An Optimally Efficient Limited Inference System*

Lokendra Shastri and Venkat Ajjanagadde

Department of Computer & Information Science
University of Pennsylvania
Philadelphia, PA 19104, USA
shastri@cis.upenn.edu

Abstract

This paper describes a knowledge representation and reasoning system that performs a limited but interesting class of inferences over a restricted class of first-order sentences with optimal efficiency. The proposed system can answer *yes-no* as well as *wh*-queries in time that is only *proportional* to the *length* of the shortest derivation of the query and is *independent* of the *size* of the knowledge base. This work suggests that the expressiveness and the inferential ability of a representation and reasoning systems may be limited in unusual ways to arrive at extremely efficient yet fairly powerful knowledge based systems.

Introduction

Research in artificial intelligence has made it abundantly clear that tremendous computational activity underlies even the most commonplace intelligent behavior. For example, language understanding, a task that we usually perform so effortlessly and effectively, depends upon the agent's ability to disambiguate word senses, recover the phrase structure of input sentences, resolve anaphoric references, impose selectional restrictions, recognize speaker's plans, perform numerous predictions, and generate explanations.

Within the knowledge representation and reasoning paradigm, most of the above computations are viewed as *inferences*. Such a view, however, leads to the following paradox: A generalized notion of inference is intractable, yet the human ability to perform cognitive tasks such as language understanding in real-time suggests that we are capable of performing a wide range of inferences with extreme efficiency.

The success of AI critically depends on resolving the above paradox. Fortunately, one is only faced with an *apparent* paradox because humans are not general purpose reasoners — we can only perform a limited (but perhaps a fairly rich) class of inferences with extreme efficiency. This suggests two alternate research strategies for addressing the paradox: One may attempt to characterize the class of inference that people perform with great efficiency and try and discover appropriate representations, algorithms, and computational architectures to perform these inferences with requisite efficiency. Unfortunately, a precise characterization of what humans can infer with ease does not appear to be in sight. An alternate strategy would be to develop a complexity theory of reasoning by identifying what classes of inference can be performed with what degree of efficiency. The work reported in this paper conforms to this latter strategy. Such a strategy has been pursued by several researchers (for example, [6, 10, 13, 9, 14, 7]), and their work has covered a wide band of the complexity spectrum: Thus we have decidability results (e.g., [13]) as well as characterizations of simple knowledge representation systems that merely perform database retrieval [10].

Most of the above results have been surprisingly negative and have shown that even fairly restricted kinds of reasoning turns out to be intractable (for example, refer to [4, 7]). Yet the human ability to understand language in real-time clearly suggests that *there does exist a fairly rich class of reasoning that humans can perform effortlessly and within seconds.* To appreciate the richness of such reasoning, consider the following sentence: "John seems to have suicidal tendencies, he has joined the Columbian drug enforcement agency". Even though the reader would have understood the above sentence spontaneously, and perhaps within a few seconds, a careful analysis would reveal that doing so requires performing a number of fairly elaborate inferences. For convenience let us agree to refer to such spontaneous and fast reasoning as *reflexive* reasoning.[1]

*This work was 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.

[1] It is as if such reasoning is a *reflex* response of our cognitive apparatus. Clearly, all human reasoning is not reflexive.

Reflexive Reasoning

In evaluating the complexity of reflexive reasoning it must be recognized that such reasoning is performed with reference to an extremely large body of knowledge. We believe that a conservative estimate of the number of 'rules' and 'facts' required to encode all relevant aspects of the domain of common sense will easily run into several million (perhaps, even more).[2] In view of the above it appears that the time complexity of an inference algorithm for reflexive reasoning should *at worst* be *sublinear* in $|KB|$ – where $|KB|$ is the size of the knowledge base – and perhaps even be *independent* of $|KB|$.

Reflexive reasoning introduces a very strong notion of effectiveness, one that seems formidable in view of the negative results cited above. In this paper we report an encouraging result. We identify a class of inference that is computable in time that is not only sublinear in $|KB|$ but is also, in a sense, *optimal*. We describe a knowledge representation and reasoning system that can encode a restricted class of first-order sentences and answer a class of queries in time that is only *proportional* to the *length* of the shortest derivation of the query and is *independent* of $|KB|$. This work suggests that there exist interesting points in the tradeoff-continuum between computational effectiveness and inferential/expressive power. It also demonstrates that it is possible to arrive at extremely efficient yet fairly powerful knowledge representation systems by explicitly recognizing the symbiotic relationship that exists between expressiveness, effectiveness of inference, the choice of representation (data-structures), and the underlying model of computational.

Functional Specification

The knowledge representation system encodes *rules* and *facts* of the following form:

$$\forall x_1, ..., x_m \; [P_1(...) \wedge P_2(...) ... \wedge P_n(...) \Rightarrow \exists z_1, ... z_l \; Q(...)]$$

The arguments of P_i's are elements of $\{x_1, x_2, ... x_m\}$. An argument of Q is either an element of $\{x_1, x_2, ... x_m\}$, or an element of $\{z_1, z_2, ... z_l\}$, or a constant. It is required that any variable occurring in multiple argument positions in the antecedent of a rule must also appear in its consequent.

Facts are assumed to be partial or complete instantiations of predicates. Thus facts are atomic formulae of the form $P(t_1, t_2 ... t_k)$ where t_i's are either constants or distinct existentially quantified variables.

A query has the same form as a fact. A query all of whose arguments are bound to constants, corresponds to a *yes-no* question. On the other hand, a query with existentially quantified variables corresponds to a *wh*-query and answering such a query involves findings variable binding(s) for which the query follows from the rules and facts encoded in the system.

It can be shown that the knowledge representation system responds soundly provided a query satisfies the following conditions:[3]

1. The number of distinct constants specified in the query does not exceed Ω, where Ω is an implementation parameter (see Section 3).

2. Any rule that participates in the derivation of the query must obey the following constraint: Any variable occurring in *multiple* argument positions in the *antecedent* of such a rule must get bound during the reasoning process (this, via backward chaining).

3. During the processing of the query, each predicate may only be instantiated with one set of argument bindings. This restriction, however, only applies to run-time or 'dynamic' instantiations of predicates and not to 'long-term' facts stored in the system. Recently, we have extended the system to allow it to represent up to k – where k is a system parameter – dynamic instantiations of *each* predicate during the processing of a query [11]. (This extension is not discussed in this paper.)

If the above conditions are met, the system answers *yes* to all *yes-no* queries that follow from the encoded rules and facts in time proportional to the *length* of the *shortest* derivation of the query. The system obeys the closed world assumption and produces a *no* answer in time proportional to d, where d equals the diameter of the inferential dependency graph associated with the rule-base (see Section 3). *wh*-queries are also answered in time proportional to d. Finally, the *space* complexity of the system is just *linear* in $|KB|$.

Cognitive Significance

Reflexive reasoning takes place with reference to a large body of knowledge and even though each reasoning episode may involve a large number of rules and facts, there is considerable psychological evidence to suggest that most cognitive tasks performed efficiently without props involve only a *small number of distinct entities*. A reasonable estimate of the maximum number of distinct entities we can deal with at a time is around seven [12]. It must be emphasized that the limit is on the number of distinct entities and not on the number of variable (i.e.,

[2] Thorpe and Imbert [16] argue that even the number of visually identifiable entities is around a hundred thousand!

[3] The system is incomplete and the above conditions characterize its incompleteness.

role) bindings that these entities participate in during an episode of reasoning. We believe that the first condition listed in the previous section is consistent with this aspect of human reasoning and a psychologically plausible value of Ω may be around seven.

We also conjecture that any given episode of reflexive reasoning does not require the same predicate to be dynamically instantiated more than k times, where a psychologically plausible value of k may be as low as three to five (observe that k greater than 1 allows for bounded recursion). Thus the third limitation of the reasoning system is also well motivated.

In our view, the KB underlying reflexive reasoning primarily encodes an agent's *long-term* and *stable* knowledge about the world. Although new rules do get added to an agent's KB, the assimilation of a new rule *in a form that allows its participation in reflexive reasoning* takes time – not seconds, but perhaps minutes, days or even months. It is in this context that one must evaluate the realization of the reasoning system described below.

Finally, the realization of the reasoning system is also biologically plausible in that it strictly adheres to the core features of the connectionist model. At the same time, there exists neurophysiological evidence to suggest that the basic mechanisms used below, namely, the propagation of rhythmic patterns of activity, play a role in the representation and processing of information in the animal brain. For a detailed discussion, refer to [15].

Realization of the Reasoning System

The representation and reasoning system is realized as a massively parallel network of simple processing elements (nodes). A major technical problem that must be solved in realizing such a 'connectionist' reasoning system is the run-time (dynamic) creation and propagation of variable bindings. The proposed system solves this problem while still using extremely simple oscillatory nodes (see next section).

Ballard [2] was the first to propose a massively parallel inference system. He however, required that *all possible variable bindings be explicitly pre-wired* into a network. This requirement is too severe and unrealistic and greatly limits the generality of his system. Touretzky and Hinton's DCPS does represent dynamic bindings [17] but its ability to do so is very limited. First, DCPS can only deal with *single variable* rules. Second, even though DCPS is a parallel system, it allows only *one* rule to fire at a time, and hence, does not satisfy the efficiency requirements of reflexive reasoning. A detailed comparison of our system with other massively parallel reasoning systems (for example, ROBIN, CONPOSIT, and NETL [8, 3, 5]) may be found in [15].

Conceptually, the proposed encoding of the knowledge base amounts to creating a directed *inferential dependency* graph: Each predicate argument is represented by a node in this graph and each rule is represented by links from nodes denoting the arguments of the consequent predicate to nodes denoting the arguments of the corresponding antecedent predicate. Facts are small networks attached to their respective predicates nodes. We describe the encoding with the help of an example. For simplicity we only describe the realization of single antecedent rules without constants and existentially quantified variables in the consequent.

Fig. 1 illustrates the encoding of the following *facts* and *rules*.

1. $\forall x, y, z\ [give(x, y, z) \Rightarrow own(y, z)]$
2. $\forall x, y\ [buy(x, y) \Rightarrow own(x, y)]$
3. $\forall x, y\ [own(x, y) \Rightarrow can\text{-}sell(x, y)]$
4. $give(John, Mary, Book1)$
5. $buy(John, x)$
6. $own(Mary, Ball1)$

The encoding makes use of two types of nodes. These are ρ-btu nodes (depicted as circles) and τ-and nodes (depicted as pentagons). The computational behavior of these nodes is as follows:

A ρ-btu is a phase-sensitive binary threshold unit. When such a node becomes active, it produces an oscillatory output in the form of a pulse train that has a period π and pulse width ω. The timing (or the *phase*) of the pulse train produced by a ρ-btu node is precisely governed by the phase of the input to the node. A τ-and node acts like a *temporal and* node. Such a node also oscillates with the same frequency as a ρ-btu node except that it becomes active only if it receives *uninterrupted* activation over a whole period of oscillation. Furthermore, the width of the pulses produced by a τ-and node equals π. The implementation parameter Ω that governs the maximum number of distinct entities that may participate in the derivation of a *yes-no* query equals ω/π (assume integer divide).

The output pulse of a node propagates along every link emanating from the node. The encoding also makes use of *inhibitory modifiers*. An inhibitory modifier is a link that impinges upon and inhibits another link. Thus a pulse propagating along an inhibitory modifier will block the propagation of a pulse propagating along the link it impinges upon. In Fig. 1, inhibitory modifiers are shown as links ending in dark blobs.

Each constant in the domain is encoded by a ρ-node. An n-ary predicate is encoded by a pair of τ-and nodes and n ρ-btu nodes, one for each of the n arguments. One of the τ-and nodes is referred to as the *enabler* and the other as the *collector*. As a matter of convention, an *enabler* always points upwards and is named *e:[predicate-name]*. A *collector* always points downwards and is named *c:[predicate-name]*.

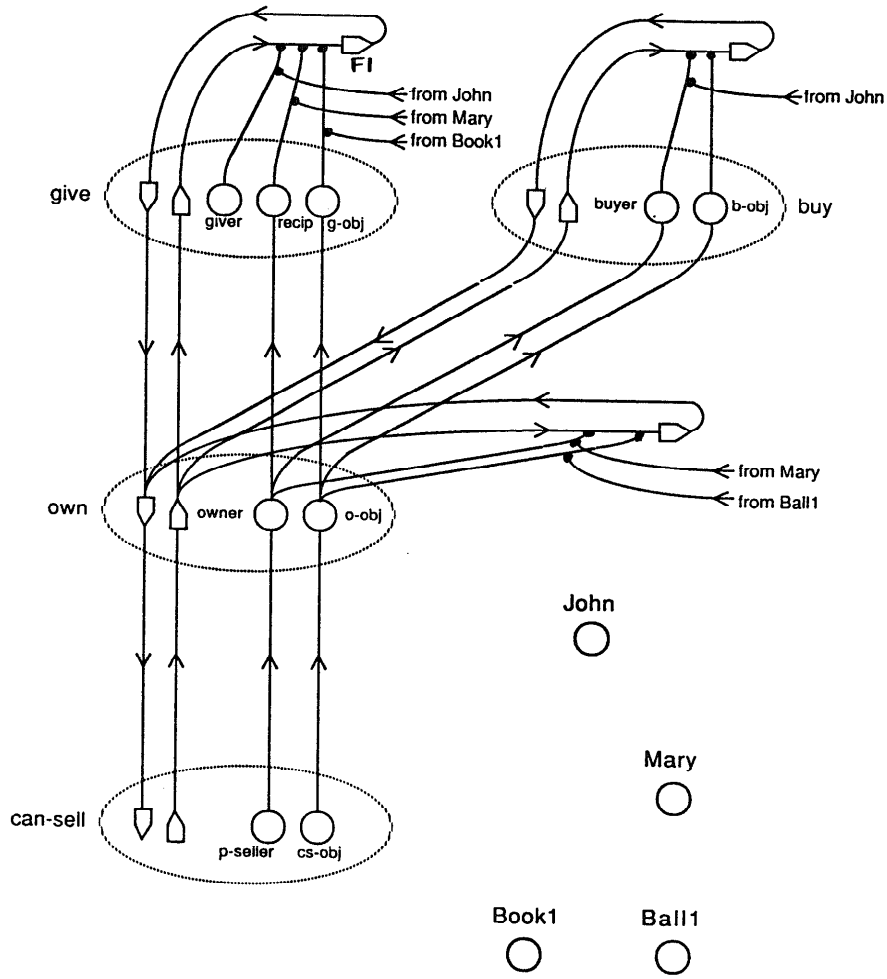

Figure 1: An example encoding of rules and facts

A rule is encoded by connecting the *collector* of the antecedent predicate to the *collector* of the consequent predicate, the *enabler* of the consequent predicate to the *enabler* of the antecedent predicate, and by connecting the argument nodes of the consequent predicate to the argument nodes of the antecedent predicate in accordance with the correspondence between these arguments specified in the rule (refer to Fig. 1.)

A fact is encoded using a τ-and node that receives an input from the enabler of the associated predicate. This input is modified by inhibitory modifiers from the argument nodes of the associated predicate. If an argument is bound to a constant in the fact then the modifier from such an argument node is in turn modified by an inhibitory modifier from the appropriate constant node. The output of the τ-and node is connected to the *collector* of the associated predicate (refer to the encoding of the fact $give(John, Mary, Book1)$ and $buy(John, x)$ in Fig. 1.)

The number of nodes required to encode a knowledge base, (i.e., the space complexity) is only *linear* in $|KB|$. Specifically, the number of nodes required are $O(r+f+a+c)$, where r is the number of rules, f is the number of facts, a is the total number of predicate arguments and c is the number of constants in the domain. The number of links required is also only linear in $|KB|$. Specifically, the number of links required is $O(r1+f1)$, where $r1$ is the number of rules weighted by the number of predicate arguments occurring in each rule, and $f1$ is the number of facts weighted by the number of arguments in the predicate associated with each fact.

Inference Process

Reasoning in the proposed system is the transient but systematic flow of *rhythmic* patterns of activation, where

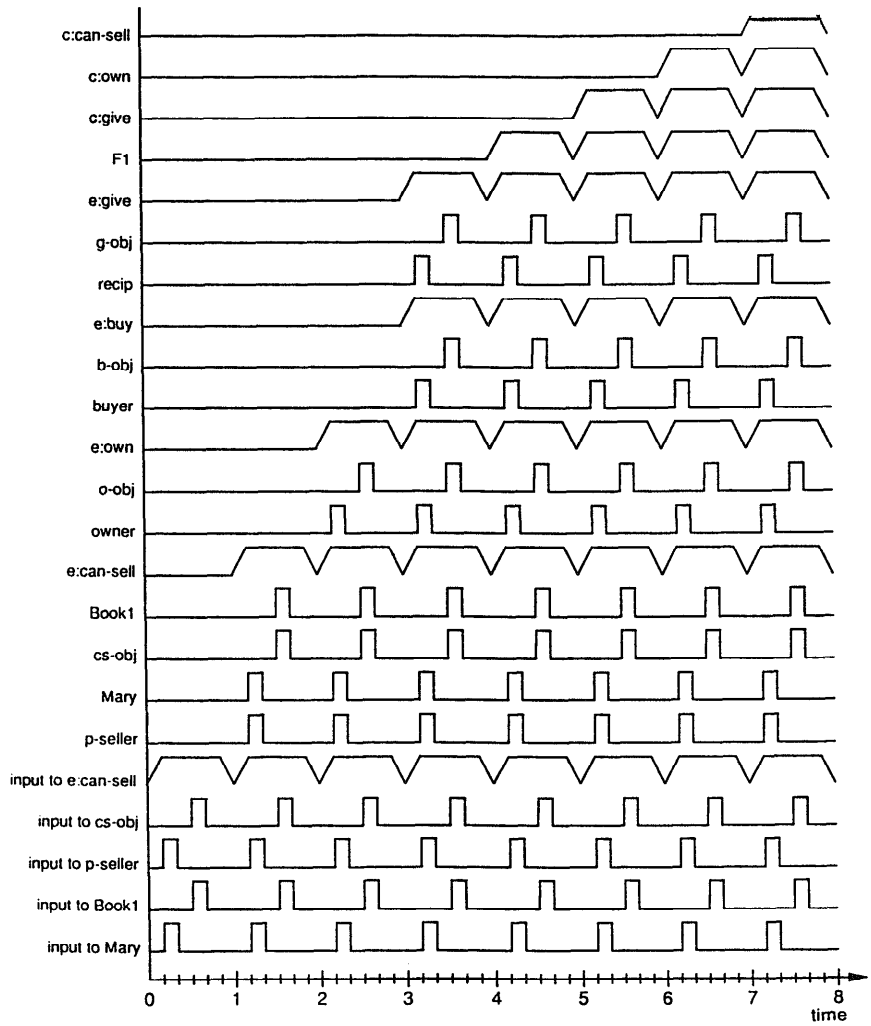

Figure 2: Activation trace for the query *can-sell(Mary, Book1)*?

each *phase* (or time-slice) in the rhythmic pattern corresponds to a distinct *constant* involved in the reasoning process and where variable bindings are represented as the *in-phase* (i.e., synchronous) firing of appropriate argument and constant nodes. A fact behaves as a temporal pattern matcher that becomes 'active' when it detects that the bindings corresponding to it are present in system's pattern of activity. Finally, rules are interconnection patterns that propagate and transform rhythmic patterns of activity. Below we describe the reasoning process in greater detail, complete details may be found in [15].

We first consider *yes-no* queries. The inference process may be thought of as consisting of two overlapping but conceptually distinct stages. The first stage corresponds to a parallel breadth-first exploration of the potentially huge inferential dependency graph. During this stage, all the facts that are relevant to the proof of the query become active. In the second stage the actual proof is constructed: activation from the relevant facts flows downwards along *collector* nodes to produce an answer to the query. A *yes* answer corresponds to the activation of the *collector* node of the query predicate.

Posing a Query: Specifying Variable Bindings

Posing a query to the system involves specifying the query predicate and the argument bindings specified in the query. In the proposed system this is done by simply activating the relevant nodes in the manner described below. In particular, *posing a query to the system does not involve any hidden time or space costs* such as rewiring of the network, or addition of new nodes and links.

Let us choose an arbitrary point in time – say, t_0 – as our point of reference for initiating the query. We

assume that the system is in a quiescent state just prior to t_0. The query predicate is specified by activating the *enabler* of the query predicate, with a pulse train of width and periodicity ω starting at time t_0.

The argument bindings specified in the query are communicated to the network as follows:

- Let the argument bindings in the query involve k distinct constants: $c_1, ..., c_k$. With each of these k constants, associate a delay δ_i such that no two delays are within ω of one another and the longest delay is less than $\pi - \omega$. Each of these delays may be viewed as a distinct *phase* within the period t_0 and $t_0 + \pi$.
- The argument bindings of a constant c_i are indicated to the system by providing an oscillatory pulse train of pulse width ω and periodicity π starting at $t_0 + \delta_i$, to c_i and all arguments to which c_i is bound. This is done for each constant c_i $(1 \leq i \leq k)$ and amounts to representing argument bindings by the *in-phase or synchronous activation of the appropriate constant and argument nodes.*

An Example

Once the query is posed, a parallel search for facts that are relevant to the proof of the query ensues. We illustrate this process with the help of an example (refer to Fig. 1.) Consider the query *can-sell(Mary,Book1)*. This query is posed by providing inputs to the constants *Mary* and *Book1*, the arguments *p-seller*, *cs-obj* and the *enabler e:can-sell* as shown in Fig. 2. (In the illustration, Ω has been assumed to be 6.) *Mary* and *p-seller* receive in-phase activation and so do *Book1* and *cs-obj*. Let us refer to the phase of activation of *Mary* and *Book1* as phase-1 and phase-2 respectively. As a result of these inputs, *Mary* and *p-seller* will fire synchronously in phase-1 of every period of oscillation, while *Book1* and *cs-obj* will fire synchronously in phase-2 of every period of oscillation. The node *e:can-sell* will also oscillate and generate a pulse train of periodicity and pulse width π.

The activations from the arguments *p-seller* and *cs-obj* reach the arguments *owner* and *o-obj* of the predicate *own*, and consequently, starting with the second period of oscillation, *owner* and *o-obj* become active in phase-1 and phase-2, respectively. At the same time, the activation from *e:can-sell* activates *e:own* (Refer to Fig. 2). The system has essentially, created dynamic bindings for the arguments of predicate *own*. *Mary* has been bound to the argument *owner*, and *Book1* has been bound to the argument *own-object*. These newly created bindings in conjunction with the activation of *e:own* can be thought of as encoding the query *own(Mary,Book1)* (i.e., 'Does Mary own Book1?')!

The τ-and node associated with the fact *own(Mary, Ball1)* does not match the query and remains inactive. Observe that during phase-2, the activation from *e:own* going into the τ-and node is blocked by the inhibitory activation from the argument *owner*.

The activations from *owner* and *o-obj* reach the arguments *recip* and *g-obj* of *give*, and *buyer* and *b-obj* of *buy* respectively. Thus beginning with the third period of oscillation, arguments *recip* and *buyer* become active in phase-1, while arguments *g-obj* and *b-obj* become active in phase-2. In essence, the system has created new bindings for the predicates *can-sell* and *buy* that can be thought of as encoding two new queries: *give(x,Mary,Book1)* (i.e., 'Did someone give Mary Book1?'), and *buy(Mary,Book1)* (i.e., 'Did Mary buy Book1?').

The τ-and node associated with the fact *buy(John, x)* does not become active because the activation from *e:buy* is blocked by the inhibitory activations from the arguments *buyer* and *b-obj*. The τ-and node associated with the fact *give(John,Mary,Book1)* (this is the τ-and node labeled F1 in Fig. 1), however, does become active as a result of the uninterrupted activation from *e:give*. The inhibitory inputs from *recip* and *g-obj* are blocked by the in-phase inputs from *Mary* and *Book1*, respectively. The activation from this τ-and node causes *c:give*, the *collector* of *give*, to become active and the output from *c:give* in turn causes *c:own* to become active and transmit an output to *c:can-sell*. Consequently, *c:can-sell*, the *collector* of the query predicate *can-sell*, becomes active resulting in an affirmative answer to the query *can-sell(Mary,Book1)*. (refer to Fig. 2).

Encoding Complex Rules

A rule with conjunctive predicates in the antecedent, i.e., a rule of the form $P_1(...) \wedge P_2(...) \wedge ...P_m(...) \Rightarrow Q(...)$, is encoded using an additional τ-and node that has a threshold of m. The outputs of the *collector* nodes of $P_1, ..., P_m$ are connected to this node which in turn is connected to the *collector* of Q. This additional node becomes active if and only if it receives inputs from the *collector* nodes of all the m antecedent predicates. The interconnections between the argument nodes of the antecedent and consequent predicates remain unchanged.

The encoding of rules and facts described in the previous section assumes that constants or existentially quantified variables do not appear in the consequent of a rule. It also assumes that the same variable does not occur in multiple argument positions in the consequent of a rule. The encoding of such rules can be carried out by very simple mechanisms that involve detecting whether appropriate nodes are firing in synchrony or not. A complete description may be found in [15].

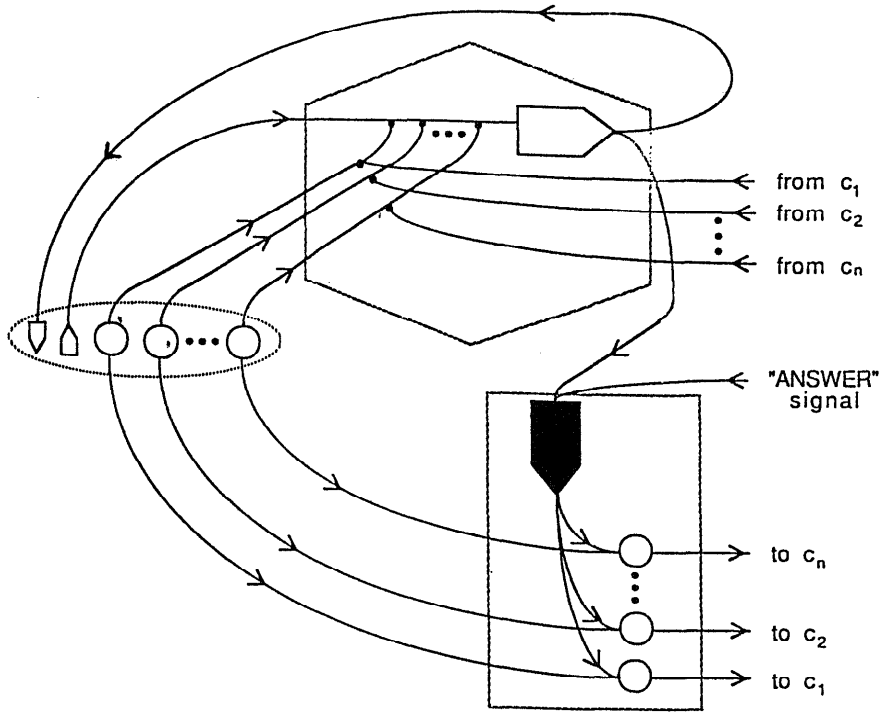

Figure 3: Augmented encoding of a fact in order to support answer extraction

Answering "wh-"queries

In this section, we will discuss a way of extending the system to answer *wh*-queries. Consider the proof of the query $can\text{-}sell(Mary, x)$ with respect to the network shown in Fig. 1. In the process of proving this query the two relevant facts $own(Mary, Ball1)$ and $give(John, Mary, Book1)$ will become active. The answer to the *wh*-query 'What can Mary sell?', can be obtained by simply identifying the constants that are bound to the arguments *g-obj* and *b-obj*, respectively, of the two active facts. This is not a coincidence — notice that the arguments *g-obj* and *b-obj* are precisely the arguments that map to the unbound argument *cs-obj* of *can-sell* via the rules encoded in the system. The system can easily extract this information by making use of the same binding propagation mechanism it uses to map arguments bound in the query. A straightforward way of doing so is to posit a separate *answer extraction* stage that is carried out after the *yes-no* query associated with the *wh*-query has produced a *yes* answer. For example, given the query 'What can Mary sell?' the system first computes the answer to the *yes-no* query 'Can Mary sell something?' and identifies the facts $own(Mary, Ball1)$ and $give(John, Mary, Book1)$ that lead to a *yes* answer. The answer extraction stage follows and picks out the constants $Ball1$ and $Book1$ as the answers.

The representation of a fact is augmented as shown in Fig. 3. in order to support answer extraction. The representation of a fact involving an n-ary predicate is modified to include $n + 1$ additional nodes: for each of the n arguments of the associated predicate there exists a ρ-btu node with a threshold of two. For convenience we will refer to such a node as a *binder* node. The other node (shown as a filled-in pentagon) is like a binder node, except that once active, it remains so — even after the inputs are withdrawn. This node, which we will refer to as a *latch* node, receives an *Answer* input in addition to an input from the τ-and node of the associated fact.

At the end of the first stage, the outputs of the τ-and nodes of all the relevant facts would be active. The output of these τ-and nodes in conjunction with the *Answer* signal will turn on the associated *latch* nodes and provide one of the two inputs to the *binder* nodes. If the associated *yes-no* query results in a *yes* answer, the answer extraction stage is initiated. Inputs relating to the first stage are withdrawn and the relevant *unbound* argument of the query predicate a_i is activated in a distinct phase. In addition a network wide *Answer* signal is also propagated. The activation of unbound query arguments results in a phase-sensitive propagation of activation and eventually leads to the activation of arguments associated with facts relevant to the query. This provides an input to the appropriate *binder* nodes of these facts. As the *binder*

nodes were already receiving an input from a *latch* node, they become active and produce a phase-sensitive output that in turn activates the associated constants in-phase with a_i. The answer to the *wh*-query will be precisely those constants that are active in-phase with a_i. The time taken by the answer extraction step is bounded by the depth of the inferential dependency graph.

Extensions

The reasoning and expressive power of the system described in this paper can be enhanced by interfacing it with specialized reasoning modules such as a semantic network or a *IS-A* hierarchy. Such an interface allows *terms* in the rules and facts to be any concept (type/instance) in the *IS-A* hierarchy. Another important extension extends the expressiveness and reasoning power of the system by allowing a limited use of function terms in rules[1].

Conclusion

The paper describes a knowledge representation and reasoning system that performs a limited but interesting class of inferences over a restricted class of first-order sentences with optimal efficiency. This work suggests that extremely efficient yet fairly powerful knowledge representation systems can be obtained by limiting the expressiveness and the inferential ability of a representation and reasoning systems in unusual ways.

References

[1] V. Ajjanagadde. Reasoning with function symbols in a connectionist network. In *Proceedings of the Cognitive Science Conference*, 1990. Submitted for publication.

[2] D.H. Ballard. Parallel logic inference and energy minimization. In *Proceedings of AAAI 86*, Philadelphia, PA, July 1986.

[3] J. Barnden. Neural-net implementation of complex symbol processing in a mental model approach to syllogistic reasoning. In *Proceedings of IJCAI 89*, Detroit, MI, August 1989.

[4] R. Brachman and H. Levesque. The tractability of subsumption in frame-based description languages. In *Proceedings of the 1984 Meeting of AAAI*, pages 34–37, Austin, TX, 1984.

[5] S. Fahlman. *NETL: A System for Representing Real-World Knowledge.* MIT Press, Cambridge, MA, 1979.

[6] A.M. Frisch and J.F. Allen. Knowledge retrieval as limited inference. In D.W. Loveland, editor, *Lecture Notes in Computer Science: Sixth Conference on Automated Deduction*, Springer-Verlag, New York, 1982.

[7] H.A. Kautz and B. Selman. Hard problems for simple default logics. *Artificial Intelligence*, 1989. Submitted.

[8] T.E. Lange and M.G. Dyer. High-level inferencing in a connectionist network. *Connection Science*, 1(2):181 – 217, 1989.

[9] H.J. Levesque. Logic and the complexity of reasoning. *Journal of Philosophical Logic*, 17, 1988.

[10] H.J. Levesque. Making believers out of computers. *Artificial Intelligence*, 30(1), 1986.

[11] D.R. Mani and L. Shastri. *Representing Multiple Dynamic Instantiations of a Predicate in a Connectionist System.* Technical Report, University of Pennsylvania, Dept. of Computer and Information Science, Philadelphia, PA, 1990. To appear.

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

[13] P. Patel-Schneider. A decidable first-order logic for knowledge representation. In *Proceedings of IJCAI-85*, pages 455–458, Los Angeles, CA, 1985.

[14] L. Shastri. *Semantic Networks: An Evidential Formulation and its Connectionist Realization.* Pitman/Morgan Kaufman, London/Los Altos, 1988.

[15] L. Shastri and V. Ajjanagadde. *From simple associations to systematic reasoning: A connectionist representation of rules, variables and dynamic bindings.* Technical Report MS-CIS-90-05, Dept. of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, January 1990.

[16] S.J. Thorpe and M. Imbert. Biological constraints on connectionist modeling. In R. Pfeiffer, editor, *Connectionism in Perspective*, Springer, 1988.

[17] D. Touretzky and G.E. Hinton. A distributed connectionist production system. *Cognitive Science*, 12(3):423 – 466, 1988.