August 1993

A Lower Bound Result for the Common Element Problem and Its Implication for Reflexive Reasoning

Paul Dietz
University of Rochester

Danny Krizanc
Carleton University

Sanguthevar Rajasekaran
University of Pennsylvania

Lokendra Shastri
University of Pennsylvania

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-93-73.

A Lower Bound Result for the Common Element Problem and Its Implication for Reflexive Reasoning

Abstract

In this paper we prove a lower bound of $\Omega(n \log n)$ for the common element problem on two sets of size n each. Two interesting consequences of this lower bound are also discussed. In particular, we show that linear space neural network models that admit *unbalanced* rules cannot draw all inferences in time independent of the knowledge base size. We also show that the *join* operation in data base applications needs $\Omega(\log n)$ time given only n processors.

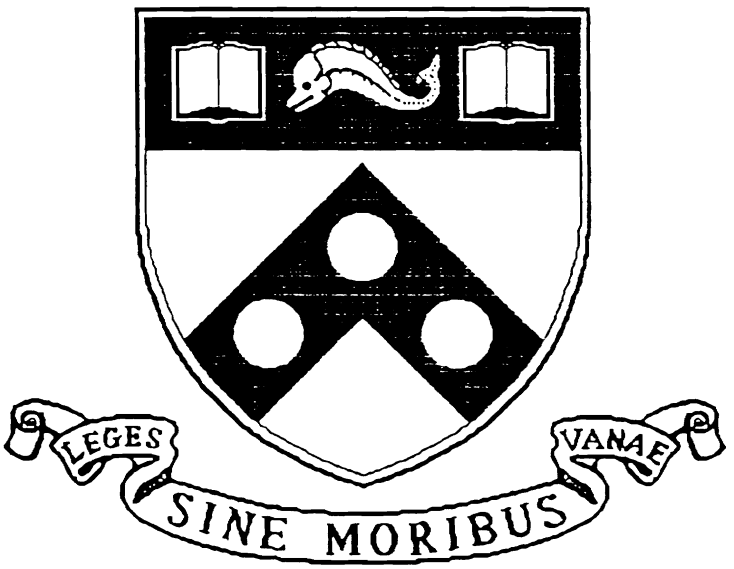
Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-93-73.

A Lower Bound Result for The Common Element Problem and Its IMplication for Reflexive Reasoning

MS-CIS-93-73
GRASP LAB 356

Paul Dietz
Danny Krizanc
Sanguthevar Rajasekaran
Lokendra Shastri



University of Pennsylvania
School of Engineering and Applied Science
Computer and Information Science Department
Philadelphia, PA 19104-6389

August 1993

A Lower Bound Result for the Common Element Problem and its Implication for Reflexive Reasoning

Paul Dietz [1], Danny Krizanc[2], Sanguthevar Rajasekaran[3], and Lokendra Shastri[3]

Abstract In this paper we prove a lower bound of $\Omega(n \log n)$ for the common element problem on two sets of size n each. Two interesting consequences of this lower bound are also discussed. In particular, we show that linear space neural network models that admit *unbalanced* rules cannot draw all inferences in time independent of the knowledge base size. We also show that the *join* operation in data base applications needs $\Omega(\log n)$ time given only n processors.

1 Introduction

In [Shastri & Ajjanagadde 1993] it is argued that there must exist a cognitively significant class of reasoning that can be processed using neural networks whose size is only linear in $|KB|$, the size of the knowledge-base, and in time that is only proportional to the length of the derivation and independent of $|KB|$. Reasoning that can be carried out within these space-time constraints has been referred to as *reflexive reasoning* [Shastri 1991].[4]

The motivation for focusing on reflexive reasoning stems from cognitive as well as biological considerations. The linear space constraint arises from the relation between the expected size of a common sense KB and the number of neurons available for encoding such a KB. The time constraint is motivated by the observation that the speed at which we perform common sense reasoning (say, during language understanding) does not slow down as the size of the KB grows significantly.

A detailed characterization of reflexive reasoning requires an answer to the following question: 'What are the formal constraints on the derivations computable by linear sized neural networks in time independent of the network size?' A partial answer to this question was provided by SHRUTI – a neurally motivated model of reflexive reasoning described

[1]Dept. of CS, Univ. of Rochester

[2]School of CS, Carleton Univ.

[3]Dept. of CIS, Univ. of Pennsylvania

[4]The label is intended to underscore the fact that such reasoning occurs spontaneously, effortlessly and without conscious effort — as it were a reflexive response of the cognitive agent.

in [Ajjanagadde & Shastri 1993]; [Shastri & Ajjanagadde 1993]; [Mani & Shastri 1992]. In particular, work on SHRUTI prompted the conjecture that when reasoning via backward chaining, any reflexive reasoning system *must* restrict itself to *balanced* rules (see Section 3.2) and lead to the identification of a class of queries that can be answered in a reflexive manner [Shastri 1993].

In this paper we prove a lower bound result for the *common-element* problem. This result establishes a lower bound of $\Omega(\log n)$ on the time required for deriving inferences involving unbalanced rules, and hence, provides a formal proof of the above conjecture. We also apply the lower bound to show that a similar result also holds for the JOIN operation in relational data bases.

Section 2 states and proves the lower bound result for the common-element problem. Section 3 provides a brief overview of SHRUTI and relate the lower bound result to the problem of reflexive reasoning with unbalanced rules. Section 4 relates the result to the JOIN operation.

2 The Lower Bound

The Common Element Problem (CEP): Given two sets S_1 and S_2 where $|S_1| = |S_2| = n$, decide if the two sets are disjoint.

Lemma 2.1 *Any comparison based algorithm for CEP takes $\Omega(n \log n)$ time on the comparison tree model.*

Proof. We'll reduce the problem of sorting to CEP.

Let $K = k_1, k_2, \ldots, k_n$ be any sequence of n numbers that we want to sort. Let $X = x_1, x_2, \ldots, x_n$ be the sorted order of the sequence K. We construct an instance of CEP as follows: Take S_1 to be $= \{(k_1, 0), (k_2, 0), \ldots, (k_n, 0)\}$ and S_2 to be $= \{(k_1, 1), (k_2, 1), \ldots, (k_n, 1)\}$.

We'll show that any algorithm for CEP would have to compare x_i with x_{i+1}, for each i, $1 \leq i \leq (n-1)$. Realize that this will imply the stated lemma on the comparison tree model (since these comparisons will yield the sorted order and sorting takes $\Omega(n \log n)$ time in the worst case).

Our claim is that any algorithm for CEP on these two sets S_1 and S_2 should have decided the relative ordering between the elements $(x_i, 1)$ and $(x_{i+1}, 0)$ for each i, $1 \leq i \leq n$. If not, we could replace $(x_i, 1)$ in S_2 with $(x_{i+1}, 0)$ and force the algorithm to output an incorrect

answer. Realize that replacing $(x_i, 1)$ with $(x_{i+1}, 0)$ does not in any way affect the orderings imposed by other comparisons made by the algorithm. □

Lemma 2.2 *Any algorithm for CEP takes $\Omega(n \log n)$ time even on the Random Access Machine model.*

Proof. The above algorithm seems to account for only n comparisons (though among the right pairs). We now show that if $T(n)$ is the run time of any algorithm for CEP on two sets of size n each, then we could sort any set of size n in time $O(n + T(n))$. We'll make use of the fact that the algorithm for CEP should have decided the relative ordering between x_i and x_{i+1}, for each i, $1 \leq i \leq n$.

Construct a graph $G(V, E)$ as follows: $V = \{k_1, k_2, \ldots, k_n\}$ and there is a directed edge from k_i to k_j if the algorithm for CEP has determined that $k_i < k_j$ (for any $1 \leq i, j \leq n$). This graph is constructible in $O(n + T(n))$ time. We could obtain the elements of K in sorted order as follows: Find the smallest element x_1 in $O(n)$ time. Find the smallest among all the neighbors of x_1; this will be x_2. Find the smallest among all the neighbors of x_2; this will be x_3, and so on. The total time spent is clearly $O(n + |V| + |E|) = O(n + T(n))$ □

Note: Realize that CEP can be solved in $O(n \log n)$ time. Also, the above lower bound can be circumvented for the following two special cases: 1) If the elements of the sets S_1 and S_2 are integers of at most a polynomial magnitude, we could sort the two sets in linear time and hence could solve CEP in linear time as well; 2) If the two sets are not of nearly the same size, the lower bound may not hold; for instance if one of the sets is of constant size, CEP can be solved in linear time. If $|S_1| \geq |S_2|$, we believe that $\Omega(|S_1| \log |S_2|)$ is a lower bound for CEP. Clearly, CEP can be solved in time $O(|S_1| \log |S_2|)$.

3 SHRUTI — a model for reflexive reasoning

SHRUTI is a neural network model that can encode a class of *rules* and *facts* (see below) using only a linear number of nodes in $|KB|$ and answer a class of queries in time proportional to the depth of the shortest derivation of the query.

Rules encoded by SHRUTI have the following form:[5]

[5]SHRUTI can also deal with soft/evidential rules, but for the purpose of this paper we will not consider such rules.

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

where the arguments of P_i's are elements of $\{x_1, ...x_m\}$, and an argument of Q is either an element of $\{x_1, ...x_m\}$, an element of $\{z_1, ...z_l\}$, or a constant.
Facts encoded by SHRUTI are 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.

Queries have the same form as facts. A query, all of whose arguments are bound to constants corresponds to the *yes-no* query: 'Does the query follow from the rules and facts encoded in the KB?' A query with existentially quantified variables, however, has two interpretations. For example, the query $P(a, x)$, where a is a constant and x is an existentially quantified argument, may be viewed as the *yes-no* query: 'Does $P(a, x)$ follow from the rules and facts for some value of x?' Alternately this query may be viewed as the *wh*-query: 'For what values of x does $P(a, x)$ follow from the rules and facts in the KB?'

3.1 An overview of SHRUTI

The following provides a simple overview of SHRUTI. It illustrates how simple rules and facts are encoded in SHRUTI and how a query is posed to and processed by SHRUTI. This example does not deal with rules with multiple antecedents and rules containing repeated variables or constants in the consequent. A detailed specification of the encoding may be found in (SA93).

The network shown in Fig. 1a encodes the following *rules* and *facts*:

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-sell(x,y)*
4. *give(John,Mary,Book1)*
5. $\exists$ *(x) buy(John,x)*
6. *own(Mary,Ball1).*

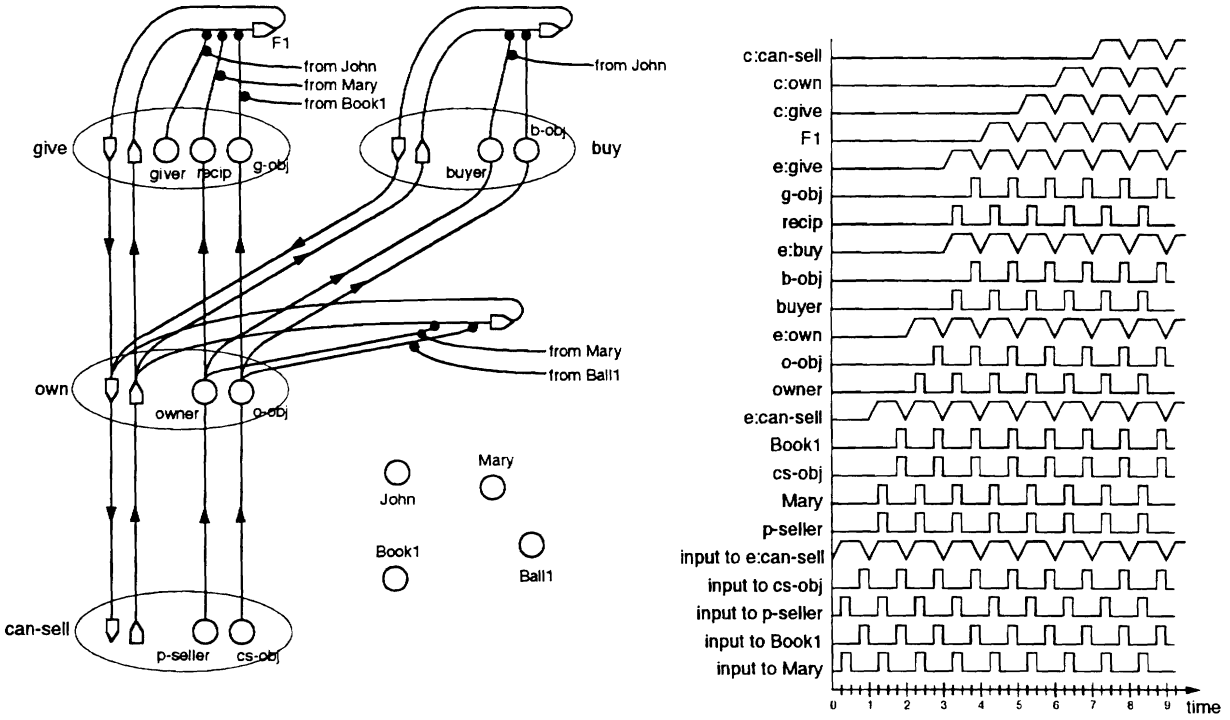


Figure 1: (**a**) An example encoding of rules and facts. (**b**) Activation trace for the query *can-sell(Mary,Book1)?*.

The encoding makes use of two types of nodes: ρ-btu nodes (depicted as circles) and τ-and nodes (depicted as pentagons). These nodes have the following idealized behavior: If a ρ-btu node A is connected to another ρ-btu node B, then the activity of node B synchronizes with the activity of node A. In particular, a periodic firing of A leads to a periodic and *in-phase* firing of B. We assume that ρ-btu nodes can respond in this manner as long as the period of firing, π, lies in the interval $[\pi_{min}, \pi_{max}]$. This interval can be interpreted as defining the frequency range over which ρ-btu nodes can sustain a synchronized response. A τ-and node behaves like a *temporal* AND node, and becomes active on receiving an uninterrupted pulse train. On becoming active, a τ-and node produces a pulse train similar to the input pulse train. A third type of node namely, the τ-or node (depicted as a triangle) is also used in SHRUTI. A τ-or node becomes active on receiving *any* activation and produces an output pulse train of width and period equal to greater than π_{max}. Fig. 2 summarizes the behavior of these nodes for the idealized case of oscillatory inputs.

The encoding also makes use of *inhibitory modifiers* — links that impinge upon and inhibit other links. A pulse propagating along an inhibitory modifier will block a pulse

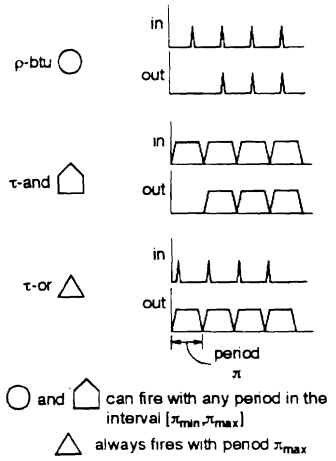


Figure 2: Behavior of the ρ-btu, τ-and and τ-or nodes in the reasoning system.

propagating along the link it impinges upon. In Fig. 1a, inhibitory modifiers are shown as links ending in dark blobs.

Each entity in the domain is encoded by a ρ-btu node. An n-ary predicate P is encoded by a pair of τ-and nodes and n ρ-btu nodes, one for each of its n arguments. One of the τ-and nodes is referred to as the **enabler**, $e{:}P$, and the other as the **collector**, $c{:}P$. In Fig. 1a, **enablers** point upward while **collectors** point downward. The **enabler** $e{:}P$ becomes active whenever the system is being queried about P. On the other hand, the system activates the **collector** $c{:}P$ of a predicate P whenever the system wants to assert that the current dynamic bindings of the arguments of P follow from the knowledge encoded in the system.

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 arguments of the consequent predicate to the arguments of the antecedent predicate in accordance with the correspondence between these arguments specified in the rule. 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 an entity in the fact then the modifier from such an argument node is in turn modified by an inhibitory modifier from the appropriate entity node. The output of the τ-and node is connected to the **collector** of the associated predicate.

...

The Inference Process

Posing a query to the system involves specifying the query predicate and the argument bindings specified in the query. This is done as follows: Choose an arbitrary point in time—say, t_0—as the point of reference for initiating the query (it is assumed that the system is in a quiescent state). The query predicate is specified by activating the **enabler** of the query predicate with a pulse train of width and periodicity π 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 n distinct entities: $c_1, ..., c_n$. With each c_i, associate a delay δ_i such that no two delays are within ω of one another and the longest delay is less than $\pi - \omega$. As mentioned earlier, ω is the width of the window of synchrony and π is the period of oscillation. Each of these delays may be viewed as a distinct *phase* within the period t_0 and $t_0 + \pi$. Now the argument bindings of an entity c_i are indicated to the system by providing an oscillatory spike train of 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 entity c_i $(1 \leq i \leq n)$ and amounts to representing argument bindings by the *in-phase* or *synchronous activation* of the appropriate entity and argument nodes.

We illustrate the reasoning process with the help of an example. Consider the query *can-sell(Mary,Book1)?* (i.e., 'Can Mary sell Book1?') This query is posed by providing inputs to the entities *Mary* and *Book1*, the arguments *p-seller*, *cs-obj* and the **enabler** *e:can-sell*, as shown in Fig. 1b. *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 ρ_1 and ρ_2 respectively. As a result of these inputs, *Mary* and *p-seller* will fire synchronously in phase ρ_1 of every period of oscillation, while *Book1* and *cs-obj* will fire synchronously in phase ρ_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 *own* predicate, and consequently, starting with the second period of oscillation, *owner* and *o-obj* become active in ρ_1 and ρ_2, respectively. At the same time, the activation from *e:can-sell* activates *e:own*. 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 *o-obj*. 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. 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 ρ_1, while arguments *g-obj* and *b-obj* become active in ρ_2. In essence, the system has created new bindings for the predicates *give* 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)?*. Observe that now the τ-and node associated with the fact *give(John,Mary,Book1)*—this is the τ-and node labeled $F1$ in Fig. 1a—becomes active as a result of the uninterrupted activation from *ε: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. 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 (refer to Fig. 1b) resulting in an affirmative answer to the query *can-sell(Mary,Book1)?*.

Conceptually, the proposed encoding of rules creates a directed *inferential dependency graph*: Each predicate argument is represented by a node in this graph and each rule is represented by links between nodes denoting the arguments of the antecedent and consequent predicates. In terms of this conceptualization, it should be easy to see that the evolution of the system's state of activity corresponds to a *parallel* breadth-first traversal of the directed inferential dependency graph. This means that i) a large number of rules can fire in parallel and ii) the time taken to generate a chain of inference is independent of the total number of rules and just equals $l\pi$ where l is the *length* of the chain of inference and π is the period of oscillatory activity. The example discussed above assumed that each predicate was instantiated at most once during the inference process. In the general case, where a predicate may be instantiated several times during an episode of reasoning, the time required for propagating bindings from a consequent predicate to antecedent predicate(s) is proportional to $k\pi$, where k is the number of dynamic instantiations of the antecedent predicate when the bindings are being propagated.

3.2 A characterization of SHRUTI's inferential power

A characterization of the class of queries that can be processed by a SHRUTI-like system in a reflexive manner is given in [Shastri 1993]. A description of this class is facilitated by the following definitions (from [Shastri 1993]):

□ Any variable that occurs in multiple argument positions in the antecedent of a rule is a *pivotal* variable.

□ A rule is *balanced* if all pivotal variables occurring in the rule also appear in its consequent.

For example, the rule $\forall x, y, z\ P(x, y) \wedge R(x, z) \Rightarrow S(x, z)$ is balanced, but the rule $\forall x, y, z\ P(x, y) \wedge R(x, z) \Rightarrow S(y, z)$ is not.

□ Consider a query Q and a KB consisting of facts and balanced rules. A derivation of Q obtained by backward chaining is *threaded* if all pivotal variables occurring in the derivation get bound and their bindings can be traced back to the bindings introduced in Q.

□ Given a KB consisting of facts and balanced rules, a *reflexive* query is one for which there exists a threaded proof.

It can be shown that the worst-case time for answering a reflexive *yes-no* query, Q, is proportional to $V|In|^V d$, where:

- V is as follows: Let V_i be the arity of the predicate P_i. Then V equals $max(V_i)$, i ranging over all the predicates occurring in the KB.
- $|In|$ is the number of *distinct* constants in Q ($|In| \leq V$).
- d equals the depth of the shallowest derivation of Q given the KB.[6]

Observe that the worst-case time is i) *independent* of the size of the KB, ii) polynomial in $|In|$ and iii) only proportional to d.

An answer to a *wh*-query can also be computed in time proportional to $V|In|^V d$, except that $|In|$ now equals the arity of the query predicate Q.

The space requirement is *linear* in the size of the KB and polynomial in $|In|$. This includes the cost of encoding the KB as well as the cost of maintaining the dynamic state of the 'working memory' during reasoning.

The above result offers a worst-case characterization which assumes that during the derivation, *all variables will get instantiated with all possible bindings involving constants in* Q. This will not be the case in a typical situation. As pointed out in (S93), in a typical episode of reasoning the actual time may seldom exceed $50d$.

3.3 CEP and reflexive reasoning

Consider the unbalanced rule:

$$P(x) \wedge Q(x, y) \Longrightarrow R(y)$$

[6]This assumes that the maximum *arity* of predicates in the KB is a constant.

for relations P, Q, and R. Let $R(a)$ be the query. Clearly, if all the tuples in Q were of the form $Q(., a)$, answering the query reduces to CEP. Thus if $|P| = |Q| = n$, $\Omega(n \log n)$ will be a lower bound on the processing time of the query. As a simple corollary it follows that any parallel algorithm for processing the query $R(a)$ will need $\Omega(\log n)$ time, given only n processors. Thus we have the following

Lemma 3.1 *Any linear sized network model for reasoning can not make inferences in time independent of the size of the KB if it admits unbalanced rules.*

The above lemma shows that the constraint that rules be balanced is a necessary constraint for reflexive reasoning and not merely an artifact of the SHRUTI design.

3.4 CEP and Database JOIN

JOIN is an important operation to be performed in relational database systems. Let R and S be two given relations with arity k and ℓ respectively. R can be thought of as a table of k-tuples, each column corresponding to some domain of values. The θ-join of R and S on columns i and j is defined to be those tuples in the cartesian product of R and S such that the ith component of R stands in relation θ to the jth component of S [Ullman 1988]. When θ stands for $=$, the JOIN operation is called *EQUIJOIN*. In the worst case, the size of the JOIN of the two relations can be $|R|\ |S|$ where $|R|$ ($|S|$) is the number of tuples in R (S).

Consider the problem of deciding if the EQUIJOIN of two given relations R and S on columns i and j is nonempty. Clearly, this problem reduces to checking if column i of R and column j of S are disjoint. If $|R| = |S| = n$, the lower bound for CEP implies that this decision problem needs $\Omega(n \log n)$ time sequentially. Also, any parallel algorithm for this problem that uses only $O(n)$ processors will need $\Omega(\log n)$ time.

4 Conclusions

In this paper we have proved a lower bound for the Common Element Problem and have demonstrated the applicability of this lower bound in two different areas of computing, i.e., reasoning and databases. The lower bound in particular implies that any model of reasoning that is of size only proportional to the size of the knowledge base, cannot hope to make inferences on all queries whose derivation involves unbalanced rules, in time independent of the size of the knowledge base. As a consequence, we conclude that the constraints imposed

on the SHRUTI model are indeed necessary and hence the SHRUTI model be a stronger predictor of the nature of reflexive reasoning processes in humans.

References

[Ajjanagadde & Shastri 1993] V.G. Ajjanagadde and L. Shastri. Rules and variables in neural nets. *Neural Computation*, **3**:121-134.

[Mani & Shastri 1992] D.R. Mani and L. Shastri. A connectionist solution to the multiple instantiation problem using temporal synchrony. In *Proceedings of the Fourteenth Conference of the Cognitive Science Society.* Lawrence Erlbaum.

[Shastri 1993] L. Shastri. A Computational Model of Tractable Reasoning — taking inspiration from cognition. In *Proceedings of IJCAI-93*, Chambery, France.

[Shastri 1991] L. Shastri. Why Semantic networks? In *Principles of Semantic Networks.* Edited by John Sowa. Morgan Kaufman Los Altos.

[Shastri & Ajjanagadde 1993] L. Shastri & V.G. Ajjanagadde, From Simple Associations to Systematic Reasoning: A connectionist representation of rules, variables and dynamic bindings using temporal synchrony. *Behavioral and Brain Sciences, 16(3).*

[Ullman 1988] J.D. Ullman, *Database and Knowledge-Base Systems, Volume I*, Computer Science Press, 1988.