Security and Inference in Multilevel Database and Knowledge-Base Systems

Matthew Morgenstern [1]

SRI International
Computer Science Laboratory
333 Ravenswood Avenue
Menlo Park, CA 94025

Abstract

This paper addresses the threat to multilevel security that arises from logical inference and the semantics of the application Such compromises of security are particularly challenging since they circumvent traditional security mechanisms and rely on a user's knowledge of the application The problems of inference and security have heretofore been amorphous and difficult to circumscribe We focus on these problems in the context of a multilevel database system and show their relevance to knowledge-based systems, sometimes referred to as expert systems Here we establish a framework for studying these inference control problems, describe a representation for relevant semantics of the application, develop criteria for safety and security of a system to prevent these problems, and outline algorithms for enforcing these criteria

1. Introduction

A user who is knowledgeable about an application area may be able to circumvent security requirements by inferring some information for which he is *not* cleared based upon other data for which he *is* cleared Analysis of this type of security leak differs from most other security work because the leakage of information arises not from penetration of the security mechanisms directly but rather from the very nature of the information and the semantics of the application

A multilevel-secure system would ensure that data at classification levels above the user's login level would be *invisible* to that user Such a system is designed to prevent higher level information from being directly encoded in other data that are available at a lower level -- such as by changing the values of lower level data in a particular pattern or by selecting certain visible data for output based upon values of higher level data

However, a system's security mechanism could not anticipate, and thus could not prevent, the user's ability to infer higher level information based upon the semantics of the application -- unless that system also had available to it relevant knowledge of the application Such knowledge must make explicit the constraints that cause different data values to be interdependent

Database systems have lacked any representation of the inter-object and inter-relational constraints that characterize the application Thus database security has been specified at the syntactic level rather than at the semantic level On the other hand, knowledge-based and expert systems have enjoyed a richer representation, but have not addressed security issues -- all data are treated at the same level

When security is of concern for rule-based expert systems, classifications will need to be applied not

[1] I would like to thank my colleague Dorothy Denning for our interesting discussions and for her helpful contributions throughout the course of this work

only to the data, but also to the rules themselves These rules represent the knowledge of experts, and privileged knowledge will be classified at higher levels than other rules Thus the results produced by the expert system may depend upon the authorization of the user!

The rule-base for an expert system defines a set of inference rules Whereas for a database we first need to make explicit the inference rules that describe the application, for expert systems the basis for important inferences already is represented in the rule-base As a consequence, the same mechanisms which we develop here to analyze the inference problem in databases also can be applied to expert systems -- we elaborate on this point below and hope to pursue this application to expert systems in future work

The problems of inference and security in multilevel systems have heretofore been amorphous and difficult to circumscribe Here we propose a framework and a methodology with which to address these inference control problems The concept of *Sphere of Influence* serves to delimit the potential for indirect disclosure This represents the scope of information that could be inferred from a collection of data by means of a forward-chained inference process Another concept of *Inference Channel* then utilizes a backward-chained process from some resultant data to isolate the specific sources of such inferences The interdependences among data are characterized in terms of constraints, and the examples are illustrated in terms of a particular constraint representation language We also develop high level algorithms for the detection and prevention of security leaks due to inference

2. The Inference Problem

The term *inference* often is used to refer to the logical process of proving or deriving some conclusions from some given facts and rules In other situations, the term *inference* is used in a more information theoretic sense For example, knowing a person's profession may localize the space of possible values for his salary We say that obtaining some information (e g profession) reduces the *degrees of freedom* of the other data values Our representation of application semantics attempts to capture both logical inference and the information theoretic measure in terms of constraint expressions

Perhaps the earliest work on inference arises from concerns of the U S Census Bureau and their desire to prevent the disclosure of personal information Their work primarily concerns statistical inference involving numeric quantities such as family income and size of household They use techniques such as linear equations to express the interrelationship among statistics from several tables For a survey of such techniques, see [DeSc83]

However, research regarding logical inference and general data and knowledge bases is in its early stages Sicherman et al [SiJR83] considered the problem of when not to answer a query -- they take into account the answers already given to prior queries so as to prevent absolute deduction of secret facts, but they do not consider partial inferences Trueblood [Tru84] illustrated how knowledge-based systems and expert-type systems could utilize unclassified data to draw conclusions about sensitive data He proposed augmenting the primary knowledge-base with a security knowledge-base that dynamically evaluates the sensitivity of the inferences that are formed utilizing the primary rules Berson and Lunt [BeLu87] give examples of security problems that could arise in expert systems' applications

More recently, Su and Ozsoyoglu [SuOz87] utilize functional dependencies (FDs) and single multi-valued dependencies (MVDs) as their basis for inference in a database Their interesting work considers first how attribute classifications can be assigned in order to avoid inference via known FD mappings They "assume that for any FD mentioned, the user also knows the related mapping" Since most FD mappings are not defined by an algorithm but rather by the tuple instances (actually their projection onto the attributes of the FD), this assumption limits their solutions For example, if Rank functionally determines Salary, they assume that the instances of this mapping are visible in an explicit relation table Thus they recommend that a person's Rank should be classified high enough to avoid inference of one's Salary However, another solution that should be considered is to classify the relation [RANK SALARY] itself higher so that it is *not*

known to users They also show how a single MVD can pose a danger when record level classifications are utilized For example, if classified tuples share the same MVD values with other tuples, then these other tuples may need to be classified higher

In an unpublished paper, Rowe [Row86] suggested using logic programming techniques to address more general forms of inference -- by capturing all facts, policies, and known conditions as clauses in logic He suggested that resolution could be used to help locate inference problems, though he did not provide a specific method of analysis nor did he consider partial inferences However, Rowe's efforts did provide motivation for joint work by the author and Dorothy Denning [DeMo86]

2.1. Example

A detailed example will help, at this point, to introduce the problems of inference and security in a database system, and it will help to outline our approach for resolving these problems Our examples utilize a constraint language which is described later in the paper -- and further details are available in [Mor84b, Mor86a] However, the ideas and techniques developed in this paper are largely independent of the particular formalism we utilize to represent the application semantics and the possible inferences

This example shows how the following inference arises and how it compromises the intended security the Secret topic of a project is inferred from Unclassified data on authors by a user who knows that Smith is an expert on lasers Security is compromised since the user is at the unclassified level

The relational database has a simple two-level classification Unclassified and Secret The application data includes projects, topics (of projects), reports, authors, employees (of projects), and specialty (primary expertise of an employee) The database includes the following two relations, R1 and R2, classified as shown

R1 [PROJ REPORT AUTHOR] Unclassified
R2 [PROJ TOPIC] Secret

The application-specific knowledge that most users already know is represented by the following two constraints Constraint C1 expresses the fact that the authors of project reports are always employees of the project The second constraint, C2, states that in this application the technical topic(s) of a project are the same as the (primary) specialties of the project's employees

C1 PROJ REPORT AUTHORS $\subseteq$ PROJ EMP
C2 PROJ EMP SPECIALTY == PROJ TOPICS

These constraints are expressed in abbreviated form [2] For example, the association among PROJ, REPORT, and AUTHORS is explicit in relation R1 The association between PROJ and EMP is not needed explicitly in the derived constraint C3 below -- but constraint C1 tells us that a subset of instances of this association can be obtained from the projection of relation R1 onto PROJ and AUTHOR This relation [PROJ EMP], as well as the relation [EMP SPECIALTY], can be just virtual relations -- they are part of what we describe below as the *universe of discourse* What is important for this example is that one or more instances of the association between EMP and SPECIALTY is part of the knowledge that a user has about the application, as we now discuss

The potential for inference of the secret data can be anticipated by symbolic composition of constraints C1 and C2, as described here informally [Mor86a, Mor84b] Each constraint consists of two *subpaths* separated by a comparator For constraint C1, we extend the right hand side of both of its subpaths with the qualification of SPECIALTY to obtain the following constraint

C1b PROJ REPORT AUTHORS SPECIALTY
$\subseteq$ PROJ EMP SPECIALTY

Since the right subpath of this constraint is identical to the left subpath of constraint C2, we easily obtain constraint C3, which expresses symbolically the critical inference that the topics of a project include the specialty areas of the authors of project reports

C3 PROJ REPORT AUTHORS SPECIALTY
$\subseteq$ PROJ TOPICS

[2] The dot between components, such as in PROJ REPORT, represents the unique association between the attributes (or objects if we were dealing with an object-oriented schema) on either side of the dot If there is more than one such association, then it must be named explicitly A dot also can represent a join if it is unambiguous

The actual inference that violates the intended security by revealing the Secret topic of certain projects is made by a user who knows that Smith is an expert on lasers The inference is represented by specializing constraint C3 with this information, yielding C3b

C3b

PROJ REPORT AUTHORS(=Smith) SPECIALTY(=Lasers)
$\subseteq$ PROJ TOPICS(=Lasers)

This result shows that projects for which Smith authored a report are projects on lasers While the database had the relation [PROJ TOPIC] classified as Secret, this classification did not adequately protect the information

A simple analysis shows how the compromise of security could be avoided one of the logical connections occuring in the left subpath of constraint C3b could be broken, relative to an unclassified user For example, relation R1 could be decomposed into two relations [PROJ REPORT] and [REPORT AUTHOR] and either of these could be classified as Secret, thereby preventing an unclassified user from making the necessary associations

2.2. Inference Function

The information theoretic measure of inference has been formalized as follows [Den82, DeMo86] The amount of information about y which can be inferred from x is the *relative equivocation* It represents the reduction in *un*certainty about y when x is known -- ranging from zero if x provides no information about y, to one if x provides full knowledge about y

We define the function INFER($x \rightarrow y$) to be this *relative equivocation* whenever this value is greater than some threshold ϵ, which also is in the range 0 to 1 While an idealistic view would prefer to guarantee that there are no information flows to lower security levels, in practice there may be information flows that are so minute that it is appropriate to disregard them The parameter ϵ gives us a handle on quantizing this tolerance It may be set to zero if no tolerance is allowed, while non-zero values quantize the degree of such tolerance

We define INFER in terms of the *uncertainty* or *entropy* of y, denoted $H(y)$, and the *relative uncertainty* of y given x, denoted $H_x(y)$, as follows

Inference Function

$$\text{INFER}(x \rightarrow y) = [H(y) - H_x(y)]/H(y) \quad \text{if } [H(y)-H_x(y)]/H(y) > \epsilon$$
$$= 0 \quad \text{otherwise}$$

For example, if y represents a U S mail ZIP code, and there are 2^{16} ZIP codes, all equally likely, then $H(zipcode)$ is 16, representing 16 bits of uncertainty If x denotes a telephone area code and there are only 2^7 equally likely ZIP codes per area code, then $H_{areacode}(zipcode)$ is 7, denoting the number of remaining bits of uncertainty about the ZIP code given the area code Then INFER($areacode \rightarrow zipcode$) = 9/16 = 56 (so long as the threshold ϵ is less than 56) In other words, the amount of information that we can infer about the ZIP code given an area code is 0 56 on a scale of 0 to 1

3. Characterizing Ingredients for Inference

When can a multi-level-secure system leak information to lower levels through inference? Assuming the system structure itself is secure, the answer lies in the way the data are assigned particular classification labels -- we sometimes informally refer to these labels just as the 'classification level' of the data If data at a low level could be used to infer a limited set of possible values (or worse, a specific value) for some more highly classified data, then the classification labels were assigned incorrectly The classifications then are *inconsistent* with respect to the application semantics

Even if classification rules were explicit and were analyzed, no inconsistency among these classification rules need be evident, as it is the application semantics which give rise to the inconsistency While we utilize constraints to represent these application semantics, as in the example above, the concepts and techniques that we develop for analyzing and preventing breaches of multilevel security through inference are substantially independent of this representation

3.1. Data Objects and Granularity of Information

In order to assign classifications, we must decide what granularity of information and what kind of information we want to classify Data takes on its meaning by virtue of its association with other data Thus ultimately we are classifying associations, either explicitly or implicitly

If we have a relational database, we may utilize any of the following classification strategies (1) classify whole relation tables, or (2) classify at the attribute level -- that is, classify all instances of a relation attribute at the same level,[3] or (3) classify individual tuples separately (instance level classification), or (4) classify at the element level, that is, allow each non-key element of a given tuple to have a separate classification[4]

Similarly, if we have an entity-based system, then the classification strategies include (1) classify an entity type, which means treating all attributes and relationships of the entity uniformly, and apply this classification to all instances; or (2) classify each attribute and relation of an entity type separately and apply this classification to all instances, or (3) do instance-based classification for the whole entity instance (including all its attributes and relationships), or (4) do element level classification where each attribute or relationship instance of an actual entity may have a separate classification

The most control is obtained if we apply classifications to the finest grained units of information that are meaningful In the literature, these units have been called *atomic facts* -- they are *non-decomposable* associations in that further decomposition would loose information [Dat86a] The atomic facts serve as independent components which capture the original dependencies and do not lose information In the case of the relational model, the atomic facts often are the functional and multi-valued dependencies Each *meaningful association* between a key and each other attribute of the relation would be treatable independently regardless of how the schema is normalized (When two or more fields constitute a key of this relation, or serve as a foreign key for linking to another relation, it may be appropriate to treat them together)

One of the advantages of entity-based schema is that the attributes of an entity and the relationships among entities are usually the atomic facts and non-decomposable associations which are the appropriate subjects of classification The entity schema makes these facts explicit, whereas a relational schema may combine several atomic facts together in one relation In any schema, we use the term **data object** to refer to those units of information that are to be treated individually and classified separately from each other

3.2. When To Classify Database Rules

Few database systems utilize complex integrity constraints Rather such semantic information about the application environment must be captured in some representation, such as rules or more expressive constraints, in order for the system to anticipate inferences that a knowledgeable user might make However, knowing when to classify a rule or a constraint and when not to is an important question If one attempts to classify a rule in the model, but most unclassified users of the system know this rule, then the model will be inaccurate, and conclusions about the security of the system may not be correct

For example, it would do no good to classify a rule for computing sales tax, since users could easily determine this tax rule elsewhere Similarly, an explicit relationship -- such as a table between the amount of purchase and the sales tax -- should not be classified when the relation instances can be computed from a known rule Constraints, rules, and algorithms often serve to describe the application how it functions and how it interacts with its environment Such commonly known

---

[3] In the case of *attribute level* classification in the relational model, the association being protected may not be obvious For a non-key attribute, the primary association being protected is the functional dependency between the key and this attribute If the attribute is part of a key, then the protection may be for a multi-valued dependency or for access to the tuple based on its key Attribute classification also protects the association between the attribute type and the *set* of current values for that attribute

[4] Each element classification actually serves to classify the association between that element and the key of the tuple Such fine grained classifications generally are not supported, though research in underway to do so [DeLS87]

constraints and rules of inference should not be classified

Even when relation instances can be generated by a rule, it might be incorrect to classify the rule and the relation table at the same level In fact, the rule might properly be classified lower than the relation instances that are derivable from this rule, The reason is that a stored relationship makes explicit the set of values that actually occur Knowledge of specific values is additional information that is not typically present in a rule Rather the rule must be given some value(s) before it can produce the other associated value(s) Thus if the rule is common knowledge but the relevant instances are not, and should not be, generally known, then the table of tuples may warrant classified status while the derivation rule might be unclassified [5]

When we are concerned with traditional databases, many of the constraints and rules that describe the application warrant unclassified status We are including these constraints in the model of the application *because* many user's know these constraints, and we want the system to be able to anticipate and prevent inferences that could compromise security Unless we are modeling multiple classes of users who know different constraints about the application, we would expect most such constraints regarding a database to be common knowledge The situation is somewhat different for expert and knowledge-based systems, as we discuss next

3.3. Knowledge-Based Systems

The best known knowledge-based systems (KBS) utilize production rule computation engines and are commonly referred to as 'expert systems' because of the so-called expertise they embody for solving certain classes of problems These systems may use either forward or backward chained activation of the rules, or a combination of these paradigms Firing of a rule serves to *propagate* information through the rule (and modify/enhance that information based upon the purpose of the rule)

Two other types of knowledge-based system technology are (1) semantic network knowledge representation systems such as KL-ONE [ScBr82], which have been utilized for tasks such as natural language understanding, and (2) truth maintenance systems [deK86], which maintain a system of constraints (often numeric) in the face of changing data values In addition, *blackboard systems* [Hay85], which were first developed for speech recognition, are now more of a framework for an A I system architecture -- they serve to integrate multiple AI modules, each of which can be a complex knowledge source

In all of these KBS systems, the knowledge, rules, and constraints go well beyond modeling commonly held knowledge about an application Rather, these systems attempt to embody knowledge that is available to only the select experts in the field Thus for applications where some information should not be equally available to all users, the individual rules and other forms of knowledge may well warrant classified status For example, the rule that calculates the expected rate of fuel consumption of a new military airplane as a function of its air speed and cargo weight probably would be classified It will be necessary to determine which rules and constraints of an application are not generally known and at what level they should be classified

Another dimension to the inference problem arises in KBS because the same problem or query, based upon the same data, could give rise to two *different* results for two different users if the authorization levels of these users differ -- examples are given in [BeLu87] In KBS (and in future intelligent databases [Mor86b]) the inferences and results will depend not only on visible data but also on which rules are available at the authorization level of the user

One aspect of analyzing inference problems actually is simplified for knowledge-based systems Whereas for databases we first have to understand and represent the relevant semantics of the application, for KBS most or all of these semantics already are captured in the rule-base On the other hand, for databases relatively few of the constraints and inference rules will warrant classified status, while for KBS a larger portion of

[5] However, an exception arises when the instances are not sensitive but a secret rule is sufficiently complex so that it could not be determined from the instances -- then it may be appropriate to classify the instances lower than the rule

the rules may be classified at levels above system low Since we have the mechanism for treating different rules at different levels, this mechanism is just as applicable for analyzing inference and security in KBS as in more traditional databases

In particular, we treat rules that are not at system low as types of objects -- these rule objects then participate in the appropriate *spheres of influence* Their contributions are evaluated in just those cases where the rules would be accessible and could be utilized to draw potential inferences Thus once the application semantics are captured -- either already available for KBS or acquired for a database application -- then the analysis of inference and security can proceed similarly in either case utilizing the mechanisms developed here

Furthermore, when the semantics of an application are available to the system, it is natural to want the database to take advantage of this knowledge to provide new and enhanced capabilities, such as intelligent responses to queries and to other database activity [Mor86b]

3.4. Universe of Discourse

We wish to model not only constraints that apply to stored data, but also constraints that involve other data objects in the application environment, even though they are not explicitly stored in the database And we wish to suppress the distinction between stored and derived data so as to provide a uniform basis for design and analysis of the conceptual schema

We thus define a **universe of discourse** U for the application as being all entities, attributes, relationships, constraints, and rules that are of interest The traditional database schema is a subset of this universe The *universe of discourse* provides the vocabulary of objects that participate in the constraints, and to which we will assign classification levels

In fact it may be the case that some stored data objects are interrelated only through information that occurs in the application environment but is not itself stored This information would be represented as virtual data objects in the universe of discourse For example, we noted earlier that the relationship between addresses and telephone area codes might be relevant for an application If the relation is not actually stored in the database, it still is important to represent it in the universe of discourse

3.5. Lattice of Classifications

Classification labels may not form a strict hierarchy of levels due to compartmentalization of information The actual classification then consists of the concatenation of the level with the compartment name, thus giving rise to a lattice of classification labels (informally, we often refer to these as levels anyway) Thus if we have two classifications c_1 and c_2, we say that c_1 *dominates* c_2 if in the lattice c_1 is an upper bound for c_2, including equality We can have two classifications where neither dominates the other because they are incomparable (due to the compartments)

We use the notation $c_1 \not\geq c_2$ to mean that c_1 either is lower than c_2 or that the classifications are incomparable Thus $\not\geq$ is not equivalent to $<$ due to the partial ordering For an object O, we will refer to its level equivalently as either *level(O)* or *O level*

4. Sphere of Influence

We introduce the concept of **sphere of influence** (SOI) as a tool for analyzing conflicts between inference and multilevel classifications, and as a tool for isolating potential security leaks due to inference The SOI delimits the scope of possible inferences given some base data We refer to the base data as the **core** -- it may consist of entities, attributes, relationships, and inference rules (constraints)

Specifically, the *sphere of influence* relative to some *core*, i e SOI(core), is the set of all information that can be inferred from that core -- often a proper subset of the core will be sufficient for a particular inference The SOI is defined in terms of the INFER function when this information theoretic measure exceeds the threshold ϵ

Sphere of Influence

$$SOI(core) = \{ Y \mid \exists X \subseteq core \wedge (INFER(X \rightarrow Y) > \epsilon) \}$$

The SOI models the process by which a user's knowledge of an application can give rise to inferences about additional information The SOI utilizes a *forward-chained* inference process from the given core to determine the scope of such inferable data -- which we refer to as the *extent* of the SOI Both the core and the extent consist of data objects defined in the universe of discourse, including data that are stored and data that may be virtual or derived The constraints serve as the inference rules, which are applied in the forward-chained process

In effect, we take the closure of potential constraint compositions Thus sphere of influence is related to the notion of inferential closure, as discussed by Goguen and Meseguer in the context of a logic interpretation of inference [GoMe84] However, the sphere of influence takes a somewhat different viewpoint it utilizes a subset of the rules/constraints -- just those constraints which involve the core objects and also are dominated by the user's clearance level -- and it emphasizes the set of related data objects rather the set of all composite inference rules The SOI approach takes into account an information theoretic notion of partial inferences in addition to the strict logic interpretation of inference

For example, the sphere of influence for a person's address may include that person's telephone number, since the address determines the single area code and a limited set of telephone exchanges Other objects in this SOI might include the person's zip code, utility company, school district, or local political representatives

A potential security leak exists if the extent of a SOI includes data objects at higher (or incompatible) classification levels than the levels of the core objects Security is compromised since inferences are possible about these higher level objects The SOI also takes into account *aggregate classifications*, which arise when a data collection as a whole is classified higher than any of its elements The classic example is a telephone book for a military base -- individual phone numbers are released as unclassified, but the whole phone book, as an *aggregate*, is classified higher

4.1. Quantizing the Degree of Influence

The sphere of influence can be given a geometrical interpretation by defining $D_{core}(y)$ as the *Distance* of some object y from the *core* of the SOI

SOI Distance

$$D_{core}(y) = [\ 1 - INFER(core \rightarrow y)\ ]$$

Distance D ranges from zero to one -- ie, we have a unit sphere where D is the radius The function $INFER(core \rightarrow y)$ represents the non-zero amount of information about objects $y \in SOI(core)$ which can be inferred given the *core* data, and knowledge about the application Thus objects with a small distance D are substantially influenced by the core objects -- that is, when we know the core object(s) then we know much about an object that is close The larger the distance D the further the objects are from the core, and the less they are influenced by the core [6]

If the tolerance parameter ϵ in the definition of the INFERence function is set to a non-zero value, then the maximum distance of the SOI is $(1-\epsilon)$ rather than one Objects of distance greater than $(1-\epsilon)$ are in the "outer fringe of width ϵ" They are trimmed off from further consideration in this SOI, since very little becomes known about them from the core objects

4.2. Safety

A data object O, and its assigned classification label, are said to be **safe** for inference if there is *no upward inference* from them -- that is, object O cannot be used to infer information about other data objects at higher or incompatible levels of the classification lattice Thus if object O could give rise to inferences about some other data objects X, then *O level* must dominate (or equal) *X level* for object O to be *safe*, as indicated in the following definition

[6] We thus have a model which is analogous to the effects of gravity, though not with an inverse square law Contour lines could be drawn to describe the different degrees of inference Then all data objects influenced by the core objects to the same degree would be connected on the same contour line

Object Safety

Data object O is *safe* if
$\forall X$ [(INFER($O \rightarrow X$) > ϵ)
$\Rightarrow$ (O level $\geq$ X level)]

While our definition of safety looks at inferences emanating from object O and giving information about other objects, we could invert the question by asking whether any other data in the database could be used to infer information about object O We call the possibility of such inferences about O an *inference channel* to object O, and we will address it shortly In part, we have chosen this way of slicing the problem because we want to analyze traditional forward reasoning so that we can determine the implications of a given set of data before we release it to a user or store it in the database

4.2.1. Safety of a Sphere of Influence

Similarly, any sphere of influence is *safe* if there is *no upward inference* from the *core* of the SOI This will be the case if the classification levels of the objects in its *extent*, SOI(core), are upper bounded by the highest level of any objects in the *core*, as we define below

We introduce some definitions to simplify further discussion The classification level for a *set* of objects is the least upper bound (LUB) of the classification levels of the individual members of the set and any *aggregate classifications* This least upper bound is relative to the partial ordering imposed by the classification lattice [Den82] An aggregate classification arises when some set of data as a whole is classified higher than any of its individual members (e g a military telephone book), as mentioned earlier We use a representative to stand for each such aggregate set Let AggregatesIn(X) be the set of zero or more such representatives for the aggregates occurring within some set X We refer to the classification level of set X equivalently as either *level*(X) or X *level*, and we define it as

level(X) = LUB{y *level* |
y $\in$ (X $\cup$ AggregatesIn(X))}

Now we can define the *safety* of a sphere of influence that arises from any chosen *core*

SOI Safety

SOI(core) is *safe* if
level(core) $\geq$ *level*(SOI(core))

Thus a SOI is safe if the classification levels of the objects in SOI(core) are upper bounded by the LUB of the levels for the objects and any aggregates within the core Note that if several SOIs were chosen so that they overlap the same object, then this overlap would impose multiple requirements on the classification levels, all of which must be satisfied

The constraints and rules of inference that describe an application often are asymmetric That is, INFER(X$\rightarrow$Y) and INFER(Y$\rightarrow$X) may have very different values For example, a ship's identity fully determines the current destination of that ship On the other hand, a particular location implies rather limited information about ships, since there may be many ships with that destination The INFER relationship is naturally reflexive, and may be transitive depending upon the application

For those few cases when the INFER relationship is symmetric and transitive, then the sphere of influence establishes an equivalence relation with regard to the classification levels in the core and in the extent of the SOI Then the above condition for safety of a SOI reduces to the requirement that all data objects in a SOI have the *same* classification level assignment

4.2.2. Safety Theorem for Classification Level

A *classification level is safe* for inference if all the data objects having that classification are safe The following theorem determines the safety of a classification level in terms of whether a particular sphere of influence yields a ***fixed point***

We denote the Universe of Discourse as U, and we let U_α = (U $\cup$ AggregatesIn(U)), that is, U augmented with a representative for each aggregate set having a higher classification than its members The set of all objects and aggregates O at and below a classification level l is expressed as U_l If all inferences derivable from U_l do not extend outside U_l, then this is a *fixed point* relative to

the SOI, namely $U_l = \mathrm{SOI}(U_l)$

THEOREM Safety Theorem for a Classification Level l

Let $U_l = \{O \in U_\alpha \mid O \text{ level} \leq l\}$

Then $[U_l = \mathrm{SOI}(U_l)] \Rightarrow l$ is *safe*

To prove the Theorem, we note that the fixed point means the SOI is *safe*, and since l is the highest level in the core, each of the objects at this level l is *safe* Actually the fact that U_l is a *fixed point* of the SOI is slightly stronger than ensuring that every object at level l is safe The above theorem also requires that an object at l does not contribute to any *upward inference* of information from other objects of U_l to a level above l -- this distinction arises when the threshold ϵ is greater than zero Note that an application of the Theorem considers the safety of only the objects at level l of the core At each of the lower levels the theorem would be reapplied to determine the safety of that level

4.3. Sanitization

As an illustration of the SOI concept, consider the effects of sanitization on the SOI Sanitization takes data at a high classification level and derives other data which may be shown at a lower level because it does not reveal anything important about the higher level input data For example, statistics released by the Census Bureau have been sanitized in advance so no personal information is revealed For an on-line multilevel system, dynamic sanitization must be done only by a *trusted subject*, that is, a program which is verified to always lose sufficient information during the downgrading to a lower classification

Consider the sphere of influence whose core consists of sanitized lower level data The data at the higher classification would appear at the fringe of the SOI since so little can be inferred about it from the low level data at the core When we realistically treat ϵ as a non-zero tolerance, we effectively remove these fringe effects Thus the original classified data is not in the SOI of the sanitized data, showing that no significant inference about the classified data is possible

5. Inference Channels

Our concept of **inference channel** serves to isolate the lower level data which could give rise to inferences about higher level data H The presence of an *inference channel* means that security could be compromised The computation of an inference channel is a *backward-chained* inference process from some resultant data H to determine all information which contributes to upward inferences about H The criteria for existence of an inference channel is

Existence of Inference Channel

InfCh exists
$\iff \exists$ sets H, C $[H \subseteq \mathrm{SOI}(C) \wedge (C \text{ level} \not\geq H \text{ level})]$

An *inference channel* exists if information about some set of data H may be inferred from data in another set C which is at a lower level than H, or at an incomparable level relative to H in the classification lattice That is, a SOI based upon C as the core would have the higher level data H in its *extent*, making that SOI *unsafe* We define the **inference channel** *InfCh*(H) to be such a set C of data objects which contribute to inferences about H -- subject to the following criteria

InfCh(H) includes all and only those data objects which allow inferences to be made about H -- but excludes any data which does not contribute to the inferences about H Since the inference threshold ϵ may be non-zero, we also include an object x if it increases another potential inference, even if it alone does not lead to an inference above the threshold *InfCh* is maximal with respect to lower level objects which could influence H, but minimal in that it does not include extraneous data objects that have no influence Letting U denote the space of all data objects, (with U_α including aggregates as above) we express these criteria for an inference channel as follows

$$\begin{aligned}
&\mathit{InfCh}(\mathrm{H}) = \\
&\quad \{ x \in U_\alpha \mid x \text{ level} \not\geq H \text{ level} \ \wedge \\
&\qquad (\mathrm{INFER}(x \rightarrow H) > \epsilon \ \vee \\
&\qquad\quad \exists Z \subseteq U_\alpha \ [Z \text{ level} \not\geq H \text{ level} \ \wedge \\
&\qquad\qquad \mathrm{INFER}((Z \cup \{x\}) \rightarrow H) > \\
&\qquad\qquad (\mathrm{INFER}(Z \rightarrow H) + \epsilon) \ ])\}
\end{aligned}$$

The *width* of this inference channel *InfCh*(H) is

defined to be the value of INFER($InfCh$(H)→H), since this is the amount of information (the reduction in degrees of freedom) we can obtain about H given the data in $InfCh$(H) The analogy here is that the wider the inference channel, the more information one can infer

5.1. Correcting Potential Inference Channels

If an inference channel is found for some assignment of classification levels, then we must take an appropriate course of action Here we consider some of the primary alternatives, where X is the inference channel $InfCh$(H) which gives us information about higher level data H

First, if the inference channel X consists of a single object, then we can simply raise the classification of this object to the level of H (or higher) If X is a set, then we may raise the level of one or more of the objects comprising X until the degree of inference is below the tolerance ϵ By raising the classification level, we effectively remove these objects from the core of this SOI -- they will be considered for some other SOI whose core is at a higher classification Note that we also must determine if inferences are now possible about this data with the newly raised classification from other lower level data

A second alternative applies when X is a set impose an aggregate classification on set X or some subset of X The level of this aggregate would be at least H level Data objects within this aggregate would be released in a controlled way, so that the whole aggregate could not be inferred from the individual values given to a lower level user This controlled release may be considered a form of sanitization, and typically is done by a trusted subject

The third alternative is to introduce noise to data released from the set X -- that is, add randomized perturbations so that H is no longer inferable with significance We characterize this process as follows for each of the objects in the set X, we derive new objects by adding a noise component to its original value -- these new objects form set X' The original set X is then replaced by this new set X' at the same classification level The original set X either would not be retained in the system or else would be classified at a higher level This process of introducing noise also can be considered as a form of sanitization of the original data X to obtain the sanitized version X'

The last option is the obvious alternative to accept the inference channel but to monitor it This option may be chosen for pragmatic reasons Inspection of audit trails may provide a sufficient threat of detection and removal of the intruder from the system to allow this alternative to be used in some cases

5.2. Criteria for System Security

A multilevel database system is **secure** from inferences if there are no *inference channels* in the system The following theorem assures us this will be the case if *every* classification level is *safe*

THEOREM Security of System
Let L be the lattice of classifications for the system
L is *secure* $\iff \forall l \in L$ (l is *safe*)

Safety of a level l means that no inferences can be made from data at that level about other data at a higher or noncomparable level -- i e level l is not part of an inference channel When this is true at all levels in L, it follows that there are no inference channels from any data objects, and thus the system is *secure* Similarly, *security* of a system implies *safety* of each of its levels, since there are no inference channels

We also can derive a simpler *sufficient* condition for *security* of a system However, since it is a stronger condition, it is not a necessary condition The following theorem states this sufficient condition in terms of the set of constraints and inference rules which constitute the model M of the application L is the lattice of all classifications for the system, as before Let objects(r) denote the set of data objects referenced in constraint or rule r

THEOREM Sufficient Condition for System Security
[$\forall$ r $\in M$
$\forall$ p,q $\in$ objects(r) (p level = q level)]
$\Rightarrow$ L is *secure*

In other words, a sufficient condition for system security is that each constraint or inference rule involves objects at just one classification level, though of course different rules may involve objects at different levels This condition is sufficient since it ensures that every data object, and thus every classification level, is *safe* But although it is simple to verify, it is overly strict For example, a sanitization rule would involve objects at two levels, and many other rules may involve only small degrees of inference in the direction from low to high level objects

It is important that all constraints and rules which are valid for an application be represented, else an inference may not be detected but still could occur Removing a constraint would not remove the problem, rather it would mislead one into thinking that the problem did not exist Rather, a problem would be corrected by adjusting the classification levels of the object types involved, as discussed above The theorems for safety and security rely on completeness of the model While completeness may be difficult to guarantee, not attempting to achieve it is much more likely to leave uncovered potentially serious violations of security

6. Representing Application Semantics

We find it advantageous to utilize a constraint language to capture the necessary application specific knowledge, and we use it for the examples in this paper We also have studied the use of a prolog-like representation [DeMo86] In either case, the concepts and techniques presented in this paper for inference control in databases and knowledge-based systems are relatively independent of the representation

Traditionally, formal logic is used to construct precise derivations of certain objects from other information -- constraints can serve this purpose too However, constraints also can be *partial*, in that they may just limit the possible values of the data objects in question Some relevant aspects of our *constraint expression* (CE) language are described here briefly (in the interests of space), before we discuss our algorithms

Our constraints express an invariant condition between two subpaths in the database or KBS schema (universe of discourse) Each subpath is described in abbreviated form, and consists of objects and relations from the schema, as we saw earlier So, for example, the constraint

```
MANAGER PROJECT ==
        MANAGER EMPLOYEE PROJECT
```

expresses the fact that for each Manager instance (the *anchor*), there is to be equality of the *target* sets of Projects arising from the two subpaths of this CE That is, a Manager is responsible for a Project if and only if one or more of his Employees are assigned to the Project The CE is implicitly iterated over all Manager instances in the database, although it would be enforced incrementally so that no overhead would occur unless a relevant relationship were updated

Some representational capabilities of our constraint expressions are not easily captured by predicate calculus [Mor84a, Mor84b] These include (1) cardinality-based quantifiers which form a continuum from $\exists$ (there exists at least one instance with the specified property) to $\forall$ (for all instances), and (2) cardinality-based set intersection with specification of lower and upper bounds on the number of common elements In cases where CEs have a predicate calculus equivalent, we have found CEs to be more compact, and more intuitive, than the equivalent, but longer, predicate calculus expressions

Constraints can interact with one another, this will occur when they involve common relationships As a result, we can express a complex action or interdependency in terms of modular and simpler constraints At the instance level, the interaction of constraints is referred to as *constraint propagation* At the symbolic level, the interaction is expressed by *constraint composition* [Mor84b], this was illustrated earlier by an example regarding projects, reports, and topics

6.1. Capturing Hard to Express Constraints

We may express the existence of an interdependence or a partial constraint even when there is insufficient knowledge to characterize it exactly An interdependence may be too complex, and may not warrant a complete description, or it

may involve additional information which is outside the application For example, the partial constraint between one's address and the telephone area code may be described as·

PERSON ADDRESS $\longleftrightarrow$ PERSON TELEPHONE

Such partial constraints are conveniently expressed with an *inferability relationship*, denoted $\longleftrightarrow$, as the comparator between the two subpaths of the CE When some quantization of the degree of inference is available, we may express it as

PERSON ADDRESS 03 $\longleftrightarrow$ 1
PERSON TELEPHONE AREACODE

In a fairly obvious way, this CE tells us that a person's address can be used to infer exactly (inference = 100%) the person's area code, whereas the area code gives us only a little (3%) information about the address (for example, it might imply the set of possible ZIP codes and the state)

The above examples have the form $E_1 \ \alpha\longleftrightarrow\beta \ E_2$ This inferability expression serves as an abbreviation for $\text{INFER}(E_1 \rightarrow E_2) = \beta$ and $\text{INFER}(E_2 \rightarrow E_1) = \alpha$ Similarly, a one directional inference can be abbreviated as $E_1 \longrightarrow \beta \ E_2$, which represents $\text{INFER}(E_1 \rightarrow E_2) = \beta$ When we omit the values α and/or β for the degree of inference (perhaps because we do not know the values in some situation), then $\longleftrightarrow$ and $\longrightarrow$ signify the existence of the respective inference in excess of the threshold ϵ

7. Algorithmic Considerations

This section describes the algorithms which are presented in the Appendix Our intention is to show the framework for analysis and the primary methods Thus for simplicity, these algorithms treat the constraints at system low -- that is, they are available without restriction We do, however, discuss how the classification levels of constraints would affect the analysis

Computation of the sphere of influence begins with the choice of *core* objects, as shown in algorithms SOI and Incremental-SOI The 'available constraints' are those which could contribute to inferences -- these include constraints at system low and other constraints which are themselves in this *core* Constraints will be in the core if access to them is limited but they are to be considered for this SOI -- this is a primary point at which the classification levels of constraints would be taken into account Of these constraints, the CEs that reference any of the objects/relationships in the *core* are selected initially (CoreCEs) In turn, these initially selected CEs are composed with other available constraints which have matching components This composition process is repeated with the available constraints until we obtain a maximal set -- all constraints which emanate from the core The set of objects that are included in any of these constraints constitutes the *extent* of this sphere of influence, i e SOI(*core*)

For efficiency, it is important to minimize the number of SOIs which need to be computed By utilizing the *Safety Theorem for a Classification Level*, we see that if the *core* is chosen to be the set U_l of all objects at and below classification level l, then from one SOI computation we can determine the *safety* of all the objects having level l

Furthermore, we can reuse the constraint compositions formed for one SOI in a subsequent SOI computation if we process lower classification levels first Algorithm Consistency Test operates in this manner, and also recognizes that the only new constraints that need to be considered in the SOI computation are those which were not utilized at a lower level, are available at this level, and involve objects in this SOI's *core*

The Theorem on Security of a System assures us that after all the levels in the classification lattice have been analyzed for safety, and if no leakage of information based upon inference is found, then the whole system is *secure* from inference-based compromises to security However, if such leaks are found, then we need to isolate the inference channels that describe the sources of these leaks Let $\{Y_i\}$ denote the subset of objects from the *extent*, SOI(core), which represent potential leaks, that is for which $\text{level}(core) \not\geq \text{level}(Y_i)$ The subset of the core that contributes to inferences about each Y_i is denoted $InfCh_{\text{core}}(Y_i)$

This inference channel can be determined efficiently by using the previously computed composite constraints Algorithm

Isolate-Inferences first determines the set of just those composite constraints which involve Y_i (call this set CC_i) Letting *objectsin(CC)* denote the data objects referenced in the set of constraints CC, we have $InfCh(Y_i)$ = *objectsin*(CC_i) (for simplicity we assume here that this yields the minimal inference channel) Finally, $InfCh_{core}(Y_i)$ = $(InfCh(Y_i) \cap core)$ -- the intersection with the core removes other objects which will be accounted for by their own inference channels

8. Conclusions and Directions for Further Research

This paper has proposed a methodology for analyzing the heretofore amorphous problems associated with inference control in multilevel database and knowledge-base systems We have developed criteria for evaluating the security and safety of a system with respect to inference problems, shown how knowledge of the application can be expressed in our constraint language, and outlined algorithms for enforcing such criteria

We offer some suggestions for future work in this area of inference control

- In order to correct a potential inference channel, selected objects can be reclassified to a higher level There may be alternative incremental adjustments which are possible Research is needed to develop a decision procedure that generates the alternatives and selects the best subset of objects to reclassify Also, it will be necessary to reevaluate potential inferences to these objects -- since they have been reclassified into a new sphere of influence Algorithms will be needed that *incrementally* evaluate the consequences of the reclassifications

- We have discussed how the techniques developed here could be applied to rule-based expert systems The clearance of the user will affect not only access to data, but also availability of the expert rules Thus the 'expert' solutions are likely to differ among users due to their different clearance levels Further work is needed to develop a procedure for consistent assignment of classification levels to the data and to the rules This process will be more complex than for databases -- due to the many consequences which must be taken into account In addition, techniques must be developed to ensure that all 'solutions' which the system could produce are valid solutions

- Some inferences become possible only when a sequence of database updates over time is analyzed This will be an interesting extension to the types of inference we can evaluate It is particularly relevant due to increasing interest in the concept of time in databases, and the increasing number of process-oriented systems which are using databases

- Classifications which are based not just on the type of data but also on its value (instance-based classification) will complicate the analysis The entry of new data into the system, and the possible reclassification of existing data also must be evaluated dynamically when needed

The knowledge about the application semantics which we capture for the analysis of inference can be used to provide other new and enhanced database capabilities [Mor86b] These include a much broader range of integrity and consistency enforcement, a higher level of interaction with a 'smart' database (e g intelligent responses to queries), and improved support for applications where a variety of constraints affect the data

9. Acknowledgments

This work has been sponsored by the U S Army CECOM at Fort Monmouth, N J , under Contract No DAAG29-81-D-0100, and by the U S Air Force, RADC, Rome, N Y , under Contract No F30602-86-C-0088 I wish to thank Dorothy Denning for her helpful comments and contributions during the course of this work

References

[BeLu87] Berson, Thomas A , and Teresa F Lunt, *Multilevel Security for Knowledge-Based Systems*, IEEE Symposium on Security and Privacy, Oakland, CA, April 1987

[Dat86a] Date, C J , *An Introduction to Database Systems*, vol 1, 4th ed , Addison-Wesley, 1986

[deK86] de Kleer, Johan, *An Assumption-based TMS*, Artificial Intelligence Jour , vol 28, no 2, March 1986, pp 127-162

[Den82] Denning, Dorothy E , *Cryptography and Data Security*, Addison-Wesley, Reading, Mass , 1982

[DeMo86] Denning, Dorothy E , and Matthew Morgenstern, *Military Database Technology Study AI Techniques for Security and Reliability*, SRI International, Technical Report, Project 1644, August 1986

[DeLS87] Denning, D E , T F Lunt, R R Schell, M Heckman, W Shockley, *A Multilevel Relational Data Model*, IEEE Computer Society, Proc of the 1987 Symp on Security and Privacy, April 1987

[DeSc83] Denning, D E and J Schlorer, *Inference Controls for Statistical Database Security*, IEEE Computer, v 16 no 7, July 1983, pp 69-82

[GoMe84] Goguen, Joseph A , and Jose Meseguer, *Unwinding and Inference Control*, Proceedings of the 1984 Symposium on Security and Privacy, Oakland, CA, May 1984, pp 75-86

[Hay85] Hayes-Roth, B , *A Blackboard Architecture for Control*, Artificial Intelligence Jour, vol 26, 1985, pp 251-321

[Mor86a] Morgenstern, Matthew, *The Role of Constraints in Databases, Expert Systems, and Knowledge Representation*, in Expert Database Systems, editor Larry Kerschberg, Benjamin/Cummings Pub , 1986, pp 351-368

[Mor86b] Morgenstern, Matthew, *Intelligent Database Systems - A Plan for Research*, SRI International, Working Paper, 1986

[Mor84a] Morgenstern, Matthew, *Constraint Equations A Concise Compilable Representation for Quantified Constraints in Semantic Networks*, National Conf on Artificial Intelligence (AAAI-84), Austin, Texas, August 1984, pp 255-9

[Mor84b] Morgenstern, Matthew, *Constraint Equations Declarative Expression of Constraints with Automatic Enforcement*, Tenth Int'l Conf on Very Large Data Bases (VLDB-84), Singapore, August 1984, pp 291-300

[Row86] Rowe, Neil C , *Security problems with inferences from the results of rule-based expert systems*, Dept of Computer Sci , Naval Postgraduate School, Monterey, CA, 1986, unpublished draft

[ScBr82] Schmolze, James G , and Ronald J Brachman, *Proceedings of the 1981 KL-ONE Workshop*, Fairchild Technical Report 618 (also BBN report 4842), May 1982, 287pp

[SiJR83] Sicherman, George L , Wiebren de Jonge, and Reind P van de Riet, *Answering Queries Without Revealing Secrets*, ACM Trans on Database Sys , v 8, no 1, March 1983, pp 41-59

[SuOz87] Su, T , and G Ozsoyoglu, *Data Dependencies and Inference Control in Multilevel Relational Database Systems*, IEEE Symposium on Security and Privacy, Oakland, CA, April 1987

[Tru84] Trueblood, Robert P , *Security Issues in Knowledge Systems*, Proceedings of 1st Int'l Workshop on Expert Database Systems, ed L Kerschberg, October 1984, vol 2, pp 834-840

Appendix of Algorithms

Algorithm: SOI (CoreObjects)

```
,Algorithm to compute SOI at schema level given Core set
CoreCEs  = {CEs which contain any objects from set of CoreObjects}
CEset    = {set of all available CEs} - CoreCEs
RETURN Incremental-SOI(CEset, CoreCEs)
******
```

Algorithm: Incremental-SOI (CEset, CoreCEs)

```
,Used by SOI and Consistency-Test algorithms
,Algorithm to compute SOI at schema level given
,    CEset   = subset of available CEs to compose with CoreCEs
,    CoreCEs = kernel set of CEs (must be basis for further compositions)
,Treats CEset as inference rules to compute all composite CEs which include CoreCEs
Todo = { <CE, Unused> |  CE in CoreCEs, Unused = CEset }    ,initialize
         ,There is one tuple <CE, Unused> per CE in CoreCEs
         ,TUPLEi Unused is set of CEs to be tried, initially is CEset
Composed = empty
FOR Next in Todo UNTIL empty DO
    FOR OtherCE in Next Unused DO
        IF New = Compose(Next CE  OtherCE) succeeds
           THEN add <New, (Next Unused - OtherCE)> to Todo
    END-FOR
    Composed = Composed + Next CE
    Todo = Todo - Next
END-FOR
SOI = Union[{objectsin(CEi)}, forall CEi in Composed]   ,is SOI(Core)
RETURN <SOI, Composed>
******
```

Algorithm: Consistency-Test

```
,Tests for Inference Channels at schema level   Returns classification Nodes
,which fail consistency test   Empty result ⇒ no inference channels
,LatticeSuccessor() returns an unmarked Node of the classification lattice such that
,all its predecessors (if any) have been marked as analyzed by this algorithm   Initially
,all nodes are unmarked  Children(Node) returns all immediate children of Node in lattice
FOR Node in LatticeSuccessor() UNTIL null DO
    Node Core = { object | classification(object) ≤ Node level }
    Composed = Union[{CN Composed}, forall CN in Children(Node)]
    PreviousCores = Union[{CN Core}, forall CN in Children(Node)]
    RelevantCEs  = {CEs containing objects which occur in Node Core}
    PreviousCEs  = {CEs containing objects which occur in PreviousCores}
    NewCEs = RelevantCEs - PreviousCEs
    Result = Incremental-SOI(NewCEs, Composed)
    IF (Result SOI - Node Core) = empty
       THEN "no inference channel"
       ELSE Node SOI = Result SOI
            Node Composed = Result Composed
            Node Inf = (Node SOI - Node Core)    ,potential inferences
            add Node to Inconsistent-Nodes
    Mark Node as analyzed
END-FOR
RETURN Inconsistent-Nodes
******
```

Algorithm: Isolate-Inferences (Node)

```
,Isolates Inference Channel(s) for Node in classification lattice which failed
,the Consistency-Test algorithm   Returns list of <Y1, MaxCore1>, where Y1 is
,object(s) at higher classification than Node Core   MaxCore1 is maximal
,subset of Node Core which contributes to inferences about object Y1
,Node Inf was set to {Y1} in Consistency-Test algorithm
InfChannels = empty
FOR Y in Node Inf DO
    MaxCore = empty
    FOR CEchain in Node Composed DO
        IF CEchain includes object Y
           THEN MaxCore = MaxCore U objectsin(CEchain)
    END-FOR
    MaxCore = MaxCore ∩ Node Core     ,limit to objectsin Node Core
    IF MaxCore already appears in an entry of InfChannels
       THEN replace that entry <Yk MaxCore>
            with <(Yk + Y)  MaxCore>      ,combine if identical MaxCore
       ELSE add <Y MaxCore> to InfChannels
END-FOR
RETURN InfChannels
******
```