Overcoming Rule-Based Rigidity and Connectionist Limitations through Massively-Parallel Case-Based Reasoning

John Barnden and Kankanahalli Srinivas

MCCS-90-187

Computing Research Laboratory
Box 30001
New Mexico State University
Las Cruces, New Mexico 88003

Overcoming Rule-Based Rigidity and Connectionist Limitations through Massively-Parallel Case-Based Reasoning[1]

John Barnden and Kankanahalli Srinivas

Computer Science Department & Computing Research Laboratory
New Mexico State University
Box 30001-3CRL,
Las Cruces, NM 88003-0001.

(505) 646-6235/4535 jbarnden/srini@nmsu.edu

Running Head: Connectionist Case-Based Reasoning

Abstract

Symbol manipulation as used in traditional Artificial Intelligence has been criticized by neural net researchers for being excessively inflexible and sequential. On the other hand, the application of neural net techniques to the types of high-level cognitive processing studied in traditional artificial intelligence presents major problems as well. We claim that a promising way out of this impasse is to build neural net models that accomplish massively parallel case-based reasoning. Case-based reasoning, which has received much attention recently, is essentially the same as analogy-based reasoning, and avoids many of the problems leveled at traditional artificial intelligence. Further problems are avoided by doing many strands of case-based reasoning in parallel, and by implementing the whole system as a neural net. In addition, such a system provides an approach to some aspects of the problems of noise, uncertainty and novelty in reasoning systems. We are accordingly modifying our current neural net system (Conposit), which performs standard rule-based reasoning, into a massively parallel case-based reasoning version.

[1] This work has been supported in part by grant AFOSR-88-0215 from the Air Force Office of Scientific Research and grant NAGW-1592 under the Innovative Research Program of the NASA Office of Space Science and Applications.

1. INTRODUCTION

Certain limitations of typical connectionist techniques arise when one attempts to apply them to the sorts of high level cognitive task attacked in traditional, symbolic Artificial Intelligence. On the other hand, there has also been much discussion on limitations of traditional symbolic computation — rule-based reasoning in particular — as used in AI and in many models in Cognitive Psychology. A connectionist who is interested, as we are, in high level cognitive tasks (commonsense inferencing, planning, natural language understanding, etc.) is therefore faced with the problem of where to look for a way of overcoming the connectionist problems without running into the traditional symbolic ones. The central theme of this paper is that a promising place to look is connectionistically implemented, massively parallel case-based reasoning.[2]

Case-based reasoning (CBR) is essentially the same as analogy-based reasoning (ABR). A good overall impression of CBR can be obtained from reference DARPA (1989), and of ABR from the papers in Helman (1988), especially Kedar-Cabelli (1988). Some largely methodological differences between the two fields are discussed in "Analogy and CBR" panel in DARPA (1989). In brief, in CBR a current problem, situation or reasoning goal is tackled by transferring advice, solutions or actions from records of similar "cases" (problems, situation-descriptions, and so on) that have been encountered in the past. A fuller account is given below. It has received a lot of attention recently because it is claimed to serve as a good framework for learning and to allow much more fluidity and flexibility of reasoning than standard types of rule-based reasoning (RBR) do. (As is customary, we concentrate on RBR in alluding to traditional AI, but we recognize that not all of traditional AI is rule-based in any strong sense. We should also note that CBR can be implemented in RBR and vice versa. Contrasts between CBR and RBR should therefore be on the basis of rules and cases which are in some sense at the same conceptual level.)

In particular, CBR provides more of a handle on problems of noise, novelty and uncertain reasoning, the three issues that were the foci of the workshop that led to this paper. For our purposes here, it is important to realize that the noise and the uncertainty can arise from imperfections in a system's knowledge base as well as from imperfections in input information. A lot of attention in case-based reasoning research goes to the task of combining the advice from many retrieved cases, under the expectation that some of the advice will be conflicting. For instance, suppose some stored cases portray episodes of insulting. If some of these cases have the insulted person being amused whereas most have the victim being not amused, then the system must deal somehow with the conflict arising when the cases are used to illuminate some new episode of insulting. One thing the system might do is to conclude that the new victim will *probably* not be amused but that he *may* be amused.

Such conflict can be viewed as a source of noise and uncertainty. Although many RBR systems also allow ways of combining advice (from different rules), the techniques are relatively primitive compared to the more advanced advice-combination facilities entertained in the CBR field. In RBR systems the combination is usually confined to some form of numerical combination

[2] This conclusion and the premises from which it was reached are similar to those of Domeshek (1989). However, he does not propose a fully connectionist case-based reasoning system, nor does he suggest using massive parallelism in the way we do.

of confidence/evidence measures attached to hypotheses that have already been created, whereas in CBR systems it is much more common to combine symbolic structures that embody advice in order to create new pieces of advice (see e.g. Barletta & Hennessy 1989, Branting 1989), as well as adapting individual pieces of advice to create new ones (see "Case Adaptation" panel in DARPA 1989).

Another type of "noise" arising in reasoning systems is that most of the system's knowledge is going to be irrelevant to any specific reasoning episode, and the rest relevant to differing extents. Much attention is given in most CBR research efforts to the problem of efficiently accessing only those cases that have promise of being relevant to the problem at hand. Again, approaches to relevance are necessary, indeed central, in RBR systems, since efficiency is severely compromised unless the system only tries to fire rules that have some chance of being useful. It is not clear that there is much advantage to either RBR or CBR with respect to their handling of relevance. However, on the issue of novelty, a system that reasons by analogy to past examples is inherently equipped to deal with significant types of novelty in its input, and to have a considerable advantage over RBR systems in this respect.

RBR as implemented in the traditional symbol processing framework, and indeed traditional symbol processing as a whole, is often criticized by connectionists for being excessively rigid. The alleged rigidity can take a number of forms, notably (a) the inability of rules to act usefully on any data other than the precisely defined type of data the rules were designed for, and (b) their inability to act usefully when not quite enough, or not quite good enough, data is provided. Both of these are failures of adaptability, with the second often described as a lack of "graceful degradation". Connectionist systems are claimed to be able to avoid these failures, even though a given system might still give the appearance of being rule-based or be *approximately* describable as acting in a rule-based way (as in e.g. Rumelhart & McClelland 1986, Smolensky 1988a).

The problem is that connectionism in its current typical forms has severe limitations with regard to effecting commonsense reasoning and other "high level" cognitive tasks. The issues have been discussed by a number of authors (including Barnden 1984, Birnbaum 1990, Dyer 1990, Fodor & Pylyshyn 1988, Smolensky 1988b), and center on the difficulty of getting connectionist systems to systematically handle complex, short-term information structures, such as plans of action, or semantic representations derived from natural language sentences. The difficulty is fundamentally that of ensuring the *efficient* and *systematic* processing of highly *arbitrary* and *temporary* associations between data items. The "variable-binding problem" is a special case of this problem. A number of systems — including but not limited to those reported in the articles in Barnden & Pollack (1990) – have attacked the difficulties raised by high-level cognitive processing. However, no connectionist system to date has been able to approach the flexibility and sophistication with which current traditional AI systems handle such structures.

We therefore wish to accomplish the following simultaneously: (i) avoid the deficiencies of traditional RBR; (ii) exploit the potential benefits of connectionist processing, such as parallel processes for associatively indexing into large knowledge bases (of rules, cases, or whatever), approximate matching of representations, perceptual processing, and some types of learning or adaptation; and yet (iii) avoid the deficiencies of current connectionist systems.

We claim that devising connectionist systems that implement CBR is a good way to achieve these goals. This is firstly because CBR, even when realized by means of conventional symbol-processing, already avoids many of the most commonly discussed problems of RBR. In particular, because reasoning proceeds by comparison of cases, allowing for mismatches and incomplete matches, CBR is much less susceptible to the two specific deficiencies, (a) and (b), that were noted above. But if we were able to devise a *connectionist* CBR system, we would have the opportunity to achieve subgoal (ii) as well. Further, assuming the connectionist CBR system that was able to manipulate cases with complex structure, we would also have achieved subgoal (iii).

This argument for connectionist CBR systems does not exclude a hybrid system involving both connectionist and non-connectionist techniques, with the latter having no connectionist implementation specified for them. However, a *fully* connectionist system is at an advantage through being a more tightly integrated system — there are richer possibilities for interaction among the various parts of the system. For example, if the system has connectionist subsystems for perception, these are more easily interfaced with reasoning subsystems if the latter are also implemented in connectionist networks. As another example, in the system described in this paper the connectionist mechanism for associative recall of cases in long-term memory is intimately related to the mechanism for matching the complex symbolic structures in cases. The intimate relationship is facilitated by the fact that the latter mechanism has a connectionist implementation. Another, independent, motive we have for being concerned with the development of *fully* connectionist CBR models is that we are ultimately interested in the construction of psychological theories and in mapping them down to biological neural networks, as well as in the engineering of reasoning artifacts.

It is often suggested that connectionist and related techniques could be used for case retrieval in CBR systems (see e.g. Domeshek 1989, Martin 1989), or, similarly, for retrieval of stored analogous problems in ABR problem-solving systems (see e.g. Thagard & Holyoak 1989). Also, Holyoak & Thagard (1989) have proposed a connectionist way of performing the mapping process in using analogies. However, such suggestions are generally for systems that are hybrid in the above sense. The few fully connectionist case-based systems that we know of (e.g. the CBR system of Becker & Jazayeri 1989) are quite limited in their ability to handle complex cases.

How are we to devise a fully connectionist system that manipulates cases containing complex structures of information? Our answer is to borrow the basic representational techniques of our existing connectionist system for high-level cognitive processing, called Conposit [Barnden 1988b, 1989, 1990]. This system allows great flexibility in variable binding, and enables very complex symbolic structures to be manipulated much as they can be in AI systems. The system gets its power from unusual ways of handling the problem of arbitrary temporary associations, and will be sketched below. Now, the current Conposit is in fact a connectionist implementation of standard RBR, so that it inherits the rigidity that has been ascribed to traditional RBR systems. Our research therefore centers on radically modifying Conposit to make it into a connectionist, massively parallel CBR system. We emphasize that we currently view the system as a vehicle for the investigation of fundamental issues, rather than as a system that could immediately put to use in some real-world domain. The new system includes promising approaches to the problems of recalling relevant knowledge (noted above) and managing some forms of noise and uncertainty (this is discussed in the concluding section). At present we merely have a partial, pilot simulation of the new version of Conposit. However, the main ideas have been elaborated in enough detail for them to provide a springboard for useful discussion and further work. Moreover, the basic representational techniques, and some crucial processing mechanisms, are inherited in largely unchanged form from the existing, simulated, RBR Conposit.

The plan of the paper is as follows. Section 2 sketches the RBR Conposit. Section 3 gives a short sketch of CBR in general. Section 4 outlines the CBR version of Conposit. Section 5 is the conclusion, and comments on how the new version of Conposit addresses problems of noise, novelty and uncertainty.

2. THE RULE-BASED VERSION OF CONPOSIT

The sketch in this section is of necessity highly compressed, but detailed accounts are given in Barnden (1988b, 1989, 1990). The most important points to appreciate for the purposes of understanding the CBR version of Conposit are the techniques used for encoding symbolic data structures (rather than for manipulating them), and it is only this aspect of Conposit that is explained in any detail here.

Conposit's main strength is the ability to encode complex, temporary, symbolic data structures for use in short-term inferencing processes. A typical type of Conposit data structure is, at an abstract level of description, a semantic network fragment expressing a proposition such as "Bill believes that John loves Mary or that Mary loves Peter". Such propositional data structures are temporary activation states of a neural subnetwork called the "Configuration Matrix" (CM). The data structures are detected, analyzed, modified, destroyed and created by means of domain-specific, hand-crafted, condition-action rules that are hardwired into another neural subnetwork. This part of the system that will disappear in the move to a CBR version of Conposit, and is therefore omitted from the present account. See Fig. 1 for the overall structure of the system.

FIG. 1 ABOUT HERE

As an example of a rule, one Conposit version contains a rule that can be paraphrased as: *if A loves B and B loves C (where C is not A) then A is jealous of C.* Here A, B and C act as variables. The "Subconfiguration Detection Module" in Fig. 1 would detect the presence in the CM of an "A loves B" proposition and a "B loves C" proposition, with a consistent binding for B. Then, the bindings for A, B and C are in essence passed to the rule's action part (in the "Action Parts" module). The action part would check that C is not A, and then create a new data structure stating that A is jealous of C, unless such a proposition already exists. The system thus pursues a course of reasoning by modifying the content of — i.e. neural activation pattern in — the CM. Currently, there is no attempt in Conposit to model any perceptual or natural language interface that could give rise to initial data structures, nor does the system produce any natural language or motor output. A simulation simply starts and ends with some set of data structures in the CM.[3]This is because the research focus has been on high-level reasoning processes.

[3] But there is a user interface allowing CM states to be derived from and converted into propositions in a textual list format.

The current versions of Conposit are RBR systems of a fairly standard type, manipulating familiar types of symbolic data structure. We emphasize that the whole system is defined at the neural net level, although for many purposes it is convenient to discuss it at the more abstract level used in this article.

Conposit's configuration matrix (CM) or working memory is a 32×32 array of *registers*. Each register is implemented as a small connectionist subnet, and is connected to its neighboring registers and to other components. A register's value consists of a "symbol" and a vector of binary "highlighting flag" values. Each highlighting flag is implemented as a connectionist network unit that is either ON (high level of activation) or OFF (zero activation). The symbol is itself implemented as a vector of high/zero activations across some other units in the register. (Conposit's rules respond to configurations of symbols and highlighting values across the CM, and change those configurations by sending signals to registers.)

A symbol may have a specific representational function, such as denoting a particular person or a particular type of relationship among people. Any symbol can be placed in any register, and all registers have the same set of highlighting flags. *Temporary structure is encoded mainly in the adjacency relationships among values in CM registers.* We therefore say that Conposit uses a *"Relative-Position Encoding"* technique. For instance, if a register contains a symbol denoting the class of all possible situations in which one person loves another, and has a certain highlighting flag in the ON state, then any *adjacent* register that has another specific highlighting flag ON is deemed to represent, temporarily, a specific loving situation.

See, for example, the representation of the proposition that John loves Mary in the upper portion of Fig. 2, which shows an 8×8 region of the CM.

FIG. 2 ABOUT HERE

Each square stands for a register, and capitalized words and letters stand for symbols. The word JOHN stands for a symbol denoting a particular person, John, who is known to the system. The LOVE symbol denotes the class of all conceivable loving situations. The X and Y symbols may be ignored for now. The registers with no symbol shown contain a "null symbol" that does not denote anything. The denotations of symbols are considered to be borrowed by the registers they occur in at any moment: a register containing a non-null symbol denotes what that symbol denotes. Hence, in the figure there are registers that — temporarily — denote John, Mary and the love-situations class. The other signs within squares show ON states of highlighting flags, which in this example are all referred to by the names of colors. An 'r' indicates that the register is red-highlighted (i.e. the red flag is currently on); similarly 'g' for green, del sign (∇) for white, and bullet ($\bullet$) for black.

One important function for highlighting is to help specify the representational relationships temporarily holding between adjacent registers. For instance, a white-highlighted register is deemed to denote a member of the class denoted by any neighboring black-highlighted register. Therefore the upper-left white register and the upper-right white register in the figure denote some man and some love situation respectively. Further, if a register denotes a love situation, then any adjacent red register (here, the one containing JOHN) denotes the "lover", and any adjacent green one (here, the one containing MARY) denotes the "lovee". Note that the *absolute* positions of the symbols and highlighting states are irrelevant, as are the *directions* of the adjacency relationships.

The upper-right white register in Figure 2 is said to be the "head" register of the love proposition. The red, green and black registers are the "role" registers. The red and green ones are also called "argument" registers.

Complex data structures can be split up into pieces by a *shared-symbol association* technique, an instance of the general "pattern-similarity association" class of techniques (Barnden 1988a,b, 1990). Shared-symbol association relies on the stipulation that two registers containing the same symbol are considered to represent the same entity. The real power comes from the sharing of variable-like **"unassigned symbols"**. By appearing within a data structure, an unassigned symbol can be viewed as having a temporary denotation dictated by the role of the symbol in the structure. The letters 'X' and 'Y' in Fig. 2 indicate unassigned symbols. Y temporarily denotes some particular but unspecified man, since that is what the register containing it denotes. Similarly, X temporarily denotes the hypothetical loving situation by being in the head register of the love proposition. Altogether, the figure shows how the proposition that *some man believes that John loves Mary* can be encoded by three separate clumps of registers that are linked by the sharing of the X and Y symbols. Much more complex data structures than the one shown in Figure 2 can be built in a similar way.

Even a simple proposition, say *John loves Mary*, can be split into pieces by the unassigned symbol sharing technique. Each piece is a clump of registers representing one or more roles of the proposition. One register in each clump acts like a local "head" register. It is highlighted in white and contains the shared symbol. Each remaining register the clump is adjacent to that register, contains the LOVES, JOHN or MARY symbol, and is highlighted in black, red or green respectively.

As stated above, the symbol in a CM register is a vector of high/zero (ON/OFF) values maintained on some units in the register. This treatment of symbols is merely for the sake of simplicity and of abstracting away from concerns orthogonal to our main goals. What we envisage ultimately is that the "ordinary" (i.e. non-unassigned) symbols like the JOHN symbol are derived by some pattern-compression mechanism from much larger activation patterns elsewhere in a total connectionist cognitive system. One of these larger patterns might, for instance, be a pattern that encodes the visual appearance of the object denoted by the symbol. As for the unassigned symbols, we have already abandoned arbitrary patterns in our design for the new, CBR Conposit. Instead, an unassigned symbol in a register is derived on the fly from the symbols in other registers by a pattern-construction mechanism. This will be explained below.

An important mechanism in the RBR Conposit is carried over into the CBR Conposit. This is the "Temporal-Winner-Take-All" (TWTA) mechanism for performing selection. In the RBR Conposit, it is often used by a rule for selecting an arbitrary register out of a set of contending CM registers (candidates for being affected by the rule). The selection is based on time differences among signals, rather than on activation differences as in the conventional "winner-take-all" styles of selection mechanism [see e.g. Feldman & Ballard 1982, Grossberg 1988, Lippmann 1987].

The TWTA mechanism in the RBR Conposit works as follows. Each contending register sends an "I'm ready" announcement to a module called the CM's "parallel distributor". The parallel distributor makes as its arbitrary choice *the one that sent the signal that was received first.* This relies on the existence of small random arrival-time differences among the announcements. The differences are caused mainly by random delays within the announcement-generating subnetworks in the CM registers. When there is a sufficiently close tie among the earliest arriving announcements, the parallel distributor orders each of the registers that sent the tying messages to try again. Thus, another round of contention starts, but usually with a reduced number of contending registers. If the new round again leads to a quasi-tie, another round of contention is initiated, and so on. The expected number of rounds is very reasonable, given a suitable choice of timing parameters — it grows approximately logarithmically with the number of initially contending registers. Further details, a connectionist implementation and the results of a mathematical analysis and simulation experiments are given in Barnden, Srinivas & Dharmavaratha (1990), where our reasons for preferring temporal-winner-take-all over conventional winner-take-all mechanisms are also given.

As an example of an application of TWTA, and of linking connectionism to CBR, we have implemented a connectionist network (unrelated to Conposit) for diagnosing automobile defects by means of a simple form of CBR. The network consists of two kinds of nodes, the concept nodes, each representing a symptom, fault, or piece of advice, and the case nodes, each representing a past failure situation. A case node is connected with positive weights to the concept nodes signifying the symptoms and faults involved in the situation and the pieces of advice that were useful in the situation. Some concept nodes are also connected to each other by links. For instance, incompatible symptom nodes are connected together by links with negative weights. In a reasoning episode, a set of of symptoms is specified to the system by virtue of high levels of activation being placed on some symptom nodes. Activation spread is started at these nodes and the network is allowed to settle down. The activations of the advice nodes are converted into time by using simple threshold units. The TWTA mechanism is used to select among the advice nodes, the effect being to select the strongest piece of advice. The TWTA based selection mechanism is ideally suited to this purpose owing to its fast convergence properties.

4. CASE-BASED REASONING

To take a basic example of the simple sort of case-based reasoning currently included in the modified Conposit, suppose a stored case (i.e. a case in the "case-base", or long-term memory) contains the following information:

Peter kissed Susan, *Susan dislikes Peter,* *Susan slapped Peter*

(This is simply an unordered set of propositions — in particular, there are no implied temporal or causal relationships among the three events/situations.) Suppose the system is presented with the following situation ("given" case):

John kissed Mary, *Mary slapped John,* *Mary is a student*

Under an obvious mapping of the participants, there is a correspondence between part of the given case and part of the stored case. As a result, the "advice" that *Mary dislikes John* would be formed. Of course, this advice is at best a plausible inference. The handling of possible inconsistencies between the conclusion and other information that might be available from other stored cases is an important research issue for the CBR field in general.

Another possibility is that there is the stored, partially-general case

X kissed Mary,   *Mary slapped X*

where *X* is a variable. (Another yet more general type of case is one that states, for example, that X kissed Y and Y slapped X.) With a given case stating just that *John kissed Mary*, the advice would be that Mary slapped John. There is no intrinsic, rule-like directionality here: if the given case had been just that *Mary slapped John*, the advice would have been that John kissed Mary. It is possible to include extra structure or information in a stored case to indicate that part of it is to be treated as a matching condition and that another part is to be treated as advice — these parts not necessarily being disjoint — and then the case acts much like a rule. However, we would still not demand that the *whole* of the matchable part of the stored case be matched by information in the given case.

The prototypical, basic CBR process can be broken down into the following stages.

(1) *Indexing:* The information in the case or cases that are the current focus of attention are used to generate indexing keys for accessing cases in long-term memory (LTM).

(2) *Retrieval:* Some or all of the indexed LTM cases are retrieved.

(3) *Matching:* The retrieved cases are compared with some or all of the current cases mentioned in (1).

(4) *Advice Creation:* When a retrieved case matches sufficiently well with a current case, portions of the retrieved case, possibly modified, are used as tentative conclusions (advice). Modifications typically involve substitutions of constants (e.g. John for Peter and Mary for Susan in the above example), or substitutions of constants for variables like the *X* above. Conflicting advice must be resolved in some way, perhaps depending of the goodness of the matching that led to the advice. Pieces of surviving, compatible advice can be combined. Advice can generally be regarded as forming a new case, which may now become an answer or a current case.

Also, new cases created may be stored in LTM, rather than merely serving as an answer or as a generator of further indexing into LTM.

This brief description of CBR leaves out many refinements and complications that can be added. Papers in DARPA (1989) provide a good idea of the variety.

4. THE CASE-BASED REASONING VERSION OF CONPOSIT

Overview

The overall architecture of the CBR version of Conposit is as shown in Fig. 3. There are now multiple configuration matrices (CMs) — we envisage hundreds of them ultimately. CMs contain cases that are currently being subject to short-term processing. The system is heavily dependent on copying operations between CMs. A copy operation is done rapidly by sending the states of all the registers in the source CM to the corresponding registers in the destination registers, in parallel.

FIG. 3 ABOUT HERE

The relatively small set of "LTM gateway" CMs provides the interaction between short-term processing and the large long-term memory of cases. (The cases in LTM are not stored in CMs. Rather, they are encoded in connection weight settings, in a way to be detailed below.) Cases in non-gateway CMs compete to have their contents copied into gateway CMs. A case in a gateway CM tends, by an associative indexing/retrieval process, to cause some more or less relevant case stored in LTM to be inserted into the CM, alongside the case already there. When this happens, the new contents of the CM are copied into a currently unused non-gateway CM, in which a matching process occurs between the two cases. If matching is successful, the unmatched part of the retrieved case, after possible symbol substitutions, acts as a collection of new assertions. These constitute the "advice" contributed by the case. The symbol substitutions take care of correspondences between John and Peter, and so on, or between John and the variable X, in the love/slap examples presented earlier. The case consisting of the modified versions of the unmatched parts of the retrieved case, together with the whole of the case that was in the CM before retrieval, is now copied into a currently unused CM. From there it may succeed in being copied into a gateway CM, where it can cause further retrievals from LTM. Therefore, we have a massively parallel and intertwined reasoning process.

There is also a small set of "primary" CMs that collect and merge advice from other CMs, and compete more strongly than other CMs do for copying into gateway CMs. Another purpose for primary CMs is for them to act ultimately as a channel or interface between the reasoning system and other systems, such as systems for non-reasoning aspects of natural language understanding/generation. At present, we simply stipulate that any "run" of the system starts with one or more of the primary CMs containing initial cases.

The case-based reasoning proceeds with no central control. In order to restrict and focus the process, however, a case which has recently been produced by case-matching, or which has recently been placed in a primary CM by an outside agency, is copied to many currently unused CMs. As a result, some proportion of those cases all cause the same LTM case to be retrieved and added into them (if they succeed in being copied into LTM gateways). They therefore tend to lead to the same advice. The more that a piece of advice is replicated in this way, the more power it has for influencing the case-retrieval process. This is a "population-based" mode of self-control, roughly reminiscent of genetic algorithms [Goldberg 1988].

In the following subsections we elaborate on this overview in considerable detail, while refraining from going down to the most detailed level of connectionist network nodes and connections. However, all the mechanisms to be mentioned have straightforward connectionist realizations.

Data Structures and Symbols

The symbolic data structures making up cases are like the data structures in the RBR Conposit (recall Fig. 2). However, the "unassigned symbols" (like X and Y in Fig. 2) are automatically and dynamically generated from the contents of CMs, rather than having unexplained origin as in the RBR Conposit. The main feature of this dynamic generation is that the unassigned symbol put into a head register of a proposition is defined by the application of a simple "hashing" transformation to the symbols and highlighting states in the proposition's role registers. The transformation is called a hashing transformation because it is reminiscent of hash functions as described in standard computer data structure textbooks —- see for example Standish (1980). It is also related to the idea of using "reduced descriptions" (Hinton 1988) as a basis for complex data structures in connectionist systems (see also Touretzky & Geva 1987 and Pollack 1988).

Our hashing transformation is implemented very efficiently as a process involving connectionist circuitry local to registers together with circuitry in the CM's parallel distributor. The hashing is a crucial substrate for the case-matching and case-retrieval operations detailed below. We are experimenting with different hashing methods, but the currently favored one is as follows.

Assume, by way of illustration, that the proposition that John loves Mary has just been created in the CM, say by virtue of input from outside the reasoning system itself. Suppose for simplicity that this proposition is not split up into separate register clumps (cf. Section 2), and is realized as a white-highlighted head register, H, and three neighboring role registers, holding the symbols JOHN, MARY and LOVES, and highlighted in red, green and black respectively (as in Fig. 2). Register H is given an unassigned symbol as follows.

Step 1: In each of H's neighboring registers R that has a role-selection highlighting flag (like black, green or red) ON, the symbol activation pattern is combined with the activation pattern over the highlighting-flag units, to get a vector V_R the same length as a symbol. (The details of this "local" combination are given below.) Note that the only registers that compute a combination vector are the role registers of the proposition.

Step 2: Each role register sends its V vector in parallel to the parallel distributor, which simply averages them and then modifies the result by adding a small, randomly generated vector, to get a result vector V_H.

Step 3: The parallel distributor sends V_H back to H and any other registers that need to contain the same symbol as H. (These registers are identified in the way explained below.) These registers adopt V_H as their unassigned symbol.

Note that any John-loves-Mary proposition will acquire the same unassigned symbol, to within the small random deviations generated in Step 2, no matter how the role registers are "geometrically" arranged around H. Also, the way the above process is modified to take care of a proposition that is split up into several clumps ensures that the computed symbol is the same as if the proposition were not split up.

The addition of a random vector in Step 2 is done to ensure that spurious symbol sharings do not arise. The work of Step 2 is done globally by the parallel distributor rather than locally by H itself, because in the more general case when the proposition is split up (see Section 2) the averaging that contributes to Step 2 involves registers that are not neighbors of H.

Hashing: Local Combination Process

The currently favored symbol/highlighting combination process in a register is as follows. Each component V_i of the result vector is computed by $V_i = S_i \sum_j W_j F_j$ where S_i is the ith component of the register's symbol vector, F_j is 0 or 1 standing for an OFF or ON state (respectively) of the register's jth role flag, and W_j is a weight associated with that role flag. Each register uses the same weight vector W. Currently, W_j is simply taken to be j itself. This appears to lead to satisfactory hashing for our purposes. First, each V_i is trivial to compute using a small number of connectionist units of a standard type. Second, the given definition of V as a function of the S and F vectors obeys the important basic constraint that it should *not* lead to the same V value under permutation of the arguments of a proposition. We want the unassigned symbol computed for John-loves-Mary to be different from that computed for Mary-loves-John.

Hashing: Further Details[4]

It is in principle *possible* in principle for radically different propositions to lead to creation of the same unassigned symbol (to within the small random deviations of the magnitude involved in Step 2 of the hashing process). However, we conjecture that it is extremely unlikely that such coincidences will actually arise. The verification of this conjecture is one important task for future experimentation with versions of the system involving large numbers of ordinary symbols and a large long-term memory of cases.

The hashing process occurs whenever a case is placed in a primary CM from outside the reasoning system, and also on other occasions mentioned below. On any such occasion, registers that are to be given an unassigned symbol must be already highlighted with one of a set of special highlighting flags called "hash" flags. (The hash flag highlighting is deleted once the unassigned symbol has been computed.) Moreover, registers that are to be given the same unassigned symbol must be highlighted with the same hash flag. Registers that have the same hash flag on are deemed to represent the same thing, just as if they already contained an unassigned symbol.

The description of hashing above used the proposition that John loves Mary as an example. We must also explain what happens when one of the role registers, say the agent register A, is itself to contain an unassigned symbol. With no further propositions involving this symbol, the effect is that of representing the proposition "something loves Mary". An unassigned symbol in a register like A is simply a small randomly generated vector (generated with same parameters as used in Step 2 of the hashing process above), as long as the symbol is not also to be shared with the head register of some proposition. This sharing would occur in the representation of a proposition like "some man loves Mary" (recall the "some man believes that ..." proposition represented as shown in Fig. 2). In this case the symbol in the argument register A of "something loves Mary" must also appear in a white register next to a black register containing the MEN symbol.

[4] This subsection can be skipped on a first reading.

Consider now the proposition "Some man believes that John loves Mary" (see Fig. 2). In this case, the symbol L in the John-loves-Mary head register H must also appear in the object role register O of the "believes" proposition's register clump. However, this does not affect L. The reason for this is that the only registers which do a local symbol/highlighting combination and transmit the result to the parallel distributor are ones that have a role flag on and are adjacent to a *white* register which is to contain the desired unassigned symbol (L in this case). Register O is not white, so none of its neighbors have any effect on L. In sum, the unassigned symbol in a proposition head register depends only on that proposition and not on higher level propositions for which that symbol acts as an argument.

Several different unassigned symbols may need to be computed in a CM. Because of embeddings of propositions within each other (as in the "believes" example just treated), the symbols cannot in general be computed in an arbitrary order: for instance, in the "believes" example the John-loves-Mary proposition's head symbol L must be computed before the head symbol B of the belief proposition. The correct ordering is obtained by allowing the computation of a symbol only if none of the registers into which that symbol is to be put has a non-white neighbor with a hash flag on. Apart from this constraint, unassigned symbols can be computed in any order. Because the computations involve the parallel distributor as a shared resource, they must be serialized. The serialization is done by means of successive applications of the TWTA mechanism.

Case Matching: Basic Process

Recall that case matching is to occur when a case C_{ltm} is retrieved from LTM and placed in a CM that already contains a case C_0. When this is done, the two cases are distinguished by having the registers they use highlighted with distinct, special flags h_{ltm} and h_0. The case matching process now seeks to establish for each proposition in C_0 whether there is a similar one in C_{ltm}.

First consider a C_0 containing only one proposition, P_0. The white, head register H of P_0 is made to broadcast its symbol throughout the CM. Then, each register R highlighted with both white and h_{ltm} compares the broadcast symbol with its own symbol. If equality is found, to within the maximum magnitude M somewhat bigger than that of the random deviations in Step 2 of the hashing process, R sends a graded "local degree of match" signal to the CM's parallel distributor. The size of the signal is dependent on the exactness of equality of R's own symbol and the broadcast one (and also, for reasons to be given later, on the magnitudes of the symbols themselves). The parallel distributor maintains a "total degree of match" activation level that is augmented on receipt of local-degree-of-match signals. The intuitive interpretation of this level is that it says how strong the C_0/C_{ltm} match is, and hence how a strong a piece of advice the contents of the unmatched propositions in the h_{ltm} portions of the CM are to be regarded as being.

A register R noticing approximate symbol equality as just described not only sends a local degree of match signal to the parallel distributor, but also sends its current symbol to it. When the local degree of match is above a certain threshold, the parallel distributor averages the incoming symbols together with the previously broadcast symbol. It then sends the result V to all registers. Registers whose current symbols are within M of V now adopt V as their new unassigned symbol. The adopting registers necessarily include H and all the registers like R, plus all registers containing the same symbol as all those registers. The intuitive effect of the symbol adoptions is to force all the registers in question to mean the same thing.

Recall that the head symbols of identical propositions have roughly identical unassigned symbols created by the hashing process, even if the clumps of registers used by the propositions are geometrically different. Therefore, if P_0 is in C_{ltm} a strong local-degree-of-match signal will be sent to the parallel distributor. Consider for instance the following example:

EXAMPLE 1:	C_0	C_{ltm}
	John kissed Mary	*John kissed Mary*
		Mary slapped John

Here the total degree of match in the parallel distributor achieves a high level, and the proposition that *Mary slapped John* is to be regarded as being fairly strong advice. More precisely, as it is not one of the propositions that was matched against one in C_0, it is what is added to C_0 to form a new case. By specially highlighting the matched parts of C_{ltm}, it is possible to cause just C_0 and *Mary slapped John* to be copied into an unused CM.

Consider now a C_0 containing several propositions. The process just described is done for each one, under some arbitrary serialization effected through the TWTA mechanism. The total degree of match is now be affected by each of the propositions. Consider:

EXAMPLE 2:	C_0	C_{ltm}
	John kissed Mary	*John kissed Mary*
	Mary was angry with John	*Mary was angry with John*
		Mary slapped John

The total degree of match in the parallel distributor a higher level than in Example 1.

Currently we take the total degree to be just the sum of the local degrees. Note in particular that this means that there is no penalty for unmatched parts of C_0, as in the following modification of Example 1.

EXAMPLE 3:	C_0	C_{ltm}
	John kissed Mary	*John kissed Mary*
		Mary slapped John
	Mary was angry with John	

By default, the system assumes that unmatched parts of C_0 do not upset the advice formed on the basis of the matched parts. However, it is certainly important to be able to upset advice on occasion. Suppose one CM contains the same C_0 and C_{ltm} as in Example 1, and another CM contains:

EXAMPLE 4:	C_0	C_{ltm}
	John kissed Mary	*John kissed Mary*
	Mary loved John	*Mary loved John*
		Mary did not slap John

From the first CM there will be the advice that *Mary slapped John.* However, from the second CM there will also be the stronger advice that *Mary did not slap John.*

Matching Variables to Constants or Other Variables

The described case-matching process works just as well on examples in which variables must be matched against variables. For instance, suppose "John loves something" is in C_0 and "John loves something" is in C_{ltm}, assuming that in each proposition the second argument of the loving is represented by an unassigned symbol (and that this symbol is not in any proposition head register in the CM). It is still the case that the hashing process will have put roughly the same unassigned symbols in the two head registers. This is because into the argument registers corresponding to the "somethings" it puts roughly the same unassigned symbols (small random vectors in this case). Therefore, matching succeeds just as before.

However, we impose the rule that a local degree of match signal sent by a register to the parallel distributor is proportional to the magnitude of the symbol vector in the register. Since the symbol corresponding to the "something" in C_{ltm} is small, the symbol in the head register of the love proposition in C_{ltm} is considerably smaller than it would have been if the object of the proposition had been Mary instead of "something". Hence, the local degree of match produced by the two instances of "John loves something" is considerably smaller than it would have been if we had had "John loves Mary" in both cases. Therefore, the system penalizes matches for lack of specificity. This effect becomes very important below.

Matching in which a constant like JOHN might be matched with (a) a variable or (b) another constant constitutes more of a problem. Our approaches to (a) and (b) are similar, and (b) is slightly more difficult, so we will discuss that. An example is as follows.

EXAMPLE 5:	C_0	C_{ltm}
	John kissed Mary	*Peter kissed Susan*
		Susan laughed

We wish to ensure that it is possible to conclude that Mary laughed, with some lower degree of confidence than if C_{ltm} consisted of *John kissed Mary* and *Mary laughed.* The problem is that the unassigned symbols used in the head registers of the two *kissed* propositions are very different, since the propositions use very different argument symbols, so that the basic matching process fails. The system now proceeds as follows.

It randomly replaces the constant symbols in some or all of the argument registers of kiss propositions in C_{ltm} by unassigned symbols. For now we will assume that *all* the argument symbols are replaced. (Both occurrences of the SUSAN symbol are replaced by the same new symbol.) In our example, none of the registers containing any of the new unassigned symbols has to share its symbol with a proposition head register. Therefore, the unassigned symbols replacing PETER and SUSAN are small random vectors.

In C_0, the system leaves the argument symbols of the current proposition, *John kissed Mary,* as they are, but highlights one or both of the argument registers with a special highlighting flag

$h_{suppress}$. It highlights the agent register in this way if any kiss-agent constant argument symbol in C_{ltm} was replaced by an unassigned symbol. It deals with the object register in the analogous way. The unassigned symbols in propositions heads are re-computed, with the extra proviso that registers highlighted with $h_{suppress}$ are treated as if they contained the zero vector (the vector for the null symbol). Hence, the new head symbols in the two "kissed" propositions will be approximately equal, so that a good match will occur by the basic case matching process.

Whenever the register clump for a proposition in C_0 receives a new head symbol as a result of a good match, it causes the symbols in its argument registers to replace the symbols that lie in the corresponding registers in the matching proposition in C_{ltm}. The replacement is done at every register in which the latter symbols appear. Intuitively the effect is to replace Peter everywhere in C_{ltm} by John, and to replace Susan everywhere in C_{ltm} by Mary. The process involves sequencing through the set of argument registers of the proposition in C_0. The sequencing is in arbitrary order and is handled by TWTA.

To sum up so far, the total effect of case matching in Example 5 is to replace the propositions in C_{ltm} by *John kissed Mary* and *Mary laughed*. Since the latter was not one that was matched, it is what it is added to C_0 to form the new case. Notice, however, that because the Peter and Susan symbols were replaced by unassigned symbols in the course of matching, the local degree of match computed by the head register of the kiss proposition in C_{ltm} is relatively small, so that the system has relatively little confidence in the new case, compared with what it would have had if C_{ltm} had contained *John kissed Mary*.

More on Example 5: Replication of Matching

What if C_{ltm} is the pair *John kissed Susan* and *Susan laughed*? If all the constant arguments in C_{ltm} were replaced by unassigned symbols as before, then we would have a match as weak as it was before, even though John is the agent of the kissing in both C_0 and C_{ltm}. It is to take care of this type of situation and related ones that the replacement process in C_{ltm} only affects a random subset of the argument registers. It is then possible that the John symbol in C_{ltm} is left untouched and only the Mary symbol is replaced. If this happens, then the magnitude of the symbol in the kiss proposition head register in C_{ltm} is bigger than in the original form of Example 5, so that we get a higher degree of match.

Naturally, it is undesirable to rely on just one random selection of constant arguments coming up with the right thing. Our technique for dealing with this is population based. The whole contents of the CM are copied into as many other unused CMs as possible. (This requires competition with other activities that are trying to grab unused CMs.) In each of these the same matching process now goes on, but with different random selections of replacements. Also, in each individual CM, if the matching process does not lead to a good match with a particular selection, then another selection is made, and so on. We therefore markedly increase the chances that a good match will be found.

The full validation of this technique awaits the construction of a simulation of a large version of the system.

Case Matching: Consistency of Matching

Consider the following variant of Example 5.

EXAMPLE 6:	C_0	C_{ltm}
	John kissed Mary	*Peter kissed Susan*
	Mary loved John	*Susan loved Peter*
		Susan laughed

The issue we address here is how to ensure that the total degree of match is boosted by the fact that there is a *consistent* match based on the mapping of John to David and Mary to Susan, compared to what would happen in a less consistent situation (in which, say, C_{ltm} stated that Susan loved someone other than Peter, a possibility to be treated in Example 7 below).

Let us assume that the system considers *John kissed Mary* before *Mary loved John*. Assume also that both the Peter symbol and the Susan symbol in C_{ltm} are replaced in the attempt to find a match for *John kissed Mary*. A (weak) match is found as in the original form of Example 5. As a result, the CM state is changed into:

C_0	C_{ltm}
John kissed Mary	*John kissed Mary*
Mary loved John	*Mary loved John*
	Mary laughed

Therefore, when the system turns to look at *Mary loved John*, it will find a strong match for this proposition. The total degree of match resulting from the two propositions in C_0 is therefore considerably greater than it was in the original form of Example 5.

Now consider what happens in the following example, a less consistent variant of Example 6:

EXAMPLE 7:	C_0	C_{ltm}
	John kissed Mary	*Peter kissed Susan*
	Mary loved John	*Susan loved Bill*
		Susan laughed

Under similar assumptions as before, the system finds a weak match for *John kissed Mary*, and will convert the CM state into:

C_0	C_{ltm}
John kissed Mary	*John kissed Mary*
Mary loved John	*Mary loved X*
	Mary laughed

where X is the unassigned symbol that replaced the Bill symbol. When the system turns to *Mary loved John*, it will find a match, but weaker than the one it found in Example 6.

Notice incidentally that this match can be found without replacing any constants in C_{ltm}. Merely putting $h_{suppress}$ highlighting at the John object register in *Mary loved John* will ensure a match. In fact, no constants in C_{ltm} that arise from C_0 should be replaced anyway. This prohibition can easily be enforced by suitably highlighting any register in C_{ltm} that adopts a symbol from C_0. In the present example, after doing the replacements caused by matching *John kissed Mary*, all the argument registers in C_{ltm} containing constants have been highlighted in this way and are protected from further symbol replacement.

Case Matching: Further Complications

Consider the problem of matching *John loves some dog* in C_0 to *John loves some film* in C_{ltm}. Each of these would be represented by means of two clumps of registers. C_0 would be represented clumps paraphrasable as *John loves D* and *D is a dog* for some unassigned symbol D. Following the technique shown in Fig. 2 for the "some man" agent of the believing, the *D is a dog* clump would consist of an adjacent pair of registers: a white one containing D and a black one containing the DOGS symbol. The two clumps for C_{ltm} would be analogous, but using a different unassigned symbol F, and using the FILMS symbol instead of the DOGS symbol.

Since black is a role flag, D and F are affected by being in the head registers for *D is a dog* and *F is a film* respectively, and are markedly different symbols. Hence, the symbols in the head registers for *John loves D* and *John loves F* are markedly different from each other. Nevertheless, it seems reasonable that a relatively weak match should be discernible between these two propositions, even though dogs are very different from films. Our approach is to rely on a technique much like the one above for matching different constants with each other. The system deletes a random subset of those propositions P in C_{ltm} whose head symbol is an argument symbol of a love proposition. Of course, in our example the only such proposition P is *F is a film*. For corresponding arguments of *John loves some dog* in C_0, the system imposes $h_{suppress}$ highlighting. Matters now proceed as in the matching of different constants.

Further, a stronger match should be discernible if we change the example by replacing "film" by "cat", in view of the greater similarity of dogs and cats than of dogs and films. The same basic process could apply as in the film example, but we are not yet sure how to ensure a stronger degree of match. One current suggestion we make is as follows, in outline. We suppose that the system, on noticing that C_0 and C_{ltm} contain the DOGS and CATS symbols respectively, has caused some a set P_{dogs} of general propositions about dogs and a set P_{cats} of general propositions about cats to be brought down from LTM into some CM. The system tries to match P_{dogs} and P_{cats} just as if they were two cases. We may suppose that a fairly good match is found. The system then causes the DOGS and CATS symbols in the original CM to be replaced by the same unassigned symbol Z, whose magnitude as a vector is proportional to the degree of match of P_{dogs} and P_{cats}. Once this replacement has been done and other unassigned symbols in the original CM recomputed, symbols D and C in the head registers of the *D is a dog* and *C is a cat* propositions are now approximately the same. This in turn means that the head registers of the *John loves some dog* and *John loves some cat* contain roughly the same symbol, and a fairly strong match can ensue. Notice, however, that the magnitude of Z affects that of D and C, and hence that of the head registers of the love propositions. Thus, there is an effect on the local degree of match reported by the *John loves cats* proposition.

This process involves an elaborate recursive use of case matching, and there remain several detailed control issues to be investigated. It is reasonable to suggest the following optimization of the process. The discovered match of P_{dogs} and P_{cats} could cause a fast association to be set up, outside the case-based reasoning system as we have described it, between the DOGS and CATS symbols. Then the appearance of DOGS in a CM could directly and rapidly cause stimulation, to some degree, of the CATS symbol, and thereby lead to the replacement of DOGS and CATS by Z as before.

A further issue on which we have only speculative suggestions at present is a problem raised by embedded propositions, such as arise, for instance, in the representation of negations and belief states. We will concentrate on the former here as they present the most extreme problem. Consider the following example.

EXAMPLE 8:	C_0	C_{ltm}
	John kissed Mary	*John kissed Mary*
	Mary was happy	*Mary was not happy*
		Mary slapped John

Assume that *Mary was not happy* is represented as a *Mary was happy* proposition embedded as the only argument in a *not* proposition. Here the *not* proposition itself is a three-register clump whose only argument register contains the same symbol X as is in the head register of *Mary is happy* proposition in C_{ltm}. The problem is that X is not affected by being in the *not* proposition, and the matching process as it has been described will find a strong match between the *Mary is happy* proposition in C_0 and the one in C_{ltm}. One simple approach to this problem is to assume that only top-level propositions take part in the match-detection process. This can be enforced by having each top-level proposition's head register marked with a special highlighting flag (as is done already in the RBR Conposit), and to allow only thus-marked registers to take part in matching. Then, nothing in C_{ltm} will be found to match the top-level proposition *Mary is happy* in C_0, since its copy in C_{ltm} is not top-level.

The total degree of match will be less than that obtained from matching C_0 against the C_{ltm} consisting of *John kissed Mary*, *Mary was happy*, and *Mary slapped John.* That is a desirable effect. However, the suggested technique does not in any way penalize the match on the basis that C_0 and C_{ltm} in Example 8 as shown above contain conflicting propositions, and such penalization might be thought desirable. For, surely the match should less strong than that arising from the C_{ltm} containing, say, *Mary was tall* instead of *Mary was not happy.* However, the issue is by no means clearcut, since it may well be, for all the system knows, that Mary's happiness or otherwise is simply irrelevant to the question of whether she slapped John. It may therefore be enough to rely on the interference from competing pieces of advice from other case matchings. Suppose, for instance, the same C_0 is being matched elsewhere against a C_{ltm} consisting of *John kissed Mary, Mary was happy* and *Mary did not slap John.* This will lead to a strong match — stronger than the one obtained in Example 8 — so that we may expect the advice that *Mary did not slap John* to win out over the advice that she did.

Long-Term Case Memory: The Neural Encoding

Complete CM states are encoded into long-term memory by means of weight change in the following simple method. Each LTM item is a node (i.e., small neural subnet) that has a bundle of connections to and from each register of some single LTM gateway. The weights on the connections to the registers are such that sufficiently high stimulation of the node causes particular symbol and highlighting activation patterns to be sent to the registers. These patterns are merely "OR'd" into the registers, in that an OFF highlighting value transmitted to a register does not switch off the corresponding highlighting flag in the register, and a null symbol transmitted to a register does not change its symbols. The intention of this is that, as long as the transmitted ON values for highlighting and the transmitted non-null symbols are in different positions from the ON highlighting and non-null symbols already in the CM, the effect is simply to *add* data structures to the CM.

To ensure that the given and retrieved cases do indeed occupy different registers in the CM, we currently rely on the simple but possibly inelegant method of "squashing" the given case into one particular half of the CM, and loading the retrieved case into the other half. Once a retrieval operation has occurred, the new state of the LTM gateway concerned is copied to possibly many other (non-gateway) CMs, which can now perform case-matching. For this copying, the LTM gateway competes with other LTM gateways for access to free, non-gateway CMs.

For creating an LTM encoding of a CM state, the state is first copied into an arbitrary LTM gateway, overwriting its previous contents. Then a new LTM node is recruited from among a pool of unused nodes that have connections to the registers of the gateway. This is done by a TWTA arbitrary-selection method (operating in this case on "ready" announcements generated by LTM nodes rather than CM registers). Then it is simple to transmit the registers' symbol and highlighting activation patterns to the node, and arrange for the weights on connections from the node to the registers to be changed appropriately (and quickly). Our current strategy as to when long-term memory items should be created is simply to have any CM that has recently performed a successful case match enter a competition to have its new state copied into an LTM gateway so that it can be stored.

LTM Indexing and Recall Competition

Our method of indexing into the case LTM relies on approximate equalities among symbols. The intention is that if an LTM gateway contains, say, the JOHN symbol in some register, then any LTM node that is connected to that register and encodes a CM state involving the JOHN symbol should be given some degree of activation. Accordingly, Conposit does the following for each non-null symbol in the gateway: (1) it broadcasts the symbol to all the registers in the gateway, and then (2) it transmits the symbol activation patterns, in parallel, from each register to each of the LTM nodes connected to the gateway, along the same connections as are used for creation of LTM nodes. It is simple to arrange for this transmission to stimulate the LTM node if it encodes a CM state involving the symbol JOHN in any position.

Since unassigned symbols with similar data structure contexts are similar, the method just described does much more than just stimulate LTM items containing particular ordinary symbols. It also tends to stimulate LTM items that have similar pieces of *complex data structure*. For instance, a *Bill believes that John loves Mary* proposition will tend to stimulate LTM items encoding CM states containing that proposition. Note that such items would also be stimulated merely by virtue of the fact that they encode CM states involving the LOVES and BELIEVES symbols. By the same token, the presence of "John loves Mary" in a gateway can cause retrieval of an LTM item that encoded "Peter loves Susan", "Peter loves someone", and so on, but the stimulation of such LTM items is relatively weak compared to that of an item containing "John loves Mary". Altogether, therefore, symbol-similarity leads to differential levels of activation rather than to a clear selection of particular LTM items.

LTM nodes stimulated (i.e. indexed) by a gateway CM compete to add their contents to that CM. The competition is governed by the strength of activation of the nodes. However, instead of using a standard Winner-Take-All mechanism, the differential activations are first converted into temporal differences (a simple matter), allowing the competition to be organized by TWTA.

Competition among CMs

Several different types of occasion have been mentioned on which CMs compete to copy their contents into another CM. The latter may be an LTM gateway, a primary CM, or just an ordinary CM. All such competitions are handled by the TWTA mechanism, which also handles contention among LTM items or nodes during recruitment and retrieval, and needed serializations of operations within CMs. The inter-CM competitions rely on "ready" announcements generated by the CMs (or more precisely by their parallel distributors).

In the case of competition among CMs containing just-formed advice, the use of TWTA tends to favor advice on the basis of the promptness with which it has been formed and of the extent to which it has been replicated in different CMs. The intention here is that the more relevant and useful lines of reasoning will lead to greater replication, and will therefore tend to be the ones to load their conclusions (advice) into other CMs. This will in turn serve to extend those lines of reasoning in preference to other possible ones, especially if the destination CM is a primary or gateway CM. This is roughly reminiscent of genetic algorithms.

The activity-to-time transformation mentioned in the previous subsection is also applied to the total-degree-of-match levels in CMs, allowing these levels to affect the various types of TWTA contention among CMs.

5. CONCLUSION

We have described the CBR version of Conposit, a preliminary, massively parallel case-based reasoning system that is connectionistically implemented. The motivation has been to achieve the ability to manipulate complex structured symbolic information — a matter which is difficult for standard connectionist models — while avoiding the rigidity problems of standard rule-based systems.

Many connectionists claim that connectionism is a way to avoid the problems with standard rule-based systems. We do not repudiate this claim, but wish to point out that many of the problems can be avoided by moving to case-based reasoning as opposed to rule-based reasoning, whether or not the case-based reasoning is implemented connectionistically. What the connectionist implementation adds is certain features and opportunities of a familiar connectionist sort. One of these is the ability to include fast parallel LTM indexing mechanisms, and to do so in a way which is thoroughly integrated with the rest of the system, rather than being grafted onto a non-connectionist case-manipulation system. In the same vein, there is the opportunity for relatively easy integration of Conposit with connectionist subsystems for perception and motor control. There is also the possibility of adding classical forms of connectionist learning, such as slow adjustment of associations among Conposit's "ordinary" symbols. For instance, if "DOGS and "CATS" symbols came to be associated through perceptual experience in the world or through reasoning about cats and dogs, a case about dogs could cause the retrieval of a case about cats.

Currently the type of learning the system is designed to do is learning by case accumulation (the basic type of learning in the CBR field). One should note that this is accomplished by (fast) connection-weight change, and so is not as distinct from standard forms of connectionist learning as it may appear at first sight. Although the system might accumulate a huge number of cases in long-term memory, each case is stored in (large) set of connection weights, *not* in a large set of nodes. A case only requires many nodes when it is brought down into a CM, which is a network containing many thousands of nodes.

Finally, we return to the issues of novelty, noise and uncertainty. Any case/analogy-based system is inherently concerned with being able to perform reasonably on inputs that are novel to some significant extent. Indeed, that is one of the main motivations for CBR, and one of the main departures from standard RBR that are claimed by CBR researchers. Although Conposit only purports at present to embody a relatively simple style of CBR, we already see an ability to deal with novelty. This is because case matches can be partial and allow different constants (e.g. the John and David symbols) to match. These features are somewhat analogous to the types of imperfect matching supported by more conventional connectionist systems, although achieved in a different way.

The particular forms of noise and uncertainty that the work addresses are largely those arising within reasoning systems, rather than those inherent in system input. CBR in general is specifically aimed at handling the conflicts (= noise and uncertainty) arising in the advice to be gleaned from past experience, and at managing the deployment of a large amount of stored information whose relevance to and usefulness for a current reasoning goal is not clearcut. A piece of information that is retrieved and tentatively applied to some problem, but which actually turns out to be irrelevant or useless, is noise of a sort. Conposit attempts to reduce the irrelevance/uselessness problem by indexing into long-term memory on the basis of small keys that can encode complex structures. These keys are the unassigned symbols computed from those structures by the hashing process. Of course, we can still expect that irrelevant or useless cases will be retrieved. This is where we get one benefit from massive parallelism at the case processing level. Many strands of processing may turn out to be unprofitable — in terms of successful matches and new conclusions — but profitable strands can coexist with them. Configuration matrices that have performed strong case matches tend to win out over other matrices when it comes to having their contents copied into "primary" configuration matrices (which can be viewed as holding results of reasoning). Notice also that only unmatched parts of retrieved cases C_{ltm} (that sufficiently strongly match a given case C_0) are deemed to be advice. By making the contention ability of a configuration matrix depend on the amount of advice it is producing, as well as on the degree of match it has computed as at present, we could guard to a large extent against the bad effects of retrieving cases that are relevant but which are useless in the sense of providing no new information.

Conposit's massive parallelism and the mechanism for resolving contentions among CMs trying to copy their contents into primary CMs potentially provides a way of handling advice conflicts (and the noise and uncertainty they entail). If many retrieved cases suggest a piece of advice A but a few suggest a conflicting piece of advice B, then the number of CMs producing A through the case-matching process is likely to be considerably greater than the number producing B. Hence, A is much more likely to be loaded into a primary CM. By trying the contention process several times and noting the relative frequency with which A and B appear in primary CMs (or perhaps by trying it once and noticing the relative frequency of instances of A and B over the primary CMs), the system can come up with confidence measures for A and B. This approach requires that further matching be done among the different propositions appearing in primary CMs, but this matching can be done by the existing type of matching mechanism, in parallel with rest of the system's operations. Another refinement of the scheme is to transmit degrees of match when advice is copied from a CM into a primary CM. These degrees can easily be made to affect the confidence measures computed from the primary CMs.

ACKNOWLEDGMENTS

We are grateful to Mike Coombs, Dwight Deugo, Chris Fields, Trent Lange, Donald Malkoff, Ganesh Mani, and Heather Pfeiffer for helpful comments, and to Coombs and Fields for organizing the conference workshop which furthered the research reported in this paper.

REFERENCES

Barletta, R. & Hennessy, D. (1989). Case adaptation in autoclave layout design. In *Procs. Case-Based Reasoning Workshop* (Pensacola Beach, Florida.) San Mateo, CA: Morgan Kaufmann, pp.203–207.

Barnden, J. A. (1984). On short-term information processing in connectionist theories. *Cognition and Brain Theory*, 7, 25–59.

Barnden, J. A. (1988a). The right of free association: relative-position encoding for connectionist data structures. In *Procs. 10th Annual Conference of the Cognitive Science Society.* Hillsdale, N.J.: Lawrence Erlbaum, pp.503–509.

Barnden, J. A. (1988b). Conposit, a neural net system for high-level symbolic processing: overview of research and description of register-machine level. *Memoranda in Computer and Cognitive Science*, No. MCCS-88-145, Computing Research Laboratory, New Mexico State University, Las Cruces, NM 88003.

Barnden, J. A. (1989). Neural-net implementation of complex symbol-processing in a mental model approach to syllogistic reasoning. In *Procs. 11th Int. Joint Conf. on Artificial Intelligence.* San Mateo, CA: Morgan Kaufmann.

Barnden, J. A. (1990). The power of some unusual connectionist data-structuring techniques. In J.A. Barnden & J.B. Pollack (Eds), *Advances in Connectionist and Neural Computation Theory*, Vol. 1. Norwood, N.J.: Ablex.

Barnden, J.A. & Pollack, J.B., Eds. (1990). *Advances in Connectionist and Neural Computation Theory, Vol. 1: High Level Connectionist Models.* Norwood, N.J.: Ablex.

Barnden, J.A., Srinivas, K. & Dharmavaratha, D. (1990 forthcoming). Winner-take-all networks: time-based versus activation-based mechanisms for various selection goals. To appear in proceedings of *IEEE International Symposium on Circuits and Systems*, New Orleans, May 1990.

Becker, L.A. & Jazayeri, K. (1989). A connectionist approach to case-based reasoning. In *Procs. Case-Based Reasoning Workshop* (Pensacola Beach, Florida.) San Mateo, CA: Morgan Kaufmann.

Birnbaum, L. (1990). Complex features in planning and understanding: problems and opportunities for connectionism. In J.A. Barnden & J.B. Pollack (Eds), *Advances in Connectionist and Neural Computation Theory*, Vol. 1. Norwood, N.J.: Ablex.

Branting, L.K. (1989). Integrating generalizationds with exemplar-based reasoning. In *Procs. Case-Based Reasoning Workshop* (Pensacola Beach, Florida.) San Mateo, CA: Morgan Kaufmann, pp.224–229.

DARPA (1989). *Procs. Case-Based Reasoning Workshop* (Pensacola Beach, Florida.) San Mateo, CA: Morgan Kaufmann.

Domeshek, E. (1989). Parallelism for index generation and reminding. In *Procs. Case-Based Reasoning Workshop* (Pensacola Beach, Florida.) San Mateo, CA: Morgan Kaufmann, pp.244–247.

Dyer, M.G. (1990). Symbolic NeuroEngineering for natural language processing: a multilevel research approach. In J.A. Barnden & J.B. Pollack (Eds), *Advances in Connectionist and Neural Computation Theory*, Vol. 1. Norwood, N.J.: Ablex.

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

Fodor, J.A. & Pylyshyn, Z.W. (1988). Connectionism and cognitive architecture: a critical analysis. In S. Pinker & J. Mehler, Eds. *Connections and Symbols.* Cambridge, Mass.: MIT Press; and Amsterdam: Elsevier. (Reprinted from *Cognition*, **28**.)

Goldberg, D. (1988). *Genctic algorithms in search, optimization and machine learning.* Reading, Mass.: Addison-Wesley.

Grossberg, S. (1988). Nonlinear neural networks: principles, mechanisms, and architectures. *Neural Networks*, 1, 17–61.

Helman, D.H. (Ed.) (1988). *Analogical reasoning.* Dordrecht: Kluwer Academic Publishers.

Hinton, G.E. (1988). Representing part-whole hierarchies in connectionist networks. In *Procs. 10th Annual Conf. of the Cognitive Science Society.* Hillsdale, N.J.: Lawrence Erlbaum.

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

Kedar-Cabelli, S. (1988). Analogy — from a unified perspective. In D.H. Helman (Ed.), *Analogical reasoning.* Dordrecht: Kluwer Academic Publishers, pp.65–103.

Lippmann, R.R. (1987). An introduction to computing with neural nets. *IEEE ASP Magazine*, 4, 4–22.

Martin, J.D. (1989). In *Procs. Case-Based Reasoning Workshop* (Pensacola Beach, Florida.) San Mateo, CA: Morgan Kaufmann, pp.300-303.

Pollack, J.B. (1988). Recursive auto-associative memory: devising compositional distributed representations. In *Procs. 10th Annual Conf. of the Cognitive Science Soc.* Hillsdale, N.J.: Lawrence Erlbaum.

Rumelhart, D.E. & McClelland, J.L. (1986). On learning the past tenses of English verbs. In J.L. McClelland, D.E. Rumelhart & The PDP Research Group (Eds), *Parallel Distributed Processing: Experiments in the Microstructure of Cognition*, Vol. 2. pp.272–325. Cambridge, MA: MIT Press,

Smolensky, P. (1988a). On the proper treatment of connectionism. *Behavioral and Brain Sciences, 11*, 1–74.

Smolensky, P. (1988b). The constituent structure of connectionist mental states: a reply to Fodor and Pylyshyn. In T. Horgan & J. Tienson (Eds), *Connectionism and the philosophy of mind*, Supplement to Vol. XXVI of *The Southern J. of Philosophy.*

Standish, T.A. (1980). *Data structure techniques.* Reading, Mass.: Addison-Wesley.

Thagard, P. & Holyoak, K.J. (1989). Why indexing is the wrong way to think about analog retrieval. In *Procs. Case-Based Reasoning Workshop* (Pensacola Beach, Florida.) San Mateo, CA: Morgan Kaufmann, pp. 36–40.

Touretzky, D.S. & Geva, S. (1987). A distributed connectionist representation for concept structures. In *Procs. 9th Annual Conf. of the Cognitive Science Society.* Hillsdale, N.J.: Lawrence Erlbaum.

FOOTNOTES

(1) This work has been supported in part by grant AFOSR-88-0215 from the Air Force Office of Scientific Research and grant NAGW-1592 under the Innovative Research Program of the NASA Office of Space Science and Applications.

(2) This conclusion and the premises from which it was reached are similar to those of Domeshek (1989). However, he does not propose a fully connectionist case-based reasoning system, nor does he suggest using massive parallelism in the way we do.

(3) But there is a user interface allowing CM states to be derived from and converted into propositions in a textual list format.

(4) This subsection can be skipped on a first reading.

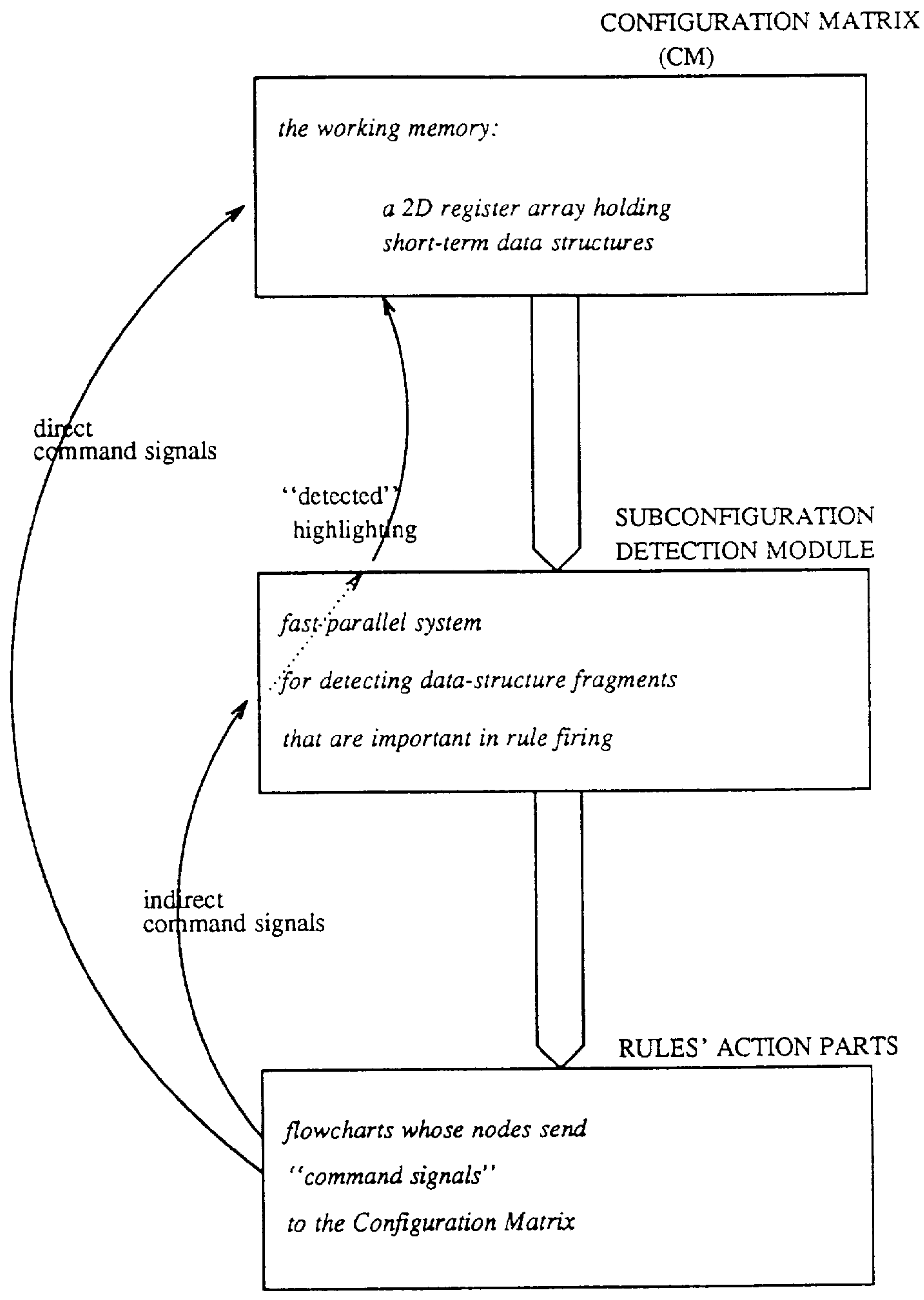
CONFIGURATION MATRIX (CM)
the working memory:
a 2D register array holding short-term data structures
direct command signals
"detected" highlighting
SUBCONFIGURATION DETECTION MODULE
fast parallel system
for detecting data-structure fragments
that are important in rule firing
indirect command signals
RULES' ACTION PARTS
flowcharts whose nodes send
"command signals"
to the Configuration Matrix

Fig. 1: Overall structure of the rule-based version of Conposit.

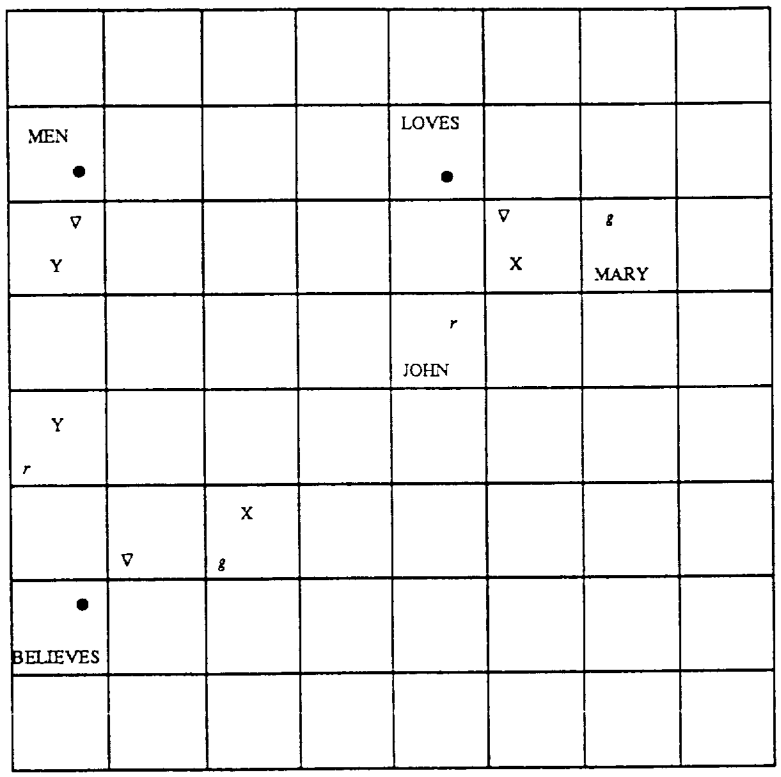

Fig. 2: CM register clumps for "Bill believes that John loves Mary".

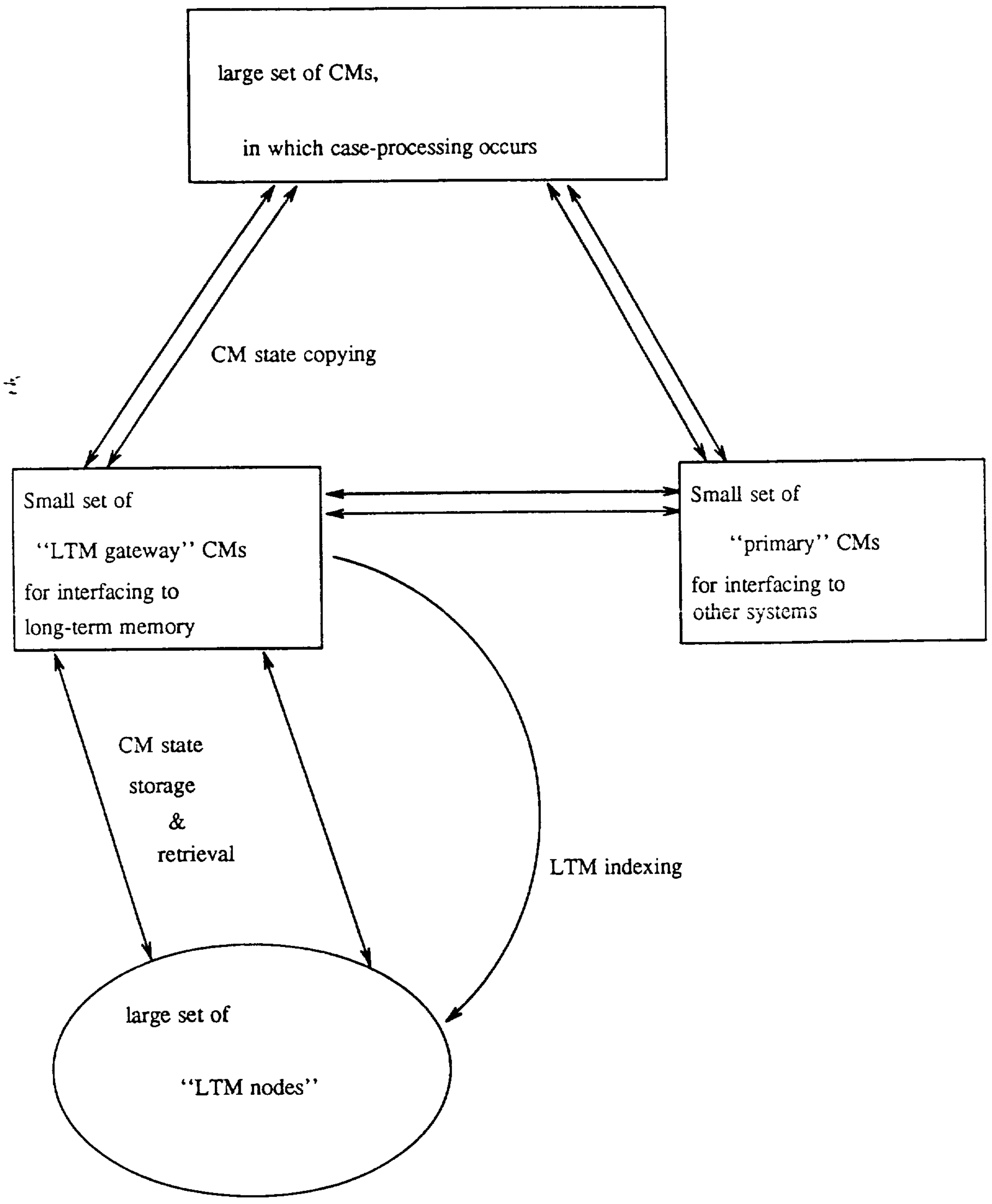

Fig. 3: Overall structure of the new, case-based version of Conposit.