Time-Bounded Reasoning in First Order Knowledge Base Systems — Source link

Yihua Shi, Setsuo Arikawa

Institutions: Kyushu University

Published on: 12 Jul 1989 - Conference on Logic Programming

Topics: First-order logic, Knowledge representation and reasoning, Axiom, Knowledge-based systems and Legal expert system

Time-Bounded Reasoning in First Order Knowledge Base Systems

Shi, Yi-Hua
Department of Information Systems, Kyushu University

Arikawa, Setsuo
Research Institute of Fundamental Information Science, Kyushu University

RIFIS-TR-CS-19

RIFIS Technical Report

Time-Bounded Reasoning in First Order Knowledge Base Systems

Yi-Hua Shi
Setsuo Arikawa

October 16, 1989

Research Institute of Fundamental Information Science
Kyushu University 33
Fukuoka 812, Japan
E-mail: shi@rifis1.sci.kyushu-u.jac.jp Phone: 092(641)1101 Ext.4484

Time-Bounded Reasoning in First Order Knowledge Base Systems

Yihua Shi* Setsuo Arikawa*

Department of Information Systems Kyushu University 39,

Kasuga, Fukuoka 816, Japan

E-mail: shi@rifis.sci.kyushu-u.ac.jp

Abstract

In first order predicate logic, it is undecidable whether a formula is deducible from a set of axioms. In order to realize a practical knowledge base system in the framework of the first order logic, we must overcome this problem.

In this paper, we propose a time-bounded reasoning and investigate the properties of the knowledge base management system based on our time-bounded reasoning. We also show that the time-bounded reasoning is sound and valid. Furthermore, we discuss the selection of time functions and give three parameters of time functions. We also outline a prototype system we have realized in K-prolog on a Sun-3.

1 Introduction

Many reasoning methods including default reasoning[11], inductive inference[13, 14], analogical reasoning [4] as well as deductive inference have been proposed and used in Artificial Intelligence. A derivation procedure to decide whether $KB \vdash \alpha$ is commonly used in all of the reasoning methods above, where KB is a knowledge base, i.e., a set of axioms and α is a formula. This decision problem, however, is known to be unsolvable in case of first order predicate logic. In order to realize a practical reasoning system, it is necessary to discuss whether the basic derivations terminate.

In this paper we present a new method called a time-bounded reasoning that forces the derivation procedure to terminate within a limited time n. That is, we use $KB \vdash_n \alpha$ instead of $KB \vdash \alpha$. Thus we can escape the undecidability. With this restriction, however, our method may infer a false result such that $KB \not\vdash_n \alpha$, but $KB \vdash \alpha$. Hence, it is

*Mailing address: Research Institute of Fundamental Information Science Kyushu University 33, Fukuoka 812, JAPAN important that there be a discussion of the reliability of conclusions thus derived from a knowledge base and of the knowledge base itself.

In this paper, we first give definitions of time-bounded reasoning and other concepts necessary for our discussion. Then, we design a special inference system which makes use of our time-bounded reasoning. In Section 3, we show this inference system is sound and valid in some sense. In Section 4, we discuss problems about truth maintenance of knowledge base, reliability and capability of our time-bounded reasoning system. In Section 5, we consider time functions for some subclasses of first order predicate logic. In Section 6, we briefly describe the implemented system.

2 Time-Bounded Reasoning

This section defines a time-bounded reasoning, and designs an inference system that uses it. The basic logic programming terminology and concepts not defined in this paper may be found in [8].

First we show the concepts of *BF-derivation* (BF stands for *Breadth-First*), *derivation tree*, and the *depth of derivation tree* as in Figure 1, where $B \leftarrow A_1, \ldots, A_m$ is a clause, and A is an atom.

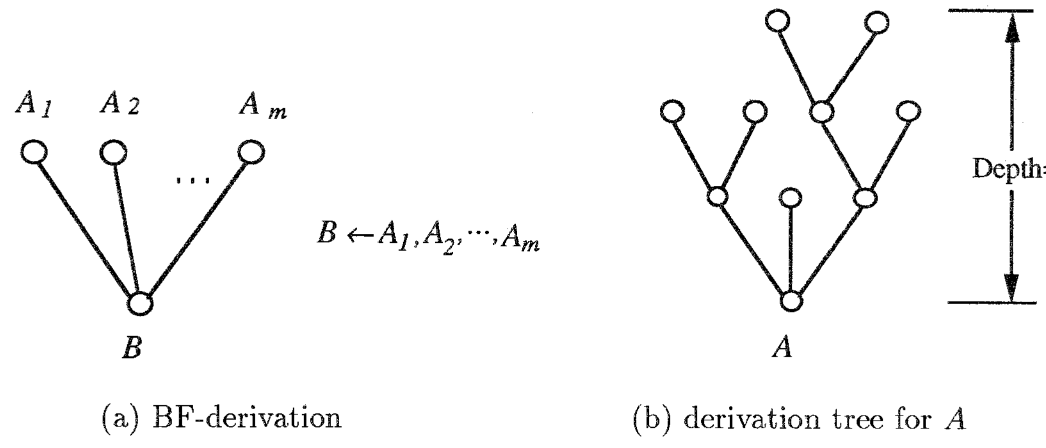
A_1
A_2
A_m
$B \leftarrow A_1, A_2, \cdots, A_m$
B
A
Depth=3
(a) BF-derivation
(b) derivation tree for A

Figure 1. BF-derivation and BF derivation tree

The derivation tree of Figure 1(b) is constructed by BF-derivation of Figure 1(a). We denote the depth of the derivation tree by the length of its longest branch, e.g. the depth of A's derivation tree is 3 in Figure 1(b). Now, we define that the derivation of an atom, say A in Figure 1(b), is rank n if the depth of its derivation tree is n.

In the above notions, n corresponds to that in $T_p \uparrow n$ [8], as proved in [18].

Proposition 2.1 (Wolfram, Maher and Lasezz[18]) Let P be a program and $A \in B(P)$ be an atom. $A \in Tp \uparrow k$ $(k > 0)$ iff there is a BF-refutation of $P \cup \{\leftarrow A\}$ of length at most k.

Definition 2.1 Let KB be a knowledge base and α be a positive ground literal called a query.

(1) α is derived in depth n from KB, denoted by $KB \vdash_n \alpha$, if it has a successful derivation tree of depth less than or equal to n.

(2) α is failed in depth n from KB, denoted by $KB \not\vdash_n^F \alpha$, if all of its derivation trees are failures and their depths are less than or equal to n.

(3) The other case is denoted by $KB \not\vdash_n^I \alpha$.

Notice that (1) and (2) correspond to the concepts of success and failure in the theory of logic programming [8]. (3) is a specific feature of our time-bounded reasoning, that is, the derivation is forced to terminate at depth n.

If α has no successful derivation tree whose depth is less than or equal to n, we denote it as $KB \not\vdash_n \alpha$. Then, clearly

$$KB \not\vdash_n \alpha \quad \Leftrightarrow \quad (KB \not\vdash_n^F \alpha \text{ , or } KB \not\vdash_n^I \alpha).$$

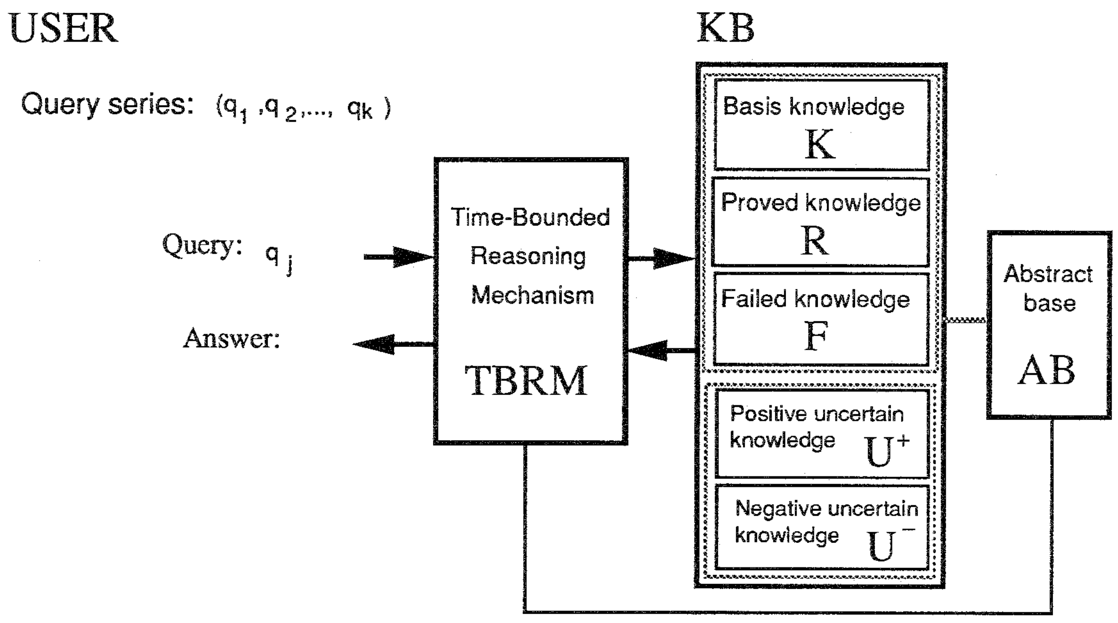
USER
KB
Query series: (q1 ,q2,..., qk)
Query: qj
Answer:
Time-Bounded Reasoning Mechanism
TBRM
Basis knowledge K
Proved knowledge R
Failed knowledge F
Positive uncertain knowledge U+
Negative uncertain knowledge U−
Abstract base
AB

Figure 2. Organization of Time-Bounded Reasoning System

Definition 2.2 Let α be a positive ground literal called a query. *Time-bounded reasoning*(TBR for short) is an inference and an addition, defined in the following way:

(1) Infer α, and add it to KB, if $KB \vdash_n \alpha$.

(2) Infer $\neg\alpha$, and add it to KB, if $KB \not\vdash_n^F \alpha$.

(3) Infer $\neg\alpha$, and add it to KB, if $KB \not\vdash_n^I \alpha$.

Since the above three kinds of derivations are different from each other in their meaning and reliability, it is necessary to classify them. In order to use TBR effectively, we design a TBR system of three parts as in Figure 2, where

(1) *TBRM* is a mechanism to infer queries within bounded time,

(2) *KB* is a knowledge base to store the basic knowledge supplied by users and inferred results, and

(3) AB is an abstract base to store the relations just among the predicate symbols in KB.

In this paper, we consider the following problem solving:

In order to solve a problem, a user puts a query to the system. TBRM infers it by using the knowledge in KB within a bounded time, and then returns inferred results to the user, and adds it to KB at the same time. Depending on the answer from the system, the user can put another query to repeat the above process.

For simplicity, we assume users are sufficiently intelligent to put no queries unnecessary to solve the problem.

We divide a *knowledge base* KB into five parts:

$$KB = K \cup R \cup F \cup U^+ \cup U^-$$

and denote it also by

$$KB =< K, R, F, U^+, U^- >,$$

where

K is the finite set of normal clauses given by users, called *basic knowledge set.*

R is the set of positive ground literals, called *proved knowledge set.*

F is the set of negative ground literals, called *failed knowledge set.*

U^+ is the set of positive ground literals, called *positive uncertain knowledge set.*

U^- is the set of negative ground literals, called *negative uncertain knowledge set.*

K works as a basis of the knowledge base KB. A change of K will propagate through the whole of the knowledge base.

A *query series* $(q_1, \ldots, q_i, \ldots)$ is a list of queries successively put to the system by a user, where $q_i (i \geq 1)$ is a positive ground literal.

In the following sections, we will consider a sequence of knowledge bases $KB_0, \ldots$, $KB_i, \ldots$ which corresponds to $(q_1, \ldots, q_i, \ldots)$, where $KB_0 =< K, \phi, \phi, \phi, \phi >$ is called an *initial knowledge base*, and $KB_i =< K, R_i, F_i, U_i^+, U_i^- >$ is successively constructed by the method defined below.

3 Properties of the System

This section describes the operations of the TBR system, and discuss its properties. By the definition of the TBR, when receiving a query, the TBR system will infer it, and add the derived results to KB. Now we give the way of adding.

Definition 3.1 Let $KB =< K, R, F, U^+, U^- >$ be a knowledge base, and q be a query. the TBR constructs $KB' =< K, R', F', U^{+\prime}, U^{-\prime} >$ by adding derived results to KB in the following way.

$$R' = R \cup \{q \mid K \cup R \cup F \vdash_n q, q \notin K\}$$

$$F' = F \cup \{\neg q \mid K \cup R \cup F \not\vdash_n^F q\}$$

$$U^{+\prime} = U^+ \cup \{q \mid KB \vdash_n q, q \notin K, q \notin R'\}$$

$$U^{-\prime} = U^- \cup \{\neg q \mid KB \not\vdash_n q, \neg q \notin F'\}$$

Now we will show that this system has many good properties. Since the TBR system may derive even a false result such as $KB \not\vdash_n q$ but $K \vdash q$, it is necessary to consider reliability and properties of this system.

In the previous section we have designed a TBR system with an organization as in Figure 2 from which we can easily obtain the following results.

Proposition 3.1 In the knowledge base KB, any knowledge in parts R and F is not deleted if K is not changed.

Proof: By the definition of R, clearly we have

$$\alpha \in R_i \Rightarrow K \vdash_{n \cdot i} \alpha \; (i \geq 1).$$

That is, every knowledge α in R is a consequence of K. Thus, if K is not changed, α cannot be deleted. Similarly we can prove it for F. $\square$

By Proposition 3.1, it is obvious that the more queries that are put to the TBR system, the stronger the proof power of the TBR system becomes. Clearly, the false knowledge is only in U^+ and U^-.

Definition 3.2 Let K be a set of basic knowledge. Then, we define three sets:

(1) $C(K) = \{\alpha \mid K \vdash \alpha\}$: a set of successful knowledge from K.

(2) $NAF(K) = \{\neg\alpha \mid K \not\vdash \alpha\}$: a set of failure knowledge from K.

(3) $DKB = K \cup R \cup F$: a definite portion of KB.

Corollary 3.2 If K is not changed, the set of proved knowledge R and the set of failed knowledge F monotonically increases. That is,

$$R_0 \subseteq \ldots \subseteq R_i \subseteq R_{i+1} \subseteq \ldots \subseteq C(K)$$

$$F_0 \subseteq \ldots \subseteq F_i \subseteq F_{i+1} \subseteq \ldots \subseteq NAF(K),$$

where $R_0, \ldots, R_i, \ldots$ is a sequence of sets of proved knowledge, and $F_0, \ldots, F_i, \ldots$ is a sequence of sets of failed knowledge that respond to $(q_1, \ldots, q_i, \ldots)$.

By the assumption that a user is intelligent enough to organize a series of queries to derive the result about α in a finite time if $K \vdash \alpha$, we have the following proposition.

Proposition 3.3

(1) $\bigcup_{i=0}^{\infty} R_i = C(K)$.

(2) $\bigcup_{i=0}^{\infty} F_i = NAF(K)$.

Since KB may have some false knowledge, it is not sound in general. By the discussion above, however, we can prove DKB, a part of KB, is sound and valid.

Theorem 3.4 (Soundness) Let α be a positive ground atom. Then

$$DKB \vdash_n \alpha \Rightarrow \alpha \in C(K).$$

Theorem 3.5 (Validness) Let α be a positive ground atom. Then under the NF rule (NF stands for negation as failure) in the TBR,

$$K \vdash \alpha \;\Leftrightarrow\; \text{there is a series of queries } q_1, \ldots, q_i (= \alpha) \text{ such that } DKB_i \vdash_2 \alpha.$$

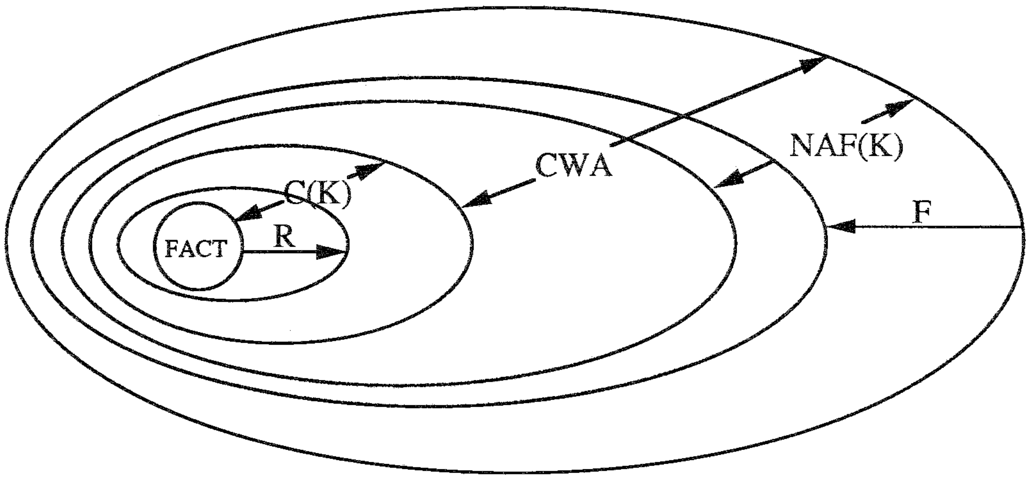

Figure 3. The power of inferences

From these discussions, we can summarize the relations between our the TBR and the usual time-unbounded reasoning as in Figure 3.

4 Truth Maintenance and Reliability of the System

4.1 Truth Maintenance

In our system, there are two kinds of truth maintenance problems. One is, as in Doyle [2], the contradiction that users want to avoid. We use a special predicate symbol *nogood* to express that a rule with "nogood" as its head and some bodies leads to a contradiction. In our TBR, three situations may arise:

$$K \vdash_n nogood, \qquad K \not\vdash_n^F nogood, \qquad K \not\vdash_n^I nogood.$$

In this paper, we say K is consistent if $K \not\vdash_n$ *nogood*. More detailed discussion about them is in Shi[16]. The other is the logical contradiction such as $KB \vdash_n \alpha$ and $KB \vdash_n \neg\alpha$, when a newly derived result is added as in Definition 3.1. In this section, we will mainly discuss this case.

With the TBR, the results of a derivation are frequently added to KB. Hence, to maintain the consistency of KB, it is necessary to delete some knowledge from KB which may cause a contradiction of the second case above.

Definition 4.1 We define a binary relation *Rel* on the set of predicate symbols of K by

$$Rel = \{(p, q) \mid p(\ldots) \leftarrow L_1(\ldots), \ldots, L_m(\ldots) \in K\},$$

where q is a predicate symbol that occurs in a literal of $L_1, \ldots, L_m$.

Definition 4.2 An influence of predicate symbol q, denoted $connect(q)$, is defined by

$$connect(q) = \{p \mid (p, q) \in Rel^+\},$$

where Rel^+ is a transitive closure of Rel.

After the adding of q or $\neg q$(a derived result), the derivation of q becomes of rank 1. Thus, the rank of the derivation of $p(\in connect(q))$ will become smaller than before. That is, some knowledge that were derived within greater ranks than n may now be derived within rank n. We give the definition of truth maintenance.

Definition 4.3 Let $KB' =< K, R', F', U^{+\prime}, U^{-\prime} >$ be a knowledge base that is defined by Definition 3.1, and q be a query. A knowledge base KB'' is defined as follows:

$$KB'' =< K, R', F', U^{+''}, U^{-''} >,$$

where

$U^{+''} = U^{+\prime} - \Delta$

$U^{-''} = U^{-\prime} - \Delta$

$\Delta = \{p \mid p \in U^{+\prime}, p \in connect(q)\} \cup \{\neg p \mid \neg p \in U^{-\prime}, p \in connect(q)\}$.

KB'' is thus obtained by correcting KB'.

Theorem 4.1 Let KB' be a knowledge base constructed by Definitions 3.1 and 4.3 from KB. If KB is consistent, then so is KB'.

Proof: It is clear from Definitions 3.1, 4.2, 4.3, and Proposition 3.1. □

Proposition 4.2 The sets K, R_k, F_k, U_k^+ and U_k^- are mutually disjoint, if $KB_k =< K, R_k, F_k, U_k^+, U_k^- >$ is constructed by Definitions 3.1 and 4.3.

Although the additions and rechecks of knowledge are carried out very frequently in the TBR system, we can see they only have influence on a small portion of knowledge base.

4.2 Abstract Base

By the TBR, the derived results are frequently added to KB. In order to maintain the consistency of KB, the system has to recheck the knowledge stored in KB. This operation may cost a huge amount of time. To save time, we introduce an abstraction procedure[19].

Definition 4.4 Let KB be a knowledge base. An *abstract base* AB is defined by

$$AB = \{ab(id, p, n, Fp, Bp, connect(p)) \mid p \text{ is an n-ary predicate symbol in } KB\},$$

is a set of atoms that express the relations among the predicate symbols in KB, where

id is a natural number, called an identifier of the knowledge,

p is a predicate symbol,

n is the arity of predicate symbol p,

$Fp = \{q \mid q(\ldots) \leftarrow \ldots, p(x_1, \ldots, x_n), \ldots \in KB\}$,

$Bp = \{r \mid p(x_1, \ldots, x_n) \leftarrow \ldots, r(\ldots), \ldots \in KB\}$,

$connect(p)$ is a set of predicate symbols (Definition 4.2), called the influence of p.

The TBR system uses AB to simplify the relations in KB in this way:

$$\text{“}p(\ldots) \text{ is related to } q(\ldots) \text{ in } KB\text{”} \implies \text{“}p \text{ is related to } q \text{ in } AB\text{”}.$$

As is easily seen, any literal which is not derived in AB is not derived even in KB. The calculations and operations on AB are much simpler than those on the original KB. Furthermore, the TBR system suffices to calculate AB only when K is changed by users. Thus, the abstraction technique contributes to making the TBR system efficient.

4.3 Reliability of System

From the discussions in Sections 2 and 3, we can now briefly summarize the reliability of our inference and knowledge in KB itself.

(1) Knowledge base KB: As defined in Section 2, KB in the TBR system has three parts, of different reliabilities:

   (a) Since K is the basic knowledge base given by the user at the starting point of inference, its reliability should be the highest among the three.
   (b) R and F are sets of knowledge that can be asserted from DKB under NF. Their reliability is medium.
   (c) U^+ and U^- are sets of derived knowledge under the TBR. Since there may be some false knowledge in it, their reliability is the lowest.

(4) Result of inference: the TBR system gives one of the following answers for a user's query in limited time.

   (a) $< yes >$: the query can be derived from K by NF.
   (b) $< no >$: the query cannot be derived from K by NF.
   (c) $< yes, uncertain >$: the query can be derived from KB by the current TBR. It may be an uncertain result.
   (d) $< no, uncertain >$: the query cannot be derived from KB by the current TBR. It may also be an uncertain result.

5 Time Functions of the TBR

When we put some queries to an inference system for solving a problem, we consciously or unconsciously require the system to return answers in a finite time. In general, there are two methods which meet the requirement: to find some safe subclasses, and to select a suitable compromise between time and reliability.

First we will discuss several subclasses which are constructed by imposing some syntactic restrictions on KB. For any KB in such classes, there is a computable time function $f(KB)$ which satisfies "$KB \vdash_{f(KB)} q \;\Leftrightarrow\; KB \vdash q$". Hence the TBR can cut derivations at depth $f(KB)$. In this meaning we say these classes are safe to the TBR. We also pay attention to the relation between the time function, reliability, query, and KB. We use the following notations:

$S(KB) = max\{m \mid (\alpha \leftarrow \beta_1, \ldots, \beta_m) \in KB\}$, denotes the size of KB.

$D(KB)$: the depth of KB, i.e., the number of rules with different heads.

5.1 Time Functions of Propositional Logic

Proposition 5.1 Let A be an atom and KB be a knowledge base in propositional logic. If $KB \vdash A$, there is a successful derivation tree of depth less than or equal to $D(KB)$.

Proof: By $KB \vdash A$, there is a finite successful derivation tree for A. First we suppose its depth is greater than $D(KB)$. Then there is a branch in the tree whose length is longer than $D(KB)$, that is,

$$p_1, p_2, \cdots, p_m \ (m > D(KB)),$$

where $p_1 = A$ is a root and p_m is a leaf. In propositional logic, by the definition of $D(KB)$, there are i and j $(1 \leq i < j \leq m)$ such that $p_i = p_j$. We can replace the subtree with a root p_i by the subtree with a root p_j. The depth of new tree is less than that of the old one. By repeating this process for every branch of length longer than $D(KB)$, a successful derivation tree of depth less than $D(KB)$ can be constructed. $\square$

By this proposition, we show that $D(KB)$ relates to a time function which guarantees the correctness of results of the TBR. From the relations between the BF-derivation and the binary derivation, we can easily see that $S(KB)$ is related to the actual time for processing, i.e., deriving a query.

5.2 Time Functions of Deductive Knowledge Bases

In first order knowledge base KB, there is no computable function f , hierarchical KB and strongly stratified KB, for which there exist time functions f which satisfy the above equivalence. By $pred(L)$ we denote the predicate symbol in the literal L.

Definition 5.1 (Lloyd[8]) *A level mapping* of a knowledge base KB, denoted by V_{level}, is a mapping from the set of predicate symbols in KB to non-negative integers. We refer to the value of a predicate symbol under this mapping as the *level* of the predicate symbol.

Definition 5.2 (Lloyd[8]) A knowledge base KB is *hierarchical* if it has a level mapping V_{level} such that

$$V_{level}(p) > V_{level}(pred(L_i))$$

for any rule $p(t_1, \ldots, t_n) \leftarrow L_1, \ldots, L_m \in KB$ and any i $(1 \leq i \leq m)$.

A hierarchical KB has several good properties. For example, there is no recursive relation in KB, and any one step of BF-derivation will decrease the level of each subgoal by at least one. Since the number of predicate symbols in KB is finite, so are the levels of predicate symbols in KB.

Proposition 5.2 Let KB be a hierarchical knowledge base, and $g(KB) = max\{V_{level}(p) \mid p \in KB\}$. Then, for any atom A,

$$KB \vdash A \;\Rightarrow\; KB \vdash_{g(KB)} A.$$

Definition 5.3 A knowledge base KB is *strongly stratified* if it has a *one-to-one level mapping* V_{level} such that

$$V_{level}(p) \geq V_{level}(pred(L_i))$$
$$V_{level}(p) > V_{level}(pred(L_j))$$

for any rule $p(t_1, \ldots, t_n) \leftarrow L_1, \ldots, L_m \in KB$, any positive literal L_i $(1 \leq i \leq m)$ and any negative literal L_j $(1 \leq j \leq m)$.

From the definition we can see that the recursive relations are only on the same predicate symbols in the strongly stratified KB. We use the following notations:

$l(p) = max\{n \mid q(t_1, \ldots, t_n)$, and $(p, q) \in Rel^*\}$, the maximum length of predicate symbol p.

U_{KB} : the domain of KB, the Herbrand universe of KB.

$\mid U_{KB} \mid$: the size of U_{KB}, the number of elements in U_{KB}.

Proposition 5.3 Let KB be a function-free strongly stratified knowledge base, and L be a literal. Then

$$KB \vdash L \Leftarrow KB \vdash_{f(KB,L)} L,$$

where $f(KB, L) = V_{level}(L) * (\mid U_{KB} \mid^{l(L)} + 1)$.

Proof In a function-free knowledge base KB, the number of constant symbols is finite, i.e., $\mid U_{KB} \mid$ is finite. Thus, we can enumerate all ground atoms of an n-ary predicate, and it is clear that the number of them is less than or equal to $\mid U_{KB} \mid^n$. Furthermore, the number of predicate symbols in any branch of a derivation tree is less than or equal to $V_{level}(L)$, and the arity of each predicate in them is less than or equal to $l(L)$. So, by the same method as in the proof of Proposition 5.1, we can prove that the depth of any successful derivation tree is less than or equal to $V_{level}(L) * (\mid U_{KB} \mid^{l(L)} + 1)$. □

5.3 Parameters in Time Functions

There may be, however, some unsafe KB's which have no such $f(KB)$ as in Sections 5.1 and 5.2. In such a case, we usually take account of a tradeoff between time and reliability. By the discussions in Sections 5.1 and 5.2, we can intuitively see that the time is related to the following parameters:

(1) $S(KB)$, $D(KB)$ (the size and depth of KB),

(2) $l(p)$ (the maximum length of predicate symbol p),

(3) $\mid U_{KB} \mid$ (the size of domain of KB).

Further, a user may have much knowledge and experience in the problem domain, so his/her decision will be very important in selecting the time bound. The detailed discussion of this will go beyond the scope of this paper. Finally we want to emphasize that the validness of the TBR system is guaranteed by Theorem 3.5 independently of the selected time bound.

6 Implementation of the TBR System

From the discussions in Sections 3 and 4, the configuration in Figure 2 can easily be implemented as a TBR system. Now we briefly describe the implemented system.

In the previous sections, we have discussed a TBR system under the assumption that K is not changed and is consistent. In realizing the TBR as a practical system, however, we must supply an approach to check and maintain the consistency of K. In our TBR system implemented on a Sun-3 in K-prolog, we consider $DKB \vdash_n nogood$, $DKB \not\vdash_n^F nogood$, and $DKB \not\vdash_n^I nogood$. When a contradiction has been found, we can search for the suspects with the contradiction backtracking algorithm (CBA for short) [13, 14].

First, we divide a knowledge base KB into five parts: K, R, F, U^+, and U^-. For simplifying the operation of rechecking for truth maintenance, we have introduced an abstract base AB as in Definition 4.4. For dealing with queries and truth maintenance of KB, we have built in a time-bounded reasoning mechanism $TBRM$. It has four main functions:

(1) *Query Treatment Function.*

For a query, $TBRM$ can return the best answer that is inferred within a limited time, and store the result in the corresponding part of KB.

(2) *Addition and Deletion of Knowledge.*

On demand of users, $TBRM$ can add or delete knowledge to or from K.

(3) *Truth Maintenance Function.*

Since (1) and (2) may destroy the consistency of KB, $TBRM$ has been designed to recheck and maintain the consistency of KB by using the techniques of AB and CBA.

(4) *Calculation and Modification of AB.*

When K is changed, $TBRM$ can automatically correct AB.

Shi[16, 17] has given algorithms for these operations and the implementation principles.

Now we show an example to illustrate the operation of our TBR system. Let a set of basic knowledge K be given as follow:

```
sunday(731).
student(shi).
student(ito).
student(arimura).
worker(sato).
worker(kawa).
holiday(101).
holiday(115).
holiday(321).
holiday(505).
climb(X,D):-nothing(X,D),noschool(X,D),asked(X,Y,D),weather(D).
nothing(X,D):-student(X),sw(D).
noschool(X,D):-holiday(D);sunday(D);(sw(D),student(X)).
asked(X,li,D):-student(X),D>=801,D=<826.
weather(D):-not(raining(D)).
raining(D):-D>=806,D=<808.
sw(D):-summer(D);winter(D).
sw(D):-D==312.
summer(D):-D>=720,D=<910.
winter(D):-D>=101,D=<120.
nogood:-climb(ayumi,820).
nogood:-student(X),worker(X).
```

The interaction 74 below shows that the set of basic knowledge K is given as an input *ex1*, and $TBRM$ calculates its abstract base at the same time.

```
74: ?- input_kb(ex1).
*** the number of predicate symbols is 14
*** the number of facts and rules is 22
*** the length of K is  4
*** the depth of K is  10
```

In order to treat queries from users, we supply a predicate $query(G, D)$ as an operation of $TBRM$, where G is a query and D is a value of time function. Its role is to decide whether $KB \vdash_D G$ as in Definition 3.1. From the following interactions 75–77, we can see that the proof power of the TBR system gradually becomes stronger.

```
75: ?- query(climb(shi,820),4).
*** <no, uncertain>
76: ?- query(sw(820),4).
*** yes
77: ?- query(climb(shi,820),4).
*** yes
```

In order to permit users to change K, we supply two more predicates, $addpro(G)$ and $delpro(G)$. The $addpro(G)$ adds a new fact or rule G to K, maintains the consistency of KB, and automatically corrects AB. The $delpro(G)$ deletes a fact or rule G from K, maintains the consistency of KB, and automatically corrects AB.

In a Prolog program, the order of the knowledge(facts and rules) is very important. For example, if its order is as in interactions 78 and 79 below, the derivation from the query, e.g. $query(nothing(X, 311), 40)$, will fall into an infinite loop. Our TBR system, however, can find all answers independently of the order of the knowledge such that interaction 80. And this system can prevent derivation from infinite loop as in interaction 81.

```
78: ?- addpro(nothing(X,Y):-nothing(X,Z)).
yes
79: ?- addpro(nothing(shi,311)).
yes
80: ?- query(nothing(X,311),40).
X=shi
*** yes
81: ?- query(nothing(ito,X),40).
*** <no, uncertain>
```

The kernel of this TBR system is a meta-interpreter $solve(M, G, D, W)$, where M is a value to express the property of the result ($M = 0$ means "success", $M = 1$ means "failure", and $M = 2$ means "the result is uncertain"), G is a query, D is a value of the time function, and W is the information about derivation, e.g. the derivation tree. W is important in debugging and constructing of a query series from the user.

For truth maintenance, we also supply a predicate $cba(X, W, E)$. According to the interaction with users, it can find a reason which causes a contradiction. The interactions 82 and 83 below show how to backtrack a contradiction by using predicates *solve* and *cba*.

```
82: ?- solve(M,climb(shi,819),7,W),M==0,cba(Y,W,S).
[(nothing(shi,819) , 0)] ? t/f  t.
[(noschool(shi,819) , 0)]?t/f  f.
[(holiday(819) , 1)]?t/f  t.
[(sunday(819) , 1)]?t/f  t.
[(sw(819) , 0)]?t/f  f.
[(summer(819) , 0)]?t/f  t.
[(sw(819):-summer(819) ; winter(819))] is an error.
83: ?- solve(M,climb(shi,819),7,W),M==0,cba(Y,W,S).
[(nothing(shi,819) , 0)]?t/f  t.
[(noschool(shi,819) , 0)]?t/f  f.
[(holiday(819) , 1)]?t/f  t.
[(sunday(819) , 1)]?t/f  t.
[(sw(819) , 0)]?t/f  t.
[(student(shi) , 0)]?t/f  f.
[(student(shi) , 0)] is an error.
```

When K is changed, the following contradictions may also be created. They may be corrected in the following way.

```
84: ?- addpro(worker(shi)).
A contradiction has been found
***CBA starts***
[(student(shi), 0)]?t/f  f.
[(student(shi), 0)] is an error
85: ?- delpro(student(shi)).
Do you want to delete it from K-base?  yes
86: ?- addpro(student(ayumi)).
A contradiction has been found
***CBA starts***
[(climb(ayumi), 0)]?t/f  f.
[(nothing(ayumi,820), 0)]?t/f  t.
[(noschool(ayumi,820), 0)]?t/f   t.
[(asked(ayumi,li,820), 0)]?t/f  f.
[(student(ayumi), 0)]?t/f  t.
[(820>=801, 0)]?t/f  t.
[(820=<826)]?t/f  t.
[asked(ayumi,li,820):-student(ayumi),820>=801,820=<826] is an error
yes
```

```
87: ?- delpro((asked(X,li,D):-student(X),D>=801,D=<826)).
Do you want to delete it from K-base?  yes
```

To supply a better environment to users, we have made many other predicates [16].

7 Conclusion

This paper has discussed the problem of terminating derivations like $KB \vdash \alpha$. To escape the undecidability of first order predicate logic, we have proposed a time-bounded reasoning, and designed a special system for it. Then we have discussed the properties, truth maintenance, and reliability of the TBR system. Although there may be some false knowledge in KB, it is only in U^+ and U^-, special parts of KB. DKB, the other part of KB, monotonically increases, and is sound and valid in some sense. We can see that the power of proof and the reliability of the TBR system will gradually become stronger and higher.

Also by using AB, we have shown our system can work effectively, although it has to recheck and modify the stored knowledge of KB. Furthermore, we have implemented a prototype system of the TBR, and with it we have also made sure of the properties above.

Our system stores the derived results in KB, and hence it has a learning function. The TBR is similar to our problem solving activities, so we can say that it is a natural reasoning method.

Further discussions on the query series and time function are left for future problems.

Acknowledgment

We would like to thank S. Miyano, A. Yamamoto, and A. Arimura for their helpful discussions.

References

[1] Blum, L., Blum, M., Towards a Mathematical Theory of Inductive Inference, *Information and Control 28*, 1975.

[2] Doyle, J., A Truth Maintenance System, *Artificial Intelligence*, Vol.12, pp.231-272, 1979.

[3] Golfesddrt, S., Micali, S. and Rackoff, C., The Knowledge Complexity of Interactive Proof-System, *Proceedings of the 17th STOC ACM*, pp. 291-303, 1985.

[4] Haraguchi, M., Arikawa, S., A Foundation of Reasoning by Analogy– Analogical Union of Logic Programs, *LNCS 264*, pp.58-69, 1987.

[5] Kitakami, H., Kunifuji, S., Miyachi, T., and Furukawa, K., Knowledge Base Management System Implemented in Logic Programming Language Prolog, *Information processing*, Vol. 26, No. 11, pp.1283-1295, in Japanese, 1985.

[6] Lloyd, J.W., Topor, R.W., A Basis for Deductive Database Systems, *J. Logic Programming*, Vol.2, No.2, pp.93-109, 1985.

[7] Lloyd, J.W., Topor, R.W., A Basis for Deductive Database Systems, *J. Logic Programming*, Vol.3, No.1, pp.55-67, 1986.

[8] Lloyd, J.W., *Foundations of Logic Programming*, (Second, Extended Edition), Springer-Verlag, 1987.

[9] Matsuda, T., Ishizuka, M., Knowledge Assimilation and Management Mechanism for Frame Knowledge-Base Including Hypothesis Knowledge, *The Transactions of the Institute of Electronics Information and Communication Engineers*, J71D,5,pp.902-908, in Japanese, 1988.

[10] Reiter, R., On Closed World Data Bases, *Logic and Data Bases*, pp. 55-76, viii+458, 1978.

[11] Reiter, R., The Default Logic, *Artificial Intelligence*, Vol.13, pp.231-272, 1979.

[12] Reiter, R., de Kleer, J., Foundations of Truth Maintenance: Preliminary Report, *Proceedings of AAAI-87 the 6th National Conference on Artificial Intelligence*, pp. 183-188, 1987.

[13] Shapiro, E.Y., Inductive Inference of Theories from Facts, *Technical Report 192*, Yale University 1981.

[14] Shapiro, E.Y., *Algorithmic Program Debugging*, MIT Press, Cambridge, Mass., 1983.

[15] Shapiro, E.Y., Alternation and the Computational Complexity of Logic Programs, *J. Logic Programming*, Vol.1, No.1, 1984.

[16] Shi, Y., *Studies on Knowledge Base Management Systems by Time-Bounded Reasoning and Contradiction Backtracking Algorithm*, The Dissertation of D. M., Interdisciplinary Graduate School of Engineering Sciences, Kyushu University, in Japanese, 1989.

[17] Shi, Y., The Principles of Time-Bounded Knowledge Base Management Systems, *Engineering Sciences Reports, Kyushu University*, Vol.11, No.1, pp.77-84, in Japanese, 1989.

[18] Wikfram, D.A., Maher, M.J., and Lasezz, J.-L., A Unified Treatment of Resolution Strategies for Logic Program, *Proceedings of the Second International Conference of Logic Programming*, pp. 263-276, 1984.

[19] Yamamoto, A., An Anatomy of Abstraction, *Bulletin of Informatics and Cybernetics*, Vol.22, No.3-4, pp.179-188, 1987.