-CM1<- ->CKVES-MD-METAL.
T=80.50  made-of-CM1<-->ckves-md-metal begins skolemization.
T=81.90, #<SM1 MADE-OF-CM1<-->CKVES-MD-METAL> received in *MATERIAL-METAL-1
T=82.10, #<SM1 *MATERIAL-METAL-1> received in MADE-OF-CM1<-->CKVES-MD-METAL
T=82.80, *material-metal-1 affiliates to sit-FDO.
T=84.20, #<SM2 MADE-OF-CM1<-->CKVES-MD-METAL> received in *MADE-OF-1.
T=85.00, #<SM2 *MADE-OF-1> received in MADE-OF-CM1<--> CKVES-MD-METAL.
T=87.60, *made-of-1 affiliates to sit-FDO.
```

Transcript 5.8.2. Transcript showing the events related to the skolemization mechanism. `#<SkI xxx>` is a Skolem incentive and `#<SM1 xxx>` and `#<SM1 xxx>` are Skolem messages of different kinds. See text for details.

Note that this in effect is a form of re-representation of the base aimed at bringing it in line with the target problem. As the target contains an explicit proposition about the material of the teapot, the source builds a corresponding counterpart. On the other hand, the proposition `shape-of(dish-FDO,` rectang-FDO) that is contained in the original description of that episode never enters the WM. This demonstrates the flexibility of the decentralized representation of AMBR situations.

After the skolemization, the mapping index of **WTP** (the competitor) rapidly goes down (see Figure 5.8.2). **FDO** is now clear and unambiguous winner. There are, however, some final rearrangements of the correspondences. In particular, the semantically grounded hypothesis `in-CM1<-->in-FDO-do` gives way to `in-CM1<-->on-FDO` under the influence of the structural constraint on mapping. The ambiguity between the two different `temperature-of` propositions in the base is also resolved. Table 5.8.1. lists the set of correspondences that are the leaders at three different times.

	T = 50	**T = 100**	**T = 200**
sit-CM1	sit-FDO	sit-FDO	sit-FDO
milk-CM1	oven-FDO	food-FDO	food-FDO
tpot-CM1	tpot-WTP	oven-FDO	dish-FDO
in-CM1	in-FDO-do	in-FDO-do	on-FDO
T-of-CM1	T-of-FDO-oven	T-of-FDO-food	T-of-FDO-food
low-T-CM1	high-T-FDO	high-T-FDO	high-T-FDO
made-of-CM1	made-of-WTP	*made-of-1	*made-of-1
metal-CM1	metal-WTP	*metal-1	*metal-1
initst-CM1	initst-FDO-1	initst-FDO-1	initst-FDO-1
goalst-CM1	goalst-FDO	goalst-FDO	goalst-FDO
to-reach-CM1	to-reach-FDO	to-reach-FDO	to-reach-FDO

Table 5.8.1. Leading correspondences for each target element at different times during the run. Target elements are listed in the left column. See text for details.

CHAPTER VI

SIMULATION EXPERIMENTS

6.1. Description of the Knowledge Base

This chapter reports the results of several simulation experiments performed with AMBR. The long-term memory of the model is the same for all experiments, with variation of some links as described below.

The LTM consists of 569 permanent agents. 273 of them are concept-agents and encode semantic knowledge about the micro-domain introduced in section 4.1. For example, it is represented that `tea`, `milk`, and `water` are subclasses of `drinkable-liquid`, which in turn is subordinate to `liquid`. The system 'knows' that `temperature-of` is a `physprop-relation` and that its first argument must be an `object` while the second one a `temperature-qualifier` such as `high-temp` or `low-temp`. The semantic memory also contains 49 instance-agents. Most of them are general propositions such as `heat-source-is-hot` and `bottle-made-of-glass`.

The remaining agents in the long-term memory represented twelve simple situations. These situations are outlined below. Appendix A contains a full description of one of them as taken directly from the source file fed to the program. Appendix B contains simplified representations in predicate calculus of all situations.

Base situation WTP (Water in a Teapot on a Plate): *There is some water in a teapot. The teapot is made of metal and its color is black. There is also a hot plate. The teapot is on the plate. The temperature of the plate is high.*

The goal is that the temperature of the water is high.

The outcome is that the temperature of the teapot is high because it is on the hot plate. In turn, this causes the temperature of the water to be high, as it is in the teapot.

Base situation BF (Bowl on a Fire burns out): *There is some water in a bowl. The bowl is made of wood. There is also a fire. The bowl is on the fire. The temperature of the fire is high.*

The goal is that the temperature of the water is high.

The outcome is that the bowl burns out because it is made of wood and is on the fire. In turn, this causes the water to dissipate, as it is in the bowl.

Base situation GP (Glass on a hot Plate breaks): *There is some water in a glass. The glass is made of [material] glass. There is also a hot plate. The glass is on the plate. The temperature of the plate is high.*

The goal is that the temperature of the water is high.

The outcome is that the glass breaks because it is made of [material] glass and is on the hot plate. In turn, this causes the water to dissipate, as it is in the glass.

Base situation IHC (Immersion Heater in a Cup)[13]: *There is some water in a cup. There is an immersion heater in the water. The immersion heater is hot. The cup is on a saucer. The cup is made of china.*

The goal is that the temperature of the water is high.

The outcome is that the temperature of the water is high due to the hot immersion heater in it.

Base situation FDO (Food on a Dish in an Oven)[14]: *There is a baking dish and some food on it. The shape of the dish is rectangular. There is also an oven. The dish is in the oven. The temperature of the oven is high.*

The goal is that the temperature of the food is high.

Since the food is on the dish which in turn is in the oven, the food is in the oven too. This causes the temperature of the food to be high, as the temperature of the oven is high.

Base situation MTF (Milk in a Teapot in a Fridge): *There is some milk in a teapot. The color of the teapot is green. There is also a fridge. The teapot is in the fridge. The temperature of the fridge is low.*

The goal is that the temperature of the milk is low.

Since the milk is in the teapot which in turn is in the fridge, the milk is in the fridge too. This causes the temperature of the milk to be low, as the temperature of the fridge is low.

Base situation ICF (Ice Cube in a Fridge)[15]: *There is an ice cube on a glass. The glass is made of [material] glass. There is also a fridge. The glass is in the fridge. The temperature of the fridge is low.*

The goal is that the temperature of the ice cube is low.

Since the ice cube is on the glass which in turn is in the fridge, the ice cube is in the fridge too. This causes the temperature of the ice cube to be low, as the temperature of the fridge is low.

[13] See Figure 6.3.1.2. for a schematic diagram.
[14] See Figure 5.1.2. for a schematic diagram.
[15] See Figure 6.3.1.3. for a schematic diagram.

Base situation BPF (Butter on a Plate in a Fridge): *There is some butter on a plate. The plate is made of china and its shape is circular. There is also a fridge. The plate is in the fridge. The temperature of the fridge is low.*

The goal is that the temperature of the butter is low.

Since the butter is on the plate which in turn is in the fridge, the butter is in the fridge too. This causes the temperature of the butter to be low, as the temperature of the fridge is low.

Base situation STC (Sugar in Tea in a Cup): *There is some tea in a cup. There is some sugar in the tea. The taste of the sugar is sweet. The cup is on a saucer.*

The goal is that the taste of the tea is sweet.

The outcome is that the taste of the tea is sweet due to sugar in it.

Base situation SFF (Salt in Food in a Fridge): *There is some food on a plate. There is some salt in the food. The taste of the salt is salty. There is also a fridge. The temperature of the fridge is low.*

The goal is that the temperature of the food is low.

The outcome is that the food is both cold and salty. Since the food is on the plate and the plate is in the fridge, the food is in the fridge too. This causes the temperature of the food to be low. In the same time, the salt that is in the food causes its taste to be salty.

Base situation ERW (Egg in Red Water): *There is some water in a teapot. The color of the water is red. The teapot is made of metal. There is also an egg which is in the water.*

The goal is that the color of the egg is red.

The outcome is that the color of the egg is red because it is in the red water.

Base situation GWB (Glass in a Wooden Box): *There is a glass. It is made of [material] glass. The glass is in a box. The box is made of wood.*

The goal is that the box protects the glass.

The outcome is that the box protects the glass.

The verbosity of these (simplified) descriptions reveal how much knowledge is involved even in the seemingly trivial task of heating water. As simple and monotonous as they are, the twelve situations are designed to highlight various subprocesses of analogy-making. The descriptions involve objects and relations in different combinations and at various levels of similarity. Many episodes involve identical objects but are not isomorphic. Others go the other way around. Some episodes fail to achieve their goal and/or have side effects besides the main goal.

6.2. Statistics Over 1000 Runs

6.2.1. Experimental Setting

This section tests the behavior of the model on ten target problems. The goal is to check whether the model can reliably access episodes from long-term memory and map them to the target.

Each target problem is run 100 times, yielding a total of 1000 runs for the ten problems. All parameters of the model are kept constant across all runs (and across all experiments reported in this Thesis in general).

The architecture DUAL is completely deterministic. The behavior of a DUAL-based model such as AMBR depends on five factors (*i*) the target problem, (*ii*) the contents of the long-term memory, (*iii*) the order of presentation of target elements (*order effect*), (*iv)* the residual activation in the long-term memory (*priming effect*), and (*v*) the external environment (*context effect*). The experiments reported in this section vary the first factor as independent variable and use the second one as source of replications. The remaining factors are kept constant. (They are explored in separate experiments. Kokinov (1994a) has demonstrated priming and context effects in an earlier version of AMBR. Order effects are explored in section 6.4. below.)

The knowledge base is replicated 100 times for the purpose of the experiments. Each variant contains the same 569 permanent agents outlined in section 6.1. Most of the links among them are the same too. There are, however, some links that vary randomly across the 100 variants. They are 'top-down' links from concepts to instances (i.e. links labeled `instance`). The sampling procedure for picking up links for each KB variant is designed to approximate the (unimplemented) mechanism for dynamic 'privileged instances' suggested in section 4.3. A small number of associative links (`a-link`) also differ randomly across KB variants. Thus it could be said that each variant represents a 'snapshot' of the long-term memory of the system. The core KB contains approximately 3000 links. Each variant adds about 100 new links (which amounts to less than 4% of the total network connectivity).

Each target problem is run on each KB variant for 200 time units. This period is enough for the model to promote a winner situation in all but one of the 1000 runs. (In this exceptional run the model failed to access any episode from LTM to a sufficient degree.) The dependent variable is the number of times that each source situation is accessed and mapped to the particular target problem.

The activation level of all permanent agents is set to zero at the beginning of each run (i.e. there is no priming). Each target situation is represented by temporary agents. Some of them are attached to the goal and input nodes of the system. All attachments are done simultaneously at the beginning. The input node does not activate any agents apart from the target elements (i.e. the external context is ignored).

6.2.2. Heating Milk

The first pair of problems that are presented to the system involve heating milk (in the micro-domain). There are complementary to each other in the sense that the first has an explicit representation of the goal but the initial conditions are incomplete. In contrast, the second problem specify the initial arrangement in full and asks about the expected outcome of this arrangement. Appendix B contains simplified representations in predicate calculus of all target situations.

Target situation HM1 (Heating Milk, variant 1): *There is a teapot and some milk in it. The teapot is made of metal.*

The goal is that the temperature of the milk is high.

Target situation HM2 (Heating Milk, variant 2): *There is a teapot and some milk in it. The teapot is made of metal. There is also a hot plate. The teapot is on the plate. The temperature of the plate is high.*

The goal, if any, is not represented explicitly.

What is the outcome of this state of affairs?

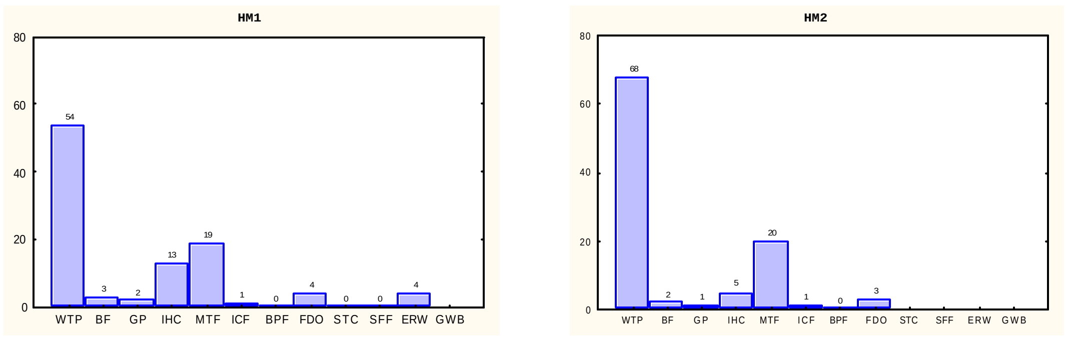


Figure 6.2.2 Bar plots showing the frequencies of mapping each long-term memory episode to target problems **HM1** and **HM2**, respectively.

The bar plots in Figure 6.2.2 demonstrate that in the majority of cases (54% of the runs) the model maps the target **HM1** to the prototypical source episode about heating liquids — situation **WTP**. In these cases AMBR notices the analogy in which `milk-HM1` maps to `water-WTP`.

Two other sources stand out against the rest. Situation **MTF** is another good match. Its liquid is the same, but it requires the reversal `high-temperature<-->low-temperature`. The fact that it is three times less frequent than **WTP** demonstrates that AMBR is sensitive to pragmatic pressures. The same pressures explain the frequency of situation **IHC** too — it represents an alternative way to heat liquids (by an immersion heater).

The second variant of the target problem (**HM2**) generates a similar pattern. The main difference is that situation **WTP** becomes even stronger (68%) at the expense of **IHC**. After all, the target problem contains a hot plate, not an immersion heater.

The bar plots reveal also that the model is not confined to the most obvious solutions to a problem. It reaches them most of the time (as it should) but occasionally it chooses more remote analogs. These are the cases with frequencies below 5% in the graphs. Most of them are episodes having some superficial similarity to the target: a teapot, goal related to high temperature, etc. These low-frequency answers are an important attestation of AMBR's flexibility.

6.2.3. Cooling Milk

The second pair of problems is similar to the first except that it deals with low temperatures. It tests whether AMBR is able to respond to a small (yet crucial) change in the target description.

Target situation CM1 (Cooling Milk, variant 1)[16]: *There is a teapot and some milk in it. The teapot is made of metal.*

The goal is that the temperature of the milk is low.

Target situation CM2 (Cooling Milk, variant 2): *There is a teapot and some milk in it. The color of the teapot is black. There is also a fridge. The teapot is in the fridge. The temperature of the fridge is low.*

What is the probable goal for this arrangement?

The outcome of this state of affairs is not known.

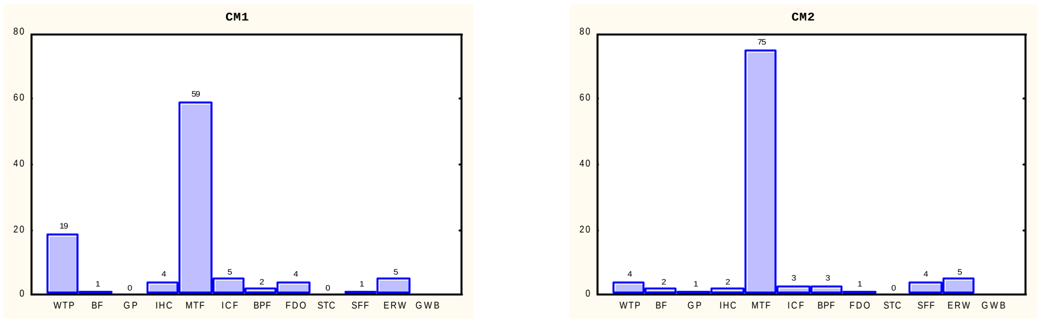


Figure 6.2.3 Bar plots showing the frequencies of mapping each long-term memory episode to target problems **CM1** and **CM2**, respectively.

A brief comparison between the left plots in Figures 6.2.2 and 6.2.3 reveals that change in the filler of a single slot in a single target agent can turn the behavior of AMBR to 180 degrees. Specifically, the `inst-of` slot of `low-T-`

16 See Figure 5.1.1. for a schematic diagram.

`CM1` (the second argument of `temperature-of-CM1`) is filled with a reference to the concept agent `low-temperature` while the respective slot in `high-T-HM1` points to `high-temperature`. (The names of the agents themselves are of course irrelevant.) This change is small but it is in a very important place — the respective agent is attached the goal node and the activation it provides to its parent concept is a major determinant of the overall content of the working memory. As a consequence, **CM1** maps to **MTF** in 59% of the runs versus 19% for **WTP**. In contrast, the respective percentages for the target problem **HM1** are 54% vs. 19%. (Recall that the experiment uses within subject design as the two targets run over the same set of knowledge bases.) Clearly, the pragmatic constraint plays an important role in AMBR.

Let us now turn to the other problem in the pair: **CM2**. It is literally similar to the base situation **MTF** (Gentner, 1983, 1989). The only difference in the two descriptions, apart from the incompleteness of the target, is the color of the teapots. As seen in Figure 6.2.3, **MTF** wins in full 75% of the cases. This is the maximal frequency among all 1000 runs. All rival episodes occur with marginally low probabilities. This suggests that AMBR models accurately the empirical finding that analog access is dominated by literal similarities (Gentner & Landers, 1985; Holyoak & Koh, 1987; Ross, 1987).

6.2.4. When the Container is Fragile

The next pair is inspired by the target problem from experimental studies on priming effects (Kokinov, 1990, 1994a). The subjects in these studies were asked how one could heat water in a wooden bowl in a forest. Kokinov (1994a) performed related simulation experiments in the micro-domain.

Target situation WB1 (Water in a Bowl): *There is a bowl and some milk in it. The bowl is made of wood.*

The goal is that the temperature of the water is high.

Target situation WG1 (Water in a Glass): *There is a glass and some water in it. The glass is made of [material] glass.*

The goal is that the temperature of the water is high.

As evident from Figure 6.2.4, the model is split between two responses to the first problem. **WTP** is the prototypical case for heating liquids. It could not generate a good solution to the problem, however, as it suggests to put the bowl on the fire where it would burn. Still, it provides a sound match to the target. The other strong episode is **BF** which is an unsuccessful past attempt to solve this problem.

Note that the source analog that provides the 'immersion heater' solution (**IHC**) works in only 6% of the cases. Incidentally, the subjects of (Kokinov, 1990) had similar difficulties in the absence of priming.

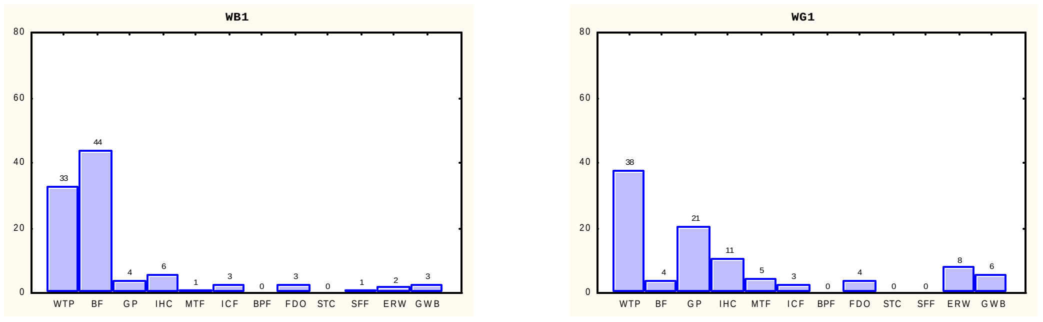


Figure 6.2.4 Bar plots showing the frequencies of mapping each long-term memory episode to target problems **WB1** and **WG1**, respectively.

In an attempt to increase the probability of using the immersion heater, target problem **WG1** replaces the wooden bowl with a glass. (Situation **IHC** involves a cup.) The attempt is moderately successful — the frequency of **IHC** increases to 11%. As a side effect, situation **GP** takes the place of **BF**. A look at the descriptions of these two episodes (see section 6.1) shows that this is to be expected.

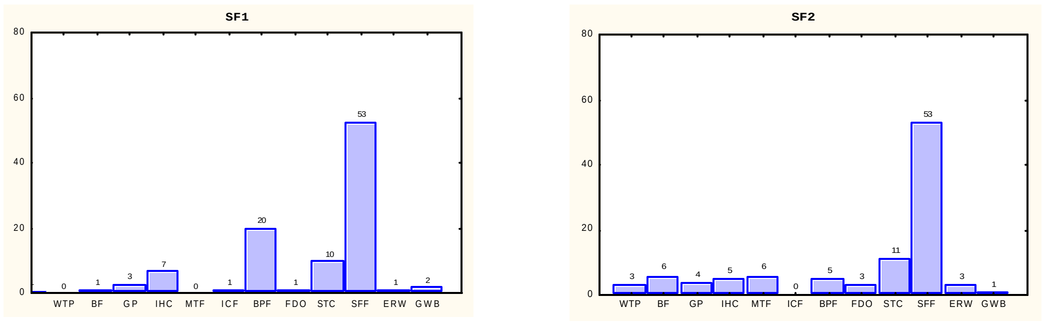


Figure 6.2.5 Bar plots showing the frequencies of mapping each long-term memory episode to target problems **SF1** and **SF2**, respectively.

6.2.5. Scaling Up: Problems Involving Taste

The problems in this subsection go away from the temperature-related focus of the current knowledge base. They deal with tastes and are intended to check whether the model is able to switch to this different thematic line. The base episodes are added to the long-term memory for similar reason.

Target situation SF1 (Salty Food, variant 1): *There is a plate and some food on it. The plate is made of china.*

The goal is that the taste of the food is salty.

Target situation SF2 (Salty Food, variant 2): *There is a plate and some food on it. There is some salt in the food.*

The goal, if any, is not represented explicitly.

What is the outcome of this state of affairs?

The bar plots in Figure 6.2.5. show that indeed the two episodes related to taste are accessed by these targets. Note also that situations **STC**, **SFF**, and **BPF** almost never show up for other target problems. This gives reasons for some optimism about the ability of AMBR to scale up to larger memory sizes. It suggests that adding more and more episodes and different 'thematic lines' will not lead to diffusion of the answers. Of course, this topic should be explored more rigorously with future (and bigger) versions of the knowledge base. We fully agree that memory of 12 episodes is very insufficient to support any serious claims about the scalability of the model.

6.2.6. Two Final Problems

Target situation EHW (Egg in Hot Water): *There is a teapot and some water in it. There is an egg in the water. The teapot is made of metal. The color of the egg is white. The temperature of the water is high.*

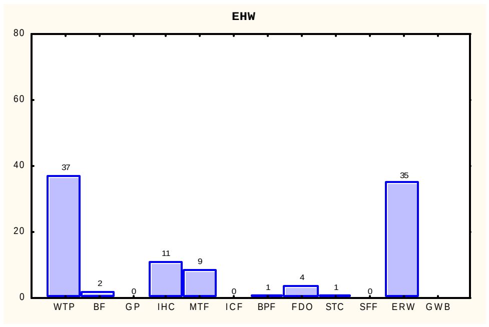
The goal, if any, is not represented explicitly.

What is the outcome of this state of affairs?

Target situation ICC (Ice Cube in Coke): *There is a glass and some coke in it. The glass is made of [material] glass. There is an ice cube in the coke. The temperature of the ice cube is low. There is also a table. The glass is on the table.*

The goal, if any, is not represented explicitly.

What is the outcome of this state of affairs?



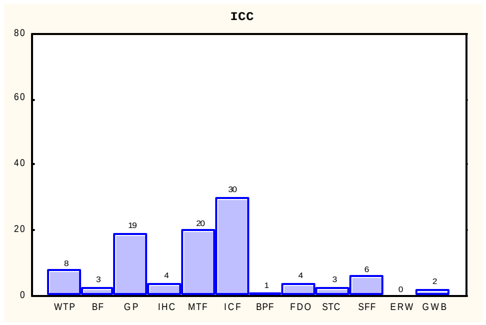


Figure 6.2.6 Bar plots showing the frequencies of mapping each long-term memory episode to target problems **EHW** and **ICC**, respectively.

The target problem **EHW** and its long-term memory counterpart **ERW** are added for similar scaling-up considerations. Problem **ICC** is used for the experiment discussed in section 6.3. The respective bar plots are shown in Figure 6.2.6.

6.2.7. Variability and Determinism

A few final remarks are in order before closing this section. Table 6.2.7. presents the joint distribution produced by all 1000 runs. The few empty cells in the table (the first column notwithstanding) indicate that the model populates all regions of its 'problem space'. That is, there is some small probability to map any source analog to almost any target. No possibilities are ruled out *a priory*. On the other hand, AMBR focuses on the episodes that best fit any given problem. It is efficient without being rigid. This is a consequence of the dynamic emergent style of computation that is characteristic of DUAL (Kokinov, Nikolov, & Petrov, 1996).

Note also that although AMBR is completely deterministic, it is still able to demonstrate the variability of behavior evident from the table. As described in subsection 6.2.1., the random factor in the experiment amounts to less than 4% of the initial links in the long-term memory. Nevertheless, each target problem generates a whole range of answers. This is again a consequence of the dynamic emergent style of computation. The macroscopic behavior of the system depends on a multitude of interrelated microscopic factors. A small change in the initial conditions can drift the global outcome far away in the problem space. Therefore, the macroscopic behavior of AMBR must be analyzed in probabilistic terms even though all microscopic mechanisms are deterministic.

	??	**WTP**	**BF**	**GP**	**IHC**	**MTF**	**ICF**	**BPF**	**FDO**	**STC**	**SFF**	**ERW**	**GWB**	**To-tal**
HM1		54	3	2	13	19	1		4			4		100
HM2		68	2	1	5	20	1		3					100
CM1		19	1		4	59	5	2	4		1	5		100
CM2		4	2	1	2	75	3	3	1		4	5		100
WB1		33	44	4	6	1	3		3		1	2	3	100
WG1		38	4	21	11	5	3		4			8	6	100
SF1	1		1	3	7		1	20	1	10	53	1	2	100
SF2		3	6	4	5	6		5	3	11	53	3	1	100
EHW		37	2		11	9		1	4	1		35		100
ICC		8	3	19	4	20	30	1	4	3	6		2	100
Total	1	264	68	55	68	214	47	32	31	25	118	63	14	1000

Table 6.2.7. Joint distribution for all 1000 runs. Each cell gives the frequency of accessing and mapping a target problem (row) to a source episode (column).

6.3. Influence of Mapping on Analog Access

6.3.1. Simulation Experiment Method

This section presents a case study exploring the integration of analog access and mapping in AMBR. It contrasts two strategies for combining access and mapping — parallel vs. serial.

6.3.1.1 Design

The experiment consists of two conditions. Both conditions involved running the model on a target problem. In the 'parallel condition', AMBR operates in its normal manner with the mechanisms for access and mapping working in parallel. In the 'serial condition', the program is artificially forced to work serially — first to access and only then to map. The target problem and the content of the long-term memory are identical in all runs. The topics of interest fall into two categories — the final mapping constructed by the program and the dynamics of the underlying computation. The latter is monitored by recording a set of variables describing the internal state of the system at regular time intervals throughout each run.

6.3.1.2. Materials

The experiment uses the knowledge base described in section 6.1. Situation **ICC** (Ice Cube in Coke) is the target problem. Its verbal description is given in subsection 6.2.6. Two of the twelve episodes are most important for the present discussion: situations **IHC** (Immersion Heater in a Cup with water) and **ICF** (Ice Cube in a Fridge).

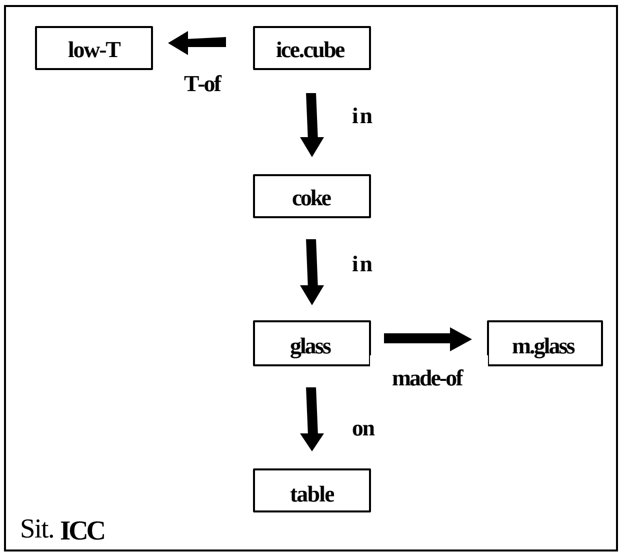


Figure 6.3.1.1. Schematized representation of target situation **ICC**. Objects are shown as boxes and relations as arrows. The actual AMBR representation is more complex and consists of 15 agents.

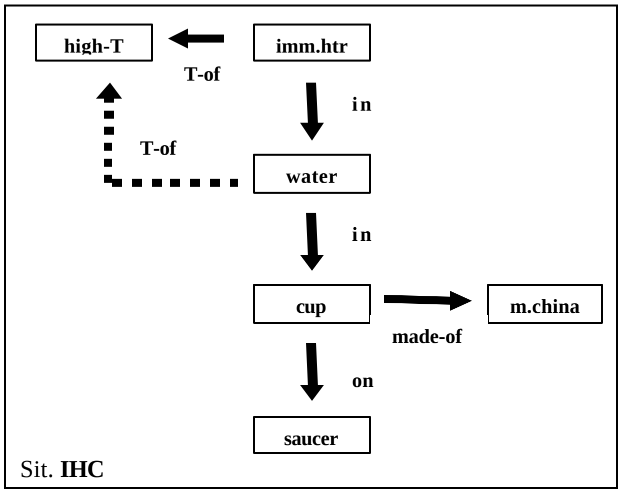


Figure 6.3.1.2. Schematized representation of situation **IHC**. Dashed arrows stand for relations in the ‘outcome’. The actual AMBR representation is more complex — it consists of 19 agents and explicates the causal structure (not shown in the figure).

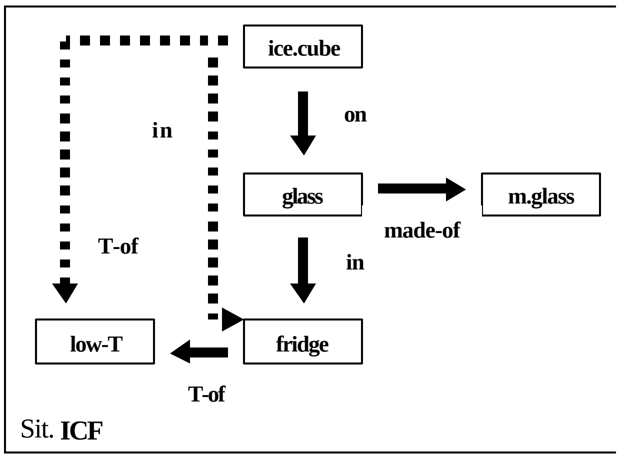


Figure 6.3.1.3. Schematized representation of situation **ICF**. The actual AMBR representation is more complex — it consists of 21 agents and explicates the causal structure (not shown in the figure).

As evident from Figures 6.3.1.1, 2, and 3, both situations **IHC** and **ICF** may be considered similar to the target problem. There are some differences, however. Situation **ICF** involves the same objects and relations as the target but the structure of the two are different. In contrast, situation **IHC** involves different objects but its system of relations is completely isomorphic to that of the target. According to Gentner (1989), the pair **IHC–ICC** may be classified as analogy while **ICF-ICC** as mere appearance. Thus it is expected that situation **ICF** would be easier to retrieve from the total pool of episodes stored in LTM. On the other hand, **IHC** would be more problematic to retrieve but once accessed it would support better mapping.

6.3.1.3. Procedure

The model is run two times on the target problem. The two runs carry out the 'parallel' and the 'serial' conditions of the experiment, respectively. The contents of the long-term memory and the parameters of the model are identical in the two conditions.

Recall that situations have decentralized representations in AMBR. The target problem is represented by a coalition of 15 agents standing for the ice-cube, the glass, two instances of the relation `in` and so on (See Appendix B). 12 of these agents are attached to the special nodes that serve as activation sources in the model. The attachment is the same in the two experimental conditions.

In the parallel condition, the model is allowed to run according to its specification. That is, all AMBR mechanisms run in parallel, interacting with each other. The program iterates until the system reaches a resting state. A number of variables are recorded at regular intervals throughout the run. Out of these many variables, the *retrieval index* is of special interest. It is computed as the average activation level of the agents involved in each situation.

In short, the data accumulated at the end of the run are the final mapping constructed by the program and a log file of the retrieval indices of all twelve situations from the LTM.

In the serial condition, the target problem is attached to the activation source in the same way and the same data were collected. However, the operation of the program is forcefully modified to separate the processes of access and mapping. To that end, the run is divided in two steps.

During step one, all mapping mechanisms in AMBR are manually switched off. Thus, spreading activation is the only mechanism that remains operational. It is allowed to work until the pattern of activation reaches asymptote. The situation with the highest retrieval index is then identified. If we hypothesize a 'retrieval module', this is the situation that it would access from LTM.

After the source analog is picked up in this way, the experiment proceeds with step two. The mapping mechanism is switched back on again but it is allowed to work only on the source situation retrieved at step one. This situation is mapped to the target. Thus, the data at the end of the second run are the final mapping constructed at step two and two logs of the retrieval indices.

6.3.2. Results and Discussion

In both experimental conditions the model settles in less than 150 time units and produces consistent mappings. By 'consistent' we mean that each element of the target problem is unambiguously mapped to an element from LTM and that all these corresponding elements belong to one and the same base situation. Stated differently, the mappings are one-to-one and there are no blends between situations.

In the parallel condition, the target problem maps to situation **IHC**, yielding the correspondences *in–in, water–coke, imm.heater–ice.cube, T.of–T.of, high.T–low.T, made.of–made.of*, etc. Four elements from the source situation remain unmapped and in particular the agent representing that the water is hot. This proposition is a good candidate for inference by analogy. *Mutatis mutandis*, it could bring the conclusion that the coke is cold.

In the serial condition, situation **ICF** wins the retrieval stage. This is explained by the high semantic similarity between its elements and those of the target — both deal with ice cubes in glasses, cold temperatures, etc. The asymptotic level of the retrieval index for **ICF** is about four times greater than that of any other situation. In particular, situation **IHC** ends up with only 5 out of 19 agents passing the working memory threshold.

According to the experimental procedure, situation **ICF** is then mapped to the target during the second stage of the run. The correspondences that emerge during the latter stage are shown in Table 6.3.2. The semantic similarity constraint dominates this run. This is not surprising given the high degree of superficial similarity between the two situations. There is, however, a serious flaw in the set of correspondences. The proposition `T-of(ice-cube-ICC, low-T-ICC)`, which belongs to the *initial* state of the target, is mapped to the proposition `T-of(ice-cube-ICF, low-T-ICF)`, which is a *consequence* in the source. Therefore, the whole analogy between the target problem and situation **ICF** could hardly generate any useful inference.

Base situation ICF	**Target situation ICC**
ice-cube	ice-cube
fridge	coke
glass	glass
in (ice-cube, fridge)	in (ice-cube, coke)
in (glass, fridge)	in (coke, glass)
on (ice-cube, glass)	on (glass, saucer)
T-of (fridge, low-T)	<unmapped>
T-of (ice-cube, low-T)	T-of (ice.cube, low-T)
low-T	low-T
made-of (glass, m.glass)	made-of (glass, m.glass)
m.glass	m.glass
initstate1	initstate
initstate2	<unmapped>
interstate	table
endstate	endstate

`goalstate`	`<unmapped>`
`follows (initstate1, endst)`	`follows (initstate, endst)`
`to-reach (initstate1, goalst)`	`<unmapped>`
`cause(initstate2,in(ic,fr))`	`<unmapped>`
`cause(interstate,T-of(ic,lT))`	`<unmapped>`

Table 6.3.2. Correspondences constructed by the model in the serial condition.

To summarize, when the mechanisms for access and mapping work together, the model constructs an analogy that can potentially solve the problem. On the other hand, when the two mechanisms are separated, the retrieval stage favors a superficially similar but inappropriate base.

The presentation so far concentrated on the final set of correspondences produced by the model. We now turn to the dynamics of the computation as revealed by the retrieval indices. Figure 6.3.2.1 plots the retrieval indices for the two critical LTM episodes during the first run of the program (i.e. when access and mapping work in parallel). Figure 6.3.2.2 concentrates on the early stage of the first run and compares it with the second run (i.e. when only the access mechanism is allowed to work). Note that the two plots are in different scales.

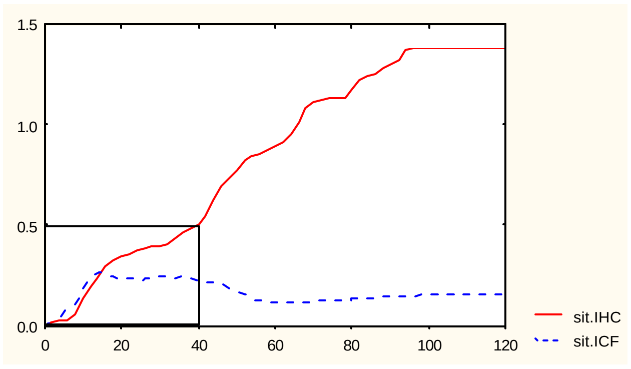


Figure 6.3.2.1. Plot of retrieval indices versus time for the parallel condition. The 'south-west' corner of the plot is reproduced in Figure 6.3.2.2. with threefold magnification.

These plots tell the following story: At the beginning of the parallel run, several situations are probed tentatively by bringing a few elements from each into the working memory. Of this lot, **ICF** (with the ice cube) looks more promising than any of its rivals as it has so many objects and relations in common with the target. Therefore, about half of the agents belonging to situation **ICF** enter the working memory and begin trying to establish correspondences between themselves and the target agents. The active members of the rival situations are doing the same thing, although with lower intensity — their symbolic processor are slower.

At about 15 time units since the beginning of the simulation, however, situation **IHC** (with the immersion heater) rapidly gains strength and eventually overtakes the original leader. At time 40, it takes the lead and gradually transforms its small advantage into an uncompromising triumph.

The final victory of situation **IHC**, despite its lower semantic similarity compared to situation **ICF**, is due to the interaction between the mechanisms of access and mapping in AMBR. More precisely, in this particular case it is the mapping that radically changes the course of access. To illustrate the importance of this influence, Figure 6.3.2.2 contrasts the retrieval indices with and without mapping.

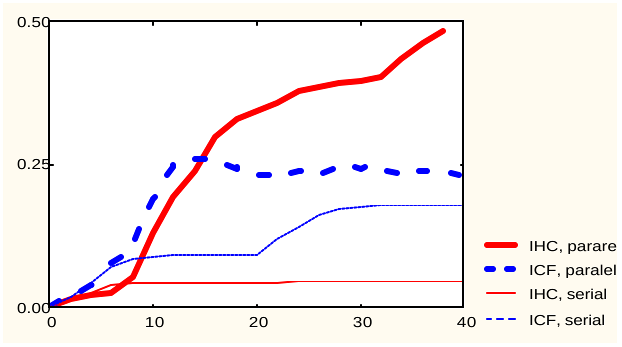


Figure 6.3.2.2. Retrieval indices for situations **IHC** and **ICF** with and without mapping influence on access. The thick lines correspond to the parallel condition and replicate (with threefold magnification) the lines from the 'south-west' corner of Figure 6.3.2.1. The thin lines show 'pure' retrieval indices.

The thin lines in the figure show the retrieval indices for the two situations when mapping mechanisms are suppressed. Thus, they indicate the 'pure' retrieval index of each situation — the value that is due to the access mechanism alone. The index for situation **ICF** is much higher than that of **IHC** and, therefore, **ICF** is used as source when the mapping is allowed to run only after the access has finished.

The step-like increases of the plots indicate moments in which an agent (or usually a tight sub-coalition of two or three agents) passes the working memory threshold (cf. Transcript 5.8.1). This happens, for instance, with situation **ICF** between time 20 and 30 of the serial condition (the thin dashed line in Figure 6.3.2.2). Thus, accessing a source episode in AMBR is not an all-or-nothing affair. Instead, situations enter the working memory agent by agent and this process extends far after the beginning of the mapping. In this way, not only can the access influence the mapping but also the other way around.

In the interactive condition the mapping mechanism boosts the retrieval index via what we call a *bootstrap cascade*. This cascade operates in AMBR in the following way. First, the access mechanism brings two or three agents of a given situation into the working memory. If the mapping mechanism then detects that these few agents can be plausibly mapped to some target elements, it constructs new correspondence nodes and links in the AMBR network. This creates new paths for the highly active target elements to activate their mates. The latter in turn can then activate their 'coalition partners', thus bringing a few more agents into the working memory and so on.

The bootstrap cascade is possible in AMBR due to two important characteristics of this model. First, situations have decentralized representations which may be accessed piece by piece. Second, AMBR is based on a parallel cognitive architecture which provides for concurrent operation of numerous interacting processes. Taken together, these two factors enable seamless integration of the subprocesses of access and mapping in analogy-making.

6.4. Order Effect on Analog Access

6.4.1. Simulation Experiment Method

This section presents an experiment testing the prediction made in subsection 5.2.4. — the order of presentation of target elements affects the frequency of accessing episodes from memory. More concretely, source analogs containing elements which are semantically similar to a given target element are accessed more frequently when this target element is attached earlier to the input node.

6.4.1.1 Design

The experiment consists of three conditions. The same target problem is presented to the system in all three conditions. In the control condition all target elements are attached simultaneously to the input and goal nodes. In the two experimental conditions the elements are attached in two different (and roughly reverse) orders. The dependent variables are frequencies of accessing and mapping the episodes in the long-term memory.

6.4.1.2. Materials

Target situation **EHW** presented in section 6.2.6. is used as a target problem. Its verbal description is reproduced below. The 100 variants of the knowledge base described in section 6.2.1. are used as replications.

Target situation EHW (Egg in Hot Water): *There is a teapot and some water in it. There is an egg in the water. The teapot is made of metal. The color of the egg is white. The temperature of the water is high.*

The goal, if any, is not represented explicitly.

What is the outcome of this state of affairs?

Note the following details of this description. On one hand, there is some water whose temperature is high. These elements are similar to the source analogs related to heating water and in particular to situations **WTP** (Water in a Teapot on a Plate) and **IHC** (Immersion Heater in a Cup). On the other hand, there is an egg whose color is white. These elements are similar to situation **ERW** (Egg in Red Water) described in section 6.1.

6.4.1.3. Procedure

The target problem is run three times on the set of 100 knowledge bases, yielding a total of 300 runs. In the control condition, all target elements are attached to the input node at the beginning of the run. The number of times that each of the twelve episodes in the long-term memory are accessed and mapped is recorded.

In the *hot water condition* the agents `water-EHW`, `T-of-EHW`, and `high-T-EHW` are attached to the input node at time zero. The remaining target elements are attached later according to the schedule shown in the left column of Table 6.4.1.3.

In the *colored egg condition* the agents `egg-EHW`, `color-of-EHW`, and `white-EHW` are attached to the input node at time zero. The remaining target elements are attached later according to the schedule shown in the right column of Table 6.4.1.3.

time	'hot-water' condition	'colored-egg' condition

0	`water`
	`high-T`
	`T-of(water,high-T)`
		`egg`
		`white`
		`color-of(egg,white)`

5	`teapot`	`teapot`

10	`metal`
	`made-of(teapot,metal)`
		`metal`
		`made-of(teapot,metal)`

15	`in(water,teapot)`	`water`
20	`egg`	`in(egg,water)`
25	`in(egg,water)`	`in(water,teapot)`

30	`white`
	`color-of(egg,white)`
		`high-T`
		`T-of(water,high-T)`

35	`endst`
	`follows(initst,endst)`
		`endst`
		`follows(initst,endst)`

Table 6.4.1.3. Time schedule for attaching different target elements in the two experimental conditions.

6.4.2. Results and Discussion

Figure 6.4.2.1. shows bar plots of the frequencies obtained in the two experimental conditions. The bar plot for the control condition is shown in Figure 6.2.6. is section 6.2

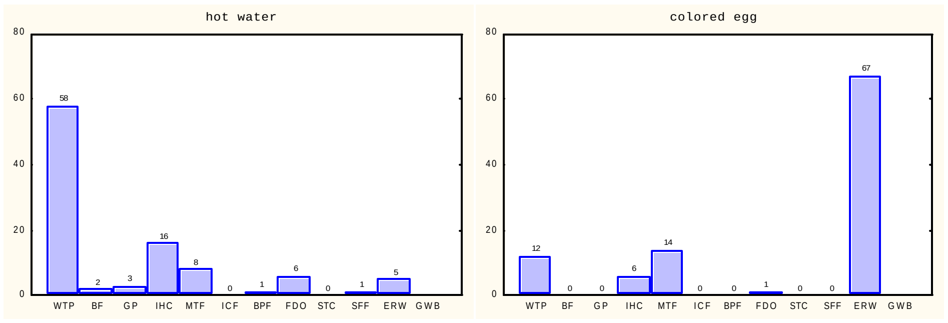


Figure 6.4.2.1. Bar plots showing the frequencies of accessing and mapping each long-term memory episode in the 'hot water' and 'colored egg' conditions, respectively.

The data show that each experimental condition differs from the control and from each other. The difference is very significant according to the chi-square test (χ^2=89.5, df=7, p<0.00001). Moreover, the effect is in the predicted direction — the two base situations about heating water appear much more frequently in the 'hot water' condition. The reverse pattern holds for the episode about coloring an egg (**ERW**).

Condition	freq(WTP)	freq(IHC)	freq(ERW)	other	total
Hot water	58 (37)	16 (11)	5 (35)	21 (17)	100
Color egg	12 (37)	6 (11)	67 (35)	15 (17)	100
Total	70 (74)	22 (22)	72 (70)	36 (34)	200

Table 6.4.2. Observed frequencies of accessing base episodes from memory for the two experimental conditions. The control condition (in parentheses) defines the expected frequencies for the chi-square test. χ^2=89.5, df=7, p<0.00001.

Thus, order of presentation of the target problem influences the process of accessing source analogs in AMBR. As the mapping process in the model is intimately intertwined with access, it is influenced too. Moreover, the direction of influence is in accord with the well-known *primacy effects* demonstrated in many studies of short-term memory (e.g. Postman & Phillips, 1965). Elements that appear earlier have greater impact than later elements.

AMBR differs from other models of analog retrieval with respect to the primacy effect (Forbus, Gentner, & Law, 1994; Hummel & Holyoak, 1997). As far as we can judge from the articles, neither MAC/FAC nor LISA predict such order effect on analog access. The first stage of MAC/FAC depends on dot products over feature vectors and, therefore, all target elements necessarily enter simultaneously. Thus the model must wait until all target ele ments are available and only then can trigger the retrieval process.

LISA do present target (or more precisely *driver*) elements in a temporal order. Indeed, this is the only way of processing available to LISA due to the limitations of the phase set. As argued in section 4.5.1, however, the model uses centralized representation of situations. Therefore, episodes are retrieved as units — either all nodes are flipped from *dormant* to *recipient* mode or none of them. In the current version of LISA this decision is taken probabilistically based on the *Luce retrieval index* computed for each episode in LTM (Hummel, personal communi cation, January 1998). The important point is that the indices are computed after multiple iterations through the whole driver set. The article does not specify the moment in which the probabilistic decision about bringing an episode to the working memory is taken (Hummel & Holyoak, 1997). If we suppose that this happens after the network has settled, the order of the driver set would have negligible effect on the retrieval indices.

CHAPTER VII

EVOLVING AMBR: AMBR4 ?

Throughout this thesis we have emphasized that analogy-making cannot be decomposed into a sequence of independent components. AMBR advocates an interactionist emergent approach and conceptualizes analogy-making in terms of overlapping subprocesses (Figure 3.2.1.2). Still, the current version of the model addresses mainly the subprocesses of access and mapping. Does this mean that AMBR assumes that these two subprocesses can be modeled separately from the rest?

The problem lies in the complexity of analogy-making. As we have argued, it is not an isolated module but emerges out of the general cognitive architecture. Therefore, any comprehensive treatment of analogy-making should cover the whole cognition. For instance, Chalmers, French, & Hofstadter (1992) have argued that analogy is inseparable from high-level perception. Without perception, the mapping between the base and the target is in effect contained in latent form in the representations of the two episodes. As AMBR starts from hand-coded descriptions, it takes the essence of the analogy from outside. This definitely is a limitation of the model. It should try to include mechanisms for high-level perception in the future. But high-level perception is obviously rooted in low-level perception. Perception at all levels involves attention, which in turn depends on motivation, which is culturally grounded, etc., etc.

It follows that *any* model of analogy-making is necessarily incomplete. AMBR makes no exception. We hope, however, that it is open-ended enough to be able to grow. This chapter suggests ways for extending the model in two directions: transfer and perception.

7.1. Possibilities for Transfer

In the current version of AMBR, each run of the model ends in the following way: One by one the *authorized secretaries* (i.e. the agents from the target, see section 5.6) pick up one of the hypotheses registered at them and send it a *promotion incentive*. The promoted agents enter the third phase of their life cycle and become *winner hypotheses*. All other hypotheses registered at the respective secretaries become losers. In this way, the model makes commitments about the correspondences between the two episodes. As there are no mechanisms that can advance the process further, the model stops. All symbolic activity comes to an end. The activation in the network reaches a resting state.

In a hypothetical future version of AMBR (*AMBR4 ?*), the mapping between the base and the target is to be used for generating inferences in the target (and possibly the base). We refer to this process as *transfer*. It involves at least three issues:

1. Which members of the two descriptions remain unmapped? These are the potential candidates for transfer? As AMBR uses decentralized representations of situations, this question cannot be answered by going through some list-like structure and crossing out mapped entities.

2. Which unmapped elements really merit transferring? This is a very difficult question. For example, suppose the target problem is to heat some milk in a teapot. The base contains water, a teapot, and a hot-plate (among other things). The color of the teapot in the base is green. The temperature of the plate is high. Neither proposition has any analog in the target and, therefore, both are candidates for transfer. Perhaps the milk will get hot if one paints the teapot green?

3. How to carry elements from the one domain to the other? Objects and propositions in the base cannot be copied literally to the target; they must be 'translated'.

The ordering above looks like a sequence of steps but it should not be understood in this way. According to the overall philosophy of AMBR, these 'steps' overlap in time. Whenever an element is identified as unmapped (point 1), the evaluation of its relevance and potential usefulness could begin (point 2). There is no need to wait for the other unmapped agents. In addition, the potential usefulness of an element depends on the quality of the inferences that this element could 'propose' (point 3). Hence, in our view the whole process should be modeled by a 'wave' similar to the one outlined in subsection 3.2.1.

How could AMBR mechanisms carry out the transfer process? Let us start with point 1. above. One possible answer is that the secretaries of the target are authorized to judge which elements are *mapped* while the secretaries of the source are authorized to judge which elements are *unmapped*.

Hummel & Holyoak (1996, 1997) propose two very useful concepts. In addition to the conventional target/base distinction, they introduce the *driver/recipient* distinction. The driver is the one that has the initiative and 'makes things happen'. In AMBR terminology, it is *authorized*. This could be the target problem or the source episode. Hummel & Holyoak (1997) suggest the following 'canonical flow of control': First the target is used as driver during the access stage. Once a source is in working memory, mapping can be performed in either direction (including successive switches between the two episodes). After the mapping stage is over, the source is used to drive inferences and schema induction in the target.

We adopt the driver/recipient terminology and agree with the main idea of the previous paragraph. However, we propose a correction — switches between 'driver mode' and 'recipient mode' should not be done in a way that serializes the process of analogy-making and cut it into separate stages (marked by interventions of the human user). Among other things, this implies that it should be possible that *both situations act as drivers simultaneously.*

For lack of better terminology, we will denote the situation (or, more precisely, the elements thereof) that drives the mapping as *driver-M*. The one that drives the transfer is *driver-T*. Each is authorized to do different and complementary things.

The target problem in AMBR typically acts as driver-M. Its agents access (partially) episodes from the long-term memory, establish correspondences, administer rating surveys, and promote winners. When a source episode emerges as winner, it becomes driver-T and its agents become authorized to identify unmapped elements, judge their potential usefulness for transfer, propose 'translations' in the target, etc. The two coalitions — driver-M and driver-T — work together, each according to its authorization. In this way, the transfer subprocess overlaps in time with mapping, potentially altering the balance in the constraint satisfaction network and affecting the correspondences that remain to be promoted.

More concretely, the driver-T secretaries could identify whether they are unmapped or not by a *constraint propagation mechanism* (e.g. Waltz, 1975). Commitment in one place (in the form of a winner promotion) propagates to other places. Consider the example in Figure 7.1.1.

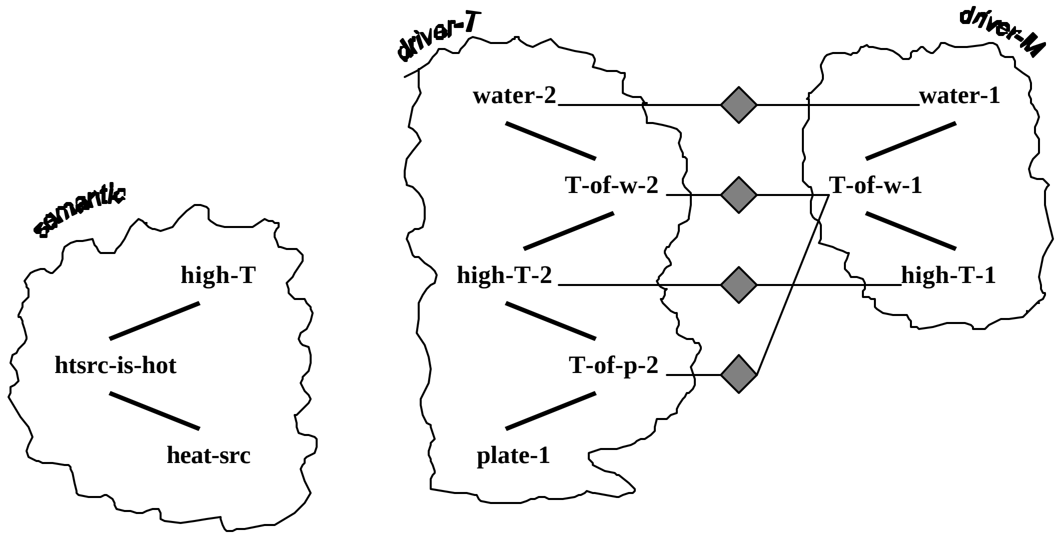


Figure 7.1.1. The *driver-M* coalition (to the right) has established four hypotheses (diamonds) with the *driver-T* coalition (in the middle). A general proposition waits in the semantic memory (to the left) and could be used for skolemization. See text for details.

Figure 7.1.1. illustrates a fragment of the network at the moment when the driver-M coalition (to the right) has established several hypotheses with the recipient coalition (in the middle). Note that two hypotheses compete for the agent `T-of-w-1`. The secretary of the latter is authorized to promote one of them as winner. Suppose the hypothesis `T-of-w-1<-->T-of-w-2` is the winner (due to structural and semantic pressures in the CSN as both propositions involve water). When it is promoted, the rival hypothesis `T-of-w-1<-->T-of-p-2` becomes a loser. It notifies the secretary of `T-of-p-2` about this. The latter agent belongs to the recipient situation (in the middle). This same situation, however, is driver-T at the same time. As such, it checks whether it has at least one non-loser hypothesis on its record. When `T-of-w-1<- ->T-of-p-2` becomes a loser, the secretary detects that `T-of-p-2` is unmapped.

Driver-T secretaries are authorized to trigger the skolemization mechanism (just as driver-M secretaries are). Thus, the general proposition that the temperature of heat-sources is high could be used to augment the driver-M situation. This could be done in the following way: The agent `T-of-p-2`, having missed the chance to map to `T-of-w-1`, now takes the initiative and issues a marker. (Note that it acts as a driver at this moment.) As described in section 5.3., this marker goes to the 'parent concept' `temperature-of`. It intersects there with the marker issued from the general proposition `htsrc-is-hot`. The marker intersection leads to a construction of a new hypothesis agent: `T-of-p-2<-->htsrc-is-hot`. The symbolic processor of this new agent could now carry out the skolemization routines (section 5.7.) and augment the description of the 'other' situation. As this particular instantiation of the skolemization mechanism has been triggered by the driver-T situation, the new Skolem instances will be added to situation driver-M. (Note that the latter acts as a recipient with respect to driver-T.)

The skolemization mechanism will re-use the agent `high-T-1` in the recipient. As the concept agent `heat-source` has received no marker from the came coalition, a new Skolem instance will be created and affiliated to the episode shown to the right in Figure 7.1.1. Suppose the name of this new agent is `sk-htsrc-1`. Finally, a Skolem proposition will be created. It binds `sk-htsrc-1` and `high-T-1` as arguments of a relation `temperature-of`. Let the name of this latter proposition is `sk-T-of-1`. Figure 7.1.2. shows the new configuration.

The new agents `sk-htsrc-1` and `sk-T-of-1` affiliate to the driver-M coalition. In this way, the description of the target problem is augmented with a heat source. The new agents now take the initiative and issue markers. These markers will create the hypotheses `sk-htsrc-1<-->plate-2` and `sk-T-of-1<-->T-of-p-2`. Hence, `plate-2` and `T-of-p-2` are not unmapped any more. After the rating mechanism does its job, the new hypotheses will be promoted as winners. In particular, the hypothesis `sk-T-of-1<-->T-of-p-2` will eliminate the general hypothesis `T-of-p-2<-->htsrc-is-hot` that has carried out the skolemization process.

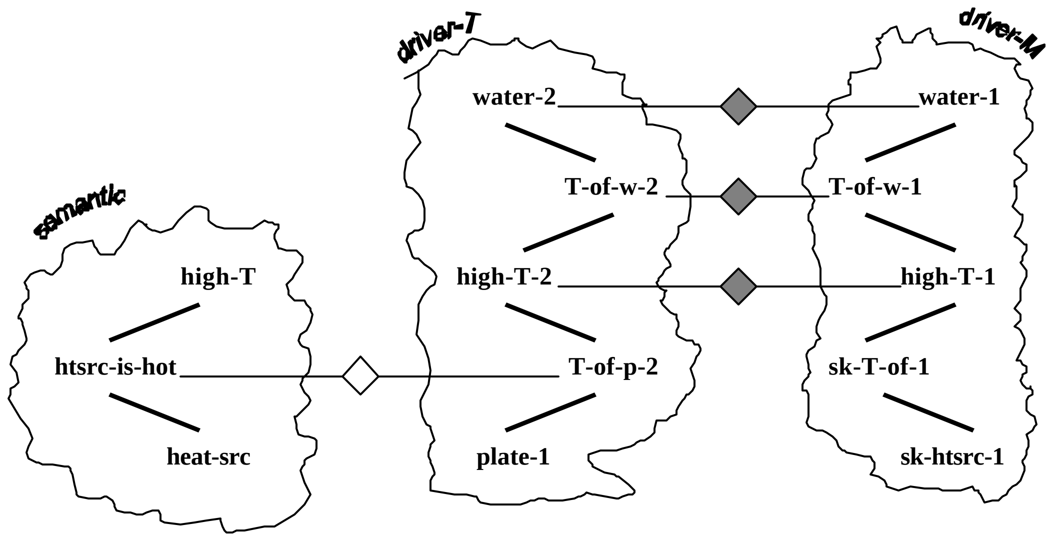


Figure 7.1.2. State of the network after the skolemization mechanism has added two new Skolem agents to the *driver-M* coalition introduced in Figure 7.1.1. The general-hypothesis agent that has carried out the skolemization is depicted by a white diamond. It will be eliminated when the new Skolem instances create hypotheses of their own. See text for details.

This example suggests that the existing AMBR mechanisms can be useful not only for the processes of analog access and mapping but for the transfer process too. The utility of the mechanisms of rating, marker passing, and skolemization is clear from the example. The other mechanisms are potentially useful too. The spreading activation is a key mechanism for estimating relevance, and such estimates will surely be needed for the selection of candidates for transfer. The constraint satisfaction is also useful when there is a need for selecting one option among alternatives.

Considerations of this kind make us believe that the AMBR model is open-ended enough and its functionality could be extended in the direction of analogical transfer. Moreover, we hope that this could be done without giving up the properties of the current version. Analog transfer could be done in a dynamic emergent way over decentralized representations. It could run in parallel with the subprocesses of mapping and access.

7.2. Possibilities for Perception

This section suggests how AMBR could be extended in the direction of high-level perception. As argued by Hofstadter (1984, 1995) and Chalmers, French, & Hofstadter (1992), the process of building representations is a crucial part of analogy-making. The same authors defend the methodological utility of micro-domains for research on high-level perception. Micro-domains allow the model to focus on building structured representations instead of dealing with low-level details such as filtering noise from images.

One such micro-domain that could be used in the research on DUAL and AMBR is the so-called TEXTSCREEN. It is based on an imaginary text processing program. TEXTSCREEN is deliberately simplified — there is plain text over a limited matrix of screen positions. There are objects like characters, marked areas, etc. The characters can be grouped in words, lines, paragraphs, columns, etc. Most of the objects are directly visible on the screen where they tend to form regular rectangular patterns.

The objects have attributes such as `long` and `vertical`. There are also a number of relations such as `left-of`, `part-of`, `aligned-with`, etc. Finally, there are various actions (or commands) to navigate through the text, to insert, delete, or move objects, and mark portions of text and thus form aggregate units for subsequent manipulation.

This material is rich enough to allow various configurations on the screen (see Figure 7.2.1). A model that operates in this environment is presented with a situation which has some defect somewhere on the screen and the task of the system is to locate the defect and correct it. To that end, the model can use previous situations (with solutions) as source analogs.

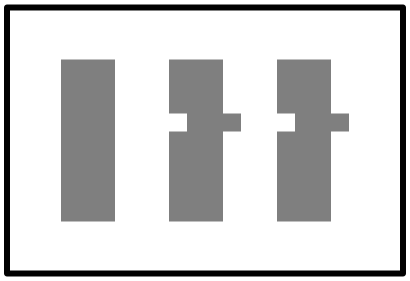


Figure 7.2.1. Sample problem in TEXTSCREEN.

Visual perception has much to do with space. Even in a simplified two-dimensional micro-world like TEXTSCREEN spatial properties, relations, and configurations are all important. This characteristic feature of the environment must be reflected somehow in the cognitive architecture DUAL. The main network cannot meet this requirement because it lacks spatial organization[1]. Therefore, we plan to augment the architecture with a large-scale structure having explicit spatial organization — the *visual array (VA)*.

As everything in DUAL, the visual array consists of agents. The defining characteristic of the array is that the agents are arranged in a rectangular matrix. Each agent in the array is associated with a particular position on TEXTSCREEN and can 'see' whether the cell is empty or not. Thus the VA is a mediator between two different worlds — the external environment of TEXTSCREEN and the internal representations in the main network. The defining principle of TEXTSCREEN is physical location. On the other hand, the defining principle of the network is interconnectivity. In the visual array these two principles meet — the visual agents have both physical locations and links to other agents. For

[1] There is a notion of *neighborhood* in the network but it is not related to spatial proximity.

instance, each agent is linked to the agents in the four neighboring cells. It can interact with them, send them symbols and activation, etc.

There are other perceptual agents that are connected to a whole row or column of the agents in the VA. These agents can detect lines, lines with defects, etc. When they locate an object in their receptive field, they create a new temporary agent in the main network that represents this external object. Other specialized perceptual agents combine lines in regions, identify various spatial relations between them, etc. They build new agents in the network to represent these regions and relations. Still other agents group things together or parse a complex object into parts. Each perceptual agent works at individual speed depending on its activation level. The activation in turns reflects two kinds of influences: bottom-up (from the VA) and top-down (from the ‘parent concept’ in the network).

The visual array is a source of activation. It will replace the input node of the current version of DUAL and AMBR. Instead of hand-coding the description of the scene and attaching it to the input list, the model should be able to build the representation itself. It would be built agent by agent. Each new agent enters the working memory, sends activation to its respective concept agent, and emits a marker to trigger the mechanisms for finding correspondences. As the simulation experiments on order effects have demonstrated (section 6.4), AMBR is capable to handle target problems that are presented over the course of time.

CHAPTER VIII

CONCLUSION

8.1. Overview of the Thesis

This thesis presents AMBR3 — a dynamic emergent integrated model of analogical access and mapping based on decentralized representations of situations. It describes in detail the knowledge structures and computational mechanisms used in the model. The behavior of the model is illustrated by many examples, diagrams, and transcripts of actual runs of the computer implementation of the model. The thesis reports the results of various simulation experiments involving more than 1,200 runs of the program on different target problems. AMBR is compared with a selection of other models and is discussed in the light of the studies of human analogy-making.

AMBR is an emergent and decentralized model. It consists of a population of small entities called *DUAL agents*. These agents are the ingredients of the DUAL cognitive architecture which is the foundation of AMBR. They represent all the knowledge and carry out all the processing in the architecture. There is no central executive that controls the operation of the system as a whole. Instead, each agent works locally and performs its simple specific task in close interaction with its immediate neighbors. The global behavior of the model emerges of the coordinated effort of these asynchronous local activities.

AMBR applies the same approach to the phenomena it is intended to model. The subprocesses of analogy-making are explained in terms of coordinated mechanisms. The main intuition behind the research reported here is that there is no 'analogy machine' that does analogies according to some fixed centralized algorithm. Instead, analogy is an emergent product of the work of general cognitive mechanisms. The thesis tries to demonstrate that such approach is feasible. Thus, analog access is based on the mechanism of spreading activation which serves a range of other purposes in the cognitive architecture. The constraint satisfaction mechanism is used for finding correspondences in the model but the same mechanism can apply to various other tasks such as perception and decision making.

AMBR representations of episodes are decentralized. The model does not maintain data structures listing the elements that belong to each situation. Instead, each situation is represented by a *coalition* of agents. This allows for greater flexibility of the representations. New elements can be added when necessary. The skolemization mechanism can augment the description of a given episode based on general semantic information. In the same time, elements that have been needed in the past and potentially belong to the description of the episode stay out of the working memory when they are irrelevant for the problem being solved. Thus the model is capable to re-represent a situation both by addition and omission of elements. Chapter V demonstrates this on a concrete example.

The theme of integration is central for AMBR research. The model conceptualizes the components of analogy-making not as sequential 'stages' but as *subprocesses* that run in parallel and interact. The version reported in this thesis integrates the subprocesses of analog access and mapping. A case study reported in Chapter VI illustrates an interaction of this kind. Other simulation experiments from the same chapter also demonstrate various aspects of these interactions. Chapter VII suggests possibilities for modeling the subprocesses of transfer and perception. It is argued that they could be added to the current version of the model without forcing radical reconsideration of the existing mechanisms.

The computational dynamics is a characteristic feature of the architecture DUAL and, consequently, of the model built on top of it. Each DUAL agent works at its own speed that varies dynamically as the activation level of the agent vary. Thus, more relevant agents work faster and contribute more to the overall behavior of the system than do less relevant (and hence less active) ones. In addition, the topology of the AMBR network is constantly changing as new nodes and links are created while others are removed. This *dynamic emergent computation* provides for flexibility and efficiency at the same time.

8.2. Contributions of This Work

The research reported in this thesis has made several extensions and improvements of the AMBR model and DUAL architecture with respect to the earlier specification (Kokinov, 1994a). In our view, the major contributions are:

From AMBR1 (Kokinov, 1994a) to AMBR2 (Petrov, 1997):

- Introduction of the *energetic analogy* and the mechanism of *consumptions* for specifying the exact relationship between the activation level of a DUAL agent and the speed of its symbolic processor.

- Introduction of the notion of *coalitions* and the intermediate level of description of the architecture (the *meso-level*). The conceptual apparatus of coalitions is an important tool for developing and communicating the ideas about emergent computation, decentralized representations, etc.

- Transition from centralized to *decentralized representation* of situations in AMBR. In turn, this led to improvements in the marker passing, structure correspondence, and constraint satisfaction mechanisms. It is also an important factor for the integration of the different subprocesses of analogy-making in the model.

- Introduction of *secretaries* for the purposes of incremental construction of the constraint satisfaction network. The presence of secretaries also prepares the ground for the rating mechanism in AMBR3.

- Disclosing the deficiencies of the *activation function* used in AMBR1 and replacing it with a more appropriate one. Detailed mathematical analysis of these functions.

- Developing, testing, and documenting a portable computer implementation of the architecture and the model. The program has been tested under two platforms: Allegro Common Lisp (Windows), and Carnegie Mellon Common Lisp (Unix).

- Enlarging the knowledge base and performing simulation experiments with AMBR2.

From AMBR2 to AMBR3:

- Introduction of the mechanisms for *rating* and *promotion*. The authorized secretaries in AMBR3 monitor the activation levels of the hypothesis registered at them. Secretaries promote *winners* and eliminate *losers* when appropriate. Thus the outcome of the mapping process is available within the model itself; there is no need for an external observer to read out the answer from the activation pattern in the constraint satisfaction network. In addition, loser elimination reduces the size of the CSN and opens new possibilities for incremental processing as discussed in subsection 5.6.1.3. The rating mechanism also performs *ballotages* to prevent implausible blendings and trigger the skolemization mechanism. The life cycle of hypothesis agents is elaborated.
- Introduction of the *skolemization* mechanism. In this way, general semantic information can be used to augment the descriptions of episodes upon necessity. This is the first attempt for *re-representation of past episodes* in analogy-making. The skolemization mechanism would also be very useful for the transfer process in analogy-making as discussed in Chapter VII.

- Extending the structure correspondence mechanism with abilities for *weak structure correspondence*. It improves the connectivity of the CSN by creating new links (but not new hypothesis agents). Combined with the differential link weighting adopted in AMBR3, this improves the structural constraint on analogical mapping.

- Elaborating the description of the episodes in the knowledge base and addition of new episodes and concepts. The total number of agents is more than doubled with respect to AMBR2. There is richer representation of the causal structure of each base situation.

- The simulation experiments with AMBR3 reported in this thesis involve more than 1,200 runs of the program and show the behavior of the model in detail. The interaction between analog access and mapping is explored. An experiment on order effects shows that AMBR3 is sensitive to the order of presentation of the target elements.

- All new mechanisms are implemented in the computer program. There are also a number of technical improvements of the old implementation. (For example, the routines performed by the symbolic processors of AMBR2 agents were interpreted. In AMBR3 they are compiled.)

8.3. Suggestions for Future Research

Each end is a new beginning.
(Bulgarian proverb)

As stated repeatedly in this thesis, AMBR3 is but an intermediate stage in a long-term research program. There are many ways in which this research can be continued. Some of them are suggested in this final section.

To begin with, much more experimentation could (and should) be done with the existing version of the model. There are a number of interesting effects that are within its scope but have not been demonstrated in rigorous simulation experiments. For example, AMBR3 could map propositions with different number of arguments, map an object to a relation, etc. The experiments on priming and context effects performed by Kokinov (1994a) could also be replicated and extended. The model should be tested on new kinds of problems in different domains. Of particular interest is whether the model will scale up to larger memory sizes. The sensitivity and robustness of the model for different values of its various parameters is another issue that has not been covered in the present thesis.

Another possibility for research is to design and implement new computational mechanisms and extend the functionality of AMBR. The subprocess of transfer seems within closest reach. The mechanisms of constraint propagation, switching the base as driver, and skolemization from base to target outlined in Chapter VII provide a starting point.

A major research direction is to add perceptual capabilities to DUAL and AMBR. This involves the visual array mentioned in Chapter VII and the TEXTSCREEN micro-domain. The integration of the perceptual mechanisms with the existing computational machinery is a very challenging and intriguing topic. Another research direction of comparable complexity and import is to add learning mechanisms to the architecture.

The research on AMBR involves psychological experimentation too. For instance, the order effect on access presented in sections 5.2.4. and 6.4. is a prediction of the model which could be tested empirically.

APPENDIX A

FULL REPRESENTATION OF A SITUATION

This appendix presents the unabridged representation of one of the twelve episodes in the current long-term memory. It is taken directly from the Lisp sources that load the knowledge base of AMBR.

The file consists of `defagent` macros. Each macro defines an agent and fills its slots. The overall syntax is:

```
(defagent agent-name agent-type
  [documentation-string]
  {G-slot-definition}*
  {S-slot-definition}*
)
```

Most slot fillers are references to other agents. Each reference is also a link and has a label and a weight (see subsection 3.1.3.1). When no explicit weight is given, it defaults to 1.0.

```
;;; -*- Mode: Lisp; Syntax: Common-Lisp; Package: AMBR -*-

;;; FILE:       AMBR/kb/episodic/b_WTP.lsp
;;; VERSION:    3.0.0   ; see AMBR/KB/VERSION.LSP
;;; PURPOSE:    Base situation WTP -- 'Water in a Teapot on a hot Plate.'
;;; DEPENDS-ON: AMBR, AMBR/kb/semantic/*.lsp
;;; PROGRAMMER: Alexander Alexandrov Petrov   (apetrov@cogs.nbu.acad.bg)
;;; VARIANTS:   none
;;; CREATED:    30-05-98 [3.0.0]  Elaboration of SIT-WTP from old LTM.LSP.
;;; UPDATED:    18-06-98  Removed IS-GOAL and RESULT propositns. Wght adjstmt
;;;                       CAUSE consequents are propositions now, not states.
;;;                       T-OF-WTP-W1 and T-OF-WTP-W2 coalesced together.
;;; UPDATED:    ...

         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
         ;;;;;;;;         SITUATION  W T P           ;;;;;;;;
         ;;;;;;;;   Water in a Teapot on a Plate     ;;;;;;;;
         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(in-package "AMBR")

;;;;;;;;;   Base situation WTP   ;;;;;;;;;;;
;;;;
;;;; There is some water in a teapot on a hot-plate.
;;;; The plate is hot.  The teapot is made of metal
;;;; and its color is black.
;;;;
;;;; The goal is to heat the water.
;;;;
;;;; The result is that the teapot becomes hot because
;;;; it is on the hot plate.  In turn, this causes the
;;;; water to become hot because it is in the teapot.
;;;;
;;;;;;;;;
;;;; Related situations:
;;;;  + GP  -- Glass on a hot Plate breaks.
;;;;  + IHC -- Imm.Heater in a Cup heats water.
;;;;  + ...
```

```
;;;;  * HM1 -- How to Heat Milk in a Teapot?
;;;;  * ...



;;;;;;;  Situation-agent
;;

(defagent   sit-WTP    instance-agent
  "Water in a Teapot on a hot Plate."
  :type      (:instance :situation)
  :inst-of   (situation 0.1)
  :a-link    ((hplate-WTP 0.5)
              (high-T-WTP 0.5)
              (T-of-WTP-w 1.0) )
)

;;;;;;;  Participating objects
;;
;;  water-WTP  :  (inst-of water)
;;  tpot-WTP   :  (inst-of teapot)
;;  hplate-WTP :  (inst-of hot-plate)
;;

(defagent   water-WTP    instance-agent
  :type       (:instance  :object)
  :modality   (:init :goal)
  :situation  (sit-WTP 0.2)
  :inst-of    water
  :c-coref    (((in-WTP . :slot1) 0.3)
               ((T-of-WTP-w  . :slot1) 1.0)
               ((goalst-WTP  . :slot3) 0.2)
               ((interst-WTP . :slot3) 0.1) )
  :a-link     (initst-WTP-1 0.1)
)

(defagent   tpot-WTP    instance-agent
  :type       (:instance  :object)
  :modality   :init
  :situation  (sit-WTP 0.2)
  :inst-of    teapot
  :c-coref    (((in-WTP . :slot2) 0.25)
               ((on-WTP . :slot2) 0.25)
               ((T-of-WTP-t   . :slot1) 0.25)
               ((made-of-WTP  . :slot1) 0.10)
               ((color-of-WTP . :slot1) 0.10)
               ((initst-WTP-2 . :slot3) 0.10) )
  :a-link     (initst-WTP-1 0.1)
  :slot1
    :type      :relation
    :inst-of   (teapot . :slot2)
    :c-coref   (made-of-WTP 0.2)
    :a-link    (mmetal-WTP  0.2)
  :slot2
    :type      :relation
    :inst-of   (teapot . :slot2)
    :c-coref   (color-of-WTP 0.1)
    :a-link    (black-WTP    0.1)
)

(defagent   hplate-WTP    instance-agent
  :type       (:instance  :object)
  :modality   :init
  :situation  (sit-WTP 0.3)
  :inst-of    hot-plate
  :c-coref    (((T-of-WTP-p . :slot1) 0.75)
               ((on-WTP . :slot1)     0.25) )
  :a-link     ((high-T-WTP   0.5)
               (initst-WTP-1 0.1) )
)
```

```
;;;;;;;  Initial relations
;;
;;  in-WTP       : (in water-WTP tpot-WTP)
;;  on-WTP       : (on hplate-WTP tpot-WTP)
;;  T-of-WTP-p   : (temperature-of hplate-WTP high-T-WTP)
;;  made-of-WTP  : (made-of  tpot-WTP mmetal-WTP)
;;  color-of-WTP : (color-of tpot-WTP black-WTP)
;;

(defagent   in-WTP    instance-agent
  "(in water-WTP tpot-WTP)"
  :type       (:instance  :relation)
  :modality   :init
  :situation  (sit-WTP 0.2)
  :inst-of    in
  :c-coref    (((initst-WTP-1 . :slot4) 0.2)
               ((interst-WTP  . :slot2) 0.1) )
  :slot1
    :type       :aspect
    :inst-of    (in . :slot1)
    :c-coref    water-WTP
  :slot2
    :type       :aspect
    :inst-of    (in . :slot2)
    :c-coref    tpot-WTP
)

(defagent   on-WTP    instance-agent
  "(on hplate-WTP tpot-WTP)"
  :type       (:instance  :relation)
  :modality   :init
  :situation  (sit-WTP 0.2)
  :inst-of    on
  :c-coref    (((initst-WTP-1 . :slot3) 0.2)
               ((initst-WTP-2 . :slot2) 0.1) )
  :a-link     (T-of-WTP-t 0.1)
  :slot1
    :type       :aspect
    :inst-of    (on . :slot1)
    :c-coref    hplate-WTP
  :slot2
    :type       :aspect
    :inst-of    (on . :slot2)
    :c-coref    tpot-WTP
)

(defagent   T-of-WTP-p    instance-agent
  "(temperature-of hplate-WTP high-T-WTP)"
  :type       (:instance  :relation)
  :modality   :init
  :situation  (sit-WTP 0.2)
  :inst-of    temperature-of
  :c-coref    (((initst-WTP-1 . :slot1) 0.3)
               ((initst-WTP-2 . :slot1) 0.1) )
  :a-link     ((T-of-WTP-t  0.1)
               (T-of-WTP-w  0.1)
               (cause-WTP-i 0.1) )
  :slot1
    :type       :aspect
    :inst-of    (temperature-of . :slot1)
    :c-coref    hplate-WTP
  :slot2
    :type       :aspect
    :inst-of    (temperature-of . :slot2)
    :c-coref    high-T-WTP
)
(defagent  high-T-WTP   instance-agent
  :type       (:instance  :object)
  :modality   (:init :goal :result)
  :situation  (sit-WTP 0.3)
  :inst-of    high-temp
  :c-coref    (((T-of-WTP-p   . :slot2) 0.5)
               ((T-of-WTP-t   . :slot2) 0.2)
               ((T-of-WTP-w   . :slot2) 0.5)
               ((initst-WTP-1 . :slot2) 0.1)
               ((goalst-WTP   . :slot2) 0.1) )
  :a-link     (hplate-WTP 0.3)
)

(defagent   made-of-WTP    instance-agent
  "(made-of tpot-WTP mmetal-WTP)"
  :type       (:instance  :relation)
  :modality   :init
  :situation  (sit-WTP 0.2)
  :inst-of    made-of
  :c-coref    (tpot-WTP . :slot1)
  :a-link     (T-of-WTP-t 0.3)
  :slot1
    :type       :aspect
    :inst-of    (made-of . :slot1)
    :c-coref    tpot-WTP
  :slot2
    :type       :aspect
    :inst-of    (made-of . :slot2)
    :c-coref    mmetal-WTP
)
(defagent  mmetal-WTP  instance-agent
  :type       (:instance  :object)
  :modality   :init
  :situation  (sit-WTP 0.2)
  :inst-of    material-metal
  :c-coref    (made-of-WTP  . :slot2)
  :a-link     (tpot-WTP 1.0)
)

(defagent   color-of-WTP    instance-agent
  "(color-of tpot-WTP black-WTP)"
  :type       (:instance  :relation)
  :modality   :init
  :situation  (sit-WTP 0.2)
  :inst-of    color-of
  :c-coref    (tpot-WTP . :slot2)
  :slot1
    :type     :aspect
    :inst-of  (color-of . :slot1)
    :c-coref  tpot-WTP
  :slot2
    :type     :aspect
    :inst-of  (color-of . :slot2)
    :c-coref  black-WTP
)
(defagent   black-WTP  instance-agent
  :type       (:instance  :object)
  :modality   :init
  :situation  (sit-WTP 0.2)
  :inst-of    black
  :c-coref    (color-of-WTP . :slot2)
  :a-link     (tpot-WTP 1.0)
)


;;;;;;;  Initial states
;;
;;  initst-WTP-1  -to-reach->  goalst-WTP
;;  initst-WTP-1  -follows->   endst-WTP
;;  initst-WTP-2  --cause-->   interst-WTP
;;
;;  initst-WTP-1  :  (init-state T-of-WTP-p high-T-WTP on-WTP in-WTP)
;;  initst-WTP-2  :  (init-state T-of-WTP-p on-WTP tpot-WTP)
;;
```

```
(defagent   initst-WTP-1  instance-agent
  "initst-WTP-1  -to-reach->  goalst-WTP"
  :type       (:instance :situation)
  :modality   :init
  :situation  (sit-WTP 0.2)
  :inst-of    init-state
  :c-coref    ((to-reach-WTP . :slot1)
               (follows-WTP  . :slot1) )
  :a-link     ((goalst-WTP   1.0)
               (initst-WTP-2 0.2)
               (water-WTP    0.2)
               (tpot-WTP     0.2)
               (hplate-WTP   0.2) )
  :slot1
    :type       :relation
    :inst-of    (init-state . :slot2)
    :c-coref    T-of-WTP-p
  :slot2
    :type       :aspect
    :inst-of    (init-state . :slot1)
    :c-coref    high-T-WTP
  :slot3
    :type       :relation
    :inst-of    (init-state . :slot2)
    :c-coref    on-WTP
  :slot4
    :type       :relation
    :inst-of    (init-state . :slot2)
    :c-coref    in-WTP
)

(defagent   initst-WTP-2  instance-agent
  "initst-WTP-2  --cause-->  interst-WTP"
  :type       (:instance :situation)
  :modality   :init
  :situation  (sit-WTP 0.2)
  :inst-of    init-state
  :c-coref    (cause-WTP-i . :slot1)
  :a-link     ((interst-WTP  1.0)
               (initst-WTP-1 0.3)
               (hplate-WTP   0.3) )
  :slot1
    :type       :relation
    :inst-of    (init-state . :slot2)
    :c-coref    T-of-WTP-p
  :slot2
    :type       :relation
    :inst-of    (init-state . :slot2)
    :c-coref    on-WTP
  :slot3
    :type       :aspect
    :inst-of    (init-state . :slot1)
    :c-coref    tpot-WTP
)


;;;;;;;  Goal state
;;
;;  goalst-WTP  <-to-reach-  initst-WTP-1
;;  goalst-WTP   :  (goal-state T-of-WTP-w high-T-WTP water-WTP)
;;
;;  T-of-WTP-w   :  (temperature-of water-WTP high-T-WTP)
;;  to-reach-WTP :  (to-reach initst-WTP-1 goalst-WTP)
;;

(defagent   T-of-WTP-w    instance-agent
  "(temperature-of water-WTP high-T-WTP)"
  :type       (:instance  :relation)
  :modality   (:GOAL   :intend-true
               :RESULT :true )
  :situation  (sit-WTP 0.3)
  :inst-of    temperature-of
  :c-coref    (((goalst-WTP  . :slot1) 0.2)
               ((endst-WTP   . :slot2) 0.2)
               ((cause-WTP-e . :slot2) 0.2) )
  :a-link     ((T-of-WTP-p 0.1)
               (T-of-WTP-t 0.1)
               (hplate-WTP 0.1) )
  :slot1
    :type       :aspect
    :inst-of    (temperature-of . :slot1)
    :c-coref    water-WTP
  :slot2
    :type       :aspect
    :inst-of    (temperature-of . :slot2)
    :c-coref    high-T-WTP
)

(defagent   goalst-WTP    instance-agent
  "goalst-WTP  <-to-reach-  initst-WTP-1"
  :type       (:instance :situation)
  :modality   :goal
  :situation  (sit-WTP 0.2)
  :inst-of    goal-state
  :c-coref    (to-reach-WTP . :slot2)
  :a-link     ((initst-WTP-1 1.0)
               (endst-WTP    0.5) )
  :slot1
    :type       :relation
    :inst-of    (goal-state . :slot2)
    :c-coref    T-of-WTP-w
  :slot2
    :type       :aspect
    :inst-of    (goal-state . :slot1)
    :c-coref    high-T-WTP
  :slot3
    :type       :aspect
    :inst-of    (goal-state . :slot1)
    :c-coref    water-WTP
)

(defagent   to-reach-WTP    instance-agent
  "(to-reach initst-WTP-1 goalst-WTP)"
  :type       (:instance  :relation)
  :modality   :goal
  :situation  (sit-WTP 0.2)
  :inst-of    to-reach
  :a-link     ((follows-WTP 0.5)
               (cause-WTP-i 0.1)
               (cause-WTP-e 0.2) )
  :slot1
    :type       :aspect
    :inst-of    (to-reach . :slot1)
    :c-coref    initst-WTP-1
  :slot2
    :type       :aspect
    :inst-of    (to-reach . :slot2)
    :c-coref    goalst-WTP
)


;;;;;;;  Intermediary state
;;
;;  interst-WTP  <-cause--  initst-WTP-2
;;  interst-WTP  --cause->  endst-WTP
;;  interst-WTP   :  (inter-state T-of-WTP-t in-WTP water-WTP)
;;
;;  T-of-WTP-t    :  (temperature-of tpot-WTP high-T-WTP)
;;  cause-WTP-i   :  (cause initst-WTP-2 T-of-WTP-t)
;;

(defagent   T-of-WTP-t    instance-agent
  "(temperature-of tpot-WTP high-T-WTP)"
  :type        (:instance  :relation)
  :modality    :result
  :situation   (sit-WTP 0.2)
  :inst-of     temperature-of
  :c-coref     (((cause-WTP-i . :slot2) 0.5)
                ((interst-WTP . :slot1) 0.2)
                ((endst-WTP   . :slot1) 0.2) )
  :a-link      ((T-of-WTP-p 0.3)
                (T-of-WTP-w 0.3)
                (hplate-WTP 0.1) )
  :slot1
    :type       :aspect
    :inst-of    (temperature-of . :slot1)
    :c-coref    tpot-WTP
  :slot2
    :type       :aspect
    :inst-of    (temperature-of . :slot2)
    :c-coref    high-T-WTP
)

(defagent   interst-WTP    instance-agent
  "interst-WTP  <-cause-  initst-WTP-2"
  :type        (:instance :situation)
  :modality    :result
  :situation   (sit-WTP 0.2)
  :inst-of     inter-state
  :c-coref     (cause-WTP-e . :slot1)
  :a-link      ((endst-WTP    0.6)
                (initst-WTP-2 0.4) )
  :slot1
    :type        :relation
    :inst-of     (inter-state . :slot2)
    :c-coref     T-of-WTP-t
  :slot2
    :type        :relation
    :inst-of     (inter-state . :slot2)
    :c-coref     in-WTP
  :slot3
    :type        :aspect
    :inst-of     (inter-state . :slot1)
    :c-coref     water-WTP
)

(defagent   cause-WTP-i   instance-agent
  "(cause initst-WTP-2 T-of-WTP-t)"
  :type        (:instance  :relation)
  :modality    :result
  :situation   (sit-WTP 0.2)
  :inst-of     cause
  :a-link      ((interst-WTP  1.0)
                (cause-WTP-e  0.5)
                (follows-WTP  0.3)
                (to-reach-WTP 0.2) )
  :slot1
    :type        :aspect
    :inst-of     (cause . :slot1)
    :c-coref     initst-WTP-2
  :slot2
    :type        :aspect
    :inst-of     (cause . :slot2)
    :c-coref     T-of-WTP-t
)


;;;;;;;  End state
;;
;;  endst-WTP  <-follows-  initst-WTP-1
;;  endst-WTP     :  (end-state T-of-WTP-t T-of-WTP-w)
;;
;;  follows-WTP   :  (follows initst-WTP-1 endst-WTP)
;;  cause-WTP-e   :  (cause interst-WTP T-of-WTP-w)
;;

(defagent   endst-WTP    instance-agent
  "endst-WTP  <-follows-  initst-WTP-1"
  :type       (:instance :situation)
  :modality   :result
  :situation  (sit-WTP 0.2)
  :inst-of    end-state
  :c-coref    (follows-WTP . :slot2)
  :a-link     ((interst-WTP 0.1)
               (goalst-WTP  0.2) )
  :slot1
    :type       :relation
    :inst-of    (end-state . :slot2)
    :c-coref    (T-of-WTP-t 0.5)
  :slot2
    :type       :relation
    :inst-of    (end-state . :slot2)
    :c-coref    T-of-WTP-w
)

(defagent   follows-WTP  instance-agent
  "(follows initst-WTP-1 endst-WTP)"
  :type       (:instance  :relation)
  :modality   :result
  :situation  (sit-WTP 0.2)
  :inst-of    follows
  :a-link     ((to-reach-WTP 0.5)
               (cause-WTP-i  0.2) )
  :slot1
    :type       :aspect
    :inst-of    (follows . :slot1)
    :c-coref    initst-WTP-1
  :slot2
    :type       :aspect
    :inst-of    (follows . :slot2)
    :c-coref    endst-WTP
)

(defagent   cause-WTP-e    instance-agent
  "(cause interst-WTP T-of-WTP-w)"
  :type       (:instance  :relation)
  :modality   :result
  :situation  (sit-WTP 0.2)
  :inst-of    cause
  :a-link     ((endst-WTP    0.5)
               (follows-WTP  0.3)
               (cause-WTP-i  0.2)
               (to-reach-WTP 0.1) )
  :slot1
    :type       :aspect
    :inst-of    (cause . :slot1)
    :c-coref    interst-WTP
  :slot2
    :type       :aspect
    :inst-of    (cause . :slot2)
    :c-coref    T-of-WTP-w
)


;;;;;;  ---- Sanity check ---- ;;;;;;
;;
(check-for-unresolved-references)


;;;;;;;;;;;;  ---  Appendix  ---  ;;;;;;;;;;
;;
;; The information below is not used by the model.
;; It is for interface purposes only.
```

```
(defcoalition  sit-WTP
  "Water in a Teapot on a hot Plate."
  :head      sit-WTP                    ; 23 agents
  :members   (sit-WTP
              water-WTP     tpot-WTP      hplate-WTP
              in-WTP        on-WTP
              T-of-WTP-p    high-T-WTP
              T-of-WTP-t    T-of-WTP-w
              made-of-WTP   mmetal-WTP
              color-of-WTP  black-WTP
              initst-WTP-1  initst-WTP-2  interst-WTP
              goalst-WTP    endst-WTP
              to-reach-WTP  follows-WTP
              cause-WTP-i   cause-WTP-e
             ))

(GENKB-template
  :herald  "Base sit.WTP -- Water in a Teapot on a hot Plate, ver.3.0.0."
  :templates '(
    (water          (:instance (water-WTP    2))
                    (:a-link   (T-of-WTP-w  0.1)) )
    (teapot         (:instance (tpot-WTP     3)) )
    (hot-plate      (:instance (hplate-WTP   5))
                    (:a-link   (T-of-WTP-p  0.2)) )
    (temperature-of (:instance (T-of-WTP-w   3) (T-of-WTP-p 3))
                    (:a-link   (high-T-WTP  0.1)) )
    (high-temp      (:instance (high-T-WTP   5))
                    (:a-link   (hplate-WTP  0.2)) )
    (in             (:instance (in-WTP       1)) )
    (on             (:instance (on-WTP       1)) )
    (made-of        (:instance (made-of-WTP  1)) )
    (material-metal (:instance (mmetal-WTP   1)) )
    (color-of       (:instance (color-of-WTP 1)) )
    (black          (:instance (black-WTP    1)) )
))


;;;;;;;;  Propositional representaion
;;
;;  sit-WTP       :  (inst-of sit-WTP situation)
;;
;;  black-WTP     :  (inst-of black-WTP black)
;;  cause-WTP-i   :  (cause initst-WTP-2 T-of-WTP-t)
;;  cause-WTP-e   :  (cause interst-WTP  T-of-WTP-w)
;;  color-of-WTP  :  (color-of tpot-WTP black-WTP)
;;  endst-WTP     :  (end-state T-of-WTP-t T-of-WTP-w)
;;  follows-WTP   :  (follows initst-WTP-1 endst-WTP)
;;  goalst-WTP    :  (goal-state T-of-WTP-w high-T-WTP water-WTP)
;;  high-T-WTP    :  (inst-of high-T-WTP high-temp)
;;  hplate-WTP    :  (inst-of hplate-WTP hot-plate)
;;  in-WTP        :  (in water-WTP tpot-WTP)
;;  initst-WTP-1  :  (init-state T-of-WTP-p high-T-WTP on-WTP in-WTP)
;;  initst-WTP-2  :  (init-state T-of-WTP-p on-WTP tpot-WTP)
;;  interst-WTP   :  (inter-state T-of-WTP-t in-WTP water-WTP)
;;  made-of-WTP   :  (made-of tpot-WTP mmetal-WTP)
;;  mmetal-WTP    :  (inst-of mmetal-WTP material-metal)
;;  on-WTP        :  (on tpot-WTP hplate-WTP)
;;  T-of-WTP-p    :  (temperature-of hplate-WTP high-T-WTP)
;;  T-of-WTP-t    :  (temperature-of tpot-WTP high-T-WTP)
;;  T-of-WTP-w    :  (temperature-of water-WTP high-T-WTP)
;;  to-reach-WTP  :  (to-reach initst-WTP-1 goalst-WTP)
;;  tpot-WTP      :  (inst-of tpot-WTP teapot)
;;  water-WTP     :  (inst-of water-WTP water)


;;;;;;  End of file  AMBR/KB/EPISODIC/B_WTP.LSP
```

APPENDIX B

PROPOSITIONAL DESCRIPTIONS OF ALL SITUATIONS

This appendix presents simplified propositional descriptions of all situations involved in the simulation experiments reported in the thesis. They appear in the order they are introduced in Chapter VI: 12 base episodes + 10 target problems.

Note that these are *simplified* representations only! The actual AMBR representations are much more complex. Generally, each line below corresponds to a whole agent with several slots. See Appendix A for an actual representation and compare it with the first group below.

```
;;;;;;  Base sit. WTP (Water in a Teapot on a Plate)

  sit-WTP       :  (inst-of sit-WTP situation)

  black-WTP     :  (inst-of black-WTP black)
  cause-WTP-i   :  (cause initst-WTP-2 T-of-WTP-t)
  cause-WTP-e   :  (cause interst-WTP  T-of-WTP-w)
  color-of-WTP  :  (color-of tpot-WTP black-WTP)
  endst-WTP     :  (end-state T-of-WTP-t T-of-WTP-w)
  follows-WTP   :  (follows initst-WTP-1 endst-WTP)
  goalst-WTP    :  (goal-state T-of-WTP-w high-T-WTP water-WTP)
  high-T-WTP    :  (inst-of high-T-WTP high-temp)
  hplate-WTP    :  (inst-of hplate-WTP hot-plate)
  in-WTP        :  (in water-WTP tpot-WTP)
  initst-WTP-1  :  (init-state T-of-WTP-p high-T-WTP on-WTP in-WTP)
  initst-WTP-2  :  (init-state T-of-WTP-p on-WTP tpot-WTP)
  interst-WTP   :  (inter-state T-of-WTP-t in-WTP water-WTP)
  made-of-WTP   :  (made-of tpot-WTP mmetal-WTP)
  mmetal-WTP    :  (inst-of mmetal-WTP material-metal)
  on-WTP        :  (on tpot-WTP hplate-WTP)
  T-of-WTP-p    :  (temperature-of hplate-WTP high-T-WTP)
  T-of-WTP-t    :  (temperature-of tpot-WTP high-T-WTP)
  T-of-WTP-w    :  (temperature-of water-WTP high-T-WTP)
  to-reach-WTP  :  (to-reach initst-WTP-1 goalst-WTP)
  tpot-WTP      :  (inst-of tpot-WTP teapot)
  water-WTP     :  (inst-of water-WTP water)


;;;;;;  Base sit. BF (Bowl on a Fire burns out)

  sit-BF       :  (inst-of sit-BF situation)

  bowl-BF      :  (inst-of bowl-BF bowl)
  cause-BF-b   :  (cause initst-BF-2 is-burnt-BF)
  cause-BF-d   :  (cause interst-BF  is-dissip-BF)
  endst-BF     :  (end-state is-burnt-BF is-dissip-BF)
  fire-BF      :  (inst-of fire-BF fire)
  follows-BF   :  (follows initst-BF-1 endst-BF)
  goalst-BF    :  (goal-state T-of-BF-w high-T-BF water-BF)
  high-T-BF    :  (inst-of high-T-BF high-temp)
  in-BF        :  (in water-BF bowl-BF)
  initst-BF-1  :  (init-state T-of-BF-f high-T-BF on-BF in-BF)
  initst-BF-2  :  (init-state T-of-BF-f made-of-BF mwood-BF on-BF)
  interst-BF   :  (inter-state is-burnt-BF in-BF bowl-BF)
  is-burnt-BF  :  (is-burnt-out bowl-BF)
  is-dissip-BF :  (is-dissipated water-BF)
  made-of-BF   :  (made-of bowl-BF mwood-BF)
  mwood-BF     :  (inst-of mwood-BF material-wood)
  on-BF        :  (on fire-BF bowl-BF)
  T-of-BF-f    :  (temperature-of fire-BF high-T-BF)
  T-of-BF-w    :  (temperature-of water-BF high-T-BF)
  to-reach-BF  :  (to-reach initst-BF-1 goalst-BF)
  water-BF     :  (inst-of water-BF water)
```

```
;;;;;;  Base sit. GP (Glass on a hot Plate breaks)

  sit-GP       :  (inst-of sit-GP situation)

  cause-GP-b   :  (cause initst-GP-2 is-broken-GP)
  cause-GP-d   :  (cause interst-GP  is-dissip-GP)
  endst-GP     :  (end-state is-broken-GP is-dissip-GP)
  follows-GP   :  (follows initst-GP-1 endst-GP)
  glass-GP     :  (inst-of glass-GP glass)
  goalst-GP    :  (goal-state T-of-GP-w high-T-GP water-GP)
  high-T-GP    :  (inst-of high-T-GP high-temp)
  hplate-GP    :  (inst-of hplate-GP hot-plate)
  in-GP        :  (in water-GP glass-GP)
  initst-GP-1  :  (init-state T-of-GP-p high-T-GP on-GP in-GP)
  initst-GP-2  :  (init-state T-of-GP-p made-of-GP mglass-GP on-GP)
  interst-GP   :  (inter-state is-broken-GP in-GP glass-GP)
  is-broken-GP :  (is-broken glass-GP)
  is-dissip-GP :  (is-dissipated water-GP)
  made-of-GP   :  (made-of glass-GP mglass-GP)
  mglass-GP    :  (inst-of mglass-GP material-glass)
  on-GP        :  (on hplate-GP glass-GP)
  T-of-GP-p    :  (temperature-of hplate-GP high-T-GP)
  T-of-GP-w    :  (temperature-of water-GP high-T-GP)
  to-reach-GP  :  (to-reach initst-GP-1 goalst-GP)
  water-GP     :  (inst-of water-GP water)
```

```
;;;;;;  Base sit. IHC (Immersion Heater in a Cup with water)

  sit-IHC       :  (inst-of sit-IHC situation)

  cause-IHC     :  (cause initst-IHC T-of-IHC-w)
  cup-IHC       :  (inst-of cup-IHC cup)
  endst-IHC     :  (end-state T-of-IHC-w)
  follows-IHC   :  (follows initst-IHC endst-IHC)
  goalst-IHC    :  (goal-state T-of-IHC-w high-T-IHC water-IHC)
  high-T-IHC    :  (inst-of high-T-IHC high-temp)
  imm-htr-IHC   :  (inst-of imm-htr-IHC immersion-heater)
  in-IHC-iw     :  (in imm-htr-IHC water-IHC)
  in-IHC-wc     :  (in water-IHC cup-IHC)
  initst-IHC    :  (init-state T-of-IHC-ih high-T-IHC in-IHC-iw imm-htr-IHC)
  made-of-IHC   :  (made-of cup-IHC mchina-IHC)
  mchina-IHC    :  (inst-of mchina-IHC material-china)
  on-IHC        :  (on saucer-IHC cup-IHC)
  saucer-IHC    :  (inst-of saucer-IHC saucer)
  T-of-IHC-ih   :  (temperature-of imm-htr-IHC high-T-IHC)
  T-of-IHC-w    :  (temperature-of water-IHC high-T-IHC)
  to-reach-IHC  :  (to-reach initst-IHC goalst-IHC)
  water-IHC     :  (inst-of water-IHC water)
```

```
;;;;;;  Base sit. FDO (Food on a Dish in an Oven)

  sit-FDO       :  (inst-of sit-FDO situation)

  cause-FDO-i   :  (cause initst-FDO-2 in-FDO-fo)
  cause-FDO-t   :  (cause interst-FDO  T-of-FDO-f)
  dish-FDO      :  (inst-of dish-FDO baking-dish)
  endst-FDO     :  (end-state T-of-FDO-f)
  follows-FDO   :  (follows initst-FDO-1 endst-FDO)
  food-FDO      :  (inst-of food-FDO food)
  goalst-FDO    :  (goal-state T-of-FDO-f high-T-FDO food-FDO)
  high-T-FDO    :  (inst-of high-T-FDO high-temp)
  initst-FDO-1  :  (init-state T-of-FDO-o high-T-FDO on-FDO in-FDO-do)
  initst-FDO-2  :  (init-state on-FDO in-FDO-do)
  interst-FDO   :  (inter-state in-FDO-fo T-of-FDO-o oven-FDO)
  in-FDO-fo     :  (in food-FDO oven-FDO)
  in-FDO-do     :  (in dish-FDO oven-FDO)
  on-FDO        :  (on dish-FDO food-FDO)
  oven-FDO      :  (inst-of oven-FDO oven)
  rectang-FDO   :  (inst-of rectang-FDO rectang-shape)
  shape-of-FDO  :  (shape-of dish-FDO rectang-FDO)
  T-of-FDO-o    :  (temperature-of oven-FDO high-T-FDO)
  T-of-FDO-f    :  (temperature-of food-FDO high-T-FDO)
  to-reach-FDO  :  (to-reach initst-FDO-1 goalst-FDO)
```

```
;;;;;;  Base sit. MTF (Milk in a Teapot in a Fridge)

  sit-MTF       :  (inst-of sit-MTF situation)

  cause-MTF-i   :  (cause initst-MTF-2 in-MTF-mf)
  cause-MTF-t   :  (cause interst-MTF  T-of-MTF-m)
  color-of-MTF  :  (color-of tpot-MTF green-MTF)
  endst-MTF     :  (end-state T-of-MTF-m)
  follows-MTF   :  (follows initst-MTF-1 endst-MTF)
  fridge-MTF    :  (inst-of fridge-MTF fridge)
  goalst-MTF    :  (goal-state T-of-MTF-m low-T-MTF milk-MTF)
  green-MTF     :  (inst-of green-MTF green)
  initst-MTF-1  :  (init-state T-of-MTF-f low-T-MTF in-MTF-mt in-MTF-tf)
  initst-MTF-2  :  (init-state in-MTF-mt in-MTF-tf)
  interst-MTF   :  (inter-state in-MTF-mf T-of-MTF-f fridge-MTF)
  in-MTF-mf     :  (in milk-MTF fridge-MTF)
  in-MTF-mt     :  (in milk-MTF tpot-MTF)
  in-MTF-tf     :  (in tpot-MTF fridge-MTF)
  low-T-MTF     :  (inst-of low-T-MTF low-temp)
  milk-MTF      :  (inst-of milk-MTF milk)
  T-of-MTF-f    :  (temperature-of fridge-MTF low-T-MTF)
  T-of-MTF-m    :  (temperature-of milk-MTF low-T-MTF)
  to-reach-MTF  :  (to-reach initst-MTF-1 goalst-MTF)
  tpot-MTF      :  (inst-of tpot-MTF teapot)
```

```
;;;;;;  Base sit. ICF (Ice Cube in a Fridge)

  sit-ICF       :  (inst-of sit-ICF situation)

  cause-ICF-i   :  (cause initst-ICF-2 in-ICF-if)
  cause-ICF-t   :  (cause interst-ICF  T-of-ICF-i)
  endst-ICF     :  (end-state T-of-ICF-i)
  follows-ICF   :  (follows initst-ICF-1 endst-ICF)
  fridge-ICF    :  (inst-of fridge-ICF fridge)
  glass-ICF     :  (inst-of glass-ICF glass)
  goalst-ICF    :  (goal-state T-of-ICF-i low-T-ICF ice-cube-ICF)
  ice-cube-ICF  :  (inst-of ice-cube-ICF ice-cube)
  initst-ICF-1  :  (init-state T-of-ICF-f low-T-ICF on-ICF-ig in-ICF-gf)
  initst-ICF-2  :  (init-state on-ICF-ig in-ICF-gf)
  interst-ICF   :  (inter-state in-ICF-if T-of-ICF-f fridge-ICF)
  in-ICF-if     :  (in ice-cube-ICF fridge-ICF)
  on-ICF-ig     :  (on ice-cube-ICF glass-ICF)
  in-ICF-gf     :  (in glass-ICF fridge-ICF)
  low-T-ICF     :  (inst-of low-T-ICF low-temp)
  made-of-ICF   :  (made-of glass-ICF mglass-ICF)
  mglass-ICF    :  (inst-of mglass-ICF material-glass)
  T-of-ICF-f    :  (temperature-of fridge-ICF low-T-ICF)
  T-of-ICF-i    :  (temperature-of ice-cube-ICF low-T-ICF)
  to-reach-ICF  :  (to-reach initst-ICF-1 goalst-ICF)
```

```
;;;;;;  Base sit. BPF (Butter on a Plate in a Fridge)

  sit-BPF       :  (inst-of sit-BPF situation)

  butter-BPF    :  (inst-of butter-BPF butter)
  cause-BPF-i   :  (cause initst-BPF-2 in-BPF-bf)
  cause-BPF-t   :  (cause interst-BPF  T-of-BPF-b)
  circular-BPF  :  (inst-of circular-BPF circular-shape)
  endst-BPF     :  (end-state T-of-BPF-b)
  follows-BPF   :  (follows initst-BPF-1 endst-BPF)
  fridge-BPF    :  (inst-of fridge-BPF fridge)
  goalst-BPF    :  (goal-state T-of-BPF-b low-T-BPF butter-BPF)
  initst-BPF-1  :  (init-state T-of-BPF-f low-T-BPF on-BPF in-BPF-pf)
  initst-BPF-2  :  (init-state on-BPF in-BPF-pf)
  interst-BPF   :  (inter-state in-BPF-bf T-of-BPF-f fridge-BPF)
  in-BPF-bf     :  (in butter-BPF fridge-BPF)
  in-BPF-pf     :  (in plate-BPF fridge-BPF)
  low-T-BPF     :  (inst-of low-T-BPF low-temp)
  made-of-BPF   :  (made-of plate-BPF mchina-BPF)
  mchina-BPF    :  (inst-of mchina-BPF material-china)
  on-BPF        :  (on plate-BPF butter-BPF)
  plate-BPF     :  (inst-of plate-BPF plate)
  shape-of-BPF  :  (shape-of plate-BPF circular-BPF)
  T-of-BPF-f    :  (temperature-of fridge-BPF low-T-BPF)
  T-of-BPF-b    :  (temperature-of butter-BPF low-T-BPF)   ; goal
  to-reach-BPF  :  (to-reach initst-BPF-1 goalst-BPF)


;;;;;;  Base sit STC (Sugar in Tea in a Cup)

  sit-STC         :  (inst-of sit-STC situation)

  cause-STC       :  (cause initst-STC taste-of-STC-t)
  cup-STC         :  (inst-of cup-STC cup)
  endst-STC       :  (end-state taste-of-STC-t)
  follows-STC     :  (follows initst-STC endst-STC)
  goalst-STC      :  (goal-state taste-of-STC-t sweet-STC tea-STC)
  in-STC-st       :  (in sugar-STC tea-STC)
  in-STC-tc       :  (in tea-STC cup-STC)
  initst-STC      :  (init-state taste-of-STC-s sweet-STC in-STC-st sugar-STC)
  on-STC          :  (on saucer-STC cup-STC)
  saucer-STC      :  (inst-of saucer-STC saucer)
  sugar-STC       :  (inst-of sugar-STC sugar)
  sweet-STC       :  (inst-of sweet-STC sweet-taste)
  taste-of-STC-s  :  (taste-of sugar-STC sweet-STC)
  taste-of-STC-t  :  (taste-of tea-STC sweet-STC)
  tea-STC         :  (inst-of tea-STC tea)
  to-reach-STC    :  (to-reach initst-STC goalst-STC)


;;;;;;  Base sit. SFF (Salt in Food in a Fridge)

  sit-SFF         :  (inst-of sit-SFF situation)

  cause-SFF-i     :  (cause initst-SFF-2 in-SFF-ff)
  cause-SFF-tmp   :  (cause interst-SFF T-of-SFF-fd)
  cause-SFF-tst   :  (cause initst-SFF-3 taste-of-SFF-f)
  endst-SFF       :  (end-state T-of-SFF-fd taste-of-SFF-f)
  follows-SFF     :  (follows initst-SFF-1 endst-SFF)
  food-SFF        :  (inst-of food-SFF food)
  fridge-SFF      :  (inst-of fridge-SFF fridge)
  goalst-SFF      :  (goal-state T-of-SFF-fd low-T-SFF food-SFF)
  in-SFF-ff       :  (in food-SFF fridge-SFF)
  in-SFF-pf       :  (in plate-SFF fridge-SFF)
  in-SFF-sf       :  (in salt-SFF food-SFF)
  initst-SFF-1    :  (init-state in-SFF-ff T-of-SFF-fr fridge
                                        in-SFF-sf T-of-SFF-fr)
  initst-SFF-2    :  (init-state on-SFF in-SFF-pf)
  initst-SFF-3    :  (init-state in-SFF-sf taste-of-SFF-s salty-SFF)
  interst-SFF    :  (inter-state in-SFF-ff T-of-SFF-fr fridge-SFF)
  low-T-SFF      :  (inst-of low-T-SFF low-temp)
  on-SFF         :  (on plate-SFF food-SFF)
  plate-SFF      :  (inst-of plate-SFF plate)
  salt-SFF       :  (inst-of salt-SFF salt)
  salty-SFF      :  (inst-of salty-SFF salt-taste)
  taste-of-SFF-s :  (taste-of salt-SFF salty-SFF)
  taste-of-SFF-f :  (taste-of food-SFF salty-SFF)
  T-of-SFF-fd    :  (temperature-of food-SFF low-T-SFF)
  T-of-SFF-fr    :  (temperature-of fridge-SFF low-T-SFF)
  to-reach-SFF   :  (to-reach initst-SFF-1 goalst-SFF)
```

;;;;;; **Base sit. ERW (Egg in Red Water)**

```
  sit-ERW        :  (inst-of sit-ERW situation)

  cause-ERW      :  (cause initst-ERW  color-of-ERW-e)
  color-of-ERW-e :  (color-of egg-ERW red-ERW)
  color-of-ERW-w :  (color-of water-ERW red-ERW)
  egg-ERW        :  (inst-of egg-ERW egg)
  endst-ERW      :  (end-state color-of-ERW-e)
  follows-ERW    :  (follows initst-ERW endst-ERW)
  goalst-ERW     :  (goal-state color-of-ERW-e egg-ERW)
  in-ERW-ew      :  (in egg-ERW water-ERW)
  in-ERW-wt      :  (in water-ERW tpot-ERW)
  initst-ERW     :  (init-state color-of-ERW-w red-ERW egg-ERW in-ERW-ew)
  made-of-ERW    :  (made-of tpot-ERW mmetal-ERW)
  mmetal-ERW     :  (inst-of mmetal-ERW material-metal)
  red-ERW        :  (inst-of red-ERW red)
  to-reach-ERW   :  (to-reach initst-ERW goalst-ERW)
  tpot-ERW       :  (inst-of tpot-ERW teapot)
  water-ERW      :  (inst-of water-ERW water)
```

;;;;;; **Base sit. GWB (Glass in a Wooden Box)**

```
  sit-GWB       :  (inst-of sit-GWB situation)

  box-GWB       :  (inst-of box-GWB box)
  cause-GWB     :  (cause in-GWB protects-GWB)
  endst-GWB     :  (end-state protects-GWB)
  follows-GWB   :  (follows initst-GWB endst-GWB)
  glass-GWB     :  (inst-of glass-GWB glass)
  goalst-GWB    :  (goal-state protects-GWB)
  in-GWB        :  (in glass-GWB box-GWB)
  initst-GWB    :  (init-state glass-GWB box-GWB in-GWB)
  made-of-GWB-b :  (made-of box-GWB mwood-GWB)
  made-of-GWB-g :  (made-of glass-GWB mglass-GWB)
  mglass-GWB    :  (inst-of mglass-GWB material-glass)
  mwood-GWB     :  (inst-of mwood-GWB material-wood)
  protects-GWB  :  (protects box-GWB glass-GWB)
  to-reach-GWB  :  (to-reach initst-GWB goalst-GWB)
```

;;;;;; **Target problem HM1 (Heating Milk, variant 1)**

```
  sit-HM1       :  (inst-of sit-HM1 situation)

  goalst-HM1    :  (goal-state T-of-HM1 high-T-HM1)
  in-HM1        :  (in milk-HM1 tpot-HM1)
  initst-HM1    :  (init-state milk-HM1 tpot-HM1 in-HM1 made-of-HM1)
  high-T-HM1    :  (inst-of high-T-HM1 high-temp)
  made-of-HM1   :  (made-of tpot-HM1 mmetal-HM1)
  mmetal-HM1    :  (inst-of mmetal-HM1 material-metal)
  milk-HM1      :  (inst-of milk-HM1 milk)
  T-of-HM1      :  (temperature-of milk-HM1 high-T-HM1)
  to-reach-HM1  :  (to-reach initst-HM1 goalst-HM1)
  tpot-HM1      :  (inst-of tpot-HM1 teapot)
```

;;;;;; **Target problem HM2 (Heating Milk, variant 2)**

```
  sit-HM2       :  (inst-of sit-HM2 situation)

  endst-HM2     :  (end-state ???)
  follows-HM2   :  (follows initst-HM2 endst-HM2)
  high-T-HM2    :  (inst-of high-T-HM2 high-temp)
  hplate-HM2    :  (inst-of hplate-HM2 hot-plate)
  in-HM2        :  (in milk-HM2 tpot-HM2)
  initst-HM2    :  (init-state hplate-HM2 on-HM2 in-HM2 T-of-HM2)
  milk-HM2      :  (inst-of milk-HM2 milk)
  on-HM2        :  (on hplate-HM2 tpot-HM2)
  T-of-HM2      :  (temperature-of hplate-HM2 high-T-HM2)
  tpot-HM2      :  (inst-of tpot-HM2 teapot)


;;;;;;  Target problem CM1  (Cooling Milk, variant 1)

  sit-CM1       :  (inst-of sit-CM1 situation)

  goalst-CM1    :  (goal-state T-of-CM1 low-T-CM1)
  in-CM1        :  (in milk-CM1 tpot-CM1)
  initst-CM1    :  (init-state milk-CM1 tpot-CM1 in-CM1 made-of-CM1)
  low-T-CM1     :  (inst-of low-T-CM1 low-temp)
  made-of-CM1   :  (made-of tpot-CM1 mmetal-CM1)
  milk-CM1      :  (inst-of milk-CM1 milk)
  mmetal-CM1    :  (inst-of mmetal-CM1 material-metal)
  T-of-CM1      :  (temperature-of milk-CM1 low-T-CM1)
  to-reach-CM1  :  (to-reach initst-CM1 goalst-CM1)
  tpot-CM1      :  (inst-of tpot-CM1 teapot)


;;;;;;  Target problem CM2  (Cooling Milk, variant 2)

  sit-CM2       :  (inst-of sit-CM2 situation)

  black-CM2     :  (inst-of black-CM2 black)
  color-of-CM2  :  (color-of tpot-CM2 black-CM2)
  goalst-CM2    :  (goal-state ???)
  to-reach-CM2  :  (to-reach initst-CM2 goalst-CM2)
  fridge-CM2    :  (inst-of fridge-CM2 fridge)
  in-CM2-mt     :  (in milk-CM2 tpot-CM2)
  in-CM2-tf     :  (in tpot-CM2 fridge-CM2)
  initst-CM2    :  (init-state fridge-CM2 in-CM2-tf in-CM2-mt T-of-CM2)
  low-T-CM2     :  (inst-of low-T-CM2 low-temp)
  milk-CM2      :  (inst-of milk-CM2 milk)
  T-of-CM2      :  (temperature-of fridge-CM2 low-T-CM2)
  tpot-CM2      :  (inst-of tpot-CM2 teapot)


;;;;;;  Target problem WB1  (Water in a wooden Bowl)

  sit-WB1       :  (inst-of sit-WB1 situation)

  bowl-WB1      :  (inst-of bowl-WB1 bowl)
  goalst-WB1    :  (goal-state T-of-WB1 high-T-WB1)
  in-WB1        :  (in water-WB1 bowl-WB1)
  initst-WB1    :  (init-state water-WB1 bowl-WB1 in-WB1 made-of-WB1)
  high-T-WB1    :  (inst-of high-T-WB1 high-temp)
  made-of-WB1   :  (made-of bowl-WB1 mwood-WB1)
  mwood-WB1     :  (inst-of mwood-WB1 material-wood)
  T-of-WB1      :  (temperature-of water-WB1 high-T-WB1)
  to-reach-WB1  :  (to-reach initst-WB1 goalst-WB1)
  water-WB1     :  (inst-of water-WB1 water)


;;;;;;  Target problem WG1  (Water in a Glass)

  sit-WG1       :  (inst-of sit-WG1 situation)

  color-of-WG1  :  (color-of glass-WG1 white-WG1)
  glass-WG1     :  (inst-of glass-WG1 glass)
  goalst-WG1    :  (goal-state T-of-WG1 high-T-WG1)
  in-WG1        :  (in water-WG1 glass-WG1)
  initst-WG1    :  (init-state water-WG1 glass-WG1 in-WG1 made-of-WG1)
  high-T-WG1    :  (inst-of high-T-WG1 high-temp)
  made-of-WG1   :  (made-of glass-WG1 mglass-WG1)
  mglass-WG1    :  (inst-of mglass-WG1 material-glass)
  T-of-WG1      :  (temperature-of water-WG1 high-T-WG1)
  to-reach-WG1  :  (to-reach initst-WG1 goalst-WG1)
  water-WG1     :  (inst-of water-WG1 water)
  white-WG1     :  (inst-of white-WG1 white)


;;;;;;  Target problem SF1  (Salty Food, variant 1)

  sit-SF1       :  (inst-of sit-SF1 situation)

  food-SF1      :  (inst-of food-SF1 food)
  goalst-SF1    :  (goal-state taste-of-SF1 salty-SF1)
  initst-SF1    :  (init-state food-SF1 plate-SF1 on-SF1 made-of-SF1)
  made-of-SF1   :  (made-of plate-SF1 mchina-SF1)
  mchina-SF1    :  (inst-of mchina-SF1 material-china)
  on-SF1        :  (on plate-SF1 food-SF1)
  plate-SF1     :  (inst-of plate-SF1 plate)
  salty-SF1     :  (inst-of salty-SF1 salt-taste)
  taste-of-SF1  :  (taste-of food-SF1 salty-SF1)
  to-reach-SF1  :  (to-reach initst-SF1 goalst-SF1)


;;;;;;  Target problem SF2  (Salty Food, variant 2)

  sit-SF2       :  (inst-of sit-SF2 situation)

  endst-SF2     :  (end-state ???)
  follows-SF2   :  (follows initst-SF2 endst-SF2)
  food-SF2      :  (inst-of food-SF2 food)
  in-SF2        :  (in salt-SF2 food-SF2)
  initst-SF2    :  (init-state salt-SF2 food-SF2 plate-SF2)
  on-SF2        :  (on plate-SF2 food-SF2)
  plate-SF2     :  (inst-of plate-SF2 plate)
  salt-SF2      :  (inst-of salt-SF2 salt)


;;;;;;  Target problem EHW  (Egg in Hot Water)

  sit-EHW       :  (inst-of sit-EHW situation)

  color-of-EHW  :  (color-of egg-EHW white-EHW)
  egg-EHW       :  (inst-of egg-EHW egg)
  endst-EHW     :  (end-state ???)
  follows-EHW   :  (follows initst-EHW endst-EHW)
  in-EHW-ew     :  (in egg-EHW water-EHW)
  in-EHW-wt     :  (in water-EHW tpot-EHW)
  initst-EHW    :  (init-state egg-EHW in-EHW-ew in-EHW-wt T-of-EHW)
  high-T-EHW    :  (inst-of high-T-EHW high-temp)
  made-of-EHW   :  (made-of tpot-EHW mmetal-EHW)
  mmetal-EHW    :  (inst-of mmetal-EHW material-metal)
  T-of-EHW      :  (temperature-of water-EHW high-T-EHW)
  tpot-EHW      :  (inst-of tpot-EHW teapot)
  water-EHW     :  (inst-of water-EHW water)
  white-EHW     :  (inst-of white-EHW white)


;;;;;;  Target problem ICC  (Ice Cube in Coke)

  sit-ICC       :  (inst-of sit-ICC situation)

  follows-ICC   :  (follows initst-ICC endst-ICC)
  coke-ICC      :  (inst-of coke-ICC coke)
  endst-ICC     :  (end-state ???)
  glass-ICC     :  (inst-of glass-ICC glass)
  ice-cube-ICC  :  (inst-of ice-cube-ICC ice-cube)
```

```
in-ICC-ic     :  (in ice-cube-ICC coke-ICC)
in-ICC-cg     :  (in coke-ICC glass-ICC)
initst-ICC    :  (init-state ice-cube-ICC in-ICC-ic in-ICC-cg T-of-ICC)
low-T-ICC     :  (inst-of low-T-ICC low-temp)
made-of-ICC   :  (made-of glass-ICC mglass-ICC)
mglass-ICC    :  (inst-of mglass-ICC material-glass)
on-ICC        :  (on table-ICC glass-ICC)
table-ICC     :  (inst-of table-ICC table)
T-of-ICC      :  (temperature-of ice-cube-ICC low-T-ICC)
```

Bibliography:

Anderson, J. (1983). *The architecture of cognition*. Cambridge, MA: Harvard University Press.

Anderson, J. (1993). *Rules of the mind*. Hillsdale, NJ: Erlbaum.

Anderson, J. & Thompson, R. (1989). Use of analogy in a production system architecture. In S. Vosniadou & A. Ortony (Eds.), *Similarity and analogical reasoning*. New York, NY: Cambridge University Press.

Barclay, C.R. (1986). Schematization of autobiographical memory. In D.C. Rubin (Ed.), *Autobiographical memory* (pp. 82-99). Cambridge, UK: Cambridge University Press.

Carbonell, J. (1983). Learning by analogy: Formulating and generalizing plans from past experience. In R. Michalski, J. Carbonell, & T. Mitchell (Eds.), *Machine learning*. Palo Alto, CA: Tioga.

Chalmers, D., French, R. & Hofstadter, D. (1992). High-level perception, representation, and analogy: A critique of artificial intelligence methodology. *Journal for Experimental and Theoretical Artificial Intelligence, 4*(3), 185-211. (Also available in edited form in (Hofstadter et al., 1995))

Charniak, E. (1983). Passing markers: A theory of contextual influence in language comprehension. *Cognitive Science, 7 (3)*, 171-190.

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

Dunker, K. (1945). On problem solving. *Psychological Monographs 38* (270).

Evans, T.G. (1968). A program for the solution of a class of geometric-analogy intelligence-test questions. In M. Minsky (Ed.), *Semantic information processing*. Cambridge, MA: MIT Press.

Fahlman, S. (1979). *NETL: A system for representing and using real world knowledge.* Cambridge, MA: MIT Press.

Falkenhainer, B. (1988). *Learning from physical analogies: A study in analogy and the explanation process.* PhD Thesis, University of Illinois.

Falkenhainer, B. (1990a). Analogical interpretation in context. *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society* (pp. 69-76). Hillsdale, NJ: Erlbaum.

Falkenhainer, B. (1990b). A unified approach to explanation and theory formation. In J. Shrager & P. Langley (Eds.), *Computational models of scientific discovery and theory formation* (pp. 157-196). Los Altos, CA: Morgan Kaufmann.

Falkenhainer, B., Forbus, K., & Gentner, D. (1986). The structure-mapping engine. *Proceedings of the Fifth Annual Conference on Artificial Intelligence* (pp. 272-277). Los Altos, CA: Morgan Kaufman.

Forbus, K., Ferguson, R., & Gentner, D. (1994a). Incremental structure mapping. *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society* (pp. 313-318). Georgia, Hillsdale, NJ: LEA.

Forbus, K., Gentner, D., & Law, K. (1994b). MAC/FAC: A model of similarity-based retrieval. *Cognitive Science, 19*, 141-205.

Forbus, K. & Oblinger, D. (1990). Making SME greedy and pragmatic. *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society* (pp. 61-68). Hillsdale, NJ: Erlbaum.

French, R. (1995). *The subtlety of sameness: A theory and computer model of analogy-making*. Cambridge, MA: MIT Press.

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science, 7*(2), 155-170.

Gentner, D. (1989). The mechanisms of analogical learning. In S. Vosniadou & A. Ortony (Eds.), *Similarity and analogical reasoning* (pp. 199-241). New York, NY: Cambridge University Press.

Gentner, D. & Landers, R. (1985). Analogical reminding: A good match is hard to find. In *Proceedings of the International Conference on Systems, Man and Cybernetics*. New York: IEEE.

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

Gick, M & Holyoak, K. (1980). Analogical problem solving. *Cognitive Psychology 12*, 306-355.

Gick, M. & Holyoak, K. (1983). Schema induction and analogical transfer. *Cognitive Psychology, 15*, 1-38.

Goshwami, U. (1992). *Analogical reasoning in children*. Hillsdale, NJ: Erlbaum.

Hall, R. (1989). Computational approaches to analogical reasoning: A comparative analysis. *Artificial Intelligence, 39*, 39-120.

Harnad, S. (1990). The symbol grounding problem. *Physica D, 42*, 335-346.

Hendler, J. (1988). *Integrating marker-passing and problem-solving: A spreading-activation approach to improved choice in planning*. Hillsdale, NJ: Erlbaum.

Hendler, J. (1989). Marker-passing over microfeatures: Towards a hybrid symbolic/connectionist model. *Cognitive Science, 13*, 79-106.

Hofstadter, D. (1983). The architecture of Jumbo. *Proceedings of the International Machine Learning Workshop*. Monticello, IL.

Hofstadter, D. (1984). *The Copycat project: An experiment in nondeterminism and creative analogies*. AI Memo No. 755, Massachusetts Institute of Technology, Cambridge, MA.

Hofstadter, D. & Mitchell, M. (1991). The Copycat project: A model of mental fluidity and analogy-making. *CRCC Technical Report No. 58*, Center for

Research on Concepts and Cognition, Indiana University, Bloomington, IN. (Also available in K. Holyoak & J. Barnden (Eds.), (1994). *Advances in connectionist and neural computation theory*, v*ol. 2: Analogical connections*. Norwood, NJ: Ablex.)

Hofstadter, D. & the Fluid Analogies Research Croup (1995). *Fluid concepts and creative analogies: Computer models of the fundamental mechanisms of thoughts*. New York, NY: Basic Books.

Holland, J., Holyoak, K., Nisbett, R., & Thagard, P. (1986). *Induction: Processes of inference, learning, and discovery*. Cambridge, MA: MIT Press.

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

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

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

Hummel, J. & Holyoak, K. (1996). LISA: A computational model of analogical inference and schema induction. *Proceedings of the Eighteenth Annual Conference of the Cognitive Science Society.* Hillsdale, NJ: Erlbaum.

Hummel, J. & Holyoak, K. (1997). Distributed representations of structure: A theory of analogical access and mapping. *Psychological Review,104*(3), 427-466.

Keane, M. (1987). On retrieving analogues when solving problems. *Quaterly Journal of Experimental Psychology, 39A*, 29-41.

Keane, M. (1988). *Analogical problem solving*. Chichester: Ellis Horwood.

Keane, M. & Brayshaw, M. (1988). The incremental analogy machine: A computational model of analogy. In D. Sleeman (Ed.), *Third European Working Session on Machine Learning*. London: Pitman.

Keane, M., Ledgeway, K. & Duff, S. (1994). Constraints on analogical mapping: A comparison of three models. *Cognitive Science, 18*, 387-438.

Kedar-Cabelli, S. (1988). Towards a computational model of purpose-directed analogy. In A. Prieditic (Ed.), *Analogica*. Los Altos, CA: Morgan Kaufmann.

Kokinov, B. (1988). Associative memory-based reasoning: How to represent and retrieve cases. In T. O'Shea & V. Sgurev (Eds.), *Artificial Intelligence III: Methodology, systems, applications*. Amsterdam: Elsevier.

Kokinov, B. (1989). About modeling some aspects of human memory. In F. Klix, N.Streitz, Y.Waern, & H.Wandke (Eds.), *MACINTER II*. Amsterdam: North-Holland.

Kokinov, B. (1990). Associative memory-based reasoning: Some experimental results. *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum.

Kokinov, B. (1992a). Inference evaluation in deductive, inductive, and analogical reasoning. *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum.

Kokinov, B. (1992b). Similarity in analogical reasoning. In B. du Boulay & V. Sgurev (Eds.), *Artificial Intelligence V: Methodology, systems, applications*. Amsterdam, Elsevier.

Kokinov, B. (1994a). A hybrid model of reasoning by analogy. In K. Holyoak & J. Barnden (Eds.), *Advances in connectionist and neural computation theory*, v*ol. 2: Analogical connections* (pp. 247-318). Norwood, NJ: Ablex.

Kokinov, B. (1994b). The DUAL cognitive architecture: A hybrid multi-agent approach. In A. Cohn (Ed.), *Proceedings of the Eleventh European Conference of Artificial Intelligence* (pp. 203-207). London: John Wiley & Sons, Ltd.

Kokinov, B. (1994c). The context-sensitive cognitive architecture DUAL. *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum.

Kokinov, B. (1995). A dynamic approach to context modeling. In P. Brezillon & S. Abu-Hakima (Eds.), *Working notes of the IJCAI-95 Workshop on "Modeling context in knowledge representation and reasoning"*. Paris: Institute B. Pascal. LAFORIA 95/11.

Kokinov, B. (1997). AMBR view on analogy. Paper presented at the *Conference on Thinking*. London, UK. March, 1997.

Kokinov, B. (1998). Analogy is like cognition: Dynamic, emergent, and context-sensitive. In K. Holyoak, D. Gentner, & B. Kokinov (Eds.), *Advances in analogy research: Integration of theory and data from the cognitive, computational, and neural sciences*. Sofia, Bulgaria: NBU Press.

Kokinov, B., Nikolov, V. & Petrov, A. (1996). Dynamics of emergent computation in DUAL. In A. Ramsay (Ed.), *Artificial Intelligence: Methodology, systems, applications* (pp. 303-311). Amsterdam:IOS Press.

Kokinov, B. & Yoveva, M. (1996). Context effects on problem solving. *Proceedings of the Eighteenth Annual Conference of the Cognitive Science Society* (pp. 586-590). La Jolla, CA: Erlbaum.

Kokinov, B., Hadjiilieva, K., & Yoveva, M. (1997). Is a hint always useful? *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum.

Kolodner, J. (1993). *Case-based reasoning*. San Mateo, CA: Morgan Kaufmann.

McCarthy, J. & Hayes, P. (1969). Some philosophical problems from the stand-point of artificial intelligence. In B. Meltzer & D. Michie (Eds.), *Machine Intelligence*, Vol. 4. Edinburgh: Edinburgh University Press.

McClelland, J. & Rumelhart, D. (1981). An interactive activation model of context effects in letter perception: Part 1. An account of basic findings. *Psychological Review, 88*, 375-407.

Michalski, R. (1989). Two-tiered concept meaning, inferential matching, and conceptual cohesiveness. In S. Vosniadou & A. Ortony (Eds.), *Similarity and analogical reasoning*. New York: Cambridge Univ. Press.

Minsky, M. (1986). *The society of mind*. New York: Simon & Schuster.

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

Postman, L. & Phillips, L.W. (1965). Short-term temporal changes in free recall. *Quaterly Journal of Experimental Psychology, 17*, 132-138.

Quillian, M. (1969). The teachable language comprehender: A simulation program and theory of language. *Communication of the ACM, 12*, 459-476.

Ross, B. (1984). Remindings and their effects in learning a cognitive skill. *Cognitive Psychology, 16*, 371-416.

Ross, B. (1987). This is like that: The use of earlier problems and the separation of similarity effects. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 13*, 629-639.

Rumelhart, D., McClelland, J., & the PDP Research Group (1986). *Parallel distributed processing, vol.1: Foundations*. Cambridge, MA: MIT Press.

Seifert, C., McKoon, G., Abelson, R., & Ratcliff, R. (1986). Memory connections between thematically similar episodes. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 12,* 220-231.

Thagard, P., Holyoak, K., Nelson, G., & Gochfeld, D. (1990). Analog retrieval by constraint satisfaction . *Aftificial Intelligence 46*, 259-310.

Turner, M. & Fauconnier, G. (1995). Conceptual integration and formal expression. *Metaphor and Symbolic Activity, 10*(3), 183-204.

Tversky,A. (1977). Features of similarity. *Psychological Review, 84*, 327-352

Veloso, M. (1994). *Planning and learning by analogical reasoning*. Berlin: Springer-Verlag.

Waltz, D.I. (1975). Understanding line drawings of scenes with shadows. In P.Winston (Ed.), *The psychology of computer vision*. McGraw-Hill.

Wharton, C., Holyoak, K., Downing, P., Lange, T., & Wickens, T. (1991). Retrieval competition in memory for analogies. *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society* (pp. 528-533). Hillsdale, NJ: Erlbaum.