NEW BULGARIAN UNIVERSITY
Graduate Program in Cognitive Science

Extensions of DUAL and AMBR

Alexander Alexandrov Petrov

Thesis submitted as a partial fulfillment of the requirements for a M.Sc. degree

Supervisor: Assistant Professor Boicho Kokinov

Sofia, July 1997

TABLE OF CONTENTS

CHAPTER I

OVERVIEW AND GOALS OF THE PROJECT

1.1. Overview

This thesis describes a cognitive architecture named DUAL and a cognitive model that has been built on its basis. It provides detailed specifications of the architecture and the model and discusses their properties from the perspective of the study of human cognition. In addition, it reports some simulation experiments performed with the model.

The research documented here is part of a large research program launched by Boicho Kokinov about ten years ago (Kokinov, 1988). The project which is the focus of this thesis aims at replication, solidification, and extension of the results obtained so far (Kokinov, 1994a,b,c; Kokinov et al., 1996). An important ingredient of the project is the development of a computer implementation of the architecture as a foundation for future research on DUAL.

The model discussed in the next chapters is called AMBR — an acronym for *Associative Memory-Based Reasoning* (Kokinov, 1988). AMBR is a cognitive model with very broad scope. Its ambition is to offer a unified account of deduction, induction, and analogy. They are treated not as different and idiosyncratic modes of reasoning but rather as manifestations of one and the same underlying mechanism.

This thesis describes AMBR2 — a revised and extended sequel of the model presented by (Kokinov, 1994a). The new version is characterized by a number of new features on the theoretical side, as well as an entirely new and portable computer implementation.

The model has been tested on a number of examples in a simulation-experiment setting. Since all the examples can be classified as analogies, AMBR2 is discussed here chiefly as a model of analogy-making. This does not imply that the model has given up the generality of the original proposal. The presentation is deliberately centered on analogy-making in order to stay focused on the concrete results that have been obtained so far.

1.2. Main Ideas of DUAL

> *Theorists are strongly influenced by their various preconceptions. The most deeply rooted preconception guiding my theorizing is a belief in the unity of human cognition, that is, that all the higher cognitive processes, such as memory, language, problem solving, imagery, deduction, and induction, are different manifestations of the same underlying system.*
>
> (Anderson 1983, p.1)

The prime mover of DUAL research is the notion of integration. This general thema (Holton, 1973) is instantiated in various ways, some of which are outlined below and will be discussed extensively in the next chapters.

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 that are brought together into a coherent whole. 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 and are present simultaneously 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 centralized executive that controls the whole system, makes large-scale decisions, allocates resources, resolves conflicts and so on. Instead, small-scale DUAL agents interact with one another, locally, and 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 that underlies DUAL's operation and to the lack of rigid and preprogrammed specification of the computation. In particular, there is no sharp boundary between the 'task' or the 'problem' given to the system to solve, and the 'context' that accompanies this problem. One and the same problem can be solved in different ways during two successive runs of a DUAL-based model, in spite of the strictly deterministic character of the architecture.

In a little more detail, each DUAL agent is a hybrid entity that serves both representational and processing purposes. It has a *micro-frame* that stores declarative and procedural information. The micro-frame has slots which are usually filled by references to other DUAL agents. These references can be conceptualized as *links* that connect the agents in a network-like structure. Correspondingly, each DUAL agent can be conceptualized as a *node* in that network. Thus, there are two metaphors related to a DUAL-based system. On one hand, it can be viewed as a population of interacting agents; on the other, it resembles a network of interconnected nodes.

Each DUAL agent is capable of performing certain simple information-processing tasks. Most of these tasks involve interaction with its neighbors. DUAL interactions are relatively simple — they boil down to exchange of *symbols* (discrete, compositional units) and *activation* (continuous, additive quantity) between two agents.

Each DUAL agent has a *connectionist processor* that is capable of receiving activation, transforming it in accordance to some simple numerical rule, and sending it further. There is an *activation level* associated with each agent and a *weight* associated with each link. The former is a measure of the degree of relevance of the particular agent to the particular task and context. The latter is a measure of the intensity of the interaction between a pair of agents.

On the other side of the coin, each agent has a symbolic processor that is capable of receiving a symbol, transforming it in accordance to some simple symbolic routine, and potentially sending (another) symbol to its peers. The symbolic processor can discriminate its neighbors on the basis on the contents of their micro-frames as well as on the *label* of the links that establish the interactions between them.

DUAL agents are heterogeneous — they differ both in their declarative components and in the characteristics of their processors (connectionist and symbolic). In addition, some of the agents are *temporary*. They are created dynamically to meet some particular demand and disappear when are no longer needed. The question whether some agent is needed (relevant) or not is answered by the connectionist aspect of the architecture: temporary agents fizzle out when their activation level drops below some lethal threshold.

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. The influence of each agent is proportional to the degree to which it is judged relevant to the particular task and context. As activation levels change continuously, the speed of various agents change accordingly, giving advantage to some and disadvantage to others. Under

different circumstances, different agents will take the lead and they will determine the global outcome of the computation.

DUAL agents (occasionally called *micro-agents*) aggregate in larger 'teams' called *coalitions* and *formations*. It is at this higher level of granularity where the emergent nature of the architecture becomes evident. Coalitions are ensembles of micro-agents that share a more or less stable pattern of interaction. They differ in size and in the degree of interdependence among their members. A DUAL agent can participate in several coalitions simultaneously and to a different extent. It can also join or renounce a given coalition, that is, establish or abandon interactions with its members. Thus, coalitions emerge dynamically out of the local interaction between micro-agents.

The knowledge in a DUAL-based system resides in individual micro-agents — in their micro-frames, built-in procedures, and the attributes (labels and weights) of their interactions. In other words, the population of all DUAL agents comprise the *long-term memory* of the architecture. It may contain thousands of agents. At any given moment, however, only a few dozens of them are active. The active portion of the long-term memory, plus the temporary agents constructed during recent computations, constitute the *working memory* of the architecture. The contents of the working memory changes dynamically as agents gain or lose activity, as well as temporary agents are added or removed.

Finally, it should be noted briefly where all this activation comes from in the first place. There are two sources of activation in the architecture — *goal node* and *input node*. A DUAL-based model is requested to begin working on a given problem by attaching some of the agents describing the problem to the goal node. Other agents are connected to the input node. The activation that comes from these nodes can then spread through the network via the links. There is also spontaneous decay of the activation which limits its propagation and restricts the size of the working memory.

1.3. Scope of the AMBR Model as Reported Here

AMBR is a cognitive 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). These three kinds of reasoning are viewed as slightly different versions of a single uniform reasoning process. In AMBR, the basic description of this process is that it establishes correspondences between two problems, schemes, or situations, 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 schemes (or situations, etc.) 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 might have broader scope.

The models of analogy-making typically decompose it into separate *stages* or *phases* (Hall, 1989; Gentner, 1983, 1989; Forbus et al, 1994a; Holyoak & Thagard, 1989, 1995; Keane, 1988; etc.). For example, one possible decomposition include: (*i*) representation of the target problem, (*ii*) retrieval of a source analog from LTM, (*iii*) mapping the two situations, (*iv*) transfer from the source to the target, (*v*) evaluation of the analogical inferences, and (*vi*) learning. Some researchers (e.g. Gentner, 1989) argue that the stages of analogy-making are relatively independent from one another and are thus susceptible to piecemeal exploration. Others (e.g. Chalmers et al., 1992) oppose to 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 devise an integrated model that realizes all these subprocesses on the uniform foundation of DUAL. This is a very ambitious goal, however. At present, there is no model that incorporates all the aspects listed above. AMBR makes no exception.

The scope of this thesis is restricted to the subprocesses of retrieval and mapping and the integration between the two. These two components of the model have been elaborated in much more detail. Moreover, the simulation experiments performed with the model so far deal with retrieving a situation from LTM and mapping it to a target situation.

To sum up, this thesis deals not with AMBR in general but with its current version denoted AMBR2. (The version documented in Kokinov (1994a) will be referred to as AMBR1.) AMBR2 is presented as an integrated model of analogical retrieval and mapping. Due to the ambitiousness of the AMBR project, even this limited version of the model is quite elaborated and deserves attention and evaluation in its own right.

1.4. Goals of the Project

This section is organized in a slightly non-canonical way. An AI veteran (McDermott, 1981) gives the following advice about developing and documenting complex computer models:

> *[...] If a thorough report on a mere actual implementation was required, or even* allowed, *as a Ph.D. thesis, progress would appear slower, but it would be real. [...]*
>
> *My proposal is that thesis research, or any other two-year effort, should be organized as follows:*
>
> *As before, a new problem, or* old problem with partial solution, *should be chosen. The part of the problem where most progress could be made (a conceptual "inner loop") should be thought about hardest. Good ideas developed here should appear in a research proposal.*
>
> *The first half of the time allotted thereafter should be applied to writing Version* n+1, *where* n *is the version number you started with (0 for virgin problems). (Substantial rewriting of Version n should be anticipated.) The second half should be devoted to writing the report and improving Version* n+1 *with enough breadth, clean code, and new user features to make it useful to the next person that needs it.*
>
> *The research report will then describe the improvements made to Version* n, *good ideas implemented, and total progress made in solving the original problem. Suggestions for further improvements should be included, in the future subjunctive tense.*
>
> (McDermott, 1981; p.160; emphasis in original)

We have taken this advice seriously. Consequently, the goals of this project are the following:

1. Developing a portable computer program that implements the architecture DUAL. This program will serve as a foundation not only for AMBR but for additional DUAL-based models in the future.

2. Extending and refining the conceptual specification of the architecture. Resolving the ambiguities of the original specification.

3. Developing a portable computer program that implements AMBR2.

4. Extending the AMBR1 model by proposing new computational mechanisms, new types of agents, refining the knowledge-representation scheme and so forth.

5. Performing simulation experiments with the model.

CHAPTER II

BACKGROUND

The current work has a lot of precursors and sources of ideas. This section tries to mention at least some of them.

2.1. Cognitive Architectures

In recent years, a number of proposals for cognitive architectures have been published in the literature. Some of them have been declared specifically as cognitive architectures (Anderson, 1983, 1993; Newell, 1991) while others are centered around a specific model but are nevertheless sufficiently general to fall into this category (Holland et al., 1986; Kokinov, 1994a; Hofstadter, 1984 1995).

One very influential proposal is Soar (Laird et al., 1987; Newell, 1991). It will not be discussed here, however, because it is relatively far from the topics that are central to this thesis. In particular, Soar belongs to (and, indeed, epitomizes) the symbolic approach to cognitive modeling, while DUAL seeks integration with connectionism. Soar also lacks separate declarative component — all long-term knowledge resides in production rules. This is quite different from the DUAL representation scheme.

John Anderson's ACT* (1983) and ACT-R (1993) have greatly influenced the research on DUAL. There are a lot of deep similarities between the two proposals. Both architectures use hybrid representation schemes with declarative and procedural components. Both architectures depend on a connectionist mechanism for delineating the scope of the working memory and for retrieval of relevant information. In both architectures, the speed of the symbolic operations varies in accordance to the activation level of the declarative aspect. This list, which can easily be prolonged, reveals that DUAL has many features in common with ACT* and its successor ACT-R.

We now turn to the differences among the two proposals. In the discussion below, we will refer to the more recent version — ACT-R — though most of the points apply to ACT* as well.

DUAL puts less emphasis on the procedural-declarative distinction than ACT-R does. DUAL seeks integration between these two aspects and,

therefore, does not separate them. In particular, the knowledge-representation scheme in DUAL is based on frames, thus putting both aspects together at the micro-level. At the macro-level, the DUAL network is a unified long-term memory that keeps declarative and procedural pieces of knowledge. In contrast, ACT-R has separate long-term memories. The arguments advanced in favor of this separation (e.g. Anderson, 1993, p. 21) are forceful but in our view their main thrust is to justify that the procedural-declarative distinction is warranted in the first place. DUAL certainly accepts the utility of this distinction. What is questioned is whether such radical divorce between declarative and procedural memories, taken as large-scale structures, is really necessary. Detailed treatment of this controversy, however, is beyond the scope of this thesis.

Another difference between DUAL and ACT-R is that the latter depends on a central executive that decides which rule to fire. The expected utilities of all matching rules is calculated in a decentralized and parallel fashion, but at the end only one rule fires. The decision about how much time to allow for matching of productions and when to pick up the winner is made centrally. This is quite different from the approach adopted in DUAL, where all active symbolic processors run in parallel. (With respect to this, DUAL is more similar to Soar than to ACT-R.)

Finally, the connectionist and symbolic aspects are treated on equal footing in DUAL, while in ACT-R the symbolic aspect seems to dominate. Thus, it could be argued that the degree of hybridization in DUAL is greater than in ACT-R.

Another proposal originates from Hofstadter (1984, 1995) and has been instantiated in two related models of analogy-making and high-level perception — Copycat (Hofstadter & Mitchell, 1991; Mitchell, 1993) and Tabletop (French, 1995). The common foundation of these two models is sufficiently general to be considered a cognitive architecture. It this thesis it is referred to as 'Copycat architecture'.

There are many similarities between DUAL and the Copycat architecture. In both of them the overall computation emerges out of local activities carried out by small agents (called *codelets* in Copycat) in the absence of central executive. Both architectures employ the idea that the speed of processing should vary dynamically, reflecting the judged relevance of the pieces of information being processed. However, Copycat uses a very different mechanism for implementing this general idea — codelets are chosen one at a time, with probabilities proportional to their *urgencies*.

The latter point highlights one of the main differences between DUAL and Copycat — the former is deterministic while the latter is stochastic. It is remarkable that the two architectures exhibit very similar properties at the macro-level despite this big underlying difference. On one hand, this is due to the fact that the non-determinism in Copycat is chiefly at the level

of small-scale local decisions, and the statistical outcome of all these micro-level random events yields stable and regular behavior of the system as a whole. On the other hand, the deterministic nature of DUAL does not imply that everything in the architecture is prescribed beforehand. Like in Copycat, the overall behavior of a DUAL-based system emerges out of small local operations which depend on many factors that vary in time. As a consequence, DUAL can generate frequency distributions that are similar to the statistics generated by Copycat (see section 5.3.).

Another difference between DUAL and Copycat is that the latter keeps instances (or *tokens*) and concepts (or *types*) separate. Instances reside in the *Workspace* while concepts reside in the *Slipnet*. In DUAL, both kinds of agents are in the same network, which allows for closer interaction between them.

In addition, DUAL is characterized by greater coupling between the procedural and the declarative aspect. In Copycat, the codelets are detached from the nodes, although the interdependence between the two is very strong. In DUAL, however, this interdependence is even stronger because everything is in one and the same hybrid agent having procedural and declarative aspects.

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 of the research on the topic 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*. For instance, one possible division include:

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

Retrieval of an appropriate analog (called *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 in the framework of the target.

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. Keeping this in mind, we start our presentation with a brief overview of some previous models. Out of the many titles, we have chosen only those which have directly influenced our work.

Dedre Gentner (1983) focuses her attention on the process of establishing a mapping between two situations represented in predicate calculus. She proposes the *systematicity principle* — interlocking relations from one description should be mapped onto interlocking relations from the other. The claim is that this can be achieved on a syntactic basis by giving special treatment to higher-order predicates like 'cause'. This general idea has been implemented in a computer program called Structure Mapping Engine (SME) (Falkenhainer, Forbus & Gentner, 1986) and is present in one form or another in all subsequent models.

Keith Holyoak and Paul Thagard argue that the structural considerations are surely necessary but not sufficient for many analogies. They propose the *multiconstraint theory* (Holyoak & Thagard, 1989, 1995) that operates with three distinct interrelated types of constraints: similarity, structure, and purpose. This theory is embodied in two computational models addressing the phase of mapping (Holyoak & Thagard, 1989) and retrieval (Thagard et al, 1990).

The connectionist computational procedure of *constraint satisfaction* is a defining feature of these models. It allows for simultaneous (partial) satisfaction of the constraints by relaxation of a neural network. The nodes of this constraint-satisfaction network (CSN) represent hypotheses about correspondences between elements of the two situations. The links represent the constraints.

Holyoak & Thagard (1989) have used this algorithm in their ACME model (Analogical Constraint Mapping Engine). This model works in two steps: (*i*) the CSN is constructed serially by a symbolic routine and (*ii*) the network is run until it reaches asymptotic state. Metaphorically, this model 'translates' the task from symbolic terms (propositional calculus) to the 'language' of localist connectionist networks. There is no genuine interaction between the symbolic and the connectionist component. Therefore, ACME can be considered as a precursor of hybrid models but in itself it does not constitute such a model.

Mark Keane (1988) introduces the notion of *incremental mapping*. His Incremental Analogy Machine (IAM) model builds correspondences by starting from a *seed match* and gradually expanding it, backtracking when necessary.

Hummel & Holyoak (1997) propose an integrated model of analogical retrieval (or *access*) and mapping called LISA (Learning and Inference

with Schemas and Analogies). This model is deeply rooted in connectionism and uses dynamic binding for the purposes of structured representations. This model, like a few others models of retrieval, is discussed in more detail in section 4.1.3.2.

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 problematic situation. There are strong arguments, however, that this latter perceptual aspect is crucial to analogy making (Chalmers et al., 1992). 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.

CHAPTER III

DUAL — A HYBRID COGNITIVE ARCHITECTURE

A cognitive architecture is a relatively complete proposal about the structure of human cognition. As such, it has many things to specify: mental representations, information processing mechanisms, flow of control, etc. Therefore, the description of an architecture is usually long and complex. DUAL makes no exception.

This chapter describes the DUAL cognitive architecture. The presentation begins with a general overview of the basic terms and concepts used in DUAL. It then continues with detailed descriptions at different levels of granularity.

3.1. Anatomy of DUAL

3.1.1. Basic Terms

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.

3.1.2. A Biological Analogy

> *We can be sure that the human mind is not to be explained by a small set of assumptions. There is no reason to suppose the mind is simpler than the body.*
>
> (Anderson 1983, p.42)

The organization of DUAL is analogous to the organization of multicellular biological organisms. This analogy brings forward many ideas that will facilitate the understanding of DUAL. Therefore, it is presented here as an introduction to the more technical subsections that follow.

The basic structural and functional unit of most living organisms is the *living cell*. Higher organisms[1] are built of a huge number of cells. Each individual cell is tiny and typically cannot live in isolation. Their coordinated activities, however, produce amazing results.

Living cells combine into larger structures of increasing complexity. Thus, *tissues* consist of similar cells specialized to perform a specific function. For instance, there are muscular, nerve, epithelial, and connective tissues. *Organs* are made up of different tissues which form a structural and functional unit. The collection of all organs with a common function is called a *system*. For example, the respiratory system in the human body consists of several organs, including the nose, the pharynx, the larynx, the trachea, and the lungs. Finally, the *organism* itself could be considered a higher-order system of all its systems. At a lower level of abstraction, however, it still consists of nothing but cells.

Most of these concepts have a counterpart in DUAL as the following table demonstrates:

biological domain	**DUAL domain**
living cell	DUAL agent
tissue	coalition
organ	coalition, formation
system	formation
organism	system, model

This analogy should not be pushed too far. Besides the similarities, there are big differences. In particular, the complexity in the biological domain overwhelmingly exceeds that in DUAL. (On the other hand, DUAL agents are more flexible than the cells in establishing interactions with their peers.) Still, the analogy has suggestive value:

[1] Unicellular organisms, prokaryotes, viruses, etc. are not considered here.

There are features common to all living cells. These are their *differentia specifica* — the characteristics that make them what they are. To illustrate, each cell has a cytoplasm, performs metabolism in order to stay alive, and originates from some other cell. Analogously, there are features shared by all DUAL agents. These features define what a DUAL agent is and thus are central to the description of the architecture. Subsequent sections deal extensively with such features.

On the other hand, biological systems exhibit a vast diversity of living cells. They differ both in structure and function. Structurally, cells are not atomic — they have parts called *organelles*. Some organelles are present in all cells while others are specific to a certain type. For example, all (eucariote) cells have nuclei but only green plant cells have chloroplasts needed for photosynthesis. (This example also illustrates that the function of a cell is closely related to its structure and vice versa.)

Similarly, DUAL agents also come in varieties, differing in their purpose and internal organization. Agents are not atomic. As will be discussed later, they have parts like slots, activation levels and so on. Some parts are common to all DUAL agents while others are specific to a certain type.

The diversity at the cellular level becomes even greater at upper levels. The same cells combine into different ways to produce a variety of tissues, organs, and so on. Thus, the overall diversity of living organisms is due not only to the diversity in cell types but also to the innumerable possibilities of their combinations.

Analogously, DUAL agents interact with one another. The same agents may build a number of different coalitions depending on the pattern of interaction between them. Moreover, this pattern changes in time, thus providing for even greater variety.

Further, different organisms have different cells. These differences are big across species and smaller within species, but nonetheless always present. This is true even for closely related individuals like a parent and a child. The genotype is unique to each organism[2]. On the other hand, the general cell structure is invariant across all individuals and species. To illustrate, all cells have nuclei but with different chromosomes.

Analogously, there could be many models, each built on top of DUAL but with individual variations. The general architectural principles stay the same; the exact structures and behaviors embedded in each model differ. This chapter of the thesis deals with the general principles of DUAL which are supposed to remain invariant across all DUAL-based models. The next chapter will describe additional features that are specific to the AMBR2 model.

---

[2] Uniovular twins are an exception.

monozygotic

3.1.3. Levels of Description

DUAL-based models are complex systems; biological organisms are even more so. Analysis of such systems must proceed at different levels of granularity. In anatomy, there are subdisciplines devoted to the particular levels: cytology studies the cells, histology — the tissues, and so on. In *dualogy* — the study of DUAL — there are no subdisciplines but still it is useful to analyze the architecture with respect to 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 *retrieval*, *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.

The table below summarizes the three levels used in the analysis of DUAL and the corresponding basic concepts:

level of analysis	basic concept
micro-level	DUAL agent
meso-level	coalition
macro-level	formation

These 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. A major challenge of DUAL's design is to establish coherence not only within but also across levels. This requires thinking at multiple levels at once.

3.2. DUAL at the Microlevel

This section describes 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 the ways of interaction with its 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. Therefore, detailed and unambiguous specification of this basic structural and functional unit is fundamental to the specification of the architecture as a whole.

3.2.1. The Hybrid Nature of DUAL agents

DUAL agents 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. This fundamental philosophical tenet penetrates the architecture and is especially evident in the hybrid nature of the micro-agents.

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 an artificial neural network (Rumelhart & McClelland, 1986). It has an *activation level* attached to it and continuously spreads activation to other agents. In this way, there is a pattern of activation over the population of DUAL agents. This pattern changes dynamically as agents gain or lose activity.

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. All this makes the symbolic aspect of DUAL agents somewhat complicated. On one hand, they *are* symbols; on the other, they can *contain* symbols and *manipulate* on them[3].

It should be emphasized that all four aspects shown in table 3.1 are tightly coordinated. The connectionist aspect influences the symbolic one and vice versa. Similarly, the procedural aspect depends on the declarative one and so on. This cohesion between the different aspects of a DUAL agent ensures its integrity. The DUAL agent is one hybrid entity rather than a combination of parts.

3.2.2. Interactions and Links

The presentation so far reveals that DUAL agents interact intensively with one another. These interactions are very important because they are the fabric that combines micro-agents into larger complexes. Thus, they form the bridge from the microlevel to the meso- and macrolevels. Moreover, interactions are the key factor for the emergence of non-local phenomena out of local activities.

Interactions in DUAL are relatively simple — they always involve only two micro-agents. Multi-agent coalitions are tied up by a number of bilateral interactions. Moreover, interactions are unidirectional. That is, only one of the participants in each interaction plays an active role, the other one being passive or even ignorant.

3.2.2.1. Types of interactions

DUAL interactions are of two kinds: a micro-agent can either 'see' something from another agent or 'say' something to it. Less metaphorically, the first agent can read or send some information to the second agent. The type of the interaction depends on which aspect of the second agent is involved. In interactions of type *read*, the first agent reads the declarative aspect of the second; in interactions of type *send*, the first agent sends some information to the procedural aspect of the second. In either case, it is the procedural aspect of the first agent that carries out the interaction.

To enable interactions of type *read*, the declarative aspect of each DUAL agent is public — all agents in the architecture can potentially read it. The agent does not 'notice' whether, when, and who reads its declarative aspect. Therefore, the interaction of type *read* is an extremely weak form of interaction. In fact, one of the participants does not even notice its involvement in it at all.

---

[3] To make the story even more complicated, many of the symbols manipulated by DUAL agents stand for other DUAL agents (which are in turn symbols themselves). That is, there are symbols which stand for other symbols.

To enable interactions of type *send*, each DUAL agent has a place where other agents can put information. This place is called *input zone* and is considered part of the procedural aspect of the DUAL agent (see figure 3.2.2.1). Each agent in the architecture can potentially send information to everyone else. The receiving agent does 'notice' whether and when something is put into its input zone but cannot trace its origin. Thus, this second type of interaction in DUAL is also relatively weak, although less weak than the first type.

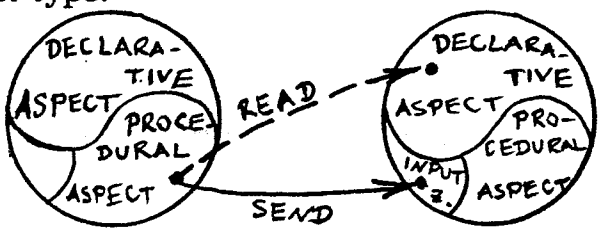


Figure 3.2.2.1. Schematic outline of the two types of DUAL interactions. The left micro-agent *reads* the declarative aspect of the right micro-agent and *sends* information to its *input zone*.

The current specification of the architecture does not provide for any other types of interactions. In particular, DUAL agents do not communicate via some elaborate protocol that involves queries and answers, acknowledgment of receipt of information, recovery from transmission failures, access regulation and the like.

3.2.2.2. The hybrid nature of DUAL interactions

DUAL interactions are hybrid just as DUAL agents are. Hence, the whole architecture is hybrid as everything ultimately boils down to DUAL agents and their interactions. DUAL is hybrid at the microlevel and this hybridity propagates to the upper levels.

The declarative/procedural dualism is represented by the interactions of type *read* and *send* respectively. A DUAL agent can read the declarative aspect of another agent and/or send something to its procedural aspect. Moreover, the two types of interaction are integrated: the black-and-white distinction between *read* and *send* is made here chiefly for descriptive purposes. In practice, DUAL agents interact through a judicious mix of both.

The symbolic/connectionist dualism of DUAL agents is straightforwardly extended to their interactions as shown 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.)

3.2.2.3. Exchange of symbols

In this section we will describe in more detail the most complex DUAL interaction — the symbolic interaction of type *send*. Other types are easier to understand once this hardest type is mastered.

The elementary act of symbolic exchange is called a *transaction*. It always involves two agents: a *sender* and a *receiver*. The sender takes the active role in the transaction: it determines what to be sent to whom. It has an explicit reference to the receiver; it 'knows' its 'name'. The receiver simply receives; it cannot trace who has sent the incoming symbol, nor can it refuse to accept it. (It can, however, choose to discard a symbol immediately after receiving it.)

In other words, transactions between DUAL agents are directional. The information flows from the sender to the receiver. To achieve a flow in the opposite direction, a second transaction with switched roles is needed. In any given transaction, however, each agent participates either as a sender or as a receiver.

A DUAL agent can participate in several transactions simultaneously. For example, some agent can receive symbols from agents A, B, and C and send other symbols to agents A, D, and E. However, it controls only those transactions in which it acts as a sender. In the example above, it can choose to send the symbol X to A and the symbol Y to D but it cannot even know who is the sender of the symbols it receives.

The specification of the architecture imposes no upper limit on the number of simultaneous transactions for an individual DUAL agent. However, this number must never approach the total number of agents in the system. As a consequence, almost all information-processing activities in DUAL are done locally. Complex tasks are carried out in small steps and ultimately reduce to simple operations performed by individual agents and simple transactions between pairs of agents. There is no central authority that can control the entire system. Instead, the macroscopic behavior of DUAL-based models *emerges* from a multitude of coordinated activities at the microlevel.

Transactions are elementary acts of exchange of symbols at a particular moment in time. They take very little time to complete. In contrast,

interactions last much longer. Usually, the same two agents engage in successive transactions over and over again. In this case, we say that they engage in an interaction.

3.2.2.4. Exchange of activation

DUAL agents also engage in connectionist interactions of type *send*. That is, they exchange activation. This exchange is done continuously (and not in discrete steps). Therefore, we do not speak of *transactions* but only of *interactions* in the connectionist context. The principles outlined in the previous subsection still apply. In particular, the exchange of activation is directional, there is a sender and a receiver, the sender takes the active role and so forth. A DUAL agent can send activation to several other agents simultaneously. Moreover, it can send different amounts of activation to the different receivers depending on the *weight* of the interaction.

To summarize, interactions of type *send* are truly hybrid — they have two aspects and these aspects obey the same overall organization. Each interaction involves a sender and a receiver, and boils down to transmission of information from the former to the latter. Activation flows continuously, while symbols are sent in discrete transactions. One micro-agent can engage in multiple interactions either as a sender or as a receiver.

3.2.2.5. The rest of interactions

Interactions of type *read* are simple — a DUAL agent can read the declarative part (both symbolic and connectionist) of another agent. The first agent actively reads, the second does not even notice that something is happening. The only prerequisite for an interaction of this type is that the former has a reference to the latter.

Each DUAL agent can potentially have a reference to any other agent in the architecture. At any particular instant in time, however, a DUAL agent can have only a limited (and relatively small) number of such references.

Permanent and temporary interactions: A large part of the interactions in the architecture are permanent. In other words, the same two DUAL agents are involved in the same interaction all the time. Permanent interactions contribute to the stability of the architecture — they bind micro-agents into coalitions and formations that persist. On the other hand, there are also temporary interactions. They contribute to the dynamics of the architecture. New coalitions emerge on the fly to meet some particular purpose. These new coalitions are formed in large part by means of temporary interactions. Thus, the organization of a DUAL-based system is stable without being static.

3.2.2.6. Node-and-link terminology

Due to their importance, interactions are often reified and called *links*. That is, they are treated as entities that exist in their own right. This is an alternative terminology which is sometimes more convenient than the interactive one. 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 using circles and lines, etc. For instance, the phrase 'a population of interacting DUAL agents' translates into 'a network of interconnected nodes.' Throughout this report, both phrases mean the same thing. We should always keep in mind, however, that the node-and-link terminology is conventional[4].

Links in DUAL are used to transmit activation and symbols. Links are directed — they have beginnings (senders) and ends (receivers). Links also represent the ability of the first agent to read the declarative aspect of the second one. An arbitrary number of links can come in and out any given node (DUAL agent). Thus, agents are connected in networks like the one illustrated in figure 3.2.2.2.

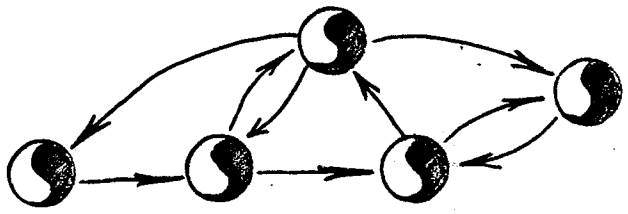


Figure 3.2.2.2. A simple network of interconnected DUAL agents. Nodes (agents) are shown as circles, and links (interactions) are shown as lines. Arrows indicate the direction of the links.

Links are hybrid — they have two aspects. The connectionist aspect of a link is called a *weight*; the symbolic aspect of a link is called a *label*. All links in DUAL have both aspects at once. Each aspect, however, is visible only to the corresponding aspect of the DUAL agents. The weight of a link is a real number and is used in the process of spreading activation. The label of a link is a symbol and is used by the symbolic aspects of the agents for various purposes. Labeled links play an important role in the symbolic representation scheme used in DUAL.

3.2.3. Symbolic Representation Scheme

The question of knowledge representation is central to the design of any cognitive system. In DUAL, it is addressed in a hybrid way and at dif-

[4] To give the word to Drew McDermott (1981): "In lucid moments, [semantic] network hackers realize that lines drawn between nodes stand for pointers [and] that almost everything in an AI program is a pointer. [...] Their lucid moments are few."

ferent levels. This subsection deals with the symbolic aspect of knowledge representation at the microlevel.

3.2.3.1. General framework

DUAL agents represent things — objects, concepts, events, situations, images, programs for action, etc. The architecture provides for at least three types of descriptions: modal-specific, motor, and conceptual. **Modal-specific** descriptions are specific for some sensory modality: visual, auditory, tactile, etc. **Motor** descriptions contain programs that control the effectors and execute actions in the environment. **Conceptual** descriptions deal with concepts, instances, relations, events, etc. For the third type of descriptions, DUAL employs a frame-like representation scheme. The other two types of descriptions are left unspecified for the time being. That is, we take into consideration the presence of such DUAL agents in the architecture and we refer to them if necessary in our theoretical discussions. However, we shall not consider their internal structure. It may be frame-like as well or have some kind of analogical representation. In this report, we refer to conceptual descriptions unless explicitly stated otherwise.

A *frame* (Minsky, 1975) is a collection of declarative and procedural knowledge describing a conceptual unit such as a concept, object, relation, action, event, situation, etc. If we concentrate on the declarative aspect of a frame, we can think of it as a complex node with many links that come in or out of it to connect it to other nodes. In fact, declarative knowledge is represented by these links, the node being merely a focus which can be referred to and where links cross. The links are implemented by *slots* and *facets* in the frame. In addition to these declarative structures, there may be *procedures* attached to the frame (or some of its slots).

The notion of frames appears in DUAL at two levels. Thus, we speak of *microlevel* and *mesolevel frames* (or, for short, *micro-* and *meso-frames*). The former are nodes which can be referred to and where links cross; the latter are whole networks of tightly interconnected nodes. The substrate of micro-frames in DUAL are the micro-agents while meso-frames are implemented by coalitions (a.k.a. meso-agents). In this subsection, we will discuss microlevel frames, postponing treatment of the mesolevel ones until section 3.3.3.

3.2.3.2. Frames, slots, and facets

Each DUAL agent is a microlevel frame. More precisely, it is the symbolic, representational aspect of a DUAL agent that is a microlevel frame (cf. table 4.1). It has *slots* which in turn may have *facets*. This forms a hierarchical structure that can be parsed and manipulated by symbolic routines. Slots and facets are placeholders — they may be (and usually are) filled up with *fillers*. Many fillers are references to other micro-frames

and thus link the given DUAL agent to its peers. Figure 3.2.3.1. gives an example.

```
color-of:
  :type      :concept
  :subc      physical-relation
  :slot1
    :type    :aspect
    :c-coref object
  :slot2
    :type    :aspect
    :c-coref color
```

Figure 3.2.3.1. An example of a micro-frame. It represents a particular concept — the relation `color-of`. The micro-frame has four slots. The first two of them describe the concept as a whole: it is an instance of the concept `physical-relation`. There are also other slots that describe the operands of the relation. `Physical-relation`, `object`, and `color` are references to separate micro-frames. Compare with figure 3.3.3.1.

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 does not depend on the particular micro-frame owning the slot. There are different kinds of general slots depending on their *labels*. The set of all possible slot labels is limited, specified in advance, and recognized by the symbolic machinery in DUAL. It includes labels like `type`, `subclass`, and `instance-of`. The architecture defines a set of slot labels that are used in all DUAL-based models. This set is the basic *vocabulary* of knowledge representation in DUAL. Each particular model may add model-specific slot labels to this basic vocabulary.

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. They are not recognized by the symbolic machinery in the architecture. By convention, frame-specific slots typically are labeled `slot1`, `slot2`, `slot3`, and so on. Each DUAL agent may have zero, one, or more such slots.

The specification of the architecture in its present state makes no commitment on the maximum number of S-slots that may be possessed by a DUAL agent. It is clear, however, that this number should be kept within reasonable limits (e.g. about seven). On the other hand, the number of G-slots is limited too as it cannot exceed the size of the vocabulary. As a con-

sequence, the total number of slots (both general and frame-specific) that may be possessed by a DUAL agent is limited.

Frame-specific slots (and only they) have *facets*. Facets can be conceived of as slots within slots. More precisely, they are like G-slots within S-slots. Facets have labels taken from the same vocabulary. That is, the same set of labels applies to both G-slots and facets. To illustrate, a micro-frame may have a G-slot labeled `type` and at the same time have a S-slot (labeled `slot1`) which has a facet labeled `type`.

3.2.3.3. Slot filler types

There are stringent restrictions on the types of objects that can serve as slot- or facet fillers in DUAL. This imposes discipline in the knowledge representation scheme used in the architecture.

Tags are simply identifiers such as `:concept`, `:instance`, and `:situation`. The vocabulary of possible tags is limited and is specified in advance on a per-slot basis. For instance, the filler of the `:type` slot of the DUAL agent illustrated at figure 3.2.3.1 is a tag.

References provide a means of referring to other micro-frames and their slots. Each micro-agent in DUAL has a *name*. Agent names are arbitrary symbols that serve to identify the agents. Agent names are unique — two different DUAL agents cannot share the same name. Therefore, a name unambiguously identifies its owner and can serve as a reference to it. There are two types of references: to a micro-frame or to some of its slots. The latter are expressed by appending the name of the slot to the name of the micro-frame, e.g. `color-of.slot1`.

```
color-of-17:
  :inst-of   color-of
  :slot1
    :inst-of color-of.slot2
  :slot2
    :inst-of color-of.slot1
```

Figure 3.2.3.2. Slot fillers that are references to other micro-frames or their slots.

Lists are heterogeneous ordered collections of elementary fillers. Using lists, a single slot may have multiple fillers. For instance, `color-of` may be a subclass of both `physical-relation` and `binary-relation`. This is represented by a `:subc` slot whose filler is a list of two references.

INTEGERS - - -

3.2.3.4. Vocabulary of slot labels

Each micro-frame describes some entity — a concept, object, event, etc. General slots describe the entity as a whole while frame-specific slots describe its parts. Each G-slot contains a particular piece of knowledge about the entity. G-slots with different labels specify different aspects. Therefore, the set of possible slot labels define the *vocabulary* of knowledge representation in DUAL. The labels from this vocabulary are recognized by the symbolic machinery in the architecture. To illustrate, a symbolic routine may examine the `type` slot of a given micro-frame and undertake different actions depending on its filler.

Any slot whose label does not belong to the vocabulary is treated as a specific slot. Such slots correspond to parts[5] of the entity being described by the micro-agent. Each facet of the S-slot contains a particular piece of knowledge about the part under consideration. Facets with different names specify different aspects of that part. Facet labels are taken from the same vocabulary as G-slot labels.

The specification of DUAL defines a basic vocabulary that is used by all models built on top of the architecture. It contains, at the time being, the following slot/facet labels: `type`, `subc`, `superc`, `inst-of`, `instance`, `c-coref`, `m-coref`, `a-link`, `t-link`, and `procedure`. Each one of them defines a specific kind of G-slot or facet. Each kind of slot (resp. facet) serves a particular purpose in the micro-frame (S-slot) which is summarized in the table below.

[5] The term *part* here should be understood broadly. It is intended to cover concepts like *parts* of objects, *elements* of situations, *participants* in relations, and so forth.

Slot name	Filler type	Purpose
type	tag	Specifies the type of a frame or slot.
subc	(list of) reference	Connects a subclass to a superclass.
superc	(list of) reference	Connects a superclass to a subclass.
inst-of	(list of) reference	Connects an instance to a class.
instance	(list of) reference	Connects a class to an instance.
c-coref	(list of) reference	Connects to some other conceptual description of the same entity.
m-coref	(list of) reference	Connects descriptions in diff. modalities.
a-link	(list of) reference	Arbitrary association.
t-link	(list of) reference	Temporary association.
procedure	(list of) reference	Attached procedure

Table 3.3 Basic vocabulary of slot and facet labels in DUAL.

3.2.3.5. Interactions and links revisited

The table of slot kinds (table 3.3) reveals that most of the slots and facets in the micro-frames are filled up by references. These references are very important. They are the fabric that combine microlevel frames into macrolevel ones. More generally, they combine DUAL agents into coalitions and formations.

Interactions in DUAL, extensive and ubiquitous as they are, are very simple. An agent X reads from or sends something to another agent Y. In both cases, X must have an explicit reference to Y. Very often, X contains this reference as a filler of one of its slots or facets. As slot fillers seldom change, X tends to engage in transactions with Y over and over again. In other words, there is a prolonged interaction between the two agents.

Using the node-and-link terminology, we say in such cases that there is a *link* between the two nodes. So, the phrase: "there is a link from

agent X to agent Y" means that agent X has a slot (or facet) that is filled up by a reference to Y. The total number of slots in a given micro-agent is limited. Therefore, the number of links going out of the agent is limited too.

A micro-frame has a number of slots as the corresponding DUAL agent interacts with many other agents. These interactions depend on the label of the slot (facet) that contains the reference to the receiver. For example, `a-link` slots do not induce any symbolic interactions. In the node-and-link terminology, we say that the link has a label. The label of the link is identical with the label of the slot or facet that contains the reference. In this way, we can speak of different kinds of links rather than of different kinds of interactions. We can also draw diagrams that conveniently visualize the organization of micro-frames into meso-frames.

3.2.4. Connectionist Aspect of DUAL agents

The symbolic representation scheme discussed so far has significant expressive power. It enables DUAL models to represent concepts, objects, events, situations, plans, etc. These representations, however, are relatively static. (More precisely, they are static if we neglect the creation of new nodes and links.) Thus, they fail to capture the dynamics that is characteristic of human cognition.

To overcome this limitation, DUAL employs a dual representation scheme. Facts are represented symbolically, 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, knowledge representation and processing in the architecture become dynamic and context-sensitive.

This subsection outlines the connectionist aspect of individual DUAL agents. It will be described following the general PDP framework (Rumelhart et al., 1986).

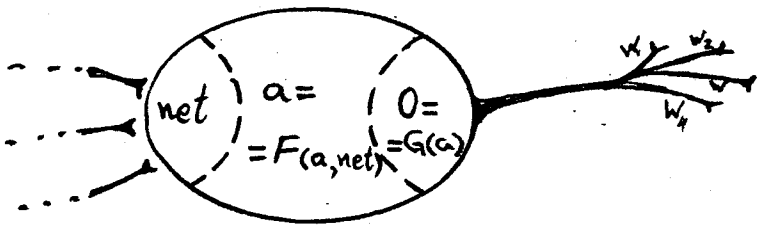


Figure 3.2.4.1. Schematic outline of the connectionist aspect of a DUAL agent (after Rumelhart et al., 1996). The *input zone* receives and accumulates the *net input* **`net(t)`**, the *activation function* **`F`** updates the *activation level* **`a(t)`**, the *output function* **`G`** transforms acti-

vation into *output* $o(t)$, which is sent farther via *weighted links*.

3.2.4.1. Activation, input, and output

Each DUAL agent is a node in a neural network. More precisely, it is the connectionist aspect of the agent that is such a node (cf. table 3.1). It continuously receives some activation from its neighbors, transforms this activation, and sends it farther. To do this, each DUAL agent is equipped with the connectionist machinery described below.

Activation level. Each DUAL agent i has an activation level $a_i(t)$ at any given moment. The activation level is a real number that varies and can take any value from the interval $[0;M)$. In other words, activation in DUAL is a non-negative continuous function of time (which is continuous too). The agent is said to be *inactive* when its activation level is zero. Inactive agents are also said to be *dormant* (Hofstadter & Mitchell, 1991).

Input zone. Each DUAL agent has an input zone which receives the incoming activation. When receiving activation, the agent is in passive position — it cannot refuse to receive the activation nor can it trace its origin. The incoming activation is summed algebraically into the *net input* $net_i(t)$. This input embodies the connectionist influence of other micro-agents. It is a real number that can take any value. Some DUAL-based models may employ agents with two separate input zones: *excitatory* and *inhibitory*. The former accumulates excitatory net input $enet_i(t)$; the latter — inhibitory net input $inet_i(t)$.

Activation function. The activation level changes continuously under the influence of the net input. The law that governs this change is specified by the activation function.

Output function. While the activation function regulates the internal state (activation) of the agent, its external output is regulated by the output function $G_i(a_i(t))$. From a connectionist perspective, the output function determines the amount of activation that the DUAL agent sends to its peers. The output function transforms the activation level $a_i(t)$ into an *output* $o_i(t) = G_i(a_i(t))$.

Different DUAL agents within a single model may have different activation and output functions. Usually, all agents of a given kind share the same functions, though the specification of the architecture does not insist on that. The exact nature of the functions is part of the specification of the concrete DUAL-based models.

3.2.4.2. Weighted links

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. When the interaction between two agents is such that the output o_j of the sender **j** increases the net input net_i of the receiver **i**, we say that there is an *excitatory link* from **j** to **i**. Conversely, when o_j decreases net_i, we speak of an *inhibitory link*. (When the receiver has two input zones, excitatory links increase $enet_i$ and inhibitory links increase $inet_i$.)

The output $o_j(t)$ of the sender is distributed unevenly among the receivers. Each one of them gets a portion of the output proportional to the *weight* of the corresponding link. The weight w_{ij} is a real number in the interval **[-1;+1]** which indicates what portion of the total output of the sender **j** will be allocated to the particular receiver **i**. To illustrate, if $w_{ij} = 2w_{kj}$ then the receiver **i** will get two times more output from **j** than the receiver **k**. Negative values indicate inhibition with the same magnitude.

Weight normalization. The numbers attached to the links are *raw weights*. They are normalized to produce the *normalized weights* that control the spread of activation. Normalization is a linear transformation of the weights so that the sum of the absolute values of normalized weights equals one. Due to the normalization, there is implicit competition between the nodes receiving activation from a given sender — the more they are, the less output is allotted to each of them.

Links and references. As it was stated in subsection 3.2.3.5., the phrase "there is a link from agent X to agent Y" means that agent X has a slot (or facet) that is filled up by a reference to Y. In other words, links between DUAL agents are actually references contained in their slots. Each reference is not only a symbol but also has a connectionist aspect, namely a number between **-1** and **+1** attached to it. This number is the raw weight of the link. Figure 3.4.2. illustrates:

```
electrical-appliance:
  :type       :concept
  :subc       (artefact 1.0)
  :superc     ((plate 0.5) (fridge 0.4) (lamp 0.4))
  :a-link     ((electricity 0.5) (instrument 0.2))
```

Figure 3.2.4.2. Example of references with different (raw) weights. Compare with figure 3.2.4.3.

This DUAL agent is connected to six other agents: `artefact`, `plate`, `fridge`, `lamp`, `electricity`, and `instrument`. The link between each pair of agents is actually a reference that fills some slot of `electrical-`

appliance. The symbolic aspect of a link is its label. It is the same as the label of the corresponding slot. The connectionist aspect of a link is its weight. It is attached to each reference. The symbolic machinery 'attends' only to the labels of the links; the connectionist one — to the weights.

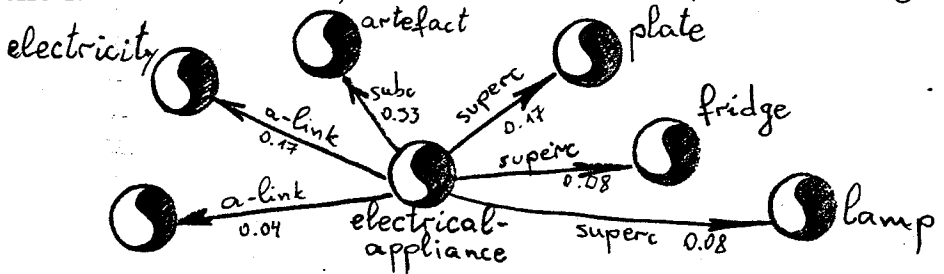


Figure 3.2.4.3. A DUAL agent that interacts with six of its peers. Each interaction (link) has a label and a weight. Compare with figure 3.2.4.2.

3.2.4.3. Availability, visibility, and speed

The connectionist aspect of DUAL agents influences the symbolic one by determining the agent's *availability* that will be discussed in this subsection.

The overall behavior of a DUAL-based system emerges out of the collective activities of a large number of individual DUAL agents. Each one of them contributes to the gross product in different degrees depending on their *availability*. The notion of availability contributes very much to the hybrid nature of DUAL agents — is merges all four aspects summarized in table 3.1. The connectionist aspect computes the availability, which is then used as power supply to the symbolic aspect. Moreover, availability, like the agent itself, has declarative and procedural aspects. The former is called *visibility*, the latter — *speed*.

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.

The visibility of a DUAL agent is measured by its activation level **a(t)**. Therefore, visibility is a non-negative real number that changes over time. Agents with **a(t)=0** are invisible to any symbolic processing that takes place at the moment. In order to be processed by the symbolic routines, the agent must first be activated by the connectionist mechanism. When its activation level **a(t)** gets high enough to pass the threshold imposed by the activation function **F**, the agent will become visible (**a(t)>0**). For example, if the micro-frame that represents the concept *color* is inactive, it

is not visible and hence will not be included in any symbolic structures that are being built. An external observer can interpret that the system ignores colors because considers them irrelevant at the moment. If, however, the micro agent `color` participates in a coalition with some other agents that are active, it may receive support from them and become visible. When this happens, it gets the chance to be heeded by the procedural machinery. Moreover, this chance increases when the activation (and hence visibility) of the DUAL agent increases. For instance, if a choice has to be made between two micro-agents, the more visible one will be preferred.

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.

The exact mechanism for incorporating speed and visibility into the symbolic machinery in the architecture is described later in this section. At the moment, we turn to the description of the symbolic processing performed by individual DUAL agents.

3.2.5. Symbolic Processing

A great deal of the computation in the architecture is *symbolic processing* — creation, interchange, and modification of symbolic structures. Although symbolic processing is in many respects similar to the connectionist processing performed by DUAL agents, there are important differences. The similarities are that both symbolic and connectionist processing involve receipt of information in the input zone, transformation of this information, and finally its redistribution. The differences are in the details but are nevertheless very important. First of all, symbolic processing is discrete while connectionist one is continuous. Second, symbolic processing is more complex and has greater diversity. This additional complexity is evident from figure 3.2.5.1. which outlines the symbolic aspect of a DUAL agent.

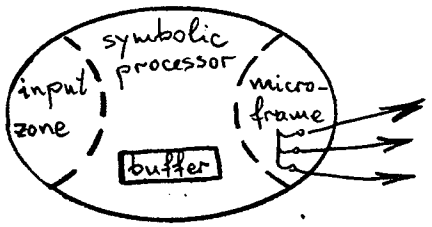


Figure 3.2.5.1. Schematic outline of the symbolic aspect of a DUAL agent. The *micro-frame* contains declarative knowledge. The *input zone* receives symbols from other DUAL agents. They are processed by a *symbolic processor* and may be stored in the *buffer* and/or sent via *labeled links* that are described in the micro-frame. Compare with figure 3.2.4.1.

3.2.5.1. Local memory

Each DUAL agent has *local memory* in which it stores symbolic information. Part of the local memory is *permanent*; the rest is *volatile memory*. When the agent loses its activation and goes to dormancy, permanent memory is retained and volatile memory is not. Thus, when the agent is reactivated later on, volatile memory is empty while the content of permanent memory is intact.

The **permanent memory** keeps the symbolic declarative aspect of the DUAL agent. That is, it stores the micro-frame with its slots, facets, and fillers. Given the importance of the frame-like symbolic representation scheme, it turns out that most of the declarative knowledge in the architecture is stored in these memories. They are the substrate of the long-term memory of DUAL-based systems.

Most slot fillers in the micro-frame are references to other DUAL agents. Therefore, in the node-and-link terminology it could be stated that the permanent memory keeps the links connecting the DUAL agent to its peers. Links are stored together with their labels and (raw) weights.

Permanent memory is available for inspection from outside the micro-agent through interactions of type *read*. This, however, is possible only when the agent is *visible*. Visibility is computed by the connectionist aspect and depends on the activation level of the agent.

In contrast to the permanent memory, **volatile memory** is wiped out when the agent's activation level drops below a certain threshold. Part of the volatile memory is the *input zone*. It is used in interactions with other DUAL agents and more specifically in interactions of type *send*. The input zone is the place where other agents can put symbols and symbolic structures. These symbols are then processed by the local symbolic processor.

The rest of the volatile memory — the *buffer* — is for agent's private use. It is inaccessible from outside the micro-agent. The symbolic processor uses it to store intermediate results of its symbolic operation and to keep track of the markers that have passed through the micro-agent. The buffer may also contain temporary links to other DUAL agents (namely `t-link` slots filled with references). These links are not used by the symbolic

machinery of the agent; they participate only in the process of spreading activation.

It is important to stress that the size of the whole local memory is limited. That is, the total number of symbols that can be stored in it is limited. The declarative memory holds only the micro-frame and thus does not exceed the maximum size of DUAL micro-frames (see subsection 3.2.3.2.). The input zone and the buffer are also postulated to be limited. The specification of the architecture in its present state makes no commitment about the actual limits nor specifies what happens when the limit is exceeded. The latter situation, however, should hardly ever occur in well-tuned DUAL models.

To sum up, the permanent memory holds the micro-frame of the agent and is available from outside through interactions of type *read*, the input zone supports interactions of type *send*, and the buffer is for private use. The latter two kinds of memory are volatile — they are wiped out when the agent's activation falls below a threshold. All aforementioned types of memory comprise the local memory of the DUAL agent. The local memory of each agent is limited in size. The symbolic processor can access all kinds of memories.

3.2.5.2. Operations, steps, and processes

A great deal of the information processing in the architecture is *symbol manipulation* — deterministic construction, transformation, storage, and interchange of symbolic structures. We use the general term *symbolic processing* to refer to these activities. They are distributed over the population of DUAL agents and are carried out by their *symbolic processors*.

Symbolic processing is discrete and can be categorized into the following segments of increasing complexity:

Symbolic operation. This is the smallest act of symbol manipulation in the architecture. Symbolic operations are simple, atomic, and deterministic. They may be conceived as elementary instructions of the symbolic processor of the corresponding DUAL agent. To illustrate, the act of decomposing the compound reference `frame235.slot2` is a symbolic operation. Another operation would be to check whether two symbols are identical, to retrieve the filler of a slot, etc.

Symbolic operations constitute the finest grain of symbolic processing. Bigger processes consist of sequences of operations. The repertoire of possible operations determines the overall reach of the symbolic processing in the architecture. A long-term goal of DUAL research is to identify a basic set of symbolic operations that are convenient to support the behavior simulated by DUAL models. An additional, even more distant goal will be to implement the basic symbolic operations with subsymbolic means. At pres-

ent, however, we do not try to explicate the exact set of operations or their subsymbolic implementation. Instead, we take them for granted and use them as building blocks.

The work of a symbolic processor is conveniently visualized in *time diagrams* like those in figure 3.2.5.2. In such diagrams, there is a line that represents the passage of time. Thin segments on this line indicate idle periods, that is, time intervals in which the processor does not execute any operation. Thick segments on the time line indicate busy periods. The dots within thick segments mark transitions between individual symbolic operations. Arrows indicate exchange of symbols with other DUAL agents. Reading and writing in the local memory is not shown.

Figure 3.2.5.2. Time diagram illustrating a typical case of symbol manipulation. The symbolic processor is idle until a symbol arrives into the input zone. This triggers a sequence of three symbolic operations that result in the production and emission of another symbol.

Symbolic step. Symbolic operations lump together into symbolic steps. These are sequences of operations performed by a single DUAL agent without intervening symbolic interactions with other DUAL agents. (Charles Hoare (1985) uses the term *disjoint process*.) Symbolic steps are the smallest units with respect to the interactions between different DUAL agents working in parallel. A symbolic step may begin, for example, with receipt of a symbol from outside the micro-agent and end with emission of another symbol. The important thing is that by definition there is no interchange *during* the symbolic step. Therefore, an external observer can assume that symbolic steps are atomic and ignore the elementary operations that constitute them. In time diagrams, symbolic steps are shown as thick (busy) segments that can have arrows only at their endings:

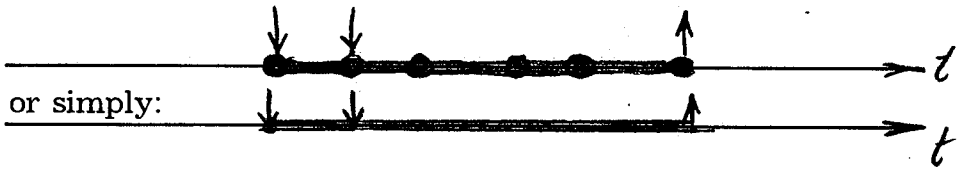


Figure 3.2.5.3. Alternative time diagrams of two consecutive symbolic steps. In the second diagram the dots delineating individual operations are omitted for simplicity.

Rigid symbolic process. This is a fixed and *a priory* specified sequence of symbolic steps. The specification of a rigid process is called a *symbolic routine*. Rigid processes are usually executed by a single DUAL agent but may also be distributed across a (tight) coalition of agents. The important

thing that makes the process rigid is the existence of an explicit routine that specifies what operations are executed, upon what conditions, when, and by which micro-agent(s). This makes the rigid process predictable and reliable. It could also be tuned for efficiency. Therefore, rigid processes are important ingredients of the overall information processing in DUAL. Important ingredients, but not the whole story: rigid processing alone cannot satisfy the demands on plausible cognitive behavior in a dynamic environment (Kokinov et al., 1996).

In time diagrams, when several processors work in parallel, there are several parallel lines in the diagram. Interactions are shown as arrows connecting two lines.

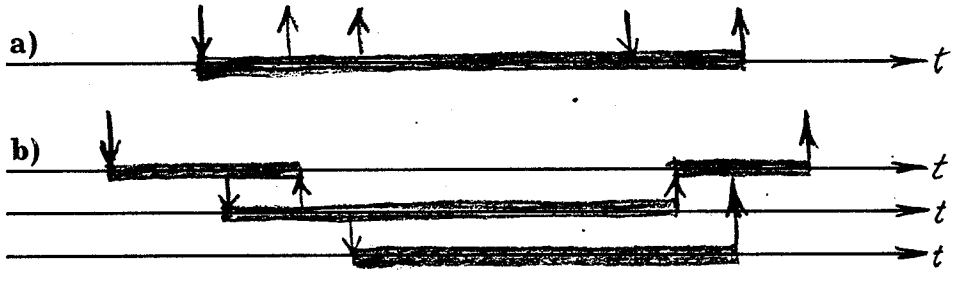


Figure 3.2.5.4. Rigid symbolic processes performed by one DUAL agent (a) or by a coalition of three agents (b).

Emergent symbolic process. This is the most complex type of symbolic processing in the architecture. The emergent symbolic process does not (and *cannot*) have any complete *a priory* specification. It is distributed over a number of interacting DUAL agents and the exact course of the computation is determined dynamically by the interplay of various factors (or *pressures* (Hofstadter, 1984)). Since these factors are numerous and their influence is intricate, their net result cannot be described or anticipated *a priory*. Thus, although the whole processing in DUAL is deterministic, the exact course of an emergent process is unpredictable. The boundary between rigid and emergent processing is fuzzy. For instance, it is difficult to tell them apart on the basis of time diagrams. As a general rule, however, the more a process is specified beforehand, the more rigid it is and vice versa.

Emergent symbolic processes by their very nature belong to the level of coalitions (the meso-level). Therefore, we postpone their discussion until section 3.3.4 and concentrate on rigid processing below.

3.2.5.3. Speed of symbolic processing

> *Activation controls the rate of information processing. It is the 'energy' that runs the 'cognitive machinery'.*
> (Anderson 1983, p.86)

DUAL symbolic processors run at different speeds depending on the activation level of the DUAL agent. Very active agents run rapidly and thus determine system's overall flow of computation, low-active agents run slowly, and inactive agents do not run at all. As the activation level of each DUAL agent changes, the speed of its symbolic processor changes accordingly. This is a key factor to the *computational dynamics* that is a characteristic feature of DUAL (Kokinov et al., 1996).

The exact mechanism that governs processing speed as a function of the activation level is based on the following **energetic analogy**. Each symbolic operation requires that the symbolic processor does a 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. The same amount of work may be done rapidly if there is enough power, slowly if power is scarce, or not at all if power is lacking completely.

Most of these concepts have a counterpart in DUAL as the following table demonstrates:

energy domain	**DUAL domain**
work	symbolic operation
amount of work **A**	consumption **C** of an operation
consumer	symbolic processor
generator (power supply)	connectionist aspect of the agent
power **P(t)**	activation **a(t)**
energy **E=** $\int$**P(t)dt**	accumulated activation $\int$**a(t)dt**
efficiency coefficient η	efficiency coefficient η

The connectionist aspect of a DUAL agent serves as a power supply to the symbolic processor. The amount of energy produced by the connectionist aspect is given by the integral $\int_{t_0}^{t_1} a(\tau)d\tau$. , where $a(\tau)$ is the activation level of the agent, t_0 is some fixed initial moment and $t>t_0$. This integral defines the *energy function* $E(t)$. When $a(\tau)$ is positive in the interval (t_0,t), $E(t)$ is an increasing function of t and thus has an inverse: E^{-1}. The inverse function expresses the time needed for production of a given amount of energy: $t-t_0 = E^{-1}(E)$.

The symbolic processor consumes energy in order to perform symbolic operations. In other words, the symbolic processor can be regarded as a machine that transforms connectionist energy into symbolic work. Not all

energy, however, is converted into useful work. There are some losses that cover the internal needs of the processor itself. The *efficiency coefficient* η is defined as the ratio of the useful work to the total energy input: $\eta = A/E$. The efficiency coefficient ranges from **0** to **1** and is a characteristic of the symbolic processor. The processors of different DUAL agents have different efficiencies. Those processors that perform highly automated tasks have η close to **1**. In contrast, processors that perform novel tasks have low efficiency.

With these definitions at hand, we are able to calculate the time needed to perform a symbolic operation. Each symbolic operation has a parameter associated with it — its *consumption* **C**. This specifies how much connectionist energy is equivalent to the amount of symbolic work embedded in the operation. Each DUAL model specifies the consumption of each kind of symbolic operations. These are parameters of the model. Another set of parameters specifies the efficiency coefficient of each (type of) symbolic processor. (The latter set of parameters can be adjusted through some sort of learning — the basic rule is that efficiency increases with practice.)

Now, in order to perform an operation with consumption **C**, a symbolic processor with efficiency coefficient η needs a total input of $E = C/\eta$ units of connectionist energy. This energy must be produced by the connectionist aspect. This, however, takes time because the power of the latter is limited. The activation $a(\tau)$ of the connectionist aspect is integrated over time: $E(t) = \int_{t_0}^{t} a(\tau)d\tau$. When enough energy is produced by the connectionist aspect and then transformed into symbolic work, the symbolic operation is completed. This happens at time $t = t_0 + E^{-1}(C/\eta)$.

Can the energetic analogy be legitimately applied to the cognitive architecture DUAL? Yes, if one takes into account two more points:

1. The specification of DUAL postulates that if the activation level of an agent drops below a threshold, its symbolic processing is terminated and all intermediate results stored in its buffer are lost. In the context of the present discussion, if $a(\tau)$ becomes zero even for a moment, the whole operation is aborted. When $a(\tau)$ stays above zero (i.e. above the threshold), the integral $E(t)$ is monotonically increasing and the inverse function E^{-1} is well defined.

2. The symbolic operations in DUAL are atomic. Therefore, the exact process that takes place during the execution of an operation is irrelevant. What matters is only the final outcome and the timing of its appearance.

Note also that when the symb. processor is idle, the energy produced by the conn. aspect goes unused and cannot be accumulated.

The quantitative law defined in this subsection meets the qualitative specifications of DUAL set forth by previous publications on the architecture (Kokinov 1994a, 1994b). Indeed, when a DUAL agent is very active,

there is plenty of power for the symbolic processor. Hence, many symbolic operations may be executed within small periods of time (see figure 3.2.5.5.a). In other words, active processors run rapidly. On the other hand, when the activation level is low, power is scarce and large time intervals are needed for the symbolic processor to complete the same sequence of operations (figure 3.2.5.5.b). Finally, inactive processors do not run at all.

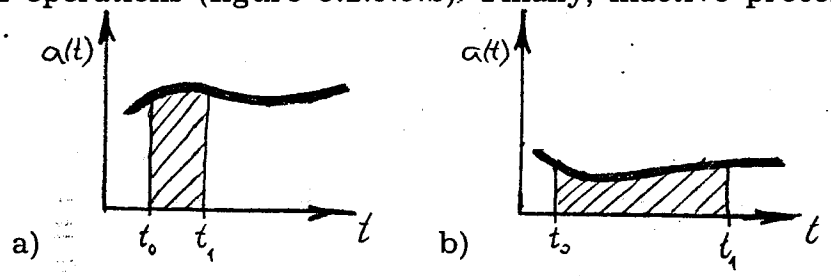


Figure 3.2.5.5. Illustration of the mechanism for determining processing speed on the basis of activation level **a(t)**. Time **t** varies along the X axis, activation **a(t)** — along the Y axis. Hatched areas represent energy or work. It is evident from the two diagrams that the same amount of work takes little time when **a(t)** is high (a) and much time when **a(t)** is low (b).

3.2.5.4. Asynchronous discrete operation

It is now possible to specify the operation of the symbolic procedural aspect of DUAL agents. Each symbolic processor has a hard-wired routine that controls its operation. The routine specifies what symbolic operations are performed, upon what conditions, and in what order. It may contain branching (e.g. `if` statements) and loops. From an external point of view, however, the routine unfolds into a linear sequence of symbolic operations. The *consumption* **C** of each operation is known — it depends on the type of the operation and on the visibility of its arguments. Moreover, **C** is determined at the start of the operation. On the other hand, the *efficiency coefficient* η of the symbolic processor is known too. Thus, the amount of connectionist energy **E** (=**C**/η) that is needed for successful completion of each operation is known prior to its execution.

The symbolic processing performed by each DUAL agent proceeds in a discrete way — operation by operation. The specification of the architecture postulates that the operations are atomic. That is, the exact process that takes place within the symbolic processor is irrelevant. What is relevant is the input/output relation of the operation and its overall duration. The latter is computed by the formula $t - t_0 = E^{-1}(C/\eta)$. The outcome of the operation takes effect exactly at the moment when enough activation is accumulated: $\int_{t_0}^{t} a(\tau)d\tau = C/\eta$.

These considerations are easily extended from individual operations to bigger sequences — steps and rigid symbolic processors. In fact, symbolic steps are at a more convenient level of granularity: they are bigger than operations and at the same time they may still be considered atomic. They are atomic in the important sense that, by definition, there are no interactions with other DUAL agents *during* the step. From that follows that the total consumption of the symbolic step can be determined prior to its execution — it simply is the total consumption of all comprising operations. (The visibility of the operands cannot change during the step because there are no interactions with other agents.)

Thus, as far as processing speed is concerned, there is no difference between operations and steps. Both are atomic and their consumption is known prior to their execution. Steps, however, have the advantage of being bigger chunks. Therefore, at this stage of our research, we concentrate on symbolic steps and postpone detailed consideration of the exact operations that carry them out. For instance, we say that the act of receiving a marker, storing it into the buffer, and sending a copy of it to another agent is one single symbolic step with consumption **c**. We do not consider the exact operations that carry it out, as long as the right marker appears at the right moment in the right place.

Symbolic steps are important as a unit of analysis because they shift the discussion from individual DUAL agents to coalitions of interacting DUAL agents. The very definition of the term 'symbolic step' refers to interactions with other agents. Each one of them works at its individual speed without synchronization with others. In other words, the agents run asynchronously and discretely.

The beginning and end of each step can happen at arbitrary instants in time. Each agent must be prepared to receive a symbol at any moment and cannot predict the exact occurrence of such symbols. Conversely, each agent is free to send a symbol to another agent without having to check whether the receiver is busy or not. (The sender does have to check whether the receiver is visible.) This considerably simplifies interactions in DUAL.

As the pattern of interactions is complex and each processor runs at individual variable speed, the overall process soon becomes unpredictable. This is true regardless of the strictly deterministic nature of DUAL agents and the architecture as a whole. This is a general fact: asynchronous parallelism is inseparable from processors' actions being random relative to one another (as pointed out by Hofstadter & Mitchell (1991), who in turn refer to Hewitt (1985)). This asynchronous parallelism is a crucial factor for the *dynamic emergent computation* in DUAL (Kokinov et al., 1996).

One of the consequences of this implicit randomness is that *exact* coincidences in the architecture are exceedingly rare. More precisely, the

probability of such coincidence is infinitesimally small (because the underlying variables are continuous). In other words, two events virtually never happen at exactly the same moment. Similarly, two DUAL agents virtually never have exactly the same activation levels. This relieves DUAL's specification of the burden of handling cases such as: "What happens when two markers come simultaneously; which one is processed first?" The specification of the architecture postulates that information is processed on a first-in first-out basis. Ties are (almost) impossible in theory, very rare in practice, and may be resolved arbitrarily.

3.2.5.5. Data visibility

Agent *availability* has two aspects — procedural *speed* and declarative *visibility*. Both depend on the activation level of the agent (subsection 3.2.4.3.).

When the activation level is below the threshold, the agent is invisible to the symbolic machinery in the architecture. It is impossible to distinguish by symbolic means whether a DUAL agent is invisible or does not exist at all. In particular, if some (active) agent tries to establish a transaction with an invisible agent, it will fail. In order for the transaction to be established, the second agent must first be activated.

The following scenario takes place in many cases: There is a highly active DUAL agent that tries to perform some action. The agent participates in a coalition of interconnected agents. Since the agent is active, its partners receive activation and become visible. This enables the symbolic processor of the first agent to gain access to the declarative knowledge stored in other agents of its coalition. The procedure activates its data. An alternative scenario is possible too — data activating their procedure (which leads to a data-driven mode of computation).

Data visibility, however, is not an all-or-nothing phenomenon. Once a DUAL agent passes the threshold (i.e. becomes visible), it may be visible to different degrees. Visibility is directly proportional to the activation level **a(t)**. Highly active agents not only are visible to other agents, they also tend to attract their 'attention'. For example, if a choice has to be made between two competing micro-agents, the more visible one will be preferred.

In addition, active data may speed up operations that work on them. To that end, the consumption **C** of the operation should depend not only on the kind of the operation itself, but also on the visibility of the operands. The specification of the architecture makes no commitment about the exact dependence — it is left to DUAL-based models.. This topic needs further experimentation. Two possible formulas are: $C = C_0/V$ and $C = C_0 - k*V$, where **C** is the consumption of the operation, C_0 is some predefined baseline consumption, **V** is the visibility of the operand, and **k** is a prede-

fined coefficient. Both formulas ensure that more visible operands are processed faster.

3.2.6. Temporary DUAL Agents and Links

Most cognitive tasks require temporary structures in memory — representations of the environment, intermediate results, hypotheses about future events, etc. Consequently, any cognitive architecture must provide a medium for building and maintaining temporary structures. In DUAL, this service is carried out by *temporary DUAL agents*. They are akin to permanent DUAL agents in all aspects but one: temporary agents disappear when their activation level falls below certain *lethal threshold*. (In contrast, permanent agents do not disappear; they simply become inactive and may be recuperated later.)

Likewise, there are *temporary links* or, more precisely, slots or facets labeled `t-link` (see subsection 3.2.3.4). They are similar to permanent associative links (`a-links`) except that `t-links` are held in the volatile memory and hence cannot survive dormant periods. Both kinds of links are ignored by the symbolic aspect of the architecture and are used for connectionist purposes only.

A permanent DUAL agent (P) can interact with a temporary one (T) *only* via a temporary link. Other agent combinations (P-P, T-P, and T-T) can employ any kind of link.

3.2.6.1. Construction of temporary agents and links

Temporary agents are created by specialized DUAL agents called *node constructors*. They are equipped with built-in routines for constructing a brand new temporary DUAL agent upon request. Each DUAL-based system has a limited number of such node constructors and other agents compete for them. In other words, node constructors are a centralized scarce resource. They are the only institution in the architecture that can create new DUAL agents. Each new agent is given unique name that will serve as a reference to it[6].

Node constructors are recruited by other DUAL agents for the needs of some computation. While a node constructor is working on some task, it is unavailable for other requests. Thus, it is possible that all node constructors in the system are busy. On such occasions, symbolic processes that need node construction are momentarily (or sometimes permanently) suspended. When a node constructor has completed its job, it is released and becomes available again.

---

[6] This uniqueness does not imply centralized administration of names. Each node constructor can maintain an internal counter or some other means for generating unique names without having to check for collisions with existing ones.

Temporary links are constructed by the general mechanism for creating new slots. Namely, each DUAL agent is able to add a slot (or facet) to its local micro-frame. The only difference with `t-link` slots is that they are stored in the buffer. In both cases it is the local symbolic processor that creates the slot by executing an appropriate sequence of symbolic operations. Non-local slots are harder to establish: if an agent wants to create such slot, it must send a request to the prospective owner of the slot.

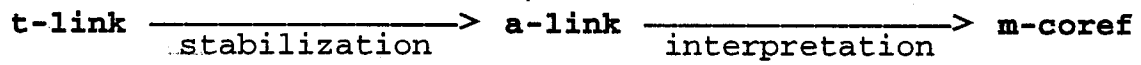
3.2.6.2. Destruction of temporary agents and links

The 'life span' of temporary agents and links depends on the connectionist mechanism. Temporary structures live as long as the activation level of the agent is maintained high enough.

Whenever the activation level of a temporary agent falls below the *lethal threshold*, the agent disappears from the system together with all its slots, the contents of its buffer, etc. All references to the agent become invalid. When the connectionist aspect of some other agent tries to follow a 'dead' reference, the absence of the referent causes that reference to be removed. In this way, all references to the temporary agent are removed shortly after its elimination.

Temporary links are stored in volatile memory and are, therefore, purged when the host becomes inactive (see subsection 3.2.5.1).

The hopeless fate of temporary agents and links might be remedied in future versions of the architecture. The projected learning mechanisms in DUAL foresee promotion of temporary structures to permanent status. The intuition behind the architecture says that most of the permanent agents and links have emerged as temporary structures that have later stabilized. For example, one particular life history might be:



3.2.7. Relation to other theories

Neither ever quite the same, nor ever quite another.
Gerard de Nerval

DUAL agents share many features with other theoretical constructs used in cognitive modeling. They have obvious relationship with production rules, connectionist units, frames, etc. At the same time, however, DUAL agents differ from any of them. This subsection tries to contrast our proposal with some of the alternatives.

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

Table 3.4. Different aspects of Dual agents. (Replication of table 3.1.)

The most important thing about DUAL agents is their hybrid nature. They put together properties that are usually held in isolation (table 3.4). Thus, the typical relation between DUAL agents and other constructs is inclusion. DUAL agents have the essential characteristics of production rules plus something more. They have the essential characteristics of connectionist units plus something more, etc. In addition to this general inclusive relationship, however, there are some details in each particular case that merit mentioning in brief.

Semantic network nodes: DUAL agents are very similar to semantic network nodes (Quillian, 1969; Anderson & Bower, 1979). They are labeled representations of concepts, objects, relations, events, and so forth. In addition, they are connected by labeled links and these links are essential to the representational scheme. Finally, DUAL agents pass markers. On the other hand, DUAL agents have many features — notably their connectionist and procedural aspects — that are alien to semantic network nodes.

Connectionist units: DUAL agents are also very similar to the units (or *neurons*) used in connectionist models and in particular localist neural networks (Feldman & Ballard, 1982; Rumelhart & McClelland, 1986). They perform numerical computations and spread activation along weighted links. Moreover, their activation level is interpreted — in DUAL it represents context relevance. On the other hand, the same activation is used in a non-canonical fashion in the architecture: as power supply for symbolic processing. DUAL agents depart radically from the connectionist movement because they deal explicitly with symbols. Distributed representations and the emphasis on learning, which are characteristic of many neural networks, are not central to DUAL research, at least for the time being.

Production rules: Turning to the procedural aspect, DUAL agents have much in common with rules (or *productions*) used in production systems (Newell & Simon, 1972; Anderson, 1983, 1993). They perform small symbolic actions when certain conditions hold. Thus, they embody efficient, rigid procedural knowledge. In most production systems, however, the rules are separated from declarative data — the latter are posted to a *blackboard* where production rules add or remove clauses. In DUAL, the symbolic processor of each micro-agent has private memory (input zone, buffer, and a micro-frame). DUAL agents send messages directly to their peers via references. They run at variable speed depending on their acti-

vation level. Finally and most importantly, there is no global executive in the architecture deciding which agent to run, resolving conflicts, etc. Instead, collective behavior is based on *dynamic emergent computation.*

Summarizing the comparisons so far, DUAL agents integrate features of semantic network nodes, connectionist units, and production rules. All these features are present in each DUAL agent at once. It should be noted, however, that in different agents these features are represented to a different degree. More concretely, some micro-agents in the architecture serve mostly as nodes — their *raison d'être* is to staticly, declaratively represent something. Nonetheless, such agents take their part in the processes of spreading activation and marker passing. In contrast, other agents serve mostly as rules: these are the agents of `type: procedure`. Still others serve mostly as distributors of activation. Despite these variations in the emphasis, however, all aspects from table 3.4. can be found in any DUAL agent.

Frames: The integration of declarative and procedural knowledge makes DUAL agents similar to *frames* (Minsky, 1975), *schemas* (Rumelhart, 1975), and *scripts* (Shank & Abelson, 1977). DUAL agents, however, are much smaller than these complex structures. A full-blown script, for example, contains a wealth of information about the participants, event sequence, preconditions, place of occurrence, and so on. This amount of information is far beyond the reach of a single DUAL agent. It would be represented by a whole *coalition* of micro-agents. DUAL coalitions, being distributed and having strong connectionist flavor, are closest to *schemata* (Rumelhart et al., 1986). To sum up, it is better to say that individual DUAL agents are *micro-frames* (section 3.2.3.1) while coalitions are *meso-frames*.

Codelets: An interesting theoretical construct is proposed by Douglas Hofstadter (1984, 1995) and elaborated by his graduate students Melanie Mitchell (1993) and Robert French (1995). Their *codelets* are 'small pieces of code' waiting in a *coderack* for the chance to run. Each codelet has a numeric attribute called *urgency* influencing the likelihood that it will be chosen. Thus, codelets fire with probability proportional to their urgencies while DUAL agents run at speed proportional to their activation levels. Both mechanisms implement the same general idea — dynamic emergent computation (*parallel terraced scan* in Hofstadter's terms).

Having listed some theoretical constructs that are similar to DUAL agents, let us now turn to two other constructs which are *not* similar, despite the appearance:

Turing machine: DUAL agents have symbolic processors and local memories, but do not have the full power of Turing machines because their memories are limited (subsection 3.2.5.1). Moreover, the symbolic as-

pect of a DUAL agent is not closed in itself — a symbolic operation can be aborted unexpectedly if the activation level drops below the threshold.

Autonomous agents: The term *agent* is used heavily in the areas of multi-agent systems and social cognition (Castelfranchi & Werner, 1994; Gilbert & Doran, 1994). In such contexts, however, the term usually refers to entities that are much more complex and autonomous than DUAL agents. The agents in a multi-agent system can, for instance, pursue explicit goals, maintain a model of the environment, negotiate with their peers via an elaborate protocol, etc. This sophisticated behavior can be achieved only by a whole *system* of DUAL agents. Individual micro-agents are nothing more than cells in a bigger organism (section 3.1).

3.3. DUAL at the Mesolevel

This subsection describes DUAL at the mesolevel. At this intermediate level of granularity, the entity of main interest is the *coalition* — a 'team' of collaborating DUAL agents.

3.3.1. The Need for Coalitions

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. The members of a coalition exchange activation and symbolic information.

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). It is at the level of coalitions where these properties appear in the architecture for the first time. Having appeared at the meso-level, these properties propagate to upper levels and become characteristic of the DUAL approach to cognitive modeling as a whole.

Coalitions play a key role in DUAL-based models. They are at the right level of abstraction for many purposes. Individual agents are so small that become meaningful only within the context of a bigger structure. On the other hand, the complexity of formations and systems (see section 3.4) make them difficult to understand and to manage. It is, therefore, necessary to have a conceptual apparatus that is at intermediate level of granularity — the meso-level (see subsection 3.1.3).

The introduction of the meso-level and the notion of coalitions is judged to be one of the important contributions of the present thesis.

3.3.2. Types of Coalitions

The coalition is a set of agents and a pattern of interaction among them. There are permanent coalitions in which the pattern of interaction is stable and changes little over time. At the opposite end of the spectrum, there are ephemeral coalitions, often involving temporary agents, which fall apart after a while. Finally, there is a whole range between these two extremes.

It is important to note that 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.

There are also '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. Again, there is a whole range between these two extremes.

To follow the metaphor proposed by Douglas Hofstadter (1995), the agents in a coalition are like molecules in a fluid. Tight coalitions are like fluids with high viscosity — their molecules are much constrained by the presence of other molecules in vicinity. For instance, if one agent from a tight coalition is highly active then it is almost certain that all other members are active too. Such coalition acts as a unit, it is like a drop of honey that keeps its spherical shape regardless of the force of gravity. By contrast, loose coalitions have low viscosity — their members are little influenced by their peers and are thus free to move around and even to 'evaporate', to abandon the coalition.

It is important to stress that all coalitions in DUAL are in fluid phase. None of them are gaseous — an isolated DUAL agent will quickly lose all its activation (due to spontaneous decay) and will drop out of the working memory. None of them are solid — it is always possible that, e.g., a new agent is included into a coalition; it is never forbidden *a priory* to have a blend between any two coalitions, etc.

3.3.3. Coalitions as Representations

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* (cf. subsection 3.2.3.1).

From the symbolic perspective, a DUAL meso-frame is a coalition of agents (micro-frames) tightly coupled with one another by many labeled links. The links of type `:subc`, `:superc`, `:inst-of`, and `:c-coref` are especially important. In particular, slot fillers are *conceptual coreferences* (`:c-coref`'s) to other frames or their slots.

To take a relatively simple example, consider figure 3.3.3.1. It depicts the core of the coalition representing the fact that some teapot (dubbed `teapot-1`) is colored in some specific shade of green. As is evident from the figure, the exact boundaries of this coalition cannot be determined — it is meshed with other coalitions representing that `teapot-4` is a teapot, that teapots are liquid holders (in addition to being artefacts), etc. In fact, figure 3.3.3.1. shows many coalitions simultaneously (or rather only one bigger and looser coalition). The figure is 'centered' around the proposition `color-of-1 (teapot-1, green-1)`, that's all. Similar figures can be drawn around `teapot-1`, for example, showing that it is green, made of metal, has a handle, etc.

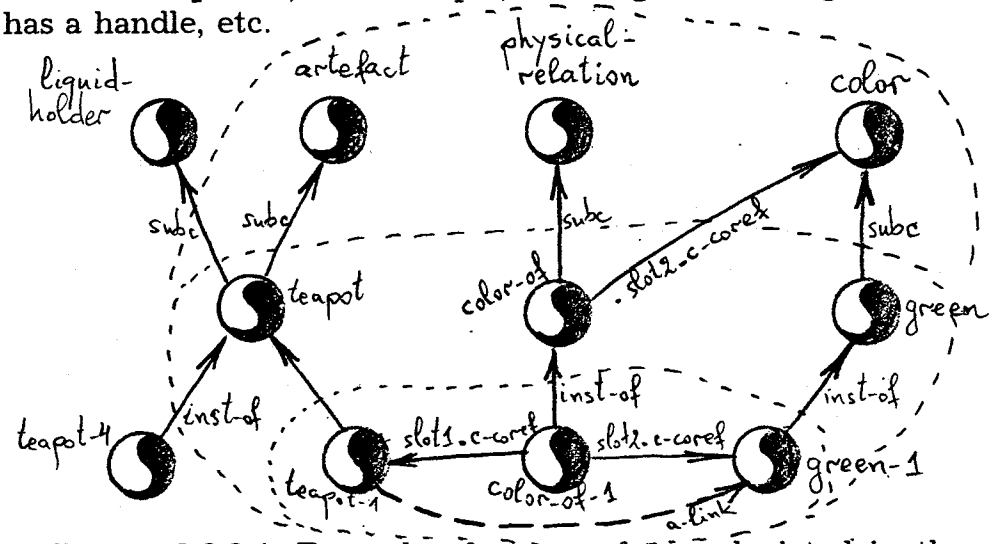


Figure 3.3.3.1. Example of a meso-frame depicted in the node-and-link notation. For simplicity, only part of the links are shown and all the connectionist aspect is omitted. The micro-frame `color-of-1` has two S-slots filled by (references to) `teapot-1` and `green-1`. It also has an `:inst-of` slot pointing to the conceptual agent `color-of`, etc. Compare with figure 4.2.3.

There is also a connectionist aspect of the meso-frame (not shown in the figure). Each link is weighted and activation spreads between the nodes via the links. For example, if the agent `color-of-1` receives activation from somewhere, it will pass it to its coalitions partners who in turn will pass it to their partners and so on. Thus, if one member of a coalition

is active, the other members tend to be active too. In tight coalitions this leads to synchronized availability of the agents involved.

It is possible to add new information to a meso-frame. For instance, some sort of perceptual mechanism[7] could construct a new micro-frame representing, e.g. that `teapot-1` is on `table-2`. This new agent will then join the coalition and thus will be involved immediately into the connectionist and symbolic activities that take place at the time.

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, two desirable properties that pose difficulties to conventional frame-based systems. These properties relate DUAL coalitions to *schemata* in some connectionist networks (Rumelhart et al, 1986).

3.3.4. Dynamic Emergent Computation

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 does (or does not) accomplish 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

[7] Perceptual mechanisms are, at the time being, completely lacking in the architecture. However, they are recognized as key ingredients of any plausible cognitive model and incorporation of such mechanisms is an important direction for future research.

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).

3.3.5. An Example: The Marker Passing Mechanism

The symbolic processors of many DUAL agents are preprogrammed to perform *local marker passing (LMP)*. It is outlined here as an illustration of the symbolic processing in the architecture as well as a basic mechanism for the AMBR model (subsection 4.3.2.).

Marker passing (MP) 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. Thus, each origin sets up a wave of markers that gradually expands until some *attenuation mechanism* stops the marking. If the waves starting from the two origins meet, one or more *paths* are found and reported. These paths can then be used for various purposes including natural language comprehension (Charniak, 1983), planning and problem solving (Hendler, 1988), similarity judgment (Kokinov, 1992b), etc. In AMBR2 the marker passing is used during the mapping subprocess: the intersection of markers justifies the hypothesis that their nodes of origin correspond (see section 4.3.2.).

The *global marker passing (GMP)* in DUAL is a dynamic emergent process that happens at the mesolevel. That is, it is a whole coalition of DUAL agents that cooperatively produce the final result. Global MP depends on local MP — the ability of each node in the network to receive and send markers. On the other hand, the important result — the intersection of two markers and the path connecting their origins — cannot be produced by any single node.

To take a concrete example, consider how AMBR2 generates the hypothesis that `teapot-1` corresponds to `glass-2` on the grounds that both are instances of the concept `liquid-holder`. Figure 3.3.5.1 shows the relevant coalition of DUAL agents (using the node-and-link notation).

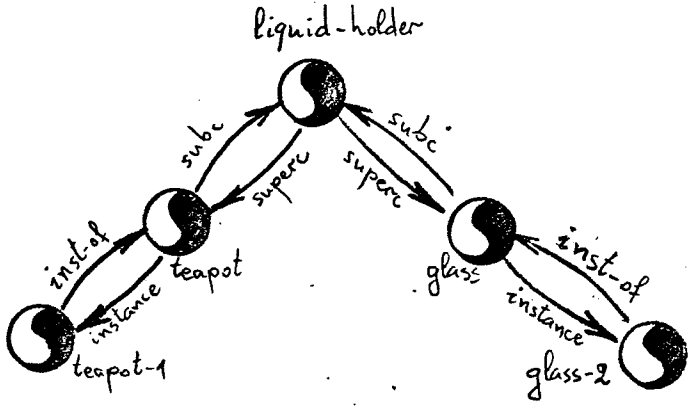


Figure 3.3.5.1. Fragment of the semantic network used in the marker-passing process. See text for details.

Suppose that at moment `t=0.0` the agent `teapot-1` is activated from some source external to the coalition. This will trigger a sequence of events over the entire coalition that eventually will produce a new temporary agent named `teapot-1<-->glass-2` at moment `t=5.3`. The following transcript, which is adapted excerpt of an actual AMBR2 transcript, reveals the major events during this process. (Agent names begin with the prefix '#$' and markers are printed as '#<M origin>'.) Compare the transcript below with the time diagram shown in figure 3.3.5.2. (See also section 4.3.3.6.)

```
At time 0.0, adding #$teapot-1 to WM.
At time 0.1, adding #$teapot to WM.
At time 0.3, adding #$liquid-holder to WM.
At time 0.5, #<M teapot-1> received in the input zone of #$teapot.
At time 0.6, adding #$glass to WM.
At time 1.0, adding #$glass-2 to WM.
At time 1.1, #<M teapot-1> received in the input zone of #$liquid-holder.
At time 1.8, #<M glass-2> received in the input zone of #$glass.
At time 1.8, #<M teapot-1> received in the input zone of #$artefact.
At time 1.9, adding #$solid-object to WM.
At time 2.3, #<M glass-2> received in the input zone of #$liquid-holder.
At time 2.4, #<M teapot-1> and #<M glass-2> intersected at #$liquid-
             holder.
At time 2.8, #<NCR liquid-holder> received in the input zone of #$nc1.
At time 5.3, creating a new agent: #$teapot-1<-->glass-2.
...
```

From this transcript, one can reconstruct the following story: Once `teapot-1` is activated, it spreads activation to its coalition partners, bringing them into the working memory (WM). They pass the threshold at different moments, reflecting their different connectedness to the source of activation which in this case is `teapot-1`. In the same time, the symbolic processor of `teapot-1` is working and sends a marker to its 'parent' in the network, namely `teapot`. (`Teapot-1` can reach its 'parent' via the link labeled `:inst-of`.) The symbolic operation of producing and sending a marker takes time and, therefore, the marker is actually sent to `teapot` at moment 0.5. In turn, `teapot` sends a copy of the marker to `liquid-holder`. The same operation (handling a marker) takes 0.6 time units because `teapot` is less active than `teapot-1` and hence its symbolic processor works more slowly. The agent `liquid-holder` (like all other agents earlier in the path) stores the marker in its local buffer. It also tries to send the marker further but its parent is not visible — it has not been activated enough to pass the threshold. Therefore, the marker stops.

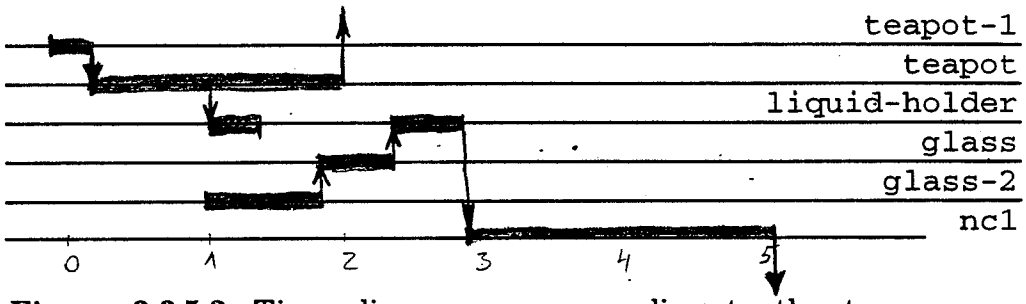


Figure 3.3.5.2. Time diagram corresponding to the transcript given in the text.

Meanwhile, the spreading activation mechanism continues to add agents to the working memory. At time 1.0, `glass-2` also passes the threshold. Its symbolic processor is preprogrammed to emit a marker whenever it enters the WM and, therefore, `glass-2` begins working on this task at time 1.0. This task takes 0.8 units of simulated time because the activation level (and hence the speed) of the agent is relatively low. The new marker is sent to `glass` at time 1.8 and eventually reaches `liquid-holder` at time 2.3. The symbolic processor of the latter detects the intersection of the new marker with the old one (which has been stored in the buffer) and produces a node-construction request. This request is then sent to a special agent (named `nc1`) capable of constructing temporary agents. Creating a whole agent from scratch is difficult — it takes 2.5 units of time even for a specialized agent like `nc1`. The new agent enters the working memory at time 5.3. It has three slots filled with references to `teapot-1`, `glass-2`, and `liquid-holder`, respectively. Now the topology of the network has changed, the new state of affairs is shown in figure 3.3.5.3. The presence of a new node (and new links) will affect the activation levels of all agents in this part of the network, which in turn will affect their speed, etc.

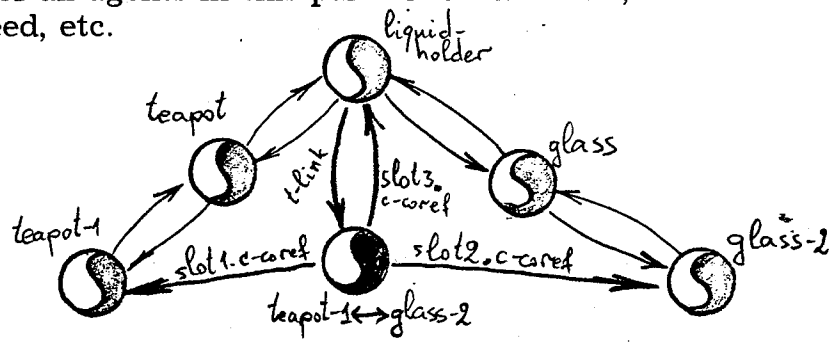


Figure 3.3.5.3. The network from figure 3.3.5.1 after addition of the new agent named `teapot-1<-->glass-2`.

The excessive detail of the transcript presented above makes it difficult to see the forest behind the trees. If one turns off all messages except the most important ones[8], however, the emergent nature of the proc-

[8] or, better, if one has access to a graphical interface

ess becomes more clear. New agents are created at irregular and virtually unpredictable moments in time. The exact course of the computation is determined by a multitude of factors, each of which has relatively minor impact on the final outcome. Yet the process exhibits certain regularities — more active areas of the network generate more marker intersections, the new agents created from these intersections are incorporated faster into the network, which in turn gives them advantage over their rivals and so on. Gradually, the process of *dynamic emergent computation* advances to completion. And the final product of the computation is 'meaningful' in most (but not all) cases.

3.4. 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. The system embodies the DUAL-based model as a whole. The behavior of the model is by definition the behavior of the system and vice versa.

Different models built on top of the architecture may involve different formations. Usually, there are only two or three formations which are richly interconnected and form a unitary system. In this subsection, we will present the formation that is present in all models — the DUAL network. AMBR2 uses one more formation — the constraint satisfaction network — which will be presented in the next chapter.

3.4.1. The DUAL Network

Most of the agents and, therefore, most of the knowledge and processing in the architecture reside in the DUAL network. Its *nodes* are the agents themselves, its *links* stand for interactions between them. There are no restrictions on the topology of the network — an arbitrary number of links may come in or out any given node.

Most nodes in the DUAL 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. It changes with time and this is of major importance for the overall dynamics of the system.

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 y when they 'drop out', they vanish altogether.

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

WM = active portion of LTM + temporary agents .

This formula, however, is potentially misleading. It suggests that the temporary agents are somehow isolated from the rest. They are not. The actual state of affairs is depicted in figure 3.4.1.

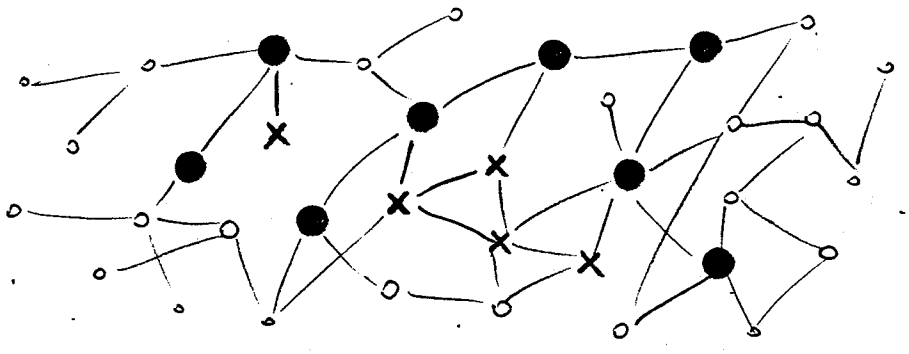


Figure 3.4.1. The long-term memory (LTM) of DUAL. Active permanent agents are shown as ●, inactive permanent ones y as **o**, and temporary nodes y as **X**.

3.4.2. The Flow of Activation in the Network

Activation can enter the DUAL network from a special node called *input node*. This node models the influence of the environment. In future models it will be replaced by a whole formation — the *visual array*. At the time being, however, the perceptual mechanisms in the architecture are extremely limited. There is only one node which is a constant (and strong) source of activation. The human user of the system attaches some agents to that node, thus allowing for the spread of activation from the input node to the DUAL network. In this case we speak of *exogenous activation* (with respect to the network). It is the medium of perceptual stimulation in DUAL models.

There are sources of *endogenous activation* too. Most importantly, there is a special *goal node* which is always active. It is, in a very rudimentary sense, the medium of the *intentions* of the system. When the model is working on some task, the agent representing the goal of the task is connected (again by the human user) to the goal node and thus receives continuous and strong support from it. In DUAL, there may be several goals connected to the goal node simultaneously and competing for the resources of the system.

Finally, small amounts of endogenous activation may be spontaneously created locally by the symbolic aspect of a given agent when the outcome of its symbolic operation is especially successful.

In short, activation in DUAL springs from some well defined sources and then spreads among the nodes via the links. The links in the network are excitatory — they have positive weights. This means that unless the activation is restricted somehow, it will soon spread throughout the entire network. The activation level of all agents will reach its maximum value and the purpose of the whole mechanism will be defeated. To prevent this, there is a *decay* process which limits the total amount of activation in the system. In the absence of external stimulation the activation level of any given agent decreases spontaneously by an exponential law. Therefore, in order for an agent to stay active, it must receive support from neighboring agents.

The decay rate, however, is not very big. This provides for a certain amount of inertia in the pattern of activation. In other words, some *residual activation* stays for a while even after the external support has been eliminated. As a consequence, recent states of the system can influence its current one. *Priming effects* and *sets* (*Einstellungen*) can be modeled in this way. The behavior of the model fits psychological data — priming effects (*i*) exist and (*ii*) decrease in the course of time (Kokinov 1990, 1994a).

As it was stated from the very beginning, DUAL is designed explicitly to model the phenomena of context effects. The *dynamic theory of context*

proposed by Boicho Kokinov (1995) is the foundation of this enterprise. In the theory, *context* is considered as "the dynamic fuzzy set of all associatively relevant memory elements (mental representations or operations) at a particular instant of time" (Kokinov & Yoveva 1996). Further, distinction is made between *reasoning-induced*, *perception-induced*, and *memory-induced contexts*. Each one of these has a straightforward counterpart in DUAL — endogenous, exogenous, and residual activation respectively.

CHAPTER IV

AN INTEGRATED MODEL OF ANALOGICAL RETRIEVAL AND MAPPING

This chapter describes AMBR2 — a cognitive model built on the basis of the DUAL architecture. 'AMBR' stands for 'Associative Memory-Based Reasoning' (Kokinov 1988, 1994a) and has been conceived as a model with very broad scope. It offers a unified account of deductive, inductive, and analogical reasoning (Kokinov, 1988, 1992a). In this thesis, however, we will concentrate only on analogical reasoning because the simulation experiments performed with the model so far fall into this category.

One of the key ideas motivating AMBR research is that it is necessary to develop *integrated* models. With respect to analogy-making, the long-term goal is to develop a model which integrates all subprocesses mentioned in chapter 2: perception, retrieval, mapping, transfer, evaluation, and learning. At the time being, however, two of them are elaborated in much greater detail than the rest. Therefore, in this thesis we will concentrate only on these aspects of analogy making — retrieving a source analog, mapping it to the target and the integration between the two.

In short, the present thesis is limited only to the AMBR2 model as it currently stands[1]. Here and now, AMBR2 is an integrated model of analogical retrieval and mapping. We fully recognize the fact that the model thus presented is incomplete and we view the current version of the model only as an intermediate stage of a bigger project: AMBR3 will add transfer and some elements of learning. Another DUAL-based project, PEAN, concentrates on perception and its integration with reasoning.

4.1. AMBR2 as a Psychological Theory

As a cognitive model, AMBR2 has two complementary aspects: (*i*) it puts forward some claims about the human cognitive system and (*ii*) it puts forward (and implements) a concrete computational scheme for solving a restricted class of problems. These two aspects correspond but do not

[1] To give the word to Drew McDermott (1981): "If a thorough report on a mere actual implementation were required, or even *allowed*, as a Ph.D. thesis, progress [in AI] would appear slower, but it would be real." (See section 1.4. for a bigger quotation.)

overlap. For instance, we do not claim that humans pass markers in their heads while solving a problem; marker passing is used only instrumentally in AMBR2. On the other hand, we do claim that the process of *human* analogy-making cannot be partitioned into successive stages, each of which works independently of the others.

From methodological point of view, it is important to keep these two aspects separate. Following the dictum of Johnson-Laird (1989) to 'make a clear distinction between a program and the theory that it is intended to model', the two aspects of AMBR2 are treated separately in this chapter. The chapter begins with an outline of the theoretical tenets behind the model. This is followed by a description of the exact computational mechanisms used by AMBR2. Finally, it is demonstrated how these computational mechanisms apply to the task of retrieving a source analog and mapping it to a target. Throughout the chapter, special emphasis is given to the points which are contributed by the present thesis and cannot be found in earlier publications.

4.1.1. Multiconstraint Theory

> *The differences among various alternative models of analogy should not obscure their commonalities. In particular, all of the models make some use of the three classes of constraints we have emphasized. The multiconstraint theory reflects the convergence of theoretical developments in the field. Analogy is undoubtedly one of the success stories in cognitive science.*
>
> (Holyoak & Thagard 1995, p.261)

AMBR2, like its predecessor AMBR1, belongs to the tradition set forth by Gentner (1983) and Holyoak & Thagard (1989). It agrees with the view that analogy-making involves alignment of the structural representations of two problems or situations. It agrees with the view that this alignment depends in one form or another on the following three constraints: (*i*) *structural constraint* — the pressure to identify and use an isomorphism between the descriptions of the two situations, (*ii*) *semantic constraint* — the pressure to identify and use correspondences between semantically similar elements of the descriptions, and (*iii*) *pragmatic constraint* — the pressure to identify and use correspondences for pragmatically important elements of the descriptions.

Further, AMBR2 agrees that the aforementioned constraints are 'soft' — they do not operate as unviolable rules but rather as competing 'pressures' (Hofstadter 1984) that restrict the space of possible solutions to a problem.

4.1.2. Flexibility, Efficiency, and Context Sensitivity

AMBR2 is aimed at achieving the following properties that are characteristic of human cognition:

Flexibility — the system should be capable of producing a broad spectrum of behaviors. In the context of the present discussion, this implies that the set of *all possible* outcomes of the analogy-making process should be as large as possible. Humans do not seem to obey the restriction that, e.g., relations are always mapped only to relations. Therefore, it is highly desirable for a cognitive model not to ban any possibilities *a priory*, however strange and infrequent they may be.

Efficiency — the system should come up with some answer within a reasonable amount of time. In the context of the present discussion, this implies that the set of *actually considered* alternatives should be quite small. This requirement could be neglected when the problems presented to the model are simplified but it quickly becomes crucial with scaling up.

Flexibility and efficiency are often in conflict with each other. Measures for increasing flexibility tend to decrease efficiency and vice versa. For instance, the use of powerful all-encompassing techniques such as the huge constraint-satisfaction network in ACME (Holyoak & Thagard, 1989) opens the door for combinatorial explosion. On the other hand, the use of optimized, resource-focusing methods such as ignoring object attributes in SME (Falkenhainer et al., 1986) entails lack of flexibility.

There is, fortunately, a way out of this dilemma — it is possible to keep the 'search space' unrestricted and open-ended and yet to explore only a small fraction of it, thus escaping the combinatorial explosion. The regions (or 'paths') of the search space that are processed on each particular case are not prescribed in advance but are determined dynamically during the run. We call this style of computation *parallel dynamic processing* (Kokinov et al., 1996). The main idea is to explore several paths simultaneously but at speeds proportional to their promise (Holland, 1975; Hofstadter, 1983; Kokinov, 1994a). This strategy guarantees that, averaged over many trials, plausible solutions will be generated most of the time and yet no solution is ruled out *a priory*. This idea has been applied in other models of analogy-making with impressive results (Hofstadter, 1995; Mitchell, 1993; French, 1995).

One more consideration remains to be clarified: how to evaluate the 'promise' of a given path. The answer adopted in DUAL is: **Context!** It is the context that gives cues about which regions of the search space are relevant and, therefore, merit exploration. Moreover, relevance comes in degrees and varies over time, thus making the computation that depends on this mechanism dynamic and context-sensitive.

There are many demonstrations that the human cognitive system is context-sensitive. Moreover, it is claimed that context-sensitivity can be found not only in processes like perception and language understanding (where it is widely accepted and unquestionable) but also at the level of so-called higher cognitive processes such as reasoning and decision making (Kokinov, 1990; Kokinov & Yoveva, 1996). One of the main objectives of the AMBR2 model is to account for this context-sensitive nature of human analogy-making.

4.1.3. Analog Retrieval

This section discusses the issue of analog retrieval and presents a novel explanation of some empirical phenomena reported in the literature.

4.1.3.1. The Task of Analog Retrieval

There is considerable evidence that the task of retrieving an appropriate source analog from long-term memory is one of the major difficulties in analogy-making. In particular, people often fail to access potentially useful analogs even though it could be verified that these analogs are in fact retained in their LTM (Gick & Holyoak, 1980, 1983). Other research has shown that, although people are often reminded of prior problems, events, or situations, these remindings are often based on similarities among objects and attributes rather than on relational similarities (Holyoak & Koh, 1987; Keane, 1987; Ross, 1984).

The task of analog retrieval can be stated roughly as follows: Given a very large pool of episodes (problems, situations, etc.) and a *probe* (a new episode), pick up one or a few episodes from the pool that are similar to the probe.

This task quickly becomes computationally very difficult as the number of old episodes increases. Moreover, a cognitive model of analog retrieval should also account for the profile of human analog retrieval:

1. Retrieval is a relatively fast process — in most cases, an episode is either retrieved rapidly or not at all. The time needed does not seem to depend on the number of episodes stored in LTM. (It could depend on the probe, however.)

2. The same person given the same probe is often reminded about different episodes if queried on different occasions. In particular, there are context and priming effects on retrieval.

3. Retrieval seems to be dominated by semantic as opposed to structural similarity (Holyoak & Koh, 1987; Keane, 1987; Ross, 1984).

4.Nevertheless, remindings based solely on shared relational structure do occur, albeit infrequently.

5. Episodes from a domain similar to that of the probe tend to be retrieved much more readily than episodes from remote domains (Keane, 1986).

6. Highly familiar episodes tend to be preferentially retrieved (Inagaki & Hatano, 1987, cited by Hummel & Holyoak, 1997).

7. Retrieval is facilitated by learning conditions that encourage induction of an abstract schema from remote analogs (Gick & Holyoak, 1983).

8. More generally, retrieval is facilitated by learning conditions that encourage intensive encoding of the original materials (Faries & Reiser, 1988, cited by Forbus et al., 1994b)

In addition, some researchers (Wharton et al., 1991) argue that retrieval is an inherently competitive process. More precisely, it is claimed that people are more likely to retrieve an episode from LTM if it is the best match available than if some other episode provides a better match. We have refrained to include this finding in the list, however, because it is not clear from the study whether the pattern of results could alternatively be explained with the interference between different episodes. Besides, the notion of 'the best match available' hides certain conceptual difficulties.

4.1.3.2. Models of Analog Retrieval

Numerous models of memory retrieval in general and analog retrieval in particular have been proposed in the literature.

In *case-based reasoning* (Carbonell, 1983; Kolodner, 1993; Veloso, 1994), retrieval is performed on the basis of specific LTM organization around a carefully crafted indexing scheme. As Kokinov (1994a) and Forbus et al. (1994b) point out, however, such indexing schemes can be very efficient but lack psychological plausibility because they fail to match the pattern outlined in the previous subsection (e.g., they are not flexible enough).

In *memory-based reasoning* (Stanfill & Waltz, 1986), retrieval is performed on the basis of a general measure of similarity between cases. Usually, this measure is simply the dot product between feature vectors, as in many mathematical models of human memory. Flat feature vectors, however, are out of favor in analogy research because analogy-making is an inherently structural problem.

Two influential models — ARCS (Thagard et al., 1990) and MAC/FAC (Forbus et al., 1994b) — use different variations of a same general idea. They address the issue of retrieval by a two-stage screening (cf. Smith et al., 1974). Initially, a cheap filter is applied to all episodes stored in LTM. Then, the episodes that have 'survived' the first test are subjected to more stringent (and computationally more costly) evaluation. (Figure 4.1.3.) The second stage in both models is based on the machinery used for mapping

— ACME and SME, respectively — although working in 'economical' mode.

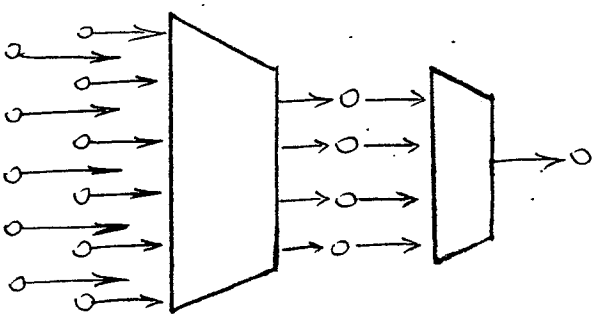


Figure 4.1.3. Memory retrieval as a two-stage screening.

In a recent proposal — LISA — Hummel & Holyoak (1997) view retrieval as 'a process of guided pattern classification, in which units representing stored propositions compete to respond to the distributed patterns generated by an active "driver" analog' (p. 47). Their model depends on distributed representations over *semantic units*, an approach that has much in common with the feature vectors mentioned above. LISA, however, escapes from the 'flatness' of such vectors by using dynamic binding to represent structured propositions. An important advantage of LISA is that the retrieval process is integrated with the process of mapping. Both operate in the same fundamental way and on the same knowledge representations. Thus, this model constitutes an interesting alternative to the AMBR approach.

Spreading activation has also been used by several independent proposals (Anderson, 1983, 1993; Anderson & Thompson, 1989; Holland et al., 1986). The main idea here is to start by activating the probe and allowing this activation to spread through the LTM. Those memory elements that become more active than others are considered as more plausible source analogs.

AMBR1 belongs to the category of models that depend on spreading activation (Kokinov, 1994a). It is also characterized by the fact that it integrates retrieval and mapping. AMBR2 follows the track of its predecessor but has modified many details. We will turn the discussion now to one particular modification that, in our view, has broader scope than that of the AMBR2 model itself.

4.1.3.3 One Problem with Retrieval Models

The models discussed so far assume explicitly or implicitly (at least to the extent that can be judged from the articles) that *all* episodes stored in the LTM are processed each time, although superficially. To begin with the more clear examples, the first stages of ARCS and MAC/FAC in fact do exhaustive parallel search of the whole episodic memory. That implies

that the system has a complete list (or some equivalent thereof) of all episodes stored in the LTM. The same implication can be made for case-based models, for feature-vector models, etc.

LISA seems to employ full connectivity between the semantic units and all predicate and object units that they affect. More precisely, if some predicate (or object) P_1 involves the semantic microfeature S, then there will be a link from the unit standing for S and the unit standing for P_1. Similar links will connect S with all other units P_2, P_2, ... P_n that depend on it. For instance, the microfeature `animate` (see Appendix B in (Hummel & Holyoak, 1997)) is connected to all nodes standing for animate beings in the LTM.

AMBR2 could use analogous strategy. At first, we intended to put `:instance` links from a conceptual node to each of its instances in the network. The problem with this approach is that the number of such links will become unacceptably big. A long-term memory of any realistic size would contain thousands of different episodes and hence hundreds of instances of a given concept. This number would be greater for high-frequency concepts such as `cause`[2]. In systems using distributed representations the number of links is even greater because each instance has to be linked to several semantic primitives instead to a single parent.

From a psychological point of view, it seems very unrealistic that there are links to *all* instances of a given concept (or, in the case of LISA, to all instances characterized by a given microfeature). A person could not possibly enumerate *all* instances of the relation `revolves-around` that participate in some episode stored in LTM. Even less likely is that some computational mechanism 'visits', however superficially and unconsciously, *all* episodes that have been accumulated from past experience.

The notion of having so many links is questionable from computational point of view as well. The spreading-activation mechanism becomes virtually useless when the fan-out factor is in the order of thousands or tens of thousands. In models using weight normalization (such as AMBR), having a thousand links is equivalent to having none. Such links fail to transmit any activation and yet they clutter the system.

It is important to stress that there is no problem with the links in the opposite direction. It is reasonable, we think, to posit a `:inst-of` link starting from each instance and pointing to the corresponding concept. Those 'bottom-up' links, however, are not very useful for retrieval of related episodes (though they are certainly useful for other purposes).

[2] With respect to this, the following excerpt from (Thagard et al., 1990) is quite instructive: 'Because of their ubiquity and context independence, the following predicates are not used as retrieval cues: `cause`, `if`, `conjoin-object`, `conjoin-event`, `become-true`, `become-false`.' (p.274)

To sum up, it seems highly unlikely that the cognitive system keeps a list (or some equivalent thereof) of all episodes stored in LTM. Consequently, it is unlikely that human memory retrieval depends on thorough search of the entire LTM (be it serial or parallel). Similarly, it is questioned whether the memory element(s) representing a given concept have direct access to all memory elements representing instances of this concept.

On the other hand, a person can usually enumerate *some* instances of any familiar concept. In addition, directed association studies reveal that activation of a concept tends to activate *some* of its instances, and in particular the more typical ones.

4.1.3.4. The AMBR2 Proposal

AMBR2 adopts the following memory organization with respect to the problem outlined above. Part of the memory elements encode information about concepts and their interrelations. In AMBR2 terminology, these are agents of `:type :concept`. Another part of the memory elements encode information about specific objects, events, situations, etc. In AMBR2 terminology, these are agents of `:type :instance`. Elements of both kinds are linked in a common network by various labeled links (see section 3.2.2.6). In addition to having a label, each link also has a weight.

As a rule, each element of `:type :instance` has a `:inst-of` link pointing to the appropriate concept. In the other direction, an element of `:type :concept` may have `:instance` links to *some* of its instances. The total number of such 'top-down' links is limited. Therefore, most of the instances of any given concept are not directly accessible from it. This relieves the model of the implausible assumptions discussed in the previous subsection.

One more question remains to be clarified: which instances are 'privileged' to be accessible by their corresponding concept and what is the mechanism responsible for favoring ones and neglecting others.

The answer adopted in AMBR2 is that top-down links (and hence the topology of the whole network) are subject to continuous restructuring. The exact mechanisms for this are open for discussion but the main principle is the following: when some instance has been activated and processed for sufficiently long time, chances are that a new memory trace will be established. In AMBR2 terminology, a new top-down link will be created or the existing one (if any) will be strengthened.

New links are established at the expense of old ones. Again, the exact mechanism is open for discussion but one natural approach is the following: the new link is added to the total set of links leaving the corresponding node with some non-zero weight. Afterwards, the total weight of the links is renormalized again, which will decrease the weights of all older links. If some weight becomes too low after the application of this procedure, the link is removed.

This mechanism ensures that at any given moment only a few links will point 'downward' from any given concept node. Stated differently, there is retroactive interference between new instances (or episodes, problems, etc.) and older ones. This interference restricts the number of reliable memory traces without ruling out any particular element on *a priory* basis.

4.1.3.5. Explanation of Empirical Facts

This subsection discusses the empirical facts outlined in subsection 4.1.3.1 in the light of the current proposal.. Related arguments can be found in (Kokinov 1994a, pp.276-8). For easy reference, we will use the numeration from subsection 4.1.3.1.

Phenomena related to episode similarity have straightforward explanation. Activation spreads from the instances of the probe along the bottom-up links and activates the corresponding concepts, which in turn activate some other instances through top-down links. Episodes (or *coalitions*) where activation converges will be retrieved and this happens either rapidly or not at all (phenomenon 1). Moreover, the time for retrieval does not depend on the total number of nodes in LTM..

It is clear that familiar episodes will be preferentially retrieved (phenomena 5. and 6.) because such episodes are more likely to be linked to the corresponding concepts. Moreover, the conceptual network for familiar domains is supposedly more elaborate, which provides more 'hooks' for the instances.

In addition, retrieval is facilitated by learning conditions that encourage intensive encoding of the original materials (phenomena 7. and 8.). Such intensive encoding, among other things, increases the probability that the representation of the problem being learned will be wired into the network. (See Kokinov (1994a, p. 277) for additional discussion about abstract schemas.)

Phenomenon 2. deserves special attention. The same person given the same probe on different occasions could be reminded about different episodes due to a combination of the following two reasons: (*i*) the connectivity among the memory elements changes from one occasion to the next as a result of the experiences happened in the interim; (*ii*) the external context on the two occasions is different. The overall pattern of activation in the network depends not only on the probe but also on the things that are

perceived at the time. (Archimedes cried "Eureka!" only after he saw the water spilling out of his bath-tub.)

We now turn to the questions that are specific to analog retrieval (phenomena 2. and 3.) and have been in the focus of previous models (Thagard et al., 1990; Forbus et al., 1994b; Kokinov, 1994a).

Why is analog retrieval difficult? According to AMBR2, at least part of the answer is the following: Spreading activation is an automatic process and is beyond the control of the reasoner. The latter can influence the outcome of the process only indirectly (Kokinov, 1994a). In addition, there are several factors that limit the spread of activation. One factor of this kind is the restriction on the number of top-down links proposed here. Thus, many episodes that would be good source analogs reside in the LTM but do not receive enough activation to enter the working memory.

Why does semantic similarity dominate retrieval? Activation spreads mostly through links which have definite semantic interpretation. In particular, many links represent class/instance and superclass/subclass relationships. In this way, activation in effect spreads mainly (though not exclusively) among semantically similar elements. The associative mechanism does not distinguish between structural and superficial features so both are used in retrieval. Since the number of superficial features used to described a situation is usually greater than the number of structural ones, retrieval as a whole is dominated by the former (Kokinov, 1994a).

There is an additional fan-out effect with respect to relations. The relative frequency of relations is greater than that of objects and attributes. A couple of dozens of relations occur again and again in each new episode. Due to retroactive interference, the 'turnover' of the `:instance` links from the relational nodes is expected to be very high. As a result, these nodes does not serve as good transmitters of activation in retrieval. They get highly activated by the probe but the activation fails to converge on a single episode in the LTM.

A prediction of the model is that low-frequency (and hence semantically more loaded) relations such as `revolves-around` will be stronger retrieval cues than high-frequency relations such as `in`, `greater-than`, and `cause`. This prediction has to be tested experimentally.

Why do 'insightful' retrievals happen, after all? According to AMBR2, the answer lies in the combination of the following four reasons. First, there always is some small baseline probability for any episode to be retrieved. This stems from the fact that AMBR2 does not reject any possibilities *a priory*. Therefore, in a big sample of trials one could expect a few 'insights' as well as a few 'blunders'. Second, retrieval in AMBR2 is context-sensitive. A particular episode could be facilitated by the external context and/or by priming effects (Kokinov, 1990, 1994a).

Third, the retrieval process in AMBR2 acts in close interaction with the mapping process. Thus, a semantically unrelated by structurally similar analog could be retrieved as a result of a 'bootstrap' sequence of events.

A fourth reason which is closely related to the third is that the retrieval in AMBR2 operates at the level of individual propositions, not at the level of whole situations. This follows from the decentralized nature of the knowledge representation scheme adopted in the model.

4.2. Knowledge Representation in AMBR2

This section begins the presentation of AMBR2 as a computational model. It shows how the DUAL representational scheme is actually put to work in AMBR2.

4.2.1. Types of AMBR2 Agents

According to the specification of the architecture (section 3.2.3.), each DUAL agent has a *micro-frame*. The micro-frame is a bundle of labeled *slots*. The semantics of a slot is determined by its label. Some labels (e.g. `type`, `inst-of`) have the same interpretation in all micro-frames. They form the basis of the so-called *general* slots (or G-slots). For instance, each DUAL agent has a G-slot labeled `type` and the interpretation of this slot is the same across all agents in the architecture. In addition to the general slots, each micro-frame may have *frame-specific* slots (or S-slots) of its own. S-slots have dummy labels like `slot1`, `slot2`, etc. and are like subframes within the frame. They have *facets* which have labels like the G-slots. For instance, the S-slot labeled `slot1` may have facet labeled `type`.

The general slots and the facets of the S-slots are filled up by *fillers*. In AMBR2 there are two kinds of fillers — *tags* and *references*. Tags are used mostly as fillers of the `type` slot to delineate the type of the micro-agent. Other slots are filled by references to other micro-agents, thus linking the micro-frame to the bigger representational structure — the meso-frame (section 3.3.3.). This subsection deals with the `type` slot and the possible tags that may serve as its fillers.

Each agent in AMBR2 has a `type` slot that describes the type of the agent. More precisely, the set of all AMBR2 type tags is the following:

- `concept` — the micro-frame represents a whole class of instances;
- `instance` — the micro-frame represents a particular instance;
- `hypothesis` — the micro-frame represents a hypothesis about a tentative correspondence between two concepts or two instances;
- `temporary` — denotes that the micro-frame (and the whole agent) is temporary (see section 3.2.6.);
- `object` — the micro-frame represents an object, an attribute value, or some other non-relational entity;
- `relation` — the micro-frame represents a relation (possibly with a single argument);

- `situation` — the micro-frame stands as a common reference point to the spatio-temporal unity of a coalition of micro-frames (see section 4.2.3.);
- `embryo` — used only for 'embryo' hypotheses (see section 4.3.3.4.);
- `mature` — used only for established hypotheses.

These tags can be used in conjunction with one another to account for the variety of agents employed by the model. For instance, the `type` slot of some agent can be filled by the list `(temporary instance relation)` thus stating that the agent under question is a temporary agent standing for an instance of some relation.

There are rules that restrict the combinations among different type tags. For example, all agents of `type hypothesis` are also `temporary`. Therefore, despite the big number of possible type combinations, there are only three major types of AMBR2 agents: *concept-agent*, *instance-agent*, and *hypothesis-agent*. Thus, the first three tags from the list above are the most important. The remaining tags mark type subdivisions.

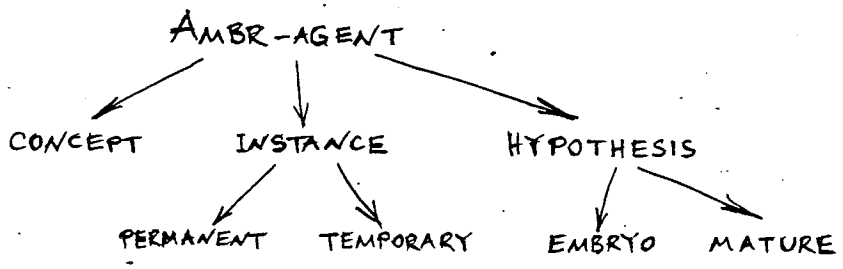


Figure 4.2.1. Main types of AMBR2 agents.

4.2.2. Representation of concepts and objects

With full awareness that AMBR2 agents are nothing but ungrounded symbols (Harnad, 1990), we follow the common AI terminology and say that they stand for 'things in the world'. We also use mnemonic agent names like `fire`, `cause`, etc. 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.[3]

Concept-agents represent classes of entities. Different concept-agents stand for different classes (or 'concepts'). The taxonomy of classes is represented by `subc` and `superc` links between concept-agents. Each class

[3] 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.

may be linked to zero, one or more super- or sub-classes, different links possibly having different weights (section 3.2.4.2.)

Instance-agents 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. The specification of AMBR2 postulates that each instance has exactly one 'parent concept'[4]. Figure 4.2.2. illustrates.

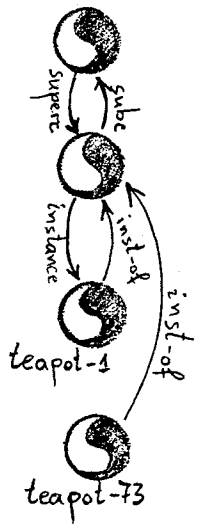
```
liquid-holder:
   :type      (concept object)
   :superc    teapot

teapot:
   :type      (concept object)
   :subc      liquid-holder
   :instance  teapot-1

teapot-1:
   :type      (entity object)
   :inst-of   teapot

teapot-73:
   :type      (entity object
               temporary)
   :inst-of   teapot
```

a)



b)

Figure 4.2.2. Example of concept-agents, instance-agents, and possible relations between them. Each micro-frame can have additional slots (not shown in the figure). All connectionist aspects are omitted.

Some instance-agents are temporary (see section 3.2.6.). They are marked by a `temporary` tag. Absence of such tag means that the agent is permanent. Temporary agents does not belong to the long-term memory of the system. Rather, (it is supposed that) they have been constructed during the recent computation by some perceptual or inference mechanism. In the current version of AMBR2, temporary instance-agents are used to represent the target situation or the problem that the model tries to solve. In contrast, permanent instance agents are used for all situations stored in the LTM. Concept-agents are always permanent.

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 4.2.1.2. belong to this category. Sometimes, the same tag is used for other non-relational categories such as colors, temperature qualifiers, etc. In contrast, the `relation` tag is

[4] This restriction does not diminish the expressive power of the representation scheme because the complex cases can be handled through `c-coref` links.

used to designate micro-frames that represent some relation. Such micro-frames usually have frame-specific slots that represent the arguments of the relation. AMBR2 treats attributes (cf. Gentner, 1983) as one-argument relations. Finally, there are agents of type `situation`. Contrary to the name of the tag, such agents do *not* represent whole situations. Rather, they represent the spatio-temporal unity of a coalition of micro-agents (see subsection 4.2.3.). This additional information is used by the structure-correspondence mechanism.

4.2.3. Representation of Propositions

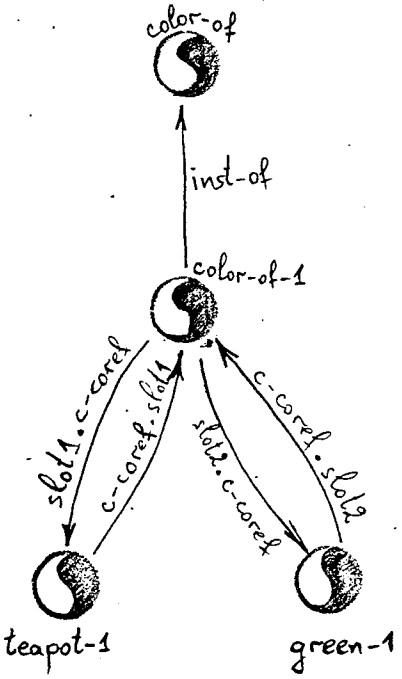
Individual AMBR agents are small and their micro-frames cannot represent much. Therefore, even relatively simple representational units like propositions need to be represented by a coalition of agents (section 3.3.3.). In the case of propositions, such coalitions are small and very tight.

```
color-of:
  :type      (concept relation)
  :subc      physical-relation
  :slot1
    :c-coref  object
  :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      (entity object)
  :inst-of   teapot
  :c-coref   (color-of-1 . :slot1)

green-1:
  :type      (entity object)
  :inst-of   green
  :c-coref   (color-of-1 . :slot2)
```



a) **b)**

Figure 4.2.3. A coalition of four micro-frames representing the proposition `color-of-1(teapot-1, green-1)`. Compare with figure 3.3.3.1.

There is an agent that represents the *head* of the proposition. In figure 4.2.3., this is the micro-agent `color-of-1`. It is of type `relation` and is an instance of the concept `color-of`. The arguments of the relation are

represented by S-slots in the heading micro-frame. More precisely, the two slots of `in-1` represents the roles of the two operands. The concept-agent defines which is which. In this case, `slot1` is an object and `slot2` is its color. (Note that the same slot labels will have completely different meaning when used in other micro-frames.)

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 4.2.3., 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 named `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`.

It is important to stress that the agents shown in the figure convey information about a number of other facts besides the proposition `color-of-1(teapot-1, green-1)`. In particular, `teapot-1` is an instance of `teapot`, `color-of` is a kind of `physical-relation` and so forth. Thus, each agent shown in figure 4.2.3. participates in a number of overlapping coalitions. Alternatively, it could be said that the small coalition participates in a bigger coalition (a `meso-frame`) representing a whole situation.

4.2.4. Representation of Situations

AMBR2 differs from its predecessor in the representation of situations (or problems). AMBR1 used centralized representation, AMBR2 — decentralized. This subsection considers the advantages and disadvantages of these two possibilities.

4.2.4.1. Centralized Representation

The centralized representation of situation is characterized by the existence of a micro-frame standing for the situation as a whole. This micro-frame is called *head* of the situation. The head brings together all agents that build up the representation of the situation. There is one S-slot for each element — object or relation. The head is linked to *all* elements and some elements are linked back to the head, thus creating a network like the one schematized in figure 4.2.4.1. In addition to the 'vertical' links between the head and its elements, there are many 'horizontal' links between the elements themselves.

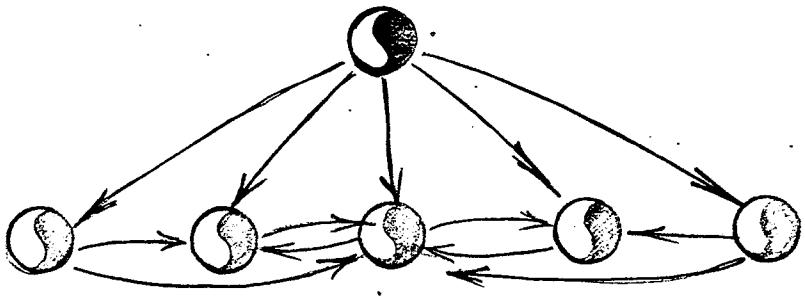


Figure 4.2.4.1. Schematic outline of centralized representation of a situation. There is one *head* that is connected to all elements of the situation. Cf. figure 4.2.4.2.

The centralized representation has a number of advantages. First, the situation has distinct identity. There is a frame that represents it as a whole. Thus, it is clear who is 'responsible' for the situation. To begin working on a problem, for example, it is sufficient to put the head on the goal list. To decide which situation 'wins' certain competition, it is sufficient to compare the activation levels of the heads, etc.

Second, all relevant aspects of a given situation are collected at one place. In other words, the frame problem is solved in advance. (Though it is the human programmer, not the program, who has solved it.)

Finally, the task of mapping one problem to another is greatly facilitated. It is transformed into a task of establishing slot-to-slot correspondences between two micro-frames. After the correspondences have been found, it is clear which elements of the source situation are left unmapped and are thus potential candidates for transfer.

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. 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 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 the assumption that they are represented in such neat and centralized fashion is warranted.

Second, centralized representations tend to be too static and inflexible. The elements of a situation are defined in advance and special steps need to be taken in order to add a new element or to remove an old one. Moreover, some researchers (Chalmers et al., 1992) argue that the models that start from hand-made representations of the problem by-pass the most difficult and essential part of analogy-making. Both AMBR1 and

AMBR2 are susceptible to this criticism but AMBR2 is a little less so due to the decentralized nature of its representations.

Finally (and very importantly), the slots in the heading micro-frame become too many. Even the simple situations used in the simulation experiments so far require that the head has at least ten S-slots. 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 re-appears again — it is necessary to specify which of the many elements of the situation are relevant to the task at hand. The big number of slots contradicts the specification of DUAL (subsection 3.2.3.2.). 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.

As a respond to these problems, AMBR2 has abandoned the centralized representation used by its predecessor. The shift to decentralized representations poses problems in its own right but also offers a number of substantial improvements.

4.2.4.2. Decentralized Representation

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 4.2.4.2. illustrates. 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.

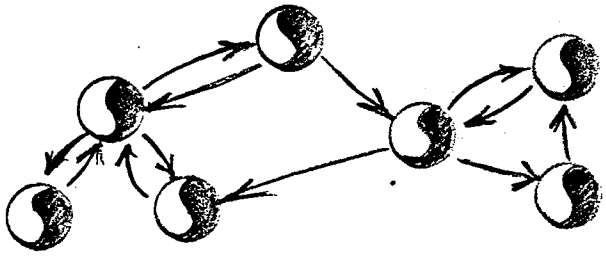


Figure 4.2.4.2. Schematic outline of decentralized representation of a situation. There are many interconnected agents, none of which is in a privileged position with respect to the others. Compare with figure 4.2.4.1.

With decentralized representations, the principal unit of analysis is the *meso-frame* (see subsection 3.3.3). Thus, micro-frames (i.e. individual

agents) can have only a few slots and yet it is possible to represent big situations. It is easier to add a new element — there is no need to 'register' it in the head.

In addition, decentralized representations are more natural for the connectionist aspect of DUAL. The representation of a situation integrates smoothly with the rest of the network. The associative mechanism can determine the relevance of the elements of a situation, activating only some of them on each particular occasion. There are many 'entry points' to the meso-frame, not just one (the head). The pattern of connectivity is evenly distributed among many agents, thus reducing the fan-out factor.

Meso-frames are emergent, flexible, and with fuzzy boundaries (subsection 3.3.3.). As there are no fixed and predefined representation rules, each particular situation can be described in a way that is most suitable for it and is not identical with other situations. Thus, it would be easier to design a perceptual mechanism that incrementally builds such representations.

Of course, all these advantages 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. In particular, the task of mapping becomes much more difficult. AMBR2 has developed a number of specialized features like secretaries, embryo hypotheses, etc. to cope with this problem.

4.3. Computational Mechanisms Used in AMBR2

This section describes the four basic computational mechanisms used in the model: spreading activation, marker-passing, constraint-satisfaction, and structure-correspondence. The next section shows how these mechanisms work together in a problem-solving task.

4.3.1. Associative Mechanism

> *One of the key factors in human intelligence is the ability to identify and to utilize the knowledge that is relevant to a particular problem.*
>
> (Anderson 1983, p.86)

The purpose of the associative mechanism is to determine the relevance of each particular piece of knowledge, bringing relevant pieces into the working memory (Anderson, 1983; Kokinov, 1994a). The associative mechanism in AMBR2 relies on the connectionist aspect of DUAL architec-

ture. The connectionist aspect is outlined in sections 3.2.4. and 3.4.2. It is not repeated here. This section is devoted to the basic formulas that are peculiar to AMBR2.

AMBR2 uses a modified version of the Grossberg activation function (Grossberg, 1978; Holyoak & Thagard, 1989). The modifications were made to meet the following requirements of DUAL's specification:

- Time is continuous. (Or, the length of one elementary connectionist cycle is negligibly small with respect to the macroscopic time scale.)
- Activation level of all agents is always non-negative and is bounded by some maximal value M.
- All links in the long-term memory are excitatory[5].
- There is a threshold θ 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 AMBR2 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.

The differential equation could be discretized using some small time step h:

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

When h is sufficiently small, the second equation can approximate the first with arbitrary precision.

In the special case of constant input $n = const$, the differential equation yields the following solution (*exp* is the exponential function):

[5] Therefore, it could also be said that AMBR2 uses a modified version of the function proposed by McClelland & Rumelhart (1981). The two functions are equivalent for non-negative inputs. See also section 4.3.3.5.

$$a(t) = \frac{E.n}{d+E.n} M.[1 - exp(-(d+En)(t-t_0))] + a_0.\ exp(-(d+En)(t-t_0))$$

Let us assume that the initial time is zero ($t_0 = 0$), and define the following terms:

$p = E/d$ — *maintenance factor* (cf. Anderson, 1983);

$a^* = \frac{E.n}{d+E.n} M = \frac{p.n}{1+p.n} M$ — asymptotic activation level for input n;

$T = \frac{1}{d+E.n}$ — *characteristic time* for net input n.

With the aid of the new terms, the activation function can be expressed in the following way:

$$a(t) = a^*.[1 - e^{-t/T}] + a_0.e^{-t/T}$$

The last equation reveals that the activation level at any given moment t can be partitioned in two components: *exogenous activation* and *residual activation* (cf. section 3.4.2.). The former reflects the external influence to the agent. The net input drives the activation level to some resting state a^* that depends on the magnitude of the input n, the parameter p, and the maximal activation level M. The agent, however, has a certain degree of inertness and, therefore, tends to keep its original state a_0. This inertness of the connectionist aspect of the agent gives rise to the residual activation in the equation above. The transition between the initial state a_0 and the resting state a* is governed by exponential law as shown in figure 4.3.1.

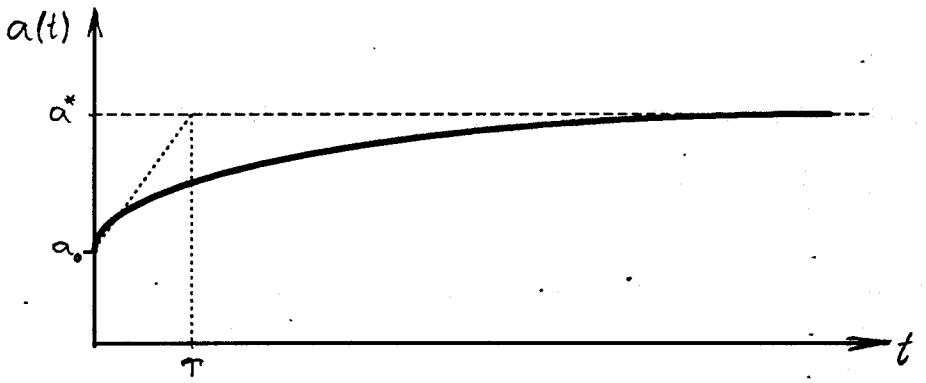


Figure 4.3.1. Plot of the change of activation of a DUAL agent under the influence of fixed external input. Activation moves from initial level a_0 to resting level a^* by an exponential function. The speed of the change is

described by the *characteristic time* T. Both a^* and T depend on the magnitude of the external input.

The function described above is the basis of the activation function of AMBR2 agents of type concept and instance. (Hypothesis-agents have a more complicated activation function described later.) There is one more complication, however — AMBR2 agents use a threshold. Whenever the activation drops below some predefined minimal level θ, it is instantaneously brought to zero (and 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 θ and then proceeds in the usual manner. The critical value n_θ is determined from the equation $\frac{p.n}{1+p.n}M = \theta$.

4.3.2. Marker-Passing Mechanism

Marker passing (MP) 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. Thus, each origin sets up a wave of markers that gradually expands until some *attenuation mechanism* stops the marking.

The attenuation mechanism is needed to restrict the spread of markers. It is one of the biggest issues in most marker-passing systems. (See Hendler (1988) for an overview.) In DUAL and AMBR, there is no need for a specialized attenuation mechanism. Instead, the spread of markers is controlled in a natural way by the following factors:

1. Markers originate only from agents of type instance. The conceptual agents do not create new markers, they only pass the existing ones.

2. Markers propagate only along links with certain labels (see below).

3. 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 associative mechanism.

4. When there is a *marker intersection*, the markers stop and do not propagate further.

5. The speed of symbolic processing in general and marker-passing in particular depends on the 'energy' supplied by the connectionist machinery (see section 3.2.5.3.). Thus, markers propagate slowly into those regions of the working memory where the activation level is low.

In AMBR2, like in its predecessor AMBR1 (Kokinov, 1994a), markers are passed only along links with labeled `inst-of` or `subc`. These are the links that go 'upward' in the class hierarchy. As a consequence, the marker-passing mechanism finds a path between two nodes in the network only when they are instances (directly or by inheritance) of one and the same concept. In AMBR2, this is interpreted as evidence that the two instances are semantically similar. In the example given in section 3.3.5., two markers originating from `teapot-1` and `glass-2` intersect at the conceptual node `liquid-holder`.

It is important to stress that the marker-passing mechanism is dynamic and context-dependent. Markers starting from the same two nodes A and B may intersect at different concepts (or not at all) depending on the activation levels of the corresponding path in the network. This makes the process of computing semantic similarity in AMBR dynamic and context-dependent (Kokinov 1994a,c).

4.3.3. Constraint-Satisfaction Mechanism

4.3.3.1. Main Points

The multiconstraint theory (Holyoak & Thagard 1989, 1995) treats analogy-making in the light of three constraints: structural, semantic, and pragmatic. AMBR2 adopts this general idea. Like ACME, AMBR uses a parallel connectionist algorithm for solving the constraint-satisfaction problem. This does not mean, however, that AMBR is a simple replication of ACME. AMBR1 (Kokinov, 1994a) makes several important contributions. Most notably, the constraint-satisfaction network (CSN) is constructed dynamically and is interleaved with the main network (i.e. the network storing conceptual and episodic information). In turn, AMBR2 makes several improvements with respect to its predecessor. They are: (*i*) decentralized representations, (*ii*) continuous marker-passing, (*iii*) *life cycle* of hypotheses, (*iv*) *secretaries*, (*v*) general and flexible treatment of symmetric relations, and (*vi*) change of the activation function.

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, a constraint satisfaction network is constructed by a sequential symbolic process. Finally, the network is allowed to settle, thus finding a (partial) solution to the constraint-satisfaction problem. This three-step process is illustrated in figure 4.3.3.1. The stages are carried out by different and independent mechanisms.

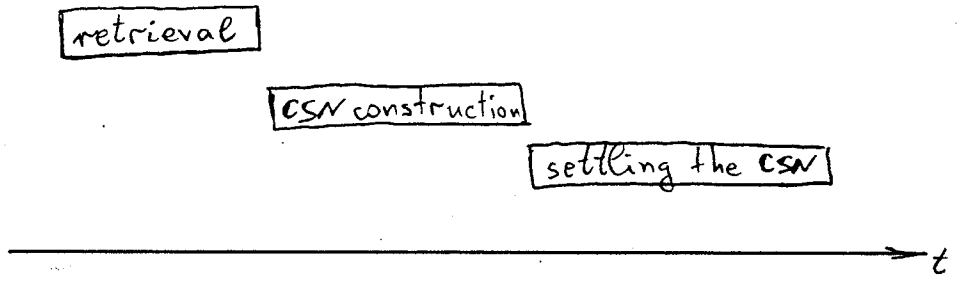


Figure 4.3.3.1. Constraint satisfaction as a three-stage process. The three stages come one after the other and do not interact.

In contrast, AMBR (Kokinov, 1994a) views constraint-satisfaction as a single integrated process that has three interacting subprocesses. The constraint-satisfaction net is constructed incrementally and in parallel with the retrieval subprocess. Moreover, the CSN is incorporated into the larger network and is continuously relaxing throughout the operation of the model. Thus, all subprocesses are evolving together, each one influencing the rest. The whole computation is performed in an integrated fashion — the same computational mechanisms are responsible for all three subprocesses and they use the same representational structures.

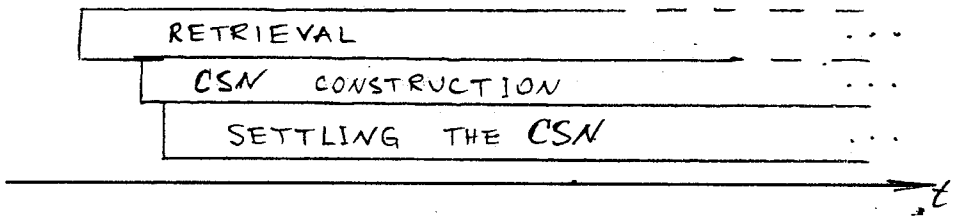


Figure 4.3.3.2. Constraint satisfaction as a set of three interacting subprocesses. Compare with figure 4.3.3.1.

This computational scheme has several important advantages:

- Mapping and retrieval are integrated.

- At any moment, the constraint satisfaction network is at least partially settled and thus the system always has some current (though incomplete) view of the problem it solves.

- 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 dynami-

cally determined, no possibilities are ignored a priory. This benefit is a direct consequence of the dynamic emergent computation that underlies AMBR's constraint satisfaction (section 4.1.2.).

- The flexibility of AMBR (and especially of AMBR2) is even greater because several mappings can proceed in parallel, thus allowing for complex analogies and blends when appropriate.

- There is no need for special nodes enforcing the semantic and pragmatic constraints. Rather, these constraints are mediated by ordinary concepts and instances in the main network.

4.3.3.2. Hypothesis Agents

The constraint satisfaction network in AMBR2 consists of *hypothesis-agents*. These agents have specific activation function and specialized symbolic processors. They interact with the agents in the main network and with one another in order to collectively carry out their task.

From declarative point of view, hypothesis-agents carry three 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 one situation) corresponds to the second element (from another situation). The activation level of the agent represents the degree of credibility that the model attributes to the hypothesis at the moment.

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 `teapot-1` corresponds to `glass-2` is that both are liquid holders. In AMBR, each hypothesis must have a justification. (This is one big difference between AMBR and ACME, which constructs hypotheses for all possible correspondences.)

There are two kinds of justifications: semantic and structural. A given hypothesis has semantic justification when its two elements are semantically similar. Such justifications are established by the marker-passing mechanism. Two instances are considered semantically similar when they have a common superclass (subsection 4.3.2.). Thus, `teapot-1` and `glass-2` are semantically similar because they both belong to the class of liquid holders. `Teapot-1` and `plate-1` are semantically similar too due to the common superclass `artefact`. On some occasions, AMBR2 can construct hypotheses between almost any two entities. This happens when the domains of the two situations are very remote and hence the marker-passing process finds an intersection in some very abstract node such as `object`, `relation`, etc. 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 and vice versa. Structural justifications are established by the structure-correspondence mechanism (subsection 4.3.4.).

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 `teapot-1<-->glass-2` could be justified by `liquid-holder` (semantic) and by `color-of-1<-->made-of-5` (structural). In AMBR2, this particular hypothesis will be represented in the following way:

```
teapot-1<-->glass-2:
  :type      (hypothesis temporary)
  :t-link    ((teapot-1<-->cup-4 -1.0)
              (teapot<-->glass  +0.5) )
  :slot1
    :c-coref  teapot-1
  :slot2
    :c-coref  glass-2
  :slot3
    :c-coref (liquid-holder color-of-1<-->made-of-5)
```

Figure 4.3.3.3. Example of a hypothesis-agent. It represents the hypothesis that `teapot-1` corresponds to `glass-2`. There are two justifications for this correspondence. Compare with figure 4.2.3.

Figure 4.3.3.3. shows only the symbolic aspect of the hypothesis. In addition, there is a connectionist aspect (as always in DUAL). The references to hypothesis elements and justifications are also links that transmit activation. In this way, the hypothesis participates in the process of spreading activation. It supports its elements and in turn is supported by them. There are also mutually excitatory links between a hypothesis and its justification(s).

Finally, there may be 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 and are used for the purposes of settling the constraint-satisfaction network only.

Temporary links with negative weights[6] 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

[6] By the way, these are the only links with negative weights in AMBR2.

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 AMBR2 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 access only to local information. 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.

4.3.3.3. 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 in which the agent is involved. Sometimes, we will use the term *boss* to refer to the other part of the entity agent, i.e. the part not related to correspondences. It should be remembered, however, that the secretary and the boss are two faces of one and the same micro-agent.

The job of a secretary is twofold: it keeps record of correspondences involving its boss and it handles *hypothesis-registration requests*. To that end, each entity-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 that have the entity-agent as element. The same references are used as links that transmit activation from the agent (e.g. teapot-1) to its hypotheses (e.g. teapot-1<-->glass-2 and teapot-1<-->cup-4).

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 exchanged between AMBR agents in the course of their interaction. Each of the two secretaries receives the request and sends a *secretary answer* to the hypothesis. There are several kinds of answers but basically all they 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 more than one justifications for a given correspondence. For example, the marker-passing mechanism could construct the hypothesis teapot-1<--> glass-2 on the grounds that both are liquid holders. Later on, the struc-

ture-correspondence mechanism could independently construct the same hypothesis on the grounds that `teapot-1` and `glass-2` 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 `teapot-1<-1->glass-2`. When it tries to register at the secretary of `teapot-1`, 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 — `teapot-1<-->glass-2` — 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-agent in favor of which to resign. Answers of type 'Establish' carry a (possibly empty) list of references to rival hypotheses.

4.3.3.4. 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'être. It may be a duplicate of an existing hypothesis.

Therefore, AMBR2 distinguishes two types of hypothesis-agents: *embryos* and *mature* hypotheses. 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. In more detail, the life cycle is the following:

The main rule for hypothesis construction in AMBR2 is that each hypothesis must have a justification. There are two possibilities for construction of a hypothesis-agent: by the marker-passing and by the structure-correspondence mechanism.

The marker-passing mechanism creates hypotheses on the basis of semantic similarity. When two markers starting from different origins (e.g. `teapot-1` and `glass-2`) intersect at some conceptual node (e.g. `liquid-holder`) the latter detects the marker intersection and initiates the process of hypothesis-construction. It recruits one of the specialized *node constructors* (see section 3.2.6.1.) and sends a *node-construction request* ■ describing the 'specifications' of the new hypothesis. The node constructor constructs a new embryo hypothesis and fills in its slots. The elements of the new hypothesis are the origins of the two markers; the justification is the concept where the markers intersected.

The structure-correspondence mechanism creates new hypotheses on the basis of some existing hypothesis (see section 4.3.4.). In such cases, the old hypothesis becomes justification of the new one. Again, new agents are constructed by sending specific request to a node constructor.

One way or the other, the new embryo hypothesis is created 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 the hypotheses in AMBR2.

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 from the secretaries.) For fair play, the new agent sends its reference to all competing hypothesis, prompting them to establish symmetric inhibitory links.

4.3.3.5. 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 (section 3.4.) of agents that cooperatively solve a constraint-satisfaction problem. Each agent in the network represents a particular correspondence between two elements. The CSN is integrated with the main network that stores conceptual and episodic information in AMBR2. Thus, they become complementary parts of the big integrated network 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 (e.g. `teapot-1`) to the respective hypothesis-agents (e.g. `teapot-1<-->glass-2`). These links are excitatory and are stored in `hypoth` slots of entity-agents.

2. LTM—>CSN: Links from concept-agents (e.g. `liquid-holder`) to the hypothesis-agents that are 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 and semantic justifications. 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, AMBR2 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 can influence the pattern of activation in the main network and hence everything in the architecture. This fact has important implications for the integration between analogical mapping and retrieval.

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* (see section 3.2.4.1.). 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.

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 have non-negative activation functions. Therefore, AMBR2 uses a linear transformation of hypothesis' activation. In this way, even hypotheses that are considered implausible have positive activation levels. The parameters of the model can be chosen in such a way that only the most suppressed hy-

potheses fall below the general working-memory threshold and are thus purged out of the system. Figure 4.3.3.5. illustrates.

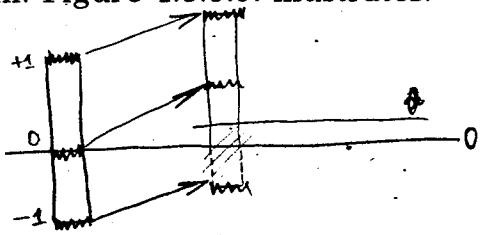


Figure 4.3.3.5. Schematic illustration of the linear transformation that makes hypothesis activation function positive. Only the most implausible hypotheses fall below the threshold θ. See text for details.

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 some embryo hypotheses resign shortly after their creation. If they do not resign, however, they become mature and their output function changes. Mature hypotheses have a threshold output function. That is, they influence their neighbors only if their (unmodified) activation level is positive.

4.3.3.6. An Example

As an example of the mechanisms discussed so far, and in preparation for the structure-correspondence mechanism that comes next, this section provides an excerpt of the transcript of an actual AMBR2 run. Only the events related to teapots and liquid holders are shown. Agent names are prefixed by '#$'. '#<M ...>' denotes a marker, '#<NCR ...>' — a node-construction request, '#<HR ...>' — a hypothesis-registration request, and '#<SA ...>' denotes a secretary answer.

The example illustrates construction of a hypothesis by the marker-passing mechanism, followed by secretary inquiries. The story begins by attaching the instance-agent `teapot-4` to the input list. The activation spreads from there and brings the concepts `teapot` and `liquid-holder` to the working memory together with a few other instances. Whenever an instance-agent enters the WM, it emits a marker (section 4.3.2.). These markers propagate through the network and the first marker intersection happens at time 3.10 :

```
At time 0.00, adding #$teapot-4 to the input list.
At time 0.20, adding #$teapot to WM.
At time 0.20, #<M1 TEAPOT-4> received in the input zone of #$teapot-4.
At time 0.20, #$teapot-4 begins working on #<M1 TEAPOT-4>.
At time 0.30, #<M1 TEAPOT-4> received in the input zone of #$teapot.
At time 0.40, adding #$teapot-1 to WM.
At time 0.40, adding #$liquid-holder to WM.
At time 0.40, #$teapot begins working on #<M1 TEAPOT-4>.
```

```
At time 0.80, #<M1 TEAPOT-4> received in the input zone of #$liquid-
              holder.
At time 1.00, #$liquid-holder begins working on #<M1 TEAPOT-4>.
At time 1.90, adding #$teapot-3 to WM.
At time 2.00, #<M0 TEAPOT-1> received in the input zone of #$teapot-1.
At time 2.10, #$teapot-1 begins working on #<M0 TEAPOT-1>.
At time 3.00, #<M0 TEAPOT-1> received in the input zone of #$teapot.
At time 3.10, #$teapot begins working on #<M0 TEAPOT-1>.
At time 3.10, #<M1 TEAPOT-4> and #<M0 TEAPOT-1> intersected at #$teapot.
```

Thus, the associative mechanism has retrieved some information related to `teapot-4` and the marker-passing mechanism has detected that `teapot-4` and `teapot-1` are semantically similar. (The names of the agents play no role in this process.) On the grounds of this semantic similarity, the agent `teapot` recruits a node-constructor and sends a node-construction request to it:

```
At time 3.40, #<NCR TEAPOT> received in the input zone of #$nc3.
At time 3.50, #$nc3 begins working on #<NCR TEAPOT>.
At time 4.20, creating a new agent: #$teapot-4<-->teapot-1
At time 4.20, adding #$teapot-4<-->teapot-1 to WM.
At time 4.20, adding a :T-LINK link from #$teapot to #$teapot-4<-->teapot-
              1 with weight 0.100
At time 4.60, #<M0 TEAPOT-3> received in the input zone of #$teapot-3.
At time 4.70, #$teapot-3 begins working on #<M0 TEAPOT-3>.
```

A new embryo-hypothesis named `teapot-4<-->teapot-1` was born. It has resulted from the joint effort of a coalition of agents: `teapot-4`, `teapot-1`, `teapot`, and `nc1`. Meanwhile, the associative and marker-passing mechanisms continue to work, preparing the ground for a rival hypothesis involving `teapot-3`.

We will focus on the first hypothesis, however. It receives activation from its parent (note the `t-link` at time stamp 4.20) and thus has the energy necessary for its symbolic processor. The embryo sends hypothesis-registration requests to the secretaries of `teapot-1` and `teapot-4`. They check their record, notice that this is the first hypothesis that registers, and reply positively[7]. This is good news for `teapot-4<-->teapot-1`. It becomes a mature hypothesis at time 6.40:

```
At time 5.40, #<HR TEAPOT-4<-->TEAPOT-1> received in the input zone of
              #$teapot-1.
At time 5.50, #$teapot-1 begins working on #<HR TEAPOT-4<-->TEAPOT-1>.
At time 5.60, #<HR TEAPOT-4<-->TEAPOT-1> received in the input zone of
              #$teapot-4.
At time 5.70, #$teapot-4 begins working on #<HR TEAPOT-4<-->TEAPOT-1>.
At time 5.70, #<SA nil> received in the input zone of #$teapot-4<--
              >teapot-1
At time 5.80, #$teapot-4<-->teapot-1 begins working on #<SA nil>.
At time 6.00, adding a :HYPOTH link from #$teapot-4 to
              (#$teapot-4<-->teapot-1 . :SLOT1) with weight 0.100
At time 6.30, #<SA nil> received in the input zone of #$teapot-4<--
              >teapot-1
At time 6.40, #$teapot-4<-->teapot-1 begins working on #<SA nil>.
At time 6.40, establishing hypothesis #$teapot-4<-->teapot-1.
```

[7] Despite the appearance, a secretary answer of the form `#<SA nil>` is a positive answer. `NIL` means in this case that the list of rival hypotheses is empty.

As it has reached maturity now, teapot-4<-->teapot-1 starts generating new hypotheses. In this particular case, the structure-correspondence mechanism (section 4.3.4.) will propose the hypothesis teapot<==> teapot on the grounds that if two instances correspond, their respective concepts should also correspond.

Note that the structure-correspondence mechanism runs in parallel and in implicit competition with marker-passing. The latter generates an alternative hypothesis at time 8.00. Meanwhile, the associative mechanism proceeds in the background, bringing the concept glass to the working memory. Glasses are liquid holders, so one can expect hypotheses mapping teapots to glasses later on. At this point, however, AMBR2 gives preference to tentative correspondences between two teapots. If for some reason these hypotheses prove inappropriate, alternative candidates (e.g. about glasses) could potentially supplant them. If, however, the initial hypotheses turn out adequate, little resources will be spent on exploring alternatives. This illustrates the idea of dynamic emergent computation and its utility for constructing models that are efficient and flexible in the same time (section 4.1.2.).

```
At time 6.70, adding #$glass to WM.
At time 6.80, #<M0 TEAPOT-3> received in the input zone of #$teapot.
At time 6.90, #$teapot begins working on #<M0 TEAPOT-3>.
At time 6.90, #<M1 TEAPOT-4> and #<M0 TEAPOT-3> intersected at #$teapot.
At time 6.90, #$teapot-4<-->teapot-1 begins bottom-up SC.
At time 7.20, #<NCR TEAPOT> received in the input zone of #$nc8.
At time 7.30, #$nc8 begins working on #<NCR TEAPOT>.
At time 8.00, #<NCR TEAPOT-4<-->TEAPOT-1> received in the input zone of
              #$nc5.
At time 8.00, creating a new agent: #$teapot-4<-->teapot-3
At time 8.00, adding #$teapot-4<-->teapot-3 to WM.
At time 8.00, adding a :T-LINK link from #$teapot to #$teapot-4<-->teapot-
              3 with weight 0.100
At time 8.10, #$nc5 begins working on #<NCR TEAPOT-4<-->TEAPOT-1>.
At time 8.20, adding a :HYPOTH link from #$teapot-1 to
              (#$teapot-4<-->teapot-1 . :SLOT2) with weight 0.100
At time 8.80, creating a new agent: #$teapot<==>teapot
At time 8.80, adding #$teapot<==>teapot to WM.
...
```

4.3.4. Structure-Correspondence Mechanism

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*.

4.3.4.1. 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 as one of the fillers of

their type slots (see section 4.2.1.). Under these circumstances, the symbolic processor of the hypothesis chases the inst-of links of the two instance-agents and retrieves their respective concepts. For example, if the two instances are teapot-1 and glass-2, the concept-agents will be teapot and glass, respectively. Then, the original hypothesis initiates a process for constructing a supplementary hypothesis stating a parallel correspondence between the two concepts. The new hypothesis is constructed in the usual way — by sending request to one of the node-constructors in the system. 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 teapot-1<-->glass-2 generates the supplementary hypothesis teapot<-->glass. After a while, another hypothesis, e.g. teapot-3<--> glass-2 constructs the same 'conceptual' hypothesis. In such cases, the duplication is detected by the secretaries and the second 'conceptual' hypothesis resigns in favor of the first. Eventually, teapot<-->glass 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 constraint-satisfaction network 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.

There is a second variant of the mechanism for bottom-up SC. It is very similar to the variant described above except that it deals with the *situations* to which the instances belong rather than with their classes.

In AMBR2, a target situation is mapped to several different bases simultaneously. For concreteness, suppose that there are two bases denoted B1 and B2, and a target T. Suppose further that the elements of B1 are a, b, c; that of B2 — m, n, p, q; and of T — x, y, z. Then, correspondences of the form a-z, c-x, b-y, etc. will support B1-T; while m-x, q-y, and m-z will support B2-T (and vice versa).

This second variant of the bottom-up SC creates a pressure that situations are mapped as units. Blends are possible but they happen only when are truly warranted. Normally, the model tries to keep the mapping within the scope of two situations only: the target and a single base.

To that end, AMBR2 has added a new slot label to the vocabulary of general-slot labels in the architecture (section 3.2.3.4). If a micro-agent possesses a slot labeled situation, this indicates that the agent belongs to

the situation (or episode, schema, problem) denoted by the filler of this slot.

The introduction of `situation` slots does not compromise the notion of *decentralized representations* discussed in section 4.2.4.2. Yes, there is an agent of `type situation` which is referenced by `situation` slots of individual members. This agent, however, does *not* represent the situation as a whole. Rather, it serves as a common reference to the spatio-temporal unity of the elements. It incarnates the fact that these particular objects and relations happened to be observed together within relatively short period of time.

The situation agent need not have links pointing 'downward' to the agents representing individual objects and propositions. (It could have links to some of them but this is not necessary.) The links are in the opposite direction: from individual elements to the situation agent. Moreover, these links are optional — it is possible to have a 'free-standing' element that does not belong to any situation (or belongs somewhere but does not 'know' that). Such agent will not have a `situation` slot and will not be affected by the mechanism for bottom-up structure-correspondence.

4.3.4.2. 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, 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 illustrated in figure 4.2.3. This greatly facilitates the structure-correspondence mechanism and relieves the model of implausible assumptions. Moreover, it makes possible to map propositions with different number of arguments (Kokinov, 1994a; Hummel & Holyoak, 1997).

The details of the top-down structure correspondence in AMBR2 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 (section 4.2.1.). If this is the case, the symbolic processor attempts to determine the slot-to-slot correspondences. To that end, 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`, `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 such 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 when this is true. Otherwise, it gives up and stops, hoping that the MP mechanism will provide the missing information later.

4.4. The Mechanisms at Work

This final section of the chapter devoted to AMBR2 pulls everything together and shows how the computational mechanisms described above can be applied to the task of analogy-making[8].

4.4.1. General Sequence During a Run

In the present version of AMBR2, 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 effect of the external context. It is possible (though not experimented yet) that target elements are presented to the system not simultaneously but incrementally. In this way, various order effects can be demonstrated (Keane et al., 1994).

[8] It is argued (Kokinov, 1994a) that the same mechanisms can be applied to other tasks as well. This, however, is beyond the scope of the present thesis.

Once the target elements are connected to the source nodes, the associative mechanism begins to operate. The activation spreads through the long-term memory of the system 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. In this way, all mechanisms in the model are put to work. They operate in parallel and in tight cooperation with one another.

Gradually, a number of agents enter the working memory. The activation does not spread unrestricted, however, and the intensity of memory retrieval 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 constraint-satisfaction network has thus become fairly elaborate and winning correspondences begin to emerge. The hypotheses standing for such correspondences become highly active and provide strong support for the respective entities in the main network. In this way, the base situations that best satisfy the constraints are fully and unambiguously retrieved.

Sooner or later, the system approaches a resting state. The pattern of activation stabilizes. The markers cover all the 'territory' that has been made available by the associative mechanism and, therefore, no more marker intersections are reported. The structure-correspondence mechanism also stops. Finally, the relaxation of the constraint-satisfaction network completes and the 'answer' of the system can be read from the activation levels of winning hypotheses. (In fact, the system maintains a 'working answer' throughout the whole run. It is often unnecessary to wait for the end.)

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 analogical retrieval, mapping, etc.

4.4.2. Integration Revisited

At the outset of this thesis, the notion of integration was declared to be the prime mover of DUAL and AMBR. It is instructive to take a look back from the point of view of the closing section. Integration is everywhere! The list below traces its manifestations in the architecture and the model.

- DUAL has two aspects — connectionist and symbolic — and they are integrated within and across all levels of the architecture.
- DUAL has two more aspects — declarative and procedural — and they are integrated within and across all levels of the architecture.
- All knowledge in the architecture is represented within a unified framework that takes into consideration all aspects cited above.
- All information-processing in the architecture operates according to unified principles that take into account all aspects cited above.
- AMBR has a handful of interlocking mechanisms which cooperatively solve the problems within its scope.
- AMBR offers a unified account of analogical retrieval and mapping.

CHAPTER V

SIMULATION EXPERIMENTS

This section reports the results of the simulation experiments performed with AMBR2. The general setting of the experiments is presented first. Then, one particular run of the program is considered in some detail. Finally, there are aggregate statistics for the performance of the model over 600 runs.

5.1. Experimental Setting

DUAL is a deterministic architecture. The behavior of a DUAL-based model (such as AMBR2) depends on three factors: (*i*) the contents of the long-term memory of the system, (*ii*) the target problem, and (*iii*) the external context. Kokinov (1994a) reports an experiment exploring the influence of the third factor — one target problem was solved several times with different preliminary settings and different agents attached to the input list. The experiments reported here concentrate on the first two factors — six target problems were run on one hundred knowledge bases, yielding a total of 600 runs.

5.1.1. Materials

The problems presented to the system belong to the domain of heating or cooling liquids. Each problem consisted of a simple situation. There was a total of 11 situations. 5 of them (denoted A, B, C, D, E) were used as source analogs:

Situation A: *There is a teapot and some water in it. The teapot is made of metal and its color is black. There also is a plate and the teapot is on it. The plate is hot. This state of affairs causes that the water becomes hot.*

Situation B: *There is a bowl and some water in it. The bowl is made of wood and is placed on a fire. The fire is hot. This state of affairs causes that the bowl burns out and the water dissipates.*

Situation C: *There is a glass and some water in it. The glass is made of glass. There is an immersion heater in the water. The immersion heater is hot. This state of affairs causes that the water becomes hot.*

Situation D: *There is a glass and some water in it. The glass is made of glass and its color is white. The glass is on a plate and the plate is hot. This state of affairs causes that the glass breaks and the water dissipates.*

Situation E: *There is a teapot and some milk in it. The teapot is made of metal and its color is green. There also is a refrigerator and the teapot is in it. The refrigerator is cold. This state of affairs causes that the milk becomes cold.*

These situations were coded by hand according to the knowledge representation scheme used in AMBR2 (section 4.2.) and included into the long-term memory of the system. The representation of one situation took 13-16 DUAL agents.

In addition to that episodic knowledge, some semantic knowledge was put into the LTM too. For example, it was represented that `hot`, `warm`, and `cold` are instances of the class `temperature-qualifier`; that `in` and `on` are spatial relations while `temperature-of` and `made-of` are physical relations; that teapots are usually made of metal, etc. Some associative links were added too. About 100 agents were used to represent this information.

Thus, the long-term memory of the system contained the 5 base situations and the semantic knowledge outlined above. This knowledge base was multiplicated one hundred times, with small variations in each case. The algorithm for generating the variants was chosen to emulate the outcome of the mechanism for connecting concepts to some of their instances proposed in section 4.1.3.4. Each knowledge base contained the same agents with some additional links. Most of these links were `instance` links from selected concept-agents to some of their respective instance-agents (selected at random). For example, in some particular variant of the knowledge base the concept `teapot` could be connected to the teapot participating in situation A (or E, or none of them). The remaining links were associative links between selected agents. Each of these links occurred with some small probability.

The variants of the knowledge base generated in this way served as replications in the experiment. All parameters of the model were kept constant across all runs. Each target problem was run on each knowledge base. Thus, it was possible to obtain a frequency distribution showing how many times a particular target situation was mapped to a particular episode from the LTM. The target situations were chosen in such a way that they can be plausibly mapped to several base analogs. The six target situations used in the experiment are outlined below:

Situation X: *There is a bowl and some water in it. The bowl is made of wood. The goal is that the water becomes hot.*

Situation Y1: *There is a teapot and some milk in it. The teapot is made of metal. The goal is that the milk becomes hot.*

Situation Y2: *There is a teapot and some milk in it. The teapot is made of metal and its color is green. There also is a hot plate and the teapot is on the plate. What will happen?*

Situation Z: *There is a glass and some water in it. The glass is made of glass and its color is white. The goal is that the water becomes hot.*

Situation U1: *There is a teapot and some milk in it. The teapot is made of metal. The goal is that the milk becomes cold.*

Situation U2: *There is a teapot and some milk in it. The teapot is made of metal and its color is green. There also is a cold refrigerator and the teapot is in the refrigerator. What will happen?*

5.1.2. Dependent Variables

A lot of data were recorded during each run. The data included the number of agents of different kinds in the working memory, the number of marker intersections, the number of agents whose symbolic processor is active, the total activation of the working memory, etc. In addition, there was a number of indices for each situation, and in particular the retrieval and mapping indices described below.

Due to the decentralized representation of situations (section 4.2.4.2), it frequently happens that only part of their members pass the threshold and enter the working memory. In addition, the agents in the WM have different activation levels. The *retrieval index* is intended to serve as an overall measure of the extent to which a given situation is present in the working memory. It is computed by the following formula:

$$R = \frac{\sum a_i}{n+0.5},$$

where the summation is taken over all members of a given situation, n is the total number of such agents, and a_i is the activation level of agent$_i$.

A complementary *mapping index* is defined to measure the strength of the mapping between two situations. It is computed on the basis the activation level of the relevant hypothesis-agents (section 4.3.3.2.). Only mature hypotheses are considered. The mapping index can be computed for each situation in a pair (the two values will generally not be the same). For example, consider the mapping between situations A and X. The mapping index from X to A is computed in the following way: (*i*) the total pool of all hypotheses is filtered, keeping only those mature hypotheses that relate an element from X to an element from A; (*ii*) these hypothesis-

agents are sorted according to their activation levels; (*iii*) for each element from X, the hypothesis with highest activation level is chosen; (*iv*) the mapping index is computed by the following formula:

$$M = \frac{\sum h_j}{n+0.5},$$

where the summation is taken over all relevant hypotheses for the particular pair of situations, n is the total number of elements of the first situation (in the example above, this would be situation X), and h_j is the decoded (see section 4.3.3.5.) activation level of hypothesis$_j$.

Each of these variables was recorded fifteen times at regular intervals during each run. This captures the dynamics of the operation of AMBR2.

5.2. A Case Study

This section presents the result of one particular run of the program. During that run, situation Y1 (boiling milk in a teapot) was used as target situation. The agents representing the goal of the problem — that the milk should become hot — was attached to the goal node of AMBR2. Some of the remaining members of the situation — the teapot, the relation that it is green, the relation that the milk is in the teapot, etc. — were attached to the input node. Then, the system was run for 150 units of simulated time, recording the values of the dependent variables at the end of each 10-unit interval.

Three of the five episodes stored in the LTM were retrieved in this case — situations C, D, and E. The retrieval indices for situations A and B stayed zero (or almost zero) throughout the run. Figure 5.2.1. plots the retrieval indices for situations C, D, and E as a function of time.

The inspection of this graph reveals that situation E initially takes precedence over its rivals. This is due to the fact that it is the only situation among the bases that involves milk, and the concept `milk` is highly active because the target problem also involves milk. In addition, situation E involves a teapot (also present in the target), while the liquid holders in situations C and D are glasses. Thus, situation E is most similar to the target if only objects are considered. Therefore, this situation is in advantageous position early in the run — when the retrieval dominates the mapping.

On the other hand, the concepts `hot` and `temperature-of` become highly active because the goal of the target problem is to heat the milk. This pragmatic pressure, implemented by the flow of activation emitted by the goal node, brings situations C and D to the working memory. Situation A (heating water on a plate) does not get activated because in

this particular knowledge base it happens to be no link from the concepts hot, temperature-of, etc. to the instances of this situation (see section 4.1.3.4.).

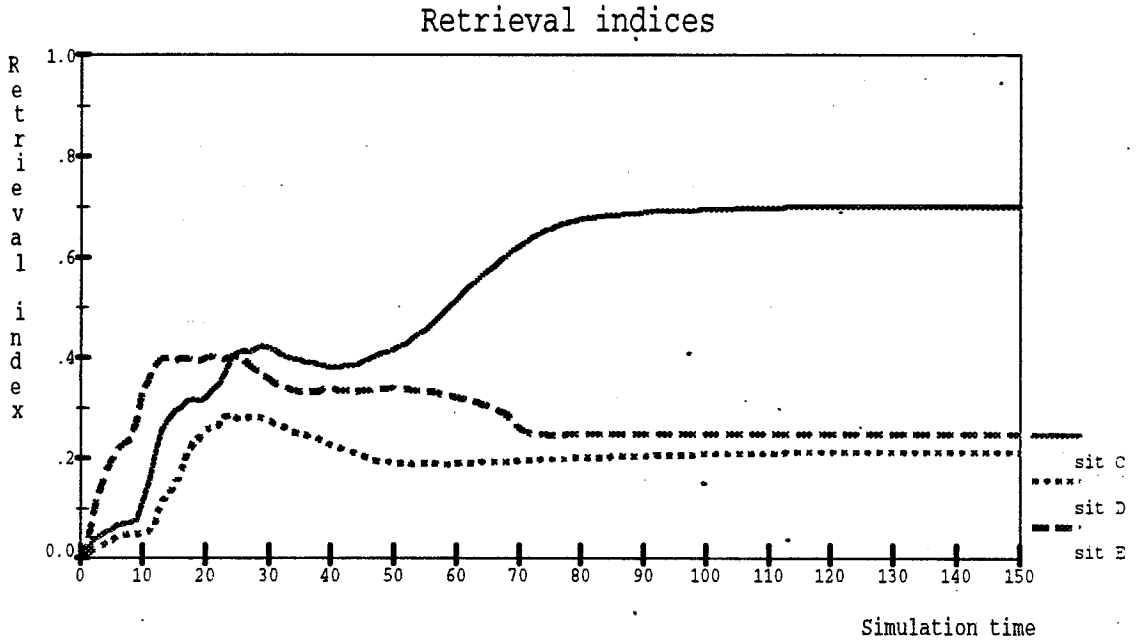


Figure 5.2.1. Retrieval indices for situations C, D, and E as a function of time. At first, situation E takes the lead, but the final winner is situation C. The crossing of the two lines at time 25 is indicative of the interaction between retrieval and mapping in AMBR2.

Meanwhile, the process of mapping gathers momentum. Each of the three episodes that have been retrieved from long-term memory is trying to map itself to the target. The dynamics of this process can be seen at figure 5.2.2. which shows the mapping indices for the three situations as a function of time.

Note the spike at time 12. It reflects the early period of the construction of the constraint-satisfaction network. During this early period, the marker-passing mechanism has constructed some hypothesis-agents but they have not yet established the mutually inhibitory links that mediate the one-to-one constraint on mapping (section 4.3.3.5.). After the construction of the inhibitory links, all mapping indices decrease a bit.

The bulk of the mapping process happens between time stamps 20 and 40. During this period, the three analogs compete with each other. Note the small jerks on the plots. They reflect the small changes in the mapping indices induced by the introduction of new hypotheses to the CSN. If all

hypotheses in the network were created beforehand (as in ACME), the plots would have been smooth.

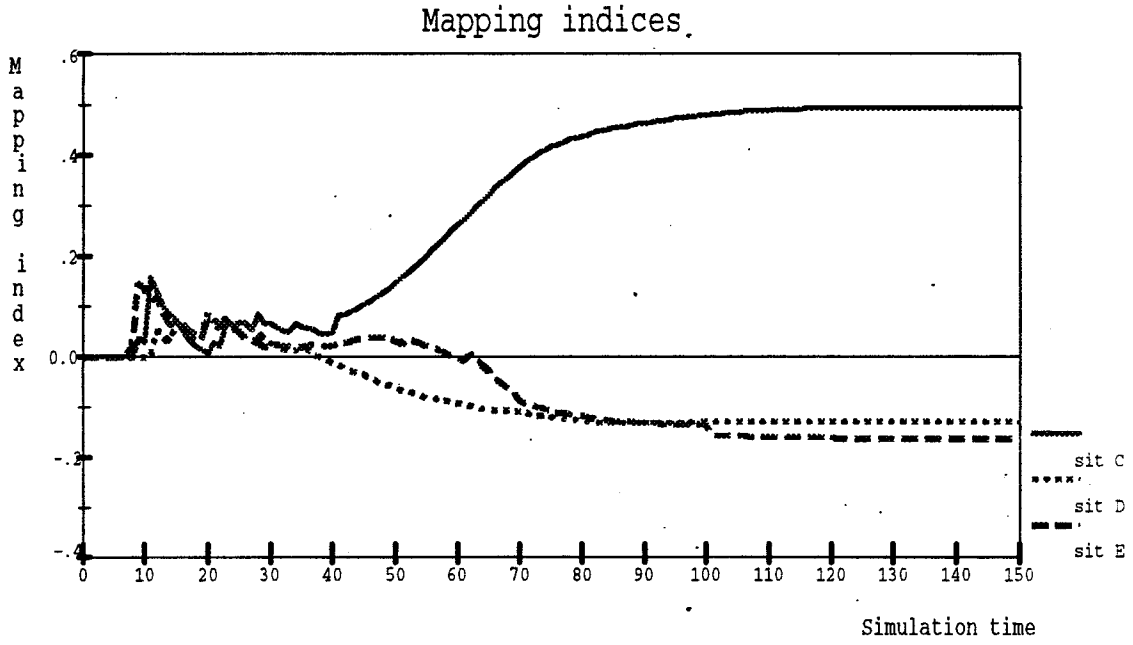


Figure 5.2.2. Mapping indices for situations C, D, and E. See text for details.

It is especially instructive to compare the two plots, keeping in mind that AMBR2 is an integrated model of retrieval and mapping. Indeed, figure 5.2.1. reveals that the trends in retrieval indices are reversed somewhere between time stamps 20 and 40. This is exactly the period when the mapping process is most intensive, as evident from figure 5.2.2.

The analysis of figures 5.2.1. and 5.2.2. can be summarized as follows: Initially, the retrieval process gives preference to situation E due to its greater semantic similarity with the target. The pragmatic and structural constraints, however, eventually overcome this trend. Somewhere at time 40 the winner becomes clear — it is situation C. After that time stamp, the winner gradually assumes its power. At time 70, the system has virtually reached asymptotic state.

Here are the hypotheses that emerge as winners at time 70:
sit-y1<-->sit-c, teapot-y1<-->glass-C, milk-Y1<-->water-C, made-of-Y1<-->made-of-C, mmetal-Y1<-->mglass-C, in-Y1<-->in-C, init-state-Y1<-->init-state-C, end-state-Y1<-->end-state-C, cause-9<-->cause-C, temper-of-Y1<-->temper-of-C, hot-Y1<--> hot-C; situation<-->situation, teapot<-->glass, milk<-->water, made-of<-->made-of, material-metal<-->material-glass, in<-->

in, init-state<-->init-state, end-state<-->end-state, cause<--> cause, temperature-of<-->temperature-of, hot<-->hot.

5.3. Summary Over All Simulations

The previous section paid great attention to one particular tree but neglected the forest. This section takes the opposite tack. It presents the aggregate frequencies over all runs.

Each of the six target problems was run on each of the 100 knowledge bases for 150 time units. This time was enough for almost all runs to reach asymptote. At the end of each run, it was cycled through the target elements, collecting the correspondence with highest activation for each one. The base situation which had the most elements collected in this way was declared as a 'winner'. Thus, a winner was identified for each run. The frequency distribution on targets versus winners is reported in table 5.1.

	A	B	C	D	E	Ambig.	Total
X	12	73	9	3	3		100
Y1	55	1	2	1	38	3	100
Y2	79	1	1		19		100
Z	29	6	38	23	4		100
U1	26		1	1	72		100
U2	13		2		85		100
Total	214	81	53	28	221	3	600

Figure 5.1. Frequencies of mappings between target situations (rows) and source analogs (columns).

Table 5.1. reveals that AMBR2 constructs reasonable mappings in most of the cases. More concretely, problems about cooling milk (U1 and U2) tend to be mapped to situation E, which can be interpreted as a literal similarity match (Gentner, 1983). In about 20% of the cases in the last two rows, however, AMBR2 maps `milk` to `water` and `cold` to `hot`, thereby constructing an analogy. (The fact that this analogy fails to solve the problem is another matter.) Problems about heating milk (Y1 and Y2) show the reverse pattern. They tend to be mapped to situation A — the prototypical case for heating liquids.

Note that the frequency of mapping Y1 or Y2 to A is smaller than that of mapping U1/U2 to E (55 or 79% vs. 72 or 85%, respectively). This reflects the influence of the similarity constraint. During both retrieval and mapping, this constraint prefers correspondences of the form `milk<-->milk` to those of the form `milk<-->water`. It is remarkable that AMBR2, which is an entirely deterministic model, is capable of producing such frequency distributions when the sole random factor in the experiment is the variation of the knowledge base.

Table 5.1. also illustrates the fact that AMBR2 visits the implausible regions of the 'problem space' with some small probability. Few cells in the table are completely empty (the 'ambiguous' column notwithstanding) thus showing that no possibilities are ruled out *a priory*. AMBR2 generates 'promising' solutions most of the time and yet occasionally goes 'off the beaten track' (see section 4.1.2.).

CHAPTER VI

CONCLUSION AND PLANS FOR THE FUTURE

This final chapter summarizes the main points of the thesis, outlines the contributions of the project, and presents some ideas about future research on DUAL and AMBR.

6.1. Summary of the Thesis

This thesis has presented the work done so far on the DUAL architecture and the AMBR2 model. Special attention has been paid to the notion of integration as a central theme of DUAL research.

Chapter III constitutes a specification of DUAL. The conceptual framework of the architecture is documented in enough detail to serve as a solid foundation for future research. In addition, there is a portable computer program (written in Common Lisp) that fully implements DUAL as defined by this specification.

Chapter IV describes AMBR2 — a cognitive model built on the basis of DUAL. In its current state, it is presented as an integrated model of analogical retrieval and mapping. The knowledge-representation scheme and the computational mechanisms used by the model are documented in detail. In addition, the model is discussed from a psychological point of view.

There is a portable computer program that fully implements AMBR2 as documented in this thesis. This program was used for performing simulation experiments with the model. The results of these experiments are reported in Chapter V.

6.2. Contributions of This Work

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

- 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 representations* of the situations in AMBR. In turn, this led to improvements in the marker-passing, structure-correspondence, and constraint-satisfaction mechanisms.

- Introduction of *secretaries* for the purposes of incremental construction of the constraint-satisfaction network.

- 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 portable computer implementations of the architecture and the model.

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

6.3. Plans for the Future

Each end is a new beginning.
(Bulgarian proverb)

As stated in the introduction, the project reported here is part of a big and ambitious research program. There are a number of open avenues for future work on the architecture and the model. They include:

- Elaboration of the mechanisms of AMBR. In particular, adding support for analogical transfer and evaluation.

- Testing the model on new kinds of problems. Experimenting with considerably larger knowledge bases (scaling up).

- Sensitivity analyses with the model. Exploring the effects of selected 'lesions' of its mechanisms.

- Comparing AMBR with other models of analogy-making.

- Performing psychological experiments to test the predictions of the model.

- Adding perceptual mechanisms to the architecture. The PEAN project is an effort in this direction.

- Adding learning mechanisms to the architecture.

Bibliography:

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

Anderson, J. (1993). *Rules of the Mind*. Hillsdale, NJ: LEA.

Anderson, J. & Bower, G. (1979). *Human Associative Memory*, Hillsdale, NJ: LEA.

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

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

Castelfranchi, C. & Werner, E. (1994). *Artificial Social Systems*. Berlin: Springer-Verlag.

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)*, pp. 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)*, pp. 171-190.

Cohen, P. (1995). *Empirical Methods for Artificial Intelligence*. Cambridge, MA: MIT Press.

Dunker, K. (1945). On problem solving. *Psychological Monographs 38 (No. 270)*.

Fahlman, S. (1979). *NETL: A System for Representing and Using Real World Knowledge*. Cambridge, MA: MIT Press.

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.

Faries, J. & Reiser, B. (1988). Access and use of previous solutions in a problem-solving situation. *Proceedings of the Tenth Annual Meeting of the Cognitive Science Society*, pp. 433-439. Montreal, Hillsdale, NJ: LEA.

Feldman, J. & Ballard, D. (1982). Connectionist models and their properties. *Cognitive Science, 6*, pp. 205-254.

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*, pp. 141-205.

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)*, pp. 155-170.

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

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

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

Gilbert, N. & Doran, J. (1994). *Simulating Societies*. London: UCL Press.

Grossberg, S. (1978). A theory of visual coding, memory, and development. In Leeuwenberg, L. & Buffart, J. (eds.) *Formal Theories of Visual Perception*. New York, NY: Wiley.

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

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

Hendler, J. (1988). *Integrating Marker-Passing and Problem-Solving: A Spreading-Activation Approach to Improved Choice in Planning*. Hillsdale, NJ: LEA.

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

Hewitt, C. (1985). The challenge of open systems. *Byte Magazine 10 (4)*, pp. 223-242.

Hoare, C. (1985). *Communicating Sequential Processes*. Englewood Cliffs, NJ: Prentice-Hall.

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 Holyoak, K. & Barnden, J. (eds.) (1994). *Advances in Connectionist and Neural Computation Theory, Vol. 2: Analogical Connections*, Norwood, NJ: Ablex.)

Hofstadter, D. & the Fluid Analogies Research Group (1995). *Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thoughts.* New York, NY: Basic Books.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems.* Ann Arbor, MI: University of Michigan Press.

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

Holton, G. (1973). *Thematic Origins of Scientific Thought.* Cambridge, MA: Harvard University Press.

Holyoak, K. & Koh, K. (1987). Surface and structural similarity in analogical transfer. *Memory & Cognition, 15(4)*, pp. 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. (1997). Distributed representations of structure: A theory of analogical access and mapping. *Psychological Review.*

Inagaki, K. & Hatano, G. (1987). Young children's spontaneous personification as analogy. *Child Development, 58*, pp. 1013-1020.

Johnson-Laird, P. (1989(?)). Mental models in cognitive science. *Cognitive Science, 13*, pp. 71-115.

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

Keane, M. (1988). *Analogical Problem Solving.* Chichester: Ellis Horwood (New York: Willey).

Keane, M., Ledgeway, K. & Duff, S. (1994). Constraints on analogical mapping: A comparison of three models. *Cognitive Science*, 18, 387-438.

Keene, S. (1989). *Object-Oriented Programming in Common Lisp: A Programmer's Guide to CLOS*. Reading, MA: Addison-Wesley.

Kokinov, B. (1988). Associative memory-based reasoning: How to represent and retrieve cases. In O'Shea, T. & Sgurev, V. (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 14th Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: LEA.

Kokinov, B. (1992b). Similarity in analogical reasoning. In du Boulay, B. & Sgurev, V. (eds.) *Artificial Intelligence V: Methodology, Systems, Applications*, Amsterdam, Elsevier.

Kokinov, B. (1994a). A hybrid model of reasoning by analogy. In Holyoak, K. & Barnden, J. (eds.) *Advances in Connectionist and Neural Computation Theory, Vol. 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: LEA.

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., Nikolov, V. & Petrov, A. (1996). Dynamics of emergent computation in DUAL. In Ramsay, A. (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: LEA.

Kolodner, J. (1993). *Case-Based Reasoning*. San Mateo, CA: Morgan Kaufmann.

Laird, J., Newell, A. & Rosenbloom, P. (1987). Soar: An architecture for general intelligence. *Aftificial Intelligence, 33*, pp. 1-64.

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*. pp. 375-407.

McDermott, D. (1981). Artificial intelligence meets natural stupidity. In Haugeland, J. (ed.) *Mind Design*, pp. 143-160. Cambridge, MA: MIT Press.

Minsky, M. (1975). A framework for representing knowledge. In Winston, P. (ed.) *The Psychology of Computer Vision*. New York, NY: McGraw-Hill. (Also available in Haugeland, J. (ed.) (1981) *Mind Design*. Cambridge, MA: MIT Press. pp. 95-128.)

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.

Newell, A. (1991). *Unified Theories of Cognition*. Cambridge, MA: Cambridge University Press.

Newell, A. & Simon, H. (1972). *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-Hall.

Palmer, S. (1979). Levels of description in information-processing theories of analogy. In Vosniadou, S. & Ortony, A. (eds.) *Similarity and Analogical Reasoning*, New York, NY: Cambridge University Press, pp. 332-345.

Quillian, M. (1969). The teachable language comprehender: A simulation program and theory of language. *Communication of the ACM, 12*, pp. 459-476.

Ross, B. (1984). Remindings and their effects in learning a cognitive skill. *Cognitive Psychology, 16*, pp. 371-416.

Rumelhart, D. (1975). Notes on a schema for stories. In Bobrow, D. & Collins, A. (eds.) *Representation and Understanding*, pp. 211-236. New York, NY: Academic Press.

Rumelhart, D., McClelland, J. & the PDP Research Group (1986). *Parallel Distributed Processing, Vol.1: Foundations*. Cambridge, MA: MIT Press.

Rumelhart, D., Smolensky, P., McClelland, J. & Hinton, G. (1986). Schemata and sequential thought processes in PDP models. In J.McClelland, D.Rumelhart & the PDP Research Group, *Parallel Distributed Processes, Vol.2: Psychological and Biological Models*. Cambridge, MA: MIT Press.

Schank, R. & Abelson, R. (1977). *Scripts, Plans, Goals, and Understanding: An Enquiry into Human Knowledge Structures*. Hillsdale, NJ: LEA.

Smith, E., Shoben, E., & Rips, L. (1974). Structure and process in semantic memory: A featural model for semantic decisions. *Psychological Review, 81*. pp. 214-241.

Stanfill, C. & Waltz, D. (1986). Toward memory-based reasoning. *Transaction of ACM, 29 (12)*, pp. 1213-1228.

Steele, G. (1990). *Common Lisp: The Language, Second Edition*. Digital Press.

Thagard, P., Holyoak, K., Nelson, G. & Gochfeld, D. (1990). Analog retrieval by constraint satisfaction . *Aftificial Intelligence 46*, pp. 259-310.

Tsang, E. (1993). *Foundations of Constraint Satisfaction*. London: Academic Press.

Veloso, M. (1994). *Planning and Learning by Analogical Reasoning*. Berlin: Springer-Verlag.

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: LEA.