Reasoning With Function Symbols In A Connectionist System

Venkat Ajjanagadde*
Department of Computer and Information Science
University of Pennsylvania

Abstract

One important problem to be addressed in realizing connectionist reasoning systems is that of dynamically creating more complex structured objects out of simpler ones during the reasoning process. In rule based reasoning, this problem typically manifests as the problem of dynamically binding objects to the arguments of a relation. In [1,7], we described a rule-based reasoning system based on the idea of using *synchronous activation* to represent bindings. As done in almost all other connectionist reasoning systems developed so far, there, we restricted our focus on the problem of binding only *static objects* to arguments. This paper describes how the synchronous activation approach can be extended to bind *dynamically created objects* to arguments, to form more complex dynamic objects. This extension allows the rule-based reasoning system to deal with function symbols. A forward reasoning system incorporating function terms is described in some detail. A backward reasoning system with similar capabilities is briefly sketched and the way of encoding long-term facts involving function terms is indicated. Several extensions to the work are briefly described, one of them being that of combining the rule based reasoner with a parallelly operating equality reasoner. The equality reasoner derives new facts by substituting equivalent terms for the terms occurring in the facts derived by the rule-based reasoner.

1 Introduction

Suppose, someone told us "At around 7PM last Thursday, as John was walking on Chestnut street,...". Virtually immediately after hearing this very small piece of explicitly stated information, we would have performed a lot of inferences, say, for example, that "John was awake", "He was moving", "His eyes were open", "His feet were touching the ground", "It was probably quite dark " and so on... . In doing this, we would have applied our general knowledge about 'walking', 'light conditions at around 7PM' etc. to make inferences about the particular case of John's walking. One of the well explored approaches of modelling this inferencing phenomenon is the following: Represent the general knowledge using rules such as[1],
walk-on(x,y)$\Rightarrow awake(x)$, $awake(x) \Rightarrow$ *open(eye-of(x))*, *walk-on(x,y)* $\Rightarrow$*touch(feet-of(x),y)* etc.. Now, given the fact *walk-on(john,chestnut-st)*, binding *john* and *chestnut-st* to appropriate variables in the rules, facts such as *awake(john)*, *touch(feet-of(john),chestnut-st)* can be derived.

We believe that this attempt to model the human inference process as reasoning with rules and facts is *a step* in the right direction. But, at the same time, we also find compelling reasons to adopt a connectionist framework in further explorations of this approach. One primary reason is that of inferencing speed. To characterize the knowledge underlying human cognition, a very very large number of rules of the form mentioned above will be

*This work was partially supported by NSF grants IRI 88-05465, MCS-8219196-CER, MCS-83-05211, DARPA grants N00014-85-K-0018 and N00014-85-K-0807, and ARO grant ARO-DAA29-84-9-0027. I am indebted to Lokendra Shastri, whose ideas and suggestions have greatly contributed to this work. John Barnden provided useful comments; my thanks to him.

[1]When not mentioned otherwise, the variables occurring in a rule are assumed to be universally quantified and have scope over the whole rule.

needed. In spite of possessing that vast a knowledge base, human beings perform a broad class of inferences involved in cognition extremely fast, within a few hundred milliseconds. To be acceptable, any computational account of cognition should be able to explain this remarkable phenomenon. One necessary (but not sufficient) condition for achieving such an efficiency is massive parallelism at the knowledge level - the kind of parallelism that is characteristic of connectionist models. Another important reason that suggests the use of connectionist approach is the brittleness of logical rules. Inherent to the connectionist methodology are elegant mechanisms for avoiding this brittleness in a natural fashion.

As a preliminary step towards the realization of a connectionist soft reasoning system, we have addressed the problem of how to reason with hard logical rules in a connectionist system[2]. That meant devising a scheme for encoding rules and facts as a connectionist network and finding a way of realizing the inference process as spread of activation in the network, there being no central controller. Unfortunately, in doing this, many challenging problems surface, which need to be solved.

One of the major problems to be addressed is "How do we represent structured objects *dynamically*[3], i.e., as temporary patterns of activity generated during network's operation?". Let us look at this problem more closely with an example of reasoning.

Consider the rules $fly(x,y,z) \Rightarrow move(x,y,z)$ and $move(x,y,z) \Rightarrow reach(x,z)$. Now, starting from the fact $fly(tweety, tree1, tree2)$ (i.e., $tweety$ flew from $tree1$ to $tree2$), using the first rule we can infer $move(tweety, tree1, tree2)$. From this newly derived fact, using the second rule, we can, in turn, infer $reach(tweety, tree2)$. In a connectionist network performing this reasoning, activity patterns representing the facts $move(tweety, tree1, tree2)$ and $reach(tweety, tree2)$ will have to be generated during the reasoning process.

Let us look at the problem of representing the dynamic objects $move(tweety, tree1, tree2)$, and $reach(tweey, tree2)$ more closely. These are structured objects formed by binding some objects to the arguments of relations. For example, $move(tweety, tree1, tree2)$ is formed by binding the objects $tweety$, $tree1$, and $tree2$ respectively to the first, second, and third arguments of the relation $move$. Thus, the problem of representing these facts boils down to the connectionist variable binding problem[3,8]: How do we represent the binding of an object to an argument of a relation?

In [1,7], we described what we refer to as the *synchronous activation approach* to represent argument bindings. Simply stated, the idea is just this: Corresponding to every argument of a predicate, have a distinct node in the network. Imagine every clock[4] cycle as divided into some number of slots, which we will refer to as *phases*. Associate distinct phases of the clock cycle with distinct objects participating in a reasoning episode[5]. Thus, in the example of reasoning given above, we would have chosen distinct phases, say, first, second, and third phases of clock cycles to correspond respectively to the objects $tweety$, $tree1$, and $tree2$. The binding of an object to an argument is represented by having the node corresponding to that argument become active in the phase associated with the object. More than one argument node can be active in a phase of a clock cycle denoting that the object associated with the phase is bound to all of those arguments.

Based on this idea of representing bindings, we developed a fairly powerful rule-based reasoning system[1,7]. This system, whose size is only linear in the size of the knowledge base, is capable of performing a large number of inferences involving rules and facts in parallel. The time taken to perform a single inference is proportional to the length of the derivation, and hence, optimal. However, as done in almost all connectionist reasoning systems developed so far, we had placed an important restriction on the kind of reasoning performed by the system. The

[2] Some ideas on how to render a degree of softness to the system thus arrived at, are mentioned later in the paper.

[3] Hence, the name *dynamic objects*, to objects represented that way.

[4] Actually, no global clock is necessary, as discussed in detail in [7]. However, for simplifying the discussion, here we are assuming that there is a global clock.

[5] Since the number of phases in a clock cycle is bounded, only a limited number of objects can participate in a particular reasoning episode. This restriction confirms with the psychological observation that human reasoning can focus only on a small number of objects at any time[6]. For detailed discussions of the neurological and psychological plausibilities of the synchronous activation approach to represent bindings, please refer to [7].

variable binding mechanism employed there was restricted to binding only *static objects* to arguments. For example, in the example of infering $move(tweety, tree1, tree2)$, and $reach(tweety, tree2)$ given above, the arguments of the relations are bound only to the static objects $tweety$, $tree1$, and $tree2$. An example of binding dynamic objects would be binding the dynamically created object $move(tweety, tree1, tree2)$ itself to an argument of some other other relation. The restriction that bound objects be static precluded us from having function symbols in rules and facts, since, to deal with function terms, the ability to bind dynamic objects to arguments is required. For example, consider the simple rule

walk-on(x,y) $\Rightarrow$*touch(feet-of(x),y)*

Now, suppose we have the fact *walk-on(john,street6)*. From this fact, we can infer the fact *touch(feet-of(john),street6)*. Notice that here, *feet-of(john)* is a dynamic object created during the reasoning process by binding the argument of the function symbol *feet-of* to the object $john$. In infering the fact *touch(feet-of(john),street6)*, the dynamic object *feet-of(john)* should itself be bound to the first argument of the relation '$touch$'.

In this paper, we describe how the synchronous activation approach can be extended to handle the above general kind of binding. That extension allows reasoning with function symbols and opens up interesting ways of enhancing the power of our earlier reasoning system, still retaining its nice features. Before we go into the details of these, we will digress a little to make a few comments on related research works.

To the best of our knowledge, no connectionist reasoning system has so far addressed the problem of reasoning with function symbols in any significant detail. Hence, our comments on related work is limited to the abilities of other proposed variable binding solutions to bind dynamic objects [6]. In [5], Lange and Dyer suggest the use of *signatures* to represent variable bindings. They permanently allocate a distinct signature to each static object and represent a binding by propagating the signature of the appropriate object to the argument to which it is bound. Now, to bind a dynamically created object to some argument, a signature will have to be recruited for the dynamic object on the fly during the reasoning process. Doing this appears very problematic. This is especially so, if signatures are learnt patterns, as suggested in [5]; that will mean that learning has to take place during an inference step. In [8], Smolensky addresses the variable binding problem in its abstract form and presents a tensor product based solution. He deals with the issue of nested bindings also. But, the way in which the proposed technique can be used in the context of reasoning remains to be seen. Barnden's 'Conposit'[2] is probably the only reasoning system other than the one presented in this paper, that is capable of handling nested bindings. However, the variable binding capabilities of Conposit come with a very significant sacrifice in knowledge level parallelism; in Conposit, only one rule can fire at a time. DCPS[9] is another reasoning system that is restrictive in its use of knowledge level parallelism. There does not appear to be any obvious way of extending the binding mechanism of DCPS to handle nested bindings.

2 Overview

Cognition requires both *forward reasoning* and *backward reasoning*. In forward reasoning, the system starts with a collection of facts and these facts trigger some rules leading to the inference of some other new facts. These newly derived facts in turn trigger some other rules and so on, the process leads to the derivation of a large set of facts that are deducible from the starting set of facts. A connectionist forward reasoning system dealing with function terms is discussed in some detail in section 3. That section describes the activity patterns chosen to represent function terms, the encoding of rules involving function symbols and how inferences involving function symbols is done in the network. Section 4 briefly sketches how a backward reasoning system incorporating function terms in rules and facts can be built. In that context, we indicate how long-term facts involving function terms (such as,

[6] Detailed comparisons of the synchronous activation approach with other proposed solutions, based on other relevant criteria, such as network size, reasoning speed, neurological plausibility etc., have been made elsewhere[7].

say, *hate(john,brother-of(tom))* are encoded in the network. Section 5 discusses some extensions to the work on reasoning with function symbols. One of the major extensions is that of combining the rule based reasoner with a parallelly operating equality reasoner.

3 A Forward Reasoning System that Deals with Function Terms

Before explaining how a reasoning system incorporating function terms is realized, let us discuss the dynamic representation of function terms.

A function term f of n arguments gets represented in the system using a relation R_f of $(n+1)$ arguments[7]. The first n arguments of R_f correspond to the n arguments of f respectively and the $(n+1)$th argument of R_f can be thought of as corresponding to the value of the function for those arguments. A dynamic object $f(c_1, c_2, ..., c_n)$[8] gets represented as follows: Recall from the discussion of section 1 that distinct objects participating in a reasoning process are associated with distinct clock phases. Suppose that the phases associated with the objects c_1,...,c_n are respectively p_1,...,p_n[9]. As said above, the first n arguments of R_f correspond to the arguments of f and hence are bound to c_1, c_2,...,c_n respectively. Using the synchronous activation approach, these bindings get represented by having the ith argument node of R_f become active in the p_ith phase of every clock cycle ($i = 1, ..., n$). Corresponding to the dynamic object $f(c_1, ..., c_n)$,[10] a free phase (i.e., a phase that is currently not assigned to any object) is assigned. The $(n+1)$th argument node of R_f is activated in this phase, thereby denoting that it is bound to the object $f(c_1, ..., c_n)$.

Summarizing, the following pattern of activity represents the function term $f(c_1, ..., c_n)$:

- The ith argument node of R_f becomes active in the p_ith phase of every clock cycle.
- A currently free phase (let it be p_*) is found and that phase is associated with the dynamic object $f(c_1, ..., c_n)$. The $(n+1)$th argument node of R_f will be active during the p_* phase of clock cycles.

Now, consider the problem of binding the object $f(c_1, ..., c_n)$ to an argument of some relation, say jth argument of the relation K. All that is to be done for this, is to activate the jth argument node of K in the phase p_* .

With that introduction, let us proceed to the encoding of rules involving function symbols. Consider the rule

$$P(x, y, z) \Rightarrow Q(f(y, x), z) \tag{1}$$

Associated with the relations P and Q, there are three and two argument nodes respectively (shown as diamond shaped nodes in Fig. 1)[11]. To represent the binary function f, a ternary relation R_f is used. The correspondences between the arguments of relations are denoted by the links between the appropriate argument nodes. For example, as per the above rule, the second argument of Q gets bound to the same object that binds the third argument of P. This is denoted by a link between the argument node a3 and the argument node a5 . Ignore the hexagonal box marked B for the time being.

Using the rule (1), starting from the fact $P(a, b, c)$, we can infer $Q(f(b, a), c)$. Let us see how this gets done with the rule encoding shown in Fig.1. Suppose that the objects a, b, and c are assigned the first, second, and third phases of clock cycles. Thus, the bindings in the starting fact are denoted by having the first, second, and third argument nodes of P active in the first, second, and third phases of clock cycles. The link from the argument node

[7] The reader is urged to remember the notation 'R_f'. In all the examples and figures, we use this notation.

[8] It is not required that c_is be distinct. In addition, c_is can be static or dynamic objects. Hence, nested function terms such as $f(g(h(a1, a2), a3))$ are allowed.

[9] If $c_i = c_j$, then, p_i will be equal to p_j.

[10] It may help to view this object as the value of the function f for the argument objects $c_1, ..., c_n$.

[11] We won't be explaining the role of the nodes drawn as pentagons in Fig.1. Due to space limitations, we have focussed on conveying the essential ideas and have omitted some details.

a3 to the node a5 causes a5 also to become active in the third phase of clock cycles - thereby, binding a5 to the object c, as desired.

The argument nodes a6 and a7 are connected respectively to the nodes a2 and a1. This causes a6 and a7 to become active in the second and first phases of clock cycles respectively. That is, a6 and a7 get bound respectively to b and a, thereby creating a part of the representation of $f(b,a)$. As discussed at the beginning of the section, now, a free phase in the clock cycle has to be found for the object $f(b,a)$. This is done by the hexagonal box labelled B, which we will refer to as *phase requester*. This box is not a single connectionist node; instead, it is a small piece of circuitry. There exists a global record maintainer that keeps track of the currently assigned phases. Box B communicates with this record maintainer to determine a free phase. We are suppressing the details of the phase requester and the record maintainer. For further discussions, it is assumed that the output of B is a pulse in the chosen free phase. The output of B goes to the third argument node of R_f. Thus, a8 becomes active in the phase chosen for the object $f(b,a)$. The link from a8 to a4, in turn, causes a4 to become active in the chosen phase. That is, the first argument of Q gets bound to the object $f(b,a)$. Now, we have the complete representation of the inferred fact $Q(f(b,a),c)$.

Above we discussed how a single rule is encoded and inference gets performed with that rule. Fig.2 shows how a collection of rules can be encoded. With this arrangement, an inferred fact in turn can spontaneously trigger other rules leading to the derivation of further facts. It is common to have one predicate occurring in the antecedent of many rules. When a fact involving a predicate is inferred, this triggers all those rules in which this predicate is the antecedent. Thus triggered rules derive further facts in parallel.

4 A Backward Reasoning System Dealing with Function Terms

Previous section described a *forward reasoner* incorporating function terms. Cognition also requires *backward reasoning*, where the task performed is essentially that of verifying whether a given fact is derivable from the rules and facts stored in the system. Two major tasks involved in performing backward reasoning are : (i) applying rules in the backward direction , (ii) checking whether a long-term fact is stored in the system.

The first of these tasks can essentially be achieved by having links in the reverse of the directions marked in Fig. 2 (A few other small changes will also be necessary.).

To see how the second task is performed, we would like to emphasize the two kinds of representations of facts: long-term facts are represented in the network as an interconnection of nodes while short-term facts get represented by a temporary pattern of activity. To check whether a particular long-term fact is present in the system, the pattern of activity corresponding to the short-term representation of that fact is induced in the network. The interconnection encoding a long-term fact is such that the activity of a particular node in the interconnection goes high when and only when the activity corresponding to the short-term representation of that fact is present in the network[12]. Such an interconnection corresponding to the long-term encoding of the fact $P(a,f(b))$ is shown in Fig.3. In figures, we use links with dark circles at their ends to denote inhibitory connections and links with solid arrows at their ends to denote enabling connections. Nodes marked with the names of the constants a and b are referred to as *constant nodes*; they will be active in the respective clock phases assigned to these objects in a reasoning episode. Node marked N is a coincidence detector; it sends a high output if and only if all of its inputs are active and synchronous. The node marked K sends a continuous high output. Node M, drawn as a horizontal pentagon is a *temporal-AND* node, which becomes active if and only if it receives activation throughout a clock cycle. Recalling the short-term representation of function terms discussed earlier, one can easily verify that the condition for M to become active gets satisfied when and only when the pattern of activity corresponding to the short-term representation of the fact $P(a,f(b))$ is present in the network.

[12]Thus, the activity of this node can be used in answering the question whether a particular long-term fact exists or not.

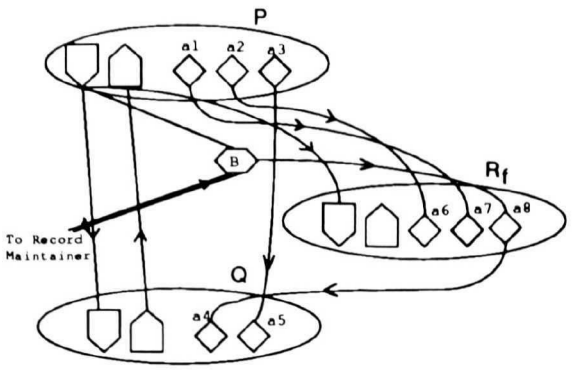

Figure 1: Encoding of the rule $P(x, y, z) \Rightarrow Q((f(y, x), z))$.

Using the above discussed technique for encoding long-term facts involving function terms, a backward reasoning system can be realized along the lines described in [1,7].

5 Extensions of the Work

5.1 Combining with an equality reasoner

In previous sections we discussed how a rule-based reasoning system incorporating function terms can be realized. Such a rule-based reasoner can be elegantly interfaced with a parallelly operating equality reasoner to obtain a more powerful reasoner. The operation of these two components is as follows: The main reasoner derives new facts and the equality reasoner substitutes equivalent terms for the terms occurring in those facts to derive some other facts. Those new facts derived by the equality reasoner can in turn trigger rules in the main reasoner. Here, we will only sketch some of the ideas involved in the realization of such a hybrid reasoner and will report the details elsewhere.

The equality reasoner operates with two kinds of equalities. Substitutions involving these two types are accomplished in different ways. The first type of equality states that an entity described using a function symbol is same as a known domain individual. An example of this kind of equality is *maternal-uncle-of(tom)=dave*. Reasoning with this kind of equality is accomplished by extending the mechanism for answering *wh-queries* reported in [7].

The second kind of equality states equalities between two entities both of them being described using function symbols. An example of this kind of equality is $\forall x$ *brother-of(mother-of(x))=maternal-uncle-of(x)*. A piece of circuitry to apply this equality to substitute *maternal-uncle-of(x)* for *brother-of(mother-of(x))* is shown in Fig. 4. This is accomplished by viewing this operation as the application of the rule *brother-of(p,q)*$\wedge$*mother-of(q,r)* $\Rightarrow$*maternal-uncle-of(p,r)*. Network realization of this is analogous to the encoding of conjunctive antecedent rules described in [1,7] and is indicated in Fig.4. There, the rectangular shaped node marked B is a coincidence detector whose output will be high if and only if its two inputs are active and synchronous. This node ensures that the same object binds the second argument of *brother-of* and the first argument of *mother-of*, as required by the above rule.

In the derivation of a particular fact both of the above two types of equalities may be made use of. Thus, if the main reasoner derives *live-in(brother-of(mother-of(tom)),boston)*, then the network may first derive *live-in(maternal-uncle-of(tom),boston)* using the second kind of equality and then derive *live-in(dave,boston)* using the first kind of equality.

5.2 Evidential reasoning

As mentioned in section 1, our primary focus so far has been on exploring how reasoning with logical rules and facts can be efficiently realized in a connectionist network. In achieving this, we assumed all the link weights to be unity. A certain degree of softness can be rendered to this reasoning system by making the rules to be probabilistic and associating a measure of certainity with the facts. These modifications can be easily realized in the network architecture developed, by having non-uniformly weighted links.

5.3 Multiple Dynamic Facts

In the description of the system presented in this paper, we have assumed that only one dynamic fact involving a relation is present at any time. Though this is a common assumption made in connectionist reasoning systems[4,5], there are situations in which this restriction is unacceptable. In [7], we discuss some techniques of relaxing this restriction in the case of our system.

6 Conclusion

We described a connectionist rule-based reasoning system incorporating function terms. The system is extremely efficient both in network size and the time taken to perform inferences. The extended variable binding capability introduced in the system opens up many interesting avenues for realizing *broad, yet, efficient, reasoning* - the characteristic feature of human cognition. They are currently being investigated.

References

[1] Ajjanagadde, V.G., and Shastri, L., Efficient inference with multi-place predicates and variables in a connectionist system, *Proceedings of the Cognitive Science Conference*, August 89, pp. 396-403.

[2] Barnden, J., Neural-net implementation of complex symbol-processing in a mental model approach to syllogistic reasoning, *Proceedings of IJCAI-89*, pp. 568-573.

[3] Feldman, J.A., Dynamic connections in neural networks, *Bio-Cybernetics*, 46:27-39, 1982.

[4] Hinton, G., Implementing semantic networks in parallel hardware, In G.Hinton and J.Anderson (Eds.),*Parallel Models of Associative Memory*, Erlbaum, 1981.

[5] Lange, T., and Dyer, M., High-Level inferencing in a Connectionist Neural Network. Technical Report, UCLA-AI-89-12, October. Computer Science Department, UCLA, 1989.

[6] Miller, G.A., The magical number seven, plus or minus two: Some limits on our capacity for processing information, *The Psychological Review*, 63(2), March 1956, pp. 81-97.

[7] Shastri, L. and Ajjanagadde, V., From simple associations to systematic reasoning: A connectionist representation of rules, variables, and dynamic bindings, Tech. Report, Dept. of Computer Science, Univ. of Pennsylvania, January 90.

[8] Smolensky, P., On variable binding and the representation of symbolic structures in connectionist systems, Technical Report CU-CS-355-87, Department of Computer Science, University of Colorado at Boulder, 1987.

[9] Touretzky, D. and Hinton, G., A Distributed Connectionist Production System. *Cognitive Science*, 12(3), pp. 423-466.

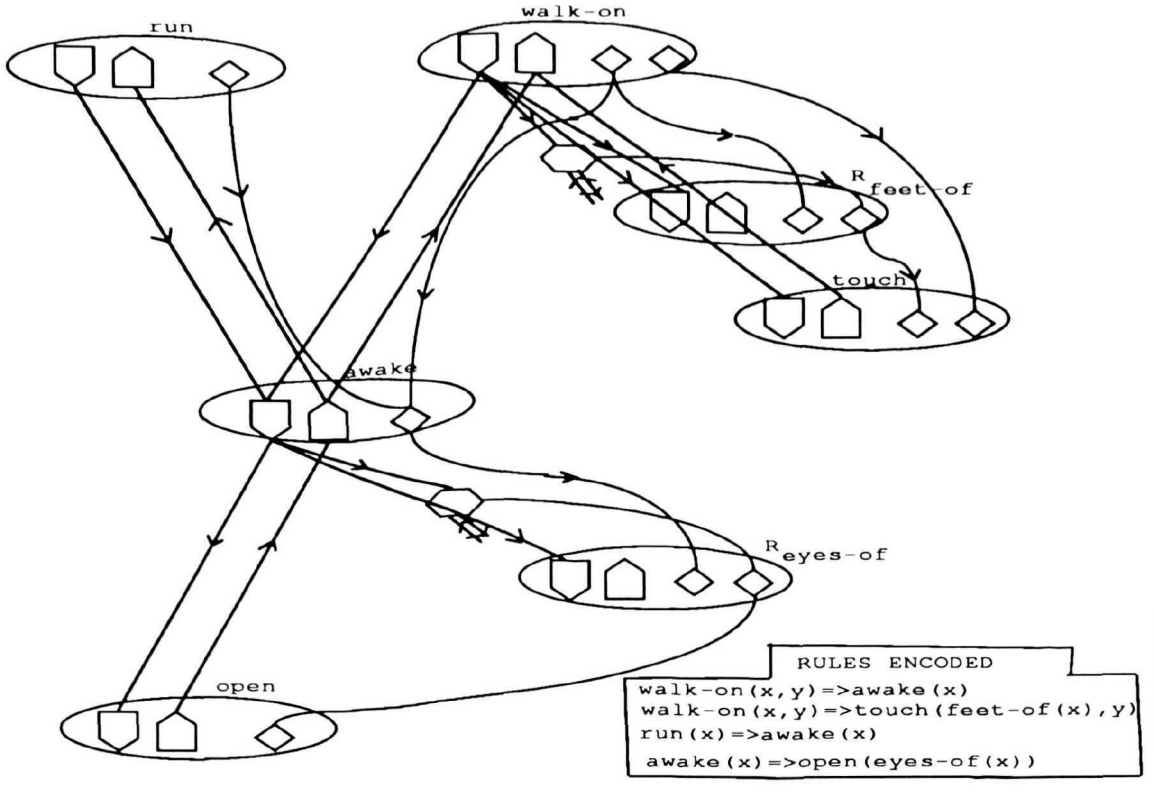

Figure 2: An example network.

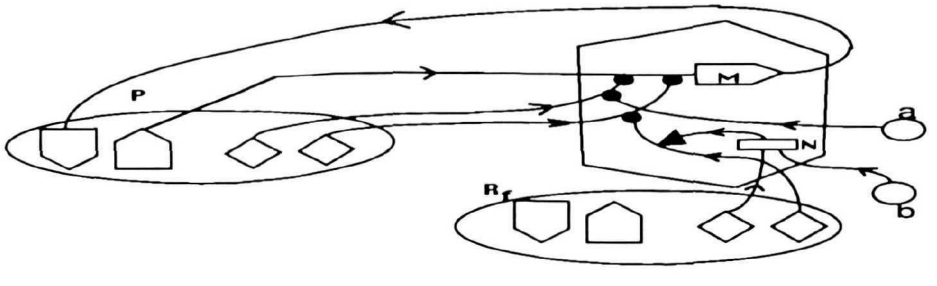

Figure 3: Encoding of the long term fact $P(a, f(b))$.

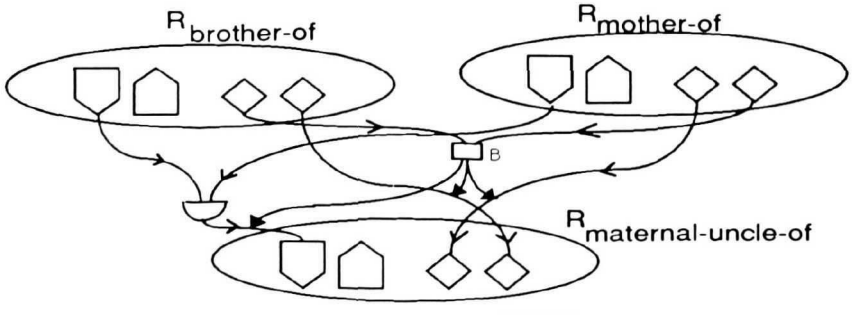

Figure 4: Interconnection for substituting $maternal\text{-}uncle\text{-}of(x)$ for *brother-of(mother-of(x))*.