ISSUES ON THE DESIGN AND IMPLEMENTATION OF AN INTELLIGENT DATABASE INFERENCE CONTROLLER

Bhavani Thuraisingham and William Ford

The MITRE Corporation, Burlington Road, Bedford, MA

ABSTRACT

The Inference Problem compromises database systems which are usually considered to be secure. Here, users pose sets of queries and infer unauthorized information from the responses that they obtain. An Inference Controller is a device that prevents and/or detects security violations via inference. This paper describes the issues involved in designing and implementing an intelligent database inference controller.

1. INTRODUCTION

It is possible for users of any database management system to draw inferences from the information that they obtain from the databases. The inferred knowledge could depend only on the data obtained from the database system or it could involve some prior knowledge possessed by the user in addition to the data obtained from the database system. The inference process can be harmful if the inferred knowledge is something that the user is not authorized to acquire. That is, a user acquiring information which he is not authorized to know has come to be known as the inference problem in database security.

We are particularly interested in the inference problem which occurs in a multilevel operating environment. In such an environment, the users are cleared at different security levels and they access a multilevel database where the data is classified at different sensitivity levels. A multilevel secure database management system (MLS/DBMS) manages a multilevel database where its users cannot access data to which they are not authorized. However, providing a solution to the inference problem, where users issue multiple requests and consequently infer unauthorized knowledge, is beyond the capability of currently available MLS/DBMSs.

A triple approach to research is needed to combat the inference problem: one is to build inference controllers which act during transaction processing,1 the other is to build inference controllers for database design, and the third is to build inference controllers to act as advisors to the Systems Security Officer (SSO). This is because the inference problem is a complex one and, therefore, an integrated and incremental approach is necessary to handle it.

This paper focuses on the issues involved in designing and implementing an intelligent database inference controller, which we will refer to as a knowledge-based inference controller. A knowledge-based inference controller should be able to detect the inference strategies that users utilize to draw unauthorized inferences and consequently protect the database from such security violations. In order for the inference controller to be effective, it must have knowledge of the various inference strategies that users could utilize. In addition, it must also have a way of determining which of the data/knowledge are sensitive. Finally, it must be able to reason about the various activities of the user and detect and/or prevent security violations via inference.

The knowledge-based inference controller that we have designed is called XINCON (eXpert INference CONtroller). XINCON is able to deduce unauthorized inferences that users could make via logical deduction, certain types of non-classical logical deduction, and analogical reasoning. XINCON is also extensible so that in the future it could reason about additional inference strategies such as inductive and heuristic reasoning. The major components of XINCON are a User Interface Manager, an Inference Engine, a Knowledge Manager, a Truth Maintenance System, and a Conflict and Contention Resolution Module. XINCON utilizes security constraints to determine the sensitivity levels of the data and knowledge.

The organization of this paper is as follows: In section 2, we discuss the need for knowledge-based inference control. We also briefly discuss the functions of an ideal knowledge-based inference controller. In section 3, we describe the issues involved in designing and implementing the knowledge-based inference controller XINCON. In particular, the modules of XINCON, knowledge representation, reasoning strategies, and a preliminary prototype implementation are described. The paper is concluded in section 4.

We assume that the reader is familiar with concepts in MLS/DBMSs. A useful starting point for MLS/DBMS concepts is the Air Force Summer Study Report [AFSB83]. Some background on the inference problem can be obtained in [MORG87, DWYE87, THUR87, HINK88, SMIT88, STAC90, THUR90, LUNT89, BUCH89, KEEF89].

2. NEED FOR KNOWLEDGE-BASED INFERENCE CONTROL

In this section, we discuss the need for knowledge-based inference control. We have identified two major needs. They are:

(i) Complex classifications of data/knowledge: If all of the data/knowledge is classified at the same security level, then there is no inference problem. In reality, data/knowledge can be classified based on content, time, associations, and
events. In section 2.1, we discuss the various rules that could be utilized to assign security levels to data/knowledge.

(ii) Complex inference strategies by users: Unless a user is intelligent enough to make deductions, it is pointless to have a sophisticated inference controller. However, even naive users have available to them several mechanisms to make complex deductions. An inference controller should be able to deduce the inference strategies utilized by various users. In section 2.2, we describe a variety of inference strategies that users could utilize to draw unauthorized inferences.

In section 2.3, we will briefly discuss the functions of an ideal knowledge-based inference controller. These constraints are called event-based constraints. (Note that logical constraints are rules that assign security levels to data and knowledge. We have defined various types of security constraints which classify complex relationships between data and knowledge. These constraints include the following:

(i) Constraints that classify a database, relation, or an attribute. These constraints are called simple constraints.

(ii) Constraints that classify any part of the database depending on the value of some data. These constraints are called content-based constraints.

(iii) Constraints that classify any part of the database depending on the occurrence of some real-world event. These constraints are called event-based constraints.

(iv) Constraints that classify associations between data (such as tuples, attributes, elements, etc.). These constraints are called association-based constraints.

(v) Constraints that classify any part of the database depending on the information that has been previously released. These constraints are called release-based constraints. We have identified two types of release-based constraints. One is the general release constraint which classifies any part of the database, depending on whether any value of another attribute has been released. The other is the individual release constraint which classifies a value of an attribute depending on whether a value of another attribute has been released.

(vi) Constraints that classify collections of data. These constraints are called aggregate constraints.

(x) Constraints which specify implications. These are constraints which assign fuzzy values to their classifications. These are called fuzzy constraints.

(xi) Constraints which specify implications. These are called logical constraints. (Note that logical constraints are rules that assign security levels to data and knowledge. They specify logical implications such as the value of an attribute A implies the value of an attribute B. The classification of A would then depend on the classification of B).

2.2 CLASSIFICATION OF INFERENCE STRATEGIES

In this section, we provide an overview of some of the various rules that users could possibly utilize to draw inferences. We will give examples of how such inference strategies can be applied to violate the security of a database system.

Inference by Deductive Reasoning: In this strategy, new information is inferred using well-formed rules. There are two types of deductions: classical logic-based deduction and non-classical logic-based deduction. We discuss each type of deduction here.

Classical Logic-Based Deduction: Rules in classical logic enable new information to be deduced (an example is the logical implication rule).

Non-Classical Logic-Based Deduction: We name the deductions not made within classical logic to be non-classical logic-based deductions. They include deductions based on probabilistic reasoning, fuzzy reasoning, non-monotonic reasoning, default reasoning, temporal logic, dynamic logic, and modal logic. Inferences based on this strategy are also according to well-formed rules.

Inference by Inductive Reasoning: In this strategy, well-formed rules are utilized to infer hypothesis from the examples observed.

Inference by Analogical Reasoning: In reasoning by analogy, statements, such as 'X is like Y', are used to infer properties of X when given the properties of Y. This type of reasoning is common to frame-based systems.

Inference by Heuristic Reasoning: Heuristics are criteria, methods, or principles for deciding which among several alternative courses of action promises to be the most effective in order to achieve some goal. In general, a heuristic is not well defined and may be a rule of thumb that is used to guide one's actions. Experts often use heuristics in order to solve a problem. Inference by

---

2 Much of the discussion in this section is obtained from THUR91.

3 We consider the relational data model [CODD70] in specifying the constraints. This is because our work has mainly been involved with handling the inference problem that occurs in a multilevel relational database management system. The specification of these constraints could be extended to include other data/knowledge representation schemes such as object-oriented data models, rules, and frames.

4 A discussion of these strategies is also given in [FORD90, THUR91]. Recently Garvey et al [GARV91] are considering an inference strategy called abductive reasoning [GARV91] in their work on the inference problem. This is an additional strategy that a user could utilize to draw inferences.
heuristic reasoning is the process of deducing new information using various heuristics.

**Inference by Semantic Association:** In this strategy, association between entities is inferred from the knowledge of the entities themselves. Various types of semantic associations have been identified. They include context-based associations, aggregation-based associations, and dependency-based associations.

**Inferred Existence:** Using the strategy of inferred existence, one can infer the existence of an entity Y from certain information on Y. For example, from the information 'Champion sails to Japan,' it is reasonable to infer that 'There is some entity called Champion.'

**Statistical Inference:** From the various statistics computed on a set of entities, information about an individual entity in the set is inferred.

### 2.3 KNOWLEDGE-BASED INFERENCE CONTROL

Ideally the inference controller should be able to use various types of reasoning techniques in order to combat the human inference strategies. Therefore, the ultimate solution to handling the inference problem is to build inference controllers which can mimic the human reasoning and thought processes. Although such a solution appears ambitious, it is not beyond the state-of-the-art. Artificial intelligence systems are being developed that use a variety of inference strategies and sophisticated learning techniques. A similar approach should be taken to developing inference controllers. A view of such an inference controller is shown in figure 1.

The knowledge base includes various rules, security constraints, heuristics, inference strategies, and input data from the MLS/DBMS. The inference controller could be used by the SSO (System Security Officer) off-line to help make decisions on the inferences that the various users have or could possibly make. Based on this advice, the SSO could modify the database, the schema or even the policy. The inference controller also gives advice to the MLS/DBMS from time to time, which the MLS/DBMS may use during transaction (either query or update) processing.

### 3. ISSUES ON DESIGNING AND IMPLEMENTING XINCON

In this section, we discuss the issues involved in developing a Knowledge-based Inference Controller which we call XINCON. In section 3.1, we provide an architectural overview of XINCON. Knowledge representation issues are discussed in section 3.2. Reasoning in XINCON is described in section 3.3. A preliminary prototype implementation is discussed in section 3.4.

#### 3.1 MODULES OF XINCON

The major modules of XINCON are shown in Figure 2. They are: User Interface (XUI), the Knowledge Manager (XKM), the Truth Maintenance System (XTMS), the Conflict/Contention Resolution System (XCCRS), and the Inference Engine (XIE), which is the heart of XINCON. A description of each module is given below.

XUI is the interface to XINCON. It can be used for updating the knowledge base, for querying XINCON, for obtaining advice from XINCON, or for requesting XINCON to solve a particular problem. XUI is also used if XINCON needs additional information from the SSO or other systems such as the MLS/DBMS.

![Figure 2. Modules of XINCON](image)

XKM is responsible for managing and structuring the knowledge base. It must also ensure the consistency of the knowledge base. Any access to the knowledge base is via XKM. It has interfaces to all of the modules of XINCON. The knowledge base stores all of the relevant information. This includes security constraints, real-world information, heuristics, and relevant information released to various users. The structure of the knowledge base will be discussed in sections 3.2.

XIE is the heart of XINCON. It has the potential for using a variety of inference strategies. Each inference strategy (or a combination of strategies) could be executed on a separate processor for efficiency. As a minimum, XIE should be able to perform logical inferences. In our design, XIE is also capable of performing fuzzy reasoning. That is, facts and rules are assigned fuzzy values and conclusions are drawn from facts and rules using Zadeh's fuzzy logic [ZADE65]. XIE uses plausible reasoning when dealing with uncertainty. Note that in a multilevel environment,
there could be different views of the same entity at different security levels. Therefore XIE should be able to reason across security levels. Issues on reasoning are discussed in sections 3.3.

XC CRS is responsible for resolving conflicts as well as determining the best choice to take when the system is presented with different options. For example, one particular reasoning strategy could potentially give results which conflict with another reasoning strategy. In such a situation, XIE would consult XC CRS to resolve the conflict. The conflict is resolved by XC CRS querying either the XKM or even the SSO if necessary. When XIE has to select between various rules XCRS is consulted. XC CRS will determine whether a particular rule will subsume the other or whether a rule could possibly be useless. It then gives appropriate advice to XIE. The knowledge needed for XCRS to function is represented as a set of rules. The following is an example rule:

\[ \text{If } R1 \text{ and } R2 \text{ are both true, then } R1 \text{ overrides } R2. \]

Therefore, when XIE concludes both R1 and R2, which are contradictory, then XC CRS will consult its knowledge base and conclude that R1 is valid. XC CRS is especially critical when reasoning across security levels where it is possible to have contradictory facts at different security levels. We discuss reasoning across security levels in section 3.3.

XTMS is the module that is responsible for maintaining the consistency of the various beliefs. Such a module is necessary for nonmonotonic reasoning. We illustrate its function with a simple example. Suppose the knowledge base includes the following information: "If CHAMPION'S destination is Libya and it carries explosives, then its destination is Libya", "CHAMPION is located in the Mediterranean and it carries explosives", and "CHAMPION carries explosives". Form this information, XIE would infer that "CHAMPION'S destination is Libya". That is, to support the fact that CHAMPION's destination is Libya, it must be located in the Mediterranean and it must carry explosive weapons. XTMS at time t1 will be maintaining the supporting evidence for the belief that CHAMPION'S destination is Libya. Suppose later, at time t2, it is found that CHAMPION is not carrying explosive weapons. Now, XTMS must retract its support for the belief that CHAMPION'S destination is Libya.

We assume that XINCON runs at system-high. The knowledge base stores all of its knowledge at system-high. However, knowledge and data are assigned different security levels. Advice given by XINCON to different users or the MLS/DBMS will depend on the users' level or the level at which the MLS/DBMS operates. If users cleared at different security levels are to use XINCON, then XINCON itself could be implemented as a multilevel knowledge based system. For example, XINCON could operate at different security levels depending on the user's level and the knowledge base could be stored at different security levels. More research needs to be done before multilevel knowledge based systems are developed, especially for controlling inferences. XINCON can be used as an advisor to SSO, it could be used to analyze the audit data for security violations via inference, or it can even be used as a real-time knowledge-based system.

3.2 KNOWLEDGE REPRESENTATION

The knowledge representation scheme used by XINCON is a combination of frames and rules. Frames are ideal to represent structured knowledge. The inheritance mechanism in frames is a powerful one which enables the representation of generic entities as well as instantiations of the generic entities. The frames used to represent the knowledge are called knowledge frames. Each knowledge frame describes a generic entity or a specific instance of a generic entity. A knowledge frame has many slots associated with it. Each slot describes some property of the entity represented or it could have rules or security constraints associated with it.

<table>
<thead>
<tr>
<th>Name of Entity: SHIP;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Type: Generic</td>
</tr>
<tr>
<td>Security Level: Unclassified</td>
</tr>
<tr>
<td>Information in Database: Ship#, Ship-name, Mission#</td>
</tr>
<tr>
<td>Other Information: None</td>
</tr>
<tr>
<td>Security Constraints: None</td>
</tr>
<tr>
<td>Instances: CHAMPION, - - - -</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name of Entity: CHAMPION;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Type: Instance of SHIP</td>
</tr>
<tr>
<td>Security Level: Unclassified</td>
</tr>
<tr>
<td>Information in Database: Inherit</td>
</tr>
<tr>
<td>Other Information:</td>
</tr>
<tr>
<td>(i) The destination is Libya with a fuzzy value of 0.2</td>
</tr>
<tr>
<td>(ii) If destination is Libya there will be war with a fuzzy value of 0.6</td>
</tr>
<tr>
<td>(iii) If ship is in the Pacific, then it cannot go to Libya.</td>
</tr>
<tr>
<td>(iv) Inherit</td>
</tr>
<tr>
<td>Security Constraints: If destination is Libya with a fuzzy value of 1.0 information related to CHAMPION is Secret</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name of Entity: CHAMPION;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Type: Instance of SHIP</td>
</tr>
<tr>
<td>Security Level: Secret</td>
</tr>
<tr>
<td>Information in Database: Inherit</td>
</tr>
<tr>
<td>Other Information:</td>
</tr>
<tr>
<td>(i) The destination is Libya with a fuzzy value of 0.6</td>
</tr>
<tr>
<td>(ii) If destination is Libya there will be war with a fuzzy value of 0.9</td>
</tr>
<tr>
<td>(iii) If ship is in the Pacific, then it cannot go to Libya.</td>
</tr>
<tr>
<td>(iv) Inherit</td>
</tr>
<tr>
<td>Security Constraint: Inherit</td>
</tr>
</tbody>
</table>

Figure 3. Knowledge Frames

Every knowledge frame has one slot which specifies the security level at which the knowledge frame is true. That is, the slot specifies the security level assigned to the knowledge frame. Note that if XINCON operates multilevel, then the knowledge frame is also stored at the level specified. The security level is necessary as entities can have different information at different security levels. Figure 3 shows Unclassified and Secret knowledge frames.

---

5 Note that any conflicts or uncertainties are handled by XIE by consulting XC CRS.

6 Detailed discussions on the theoretical foundations of truth maintenance can be found in [DOYL82].

7 System-high (System-low) is the highest (lowest) security level supported by the system.

8 For a discussion on level of assurance provided by computing systems, we refer to [TCSEC85].

9 The performance implications of using XINCON as a real-time system are yet to be determined.
which have information on the ship CHAMPION. Since CHAMPION is a ship, it inherits information from the knowledge frame which has information on the generic entity SHIP. Each knowledge frame also has real-world information and security constraints associated with it. Note that whenever the word "inherit" is used for a slot, it means that the value for that slot is inherited from the knowledge frame representing the generic entity of the specific instance.

In addition to representing knowledge as frames, rules are also used to represent some of the knowledge such as constraints, real-world data, and conflict resolution rules. The rules could be specified as Horn clauses, in which case the techniques developed for maintaining the consistency and completeness of horn clause logic programs could be used for the rules also. Rules could also be specified as clauses of the Logical Data Language developed at MCC [NAQV89] or as statements of the constraint language developed in the Cyc project at MCC [LENA89]. Each rule is assigned a security level.

### 3.3 REASONING

XINCON uses rule-based reasoning, frame-based reasoning, and fuzzy reasoning. In addition, it also reasons across security levels and under uncertainty. That is, XIE should be able to reason across security levels. In our design, XIE reasons as follows. In the example of figure 3, when XIE is reasoning with respect to users at the Unclassified level (that is when it attempts to detect the unauthorized inferences that the Unclassified users could draw), it considers the knowledge frame on CHAMPION at the Unclassified security level. If it is reasoning with respect to users at the Confidential level, then it still considers the knowledge frame at the Unclassified level as there is no knowledge frame on CHAMPION at the Confidential level. It is reasoning with respect to users at the Secret level, then it could do one of the following:

- Consider only the knowledge frame on CHAMPION at the Secret level.
- Consider both the knowledge frames on CHAMPION at the Unclassified and Secret levels.
- Consult with XCCRS as to which frame to consider

A simple solution would be to take the first action. That is, assume that information at level L is more accurate than the information at level L-1.1 In reality, however, information at a lower level could be more current than the one at the higher level. XCCRS could resolve the conflicts either by (i) checking the knowledge base for an appropriate conflict resolution rule, (ii) querying the user to give more up-to-date information, (iii) in the absence of appropriate information, make plausible guesses based on recent experiences, (iv) reason using the rules of a theory, such as plausibility theory.

### 3.4 A PROTOTYPE IMPLEMENTATION

XINCON could be implemented using a high level shell, a conventional language such as C, or an AI language such as LISP or Prolog. In this section, we discuss a simple inference controller that we have implemented in C.12

```
Query in SQL

User Interface

Parsed Query

Inference Engine
(Also the Inference Controller)

Response

Query/Update Knowledge Base

Multilevel Database and
Knowledge Base

Figure 4. Architecture of the Prototype
```

The architecture of the prototype is shown in figure 4. In this architecture, a commercial MLS/DBMS is augmented by a deductive manager. The inference engine was implemented entirely in C. It handles simple logical inferences that users could utilize to draw unauthorized inferences. The components of the deductive manager are the query modifier (which performs query modifications), the constraint manager (which manages the knowledge base), the response processor (which sanitizes the response), and the release database updater (which updates the knowledge base with information released to users). The implementation of the deductive manager consisted of about 8500 lines of code. The modules which handle conflicts and maintain the consistency of beliefs were not implemented. In addition, the inference engine did not handle any complex inference strategies. The knowledge consisted of the security constraints represented as horn clauses, release data information, and real-world events.

### 4. CONCLUSION

In this paper, we first described the inference problem in multilevel database management systems. We then discussed two major needs for developing a knowledge-based inference controller. These needs are the following:

(i) In a multilevel world, complex associations between data and knowledge could be classified using security constraints. An inference controller should be able to reason about the world and be able to detect and/or prevent the direct or indirect violation of these constraints.

---

10 Much of our work on knowledge-based inference control has been influence by the Cyc project [LENA90].

11 Note that in general the set of security levels forms a lattice. That is, the security levels are not strictly hierarchical.

12 We feel that ultimately XINCON should be implemented in a language such as Prolog. Issues on building expert systems in Prolog are discussed in [MERR89].

13 The commercial MLS/DBMS that we used is the Secure SQL DataServer targeted to be evaluated at the B1 level [SYBA89].

1701
Even naive users have available to them a complex set of inference strategies that could utilize to make deduction. For an inference controller to be effective, it should be able to reason about the inference strategies utilized by the users.

Next, we described the issues involved in designing a knowledge-based inference controller which satisfies the needs. The inference controller has all the essential components for a knowledge-based management system. That is, the user interface component manages all access to the inference controller. The knowledge manager component manages the knowledge base which is represented as collections of rules and frames. The inference engine uses a variety of reasoning strategies. In addition, it reasons across security levels and copes with uncertainty. The conflict and contention resolution module is responsible for handling contradictory situations. The truth maintenance module maintains the consistency of beliefs at different security levels. Finally, we described a preliminary prototype implementation.

We believe that all of the technology is available in place to develop a proof-of-concept prototype of the inference controller XINCON. As more progress is made in expert system technology, as well as in artificial intelligence technology, it will increase the possibility of developing inference controllers that can mimic human reasoning and thought processes.

ACKNOWLEDGEMENTS

We gratefully acknowledge the Department of the Navy (SPAWAR) for sponsoring this work under contract F19628-89-C-0001.

REFERENCES


[PEAR84] Pearl, J., 1984, Heuristics, Addison Wesley, MA.


