J. Software Engineering & Applications, **2010, 3: 255-267**
doi:10.4236/jsea.2010.33031 Published Online March 2010 (http://www.SciRP.org/journal/jsea)

Information Content Inclusion Relation and its Use in Database Queries

Junkang Feng[1], **Douglas Salt**[2]

[1]Business College of Beijing Union University, Beijing, China; [1,2]Database Research Group School of Computing, University of the West of Scotland, Paisley, UK.
Email: {junkang.feng, douglas.salt}@uws.ac.uk

Received October 31st, 2009; revised November 19th, 2009; accepted November 25th, 2009.

ABSTRACT

A database stores data in order to provide the user with information. However, how a database may achieve this is not always clear. The main reason for this seems that we, who are in the database community, have not fully understood and therefore clearly defined the notion of "the information that data in a database carry", in other words, "the information content of data". As a result, databases' capability is limited in terms of answering queries, especially, when users explore information beyond the scope of data stored in a database, the database normally cannot provide it. The underlying reason of the problem is that queries are answered based on a direct match between a query and data (up to aggregations of the data). We observe that this is because the information that data carry is seen as exactly the data per se. To tackle this problem, we propose the notion of information content inclusion relation, and show that it formulates the intuitive notion of the "information content of data" and then show how this notion may be used for the derivation of information from data in a database.

Keywords: *Information Content, Databases, Knowledge Discovery from Databases, Semantic Theory of Information*

1. Introduction

When we query a database, it is said that we are retrieving information from it. This is taken for granted. But, how this happens is not always fully understood. As a result, when a user queries a database [1], the query can only be answered through a "direct match" between the selection criteria within a query and data (up to aggregations of the data) [2]. In a case of querying a database beyond this, the system is unlikely to answer the query. A conventional query is, in essence, concerned with only the *propositional content* of data [3]. We believe that data carries information [4–6]. A piece of data may carry information about another, and moreover it may carry information about a real world situation [7,8]. Therefore, if we can define and formulate the notion of "the information content of data", not only may we obtain insight about the essence of conventional queries, but also we may derive more information beyond "direct match".

However, it would appear that the notion of "information content of data" is elusive. It has been taken as the instance of a database and the information capacity of a data schema as the collection of instances of the schema [9–11]. Another view on the topic of the relationship between information and data is that if it is truthful, meaningful data is semantic information [12]. We argue that such views miss two fundamental points. One is a convincing conception of "information content of data". To equate data with information overlooks the fact that data in a database is merely raw material for bearing and conveying information. Information must be veridical [7], that is, it must relate to a contingent truth [12], while for data there is no such requirement. The other is a framework for approaching the information content of data whereby to reveal information.

That is to say, we define the following research question that we tackle in this paper: how the "information content" of data in a database may be defined with mathematical rigor, and how this notion after have been defined may help retrieve information through reasoning that cannot otherwise be possible through conventional queries.

To answer this research question, we purpose to look at the relationships between the information content of data, database structure and domain knowledge, which may be captured as business rules. These include how tacit domain knowledge may be explicitly expressed and used.

In this paper, we present a novel framework for approaching the information content of data in a database, which is centered on the notion of *information content inclusion relation*. It helps us understand how a database does its job, *i.e.*, providing information, and helps a database system improve its capability of providing information through inference. The latter is achieved by introducing a variety of information sources such as domain knowledge. With the help of external information sources, queries that deal with a wider range of information than the propositional content of data within a database may be answered. The underlying thought of the framework is based on a concept of *information content* of a signal. Dretske [4] firstly introduced the concept. Then Xu *et al.* [13] extended Dretske's idea and gave a more detailed definition of the information content of a state of affairs. Our thoughts are based on the latter definition.

The next section gives a number of foundational concepts. Then the framework and a prototype of implementation are presented in the third section. The last section concludes the paper.

2. Foundational Concepts

A number of concepts are defined in this section and they are foundational for defining the notion of "information content inclusion relation".

2.1 Information Content

Fred Dretske [4] gave the definition of information content as follow:

"A state of affairs contains information about X to just that extent to which a suitably placed observer could learn something about X by consulting it."

Then he formalized the above as

"Information Content: A signal r carries the information that s is F = The conditional probability of *s*'s being F, given r (and k), is 1(but, given k alone, less than 1)."

Note that k stands for prior knowledge about information source s.

Here is an example: That John is awarded a grade "A" for his Programming course contains the information that he has scored 70% or above for that course.

Dretske's above definition needs to be extended, however, as it does not capture explicitly the crucial role that individual objects, situations and events play in carrying information, and it is these individual things that actually carry information. In the above example, it is the individual event namely "John is awarded a grade "A" for his Programming course" that carries the information "John has scored 70% or above for that course". Dretske's definition is based on probability, and a single event does not have a probability [7], and a type of events has. To extend Dretske's definition and therefore make such a concept accurate, let us define a few very basic notions first.

2.2 Random Variables

Definition 1

Let *s* be a selection process under a set C of conditions, O a possible outcome of *s*, O can therefore be of one of a number values, *i.e.*, the possible outcomes. O is said to be a random variable.

That is to say, a random variable is a variable that can hold one of a number of possible values at a time and which one of the values to be hold is determined randomly. For example, in a database, table Students contains attributes such as ID, Name and DOB. A random variable could be any one attribute or a collection of attributes of the Students table in the sense that for a randomly chosen tuple, the value of its ID cannot be pre-determined and can only be one of all the possible values for ID.

2.3 Random Events

Definition 2

Let *s* be a selection process under a set C of conditions, O a possible outcome of *s*, and such an outcome is called a *state*, and E the power set of all the possible values for O, *i.e.*, all the *states*, X is a random event if $E \ni X$ and there is a probability of X, *i.e.*, P(X).

For example, to select a student record from table Students randomly in database and the record being concerned with a particular student is a random event.

A random event has to occur within a "probability space", which we define below:

Definition 3

Let *s* be a selection process under a set C of conditions, O a possible outcome of *s*, E the power set of all the possible values for O, *i.e.*, all the *states*, and $E \ni Xi$ for $i = 1,\ldots, n$, P_s is the probability space of the random events X_i for $i = 1,\ldots, n$, if $P_s = \{P(X_1), P(X_2),\ldots, P(X_n)\}$ and $\sum P(X_i) = 1$.

Note that this notion is also useful for explaining what it means by "probability distribution" and the change of "probability distribution", which is necessary and sufficient for information to flow.

2.4 Particulars of Random Events

Furthermore, as mentioned earlier, Xu *et al.* pointed out that even though Dretske's definition was plausible, the role that individual events play in our looking at the information content of a state of affairs was overlooked. To amend this, Xu *et al.* [13] put forward a definition of particulars of a random event as follow:

Definition 4

Let *s* be a selection process under a set C of conditions, X a random event concerning *s*, X_i an instance of *s*, X_i is a particular of X if X_i is in a state Ω, written $\Omega = \text{state}(X_i)$, and $X \ni \Omega$.

As in the example above, to select a student record from table Students is a random variable, the record happens to be John's is a random event, and one occurrence of John's record is a particular of the random event.

3. Information Content Inclusion Relations

Having defined the foundational concepts, we can now define the notion of "information content inclusion relation". As this notion formulates the intuitive notion of "the information content of a signal/data", we formulate the latter first.

3.1 Information Content of a State of Affairs

Data in a database may be seen as a type of signals, and as said earlier data may be seen as random events and random variables. A random event is also informally called state of affairs by Dretske in [4].

Definition 5

Let *s* be some selection process or mechanism the result of which is reduction of possibilities, and therefore be an information source, and *k* prior knowledge about *s*[1]

Let *r* be a random event, and r_i a particular of *r* at time t_i and location l_i;

Let *s*'s being *F* be a random event concerning *s*, and s_j some particular of *s*'s being *F* at time t_j and location l_j;

r_i carries the information that there must be some s_j existing at time t_j and location l_j, that is, the state of affairs of *s* is *F* at t_j and l_j, if and only if the conditional probability of *s*'s being *F* given *r* is 1 (and less than 1 given *k* alone).

Definition 6

That a particular r_i carries the information that a particular s_j exists can also be termed that the *information content* of r_i includes s_j, or in other words, s_j is in the *information content* of r_i.

3.2 Information Content Inclusion Relations (IIR)

The term, *information content inclusion relation*, was firstly put forward by Feng in 1998 [14]. We now give an amended definition below:

Definition 7

Let *X* and *Y* be a random event respectively, there exists an *information content inclusion relation*, IIR for short, from *X* to *Y*, if every possible particular of *Y* is in the information content of at least one particular of *X*.

3.3 Types and Sources of IIR

We observe that there are four types of IIR in terms of where a state of affairs takes place, and we list them and some of their sources in the table below:

[1] Note that *k* here goes only as far as what counts as a possibility involved in *s*, and it is not concerned with whether an observer is able to learn and actually learns something about *s* by consulting something else such as *r*.

Information Inclusion Relation - Information content of X includes Y, denoted IIR(X, Y)	**Sources**
X, Y are random events both in the database world	Syntactic relations between data constructs and data values
X is a database random event. Y is random event in the real world	"Semantic values" [15] of data
X is a real world random event. Y is a database random event	Rules and processes of database design and database operations
X,Y are random events both in the real world	Relations between real world objects and events, business rules

The first two types of IIRs above constitute the information content of data in a database. Furthermore, we observe that for a database to provide information and nothing else, all the four types and all IIRs must be consistent with one another. To elaborate this observation would require much more work, and thus we leave it till another paper later.

3.4 Rules for Inferences on and of IIR

IIR can be formally reasoned about. Modifying those presented in Xu *et al* [13], we present the following inference rules for reasoning about IIR.

"Sum": If $Y = X_1 \cup X_2 \ldots \cup X_n$, then IIR($X_i$, Y) for i = 1,…,n.

This rule says that if it is the disjunction of a number of random events, then a random event *X* is in the information content of any of the latter. A trivial case is where *X* and *Y* above are not distinct. The rest of the rules can be interpreted similarly.

"Product": If $X = X_1 \cap X_2 \ldots \cap X_n$, $Y = X_i$ for i = 1, …, n, then IIR(X, Y).

Transitivity: If IIR(X, Y), IIR(Y, Z), then IIR(X, Z).

Union: If IIR(X, Y), IIR(X, Z), then IIR(X, $Y \cap Z$).

Augmentation: If $W = W_1 \cap W_2 \ldots \cap W_n$, Z is the product of a subset of $\{W_1, W_2, \ldots, W_n\}$, IIR($X$, Y), then IIR($W \cap X$, $Z \cap Y$).

Decomposition: If IIR(X, $Y \cap Z$) then IIR(X, Y), IIR(X, Z).

The above set of rules is proven sound and complete. The proofs can be found in [13].

4. Preparing the Information Base for Database Queries

Given a set of IIRs, all IIRs that are logically implied by them and therefore are derivable, which we call the "closure" of the former, constitute the information base for answering queries that are posed to a database.

4.1 The Closure of a Set of IIRs

Definition 7

Let F be a set of IIRs. F closure (denoted F^+) is the set of IIRs logically implied by F. F $\subseteq$ F^+. If F= F^+, F is called a complete set of IIRs in the sense that no more IIRs that are logically implied by F can be derived from it by using the IIR inference rules.

The F above are called the original IIR, which are identified by applying the definition of IIR directly to a variety of sources such as the real world, database systems and domain knowledge, and which are not those that are derivable by using the inference rules on known IIR. For example, the *referential integrity* of a relational database is a kind of constraints in a relational database, from which, original IIR can be derived.

To compute F^+ given F, we can compute instead X^+ for all X, where X is a random event, which is normally easier than computing F^+ directly. Once X closure is known, to know if IIR(X, Y) holds given F (*i.e.*, whether it is implied by F) is a matter of verifying if Y is in the X closure or not. If so, IIR(X, Y) holds. Otherwise, as far as the given F goes, IIR(X, Y) does not exist.

4.2 IIR Closure of a Random Event

All random events that are derivable by using the IIR inference rules on a given set of original IIR and therefore are in the information content of the given random event constitute the IIR closure of the random event. For example, "Student ID = B001" is a random event, and "Student Address = 1 High Street" is in its information content. Likewise, "Student Postcode = PA1 2BE" is in that of "Student Address = 1 High Street". Through Transitivity (see Subsection 3.4), "Student Postcode = PA1 2BE" is also in the information content of "Student ID = B001". All such random events as "Student Postcode = PA1 2BE" and "Student Address = 1 High Street" would constitute the IIR closure of "Student ID = B001". Let X denote "Student ID = B001", then we use X^+ to done the IIR closure of X.

Figure 1 shows how the information base for answering queries is identified and formulated by means the foundational concepts, IIR and inference rules for IIR.

5. A System for Querying a Database with IIR

With the idea of IIR and other associated notions just presented, we have created a system for reasoning about the *information content* of data whereby to help derive information in a database by drawing on Wang and Feng [16] and Eessaar [17]. Intuitively, the system works like this.

Let us re-iterate that to select a student from table Students is seen as a random event, and the term "particulars of a random event" is used to describe a single occurrence of a random event. For example, student John's record happens to be selected from table Students, and this particular occurrence of John's record being selected is a "particular" of the random event that the record happens to be John's. A random variable may be seen as an aggregation of random events. In a table, an attribute can be seen as a random variable because it normally contains many random events in it. For example, Student Name is a random variable, which contains Student Name being John and Student Name being Herman, among others. The IIR closure of Student ID being B001, for example, contains Student Name being "John", Student Major being "history" and Class Name being "BD445". If a user queries about the class name about John, the query can be answered by searching in this IIR closure of Student ID being B001. That is, once IIR closures are known, queries can be checked against these closures. This way some information that cannot be found by conventional queries may be discovered.

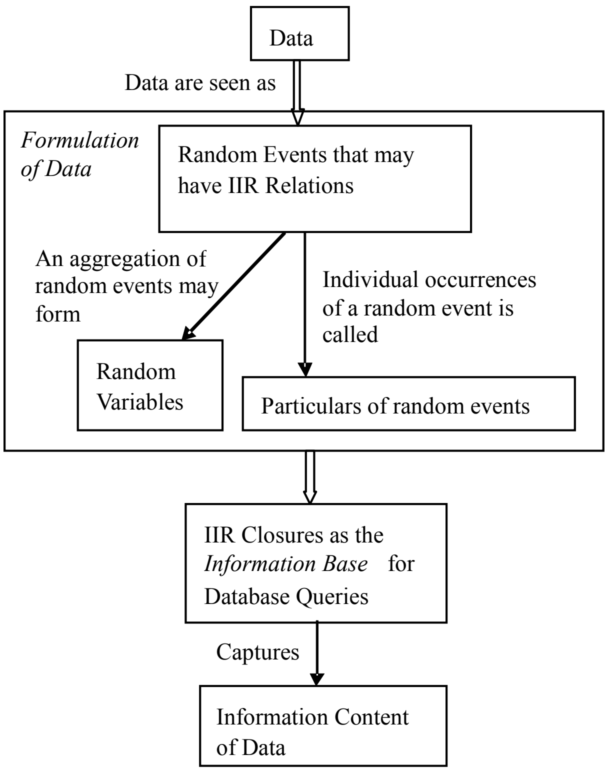
Data
Data are seen as
Formulation of Data
Random Events that may have IIR Relations
An aggregation of random events may form
Individual occurrences of a random event is called
Random Variables
Particulars of random events
IIR Closures as the Information Base for Database Queries
Captures
Information Content of Data

Figure 1. The information base for database queries

Figure 2 illustrates the structure of our experimental system. It consists of three main parts. The upper part is where users pose queries to the system. The middle part is the Datalog implementation of information content reasoning. The lower part shows a variety of sources of original IIR, namely domain knowledge and the syntactic and semantic properties of the database that are inherent to it.

The form of the queries is the conventional SQL. Most programming efforts were made on computing the IIR closures. The core algorithm is based on the IIR rules. Original IIRs were then added into the unit. This is one of the most difficult tasks in the programming required for the construction of the system as when more original IIR were discovered more computation capability has to be added into the program such that the closures can continually increase accordingly. The output of the unit is simple however, which are IIR closures. User queries, then, are checked against these closures. Thus, more information can be discovered through queries.

The process of discovering original IIRs could be hard. There is a variety of sources out there that could potentially contain huge amount of original IIRs [13]. The two main sources though are domain knowledge and the properties of the database *per se*. The latter can be further divided into those of semantic and syntactic levels respectively. Hereinto, the syntactic level includes plenty of constraints such as data dependencies, integrity rules and the cardinality ratio between tables.

We now wish to demonstrate how the experimental system was created using IIR. The previous version of the system was coded in Oracle's PL/SQL [18]. It is now coded in the deductive database language, Datalog (Datalog Learning System, Universidad Complutense de Madrid, System v.1.6.2). First, we show how our IIR inference rules may be implemented by using Datalog in order to make use of the deduction power of it.

5.1 Datalog Implementation of IIR Inference Rules

It will now be shown that the above inference rules may be coded as *rules* (with example facts) in the Datalog language.

5.1.1 Conventions

1) Random Events

Any lower case literal is considered a constant in Datalog. We shall conflate these to random events or products of random events. In general, a Datalog constant, x ≡ *X*, where x is a Datalog constant, and *X* is a random event, which in the case of a database could be an attribute being a particular value extant in a database.

2) Information Content Inclusion Relation (IIR)

IIRs may be expressed as a relationship between either random events (Datalog constants), or Datalog variables, where the Datalog variable, when evaluated will contain a Datalog constant, and therefore, by extension a representation of a random event. We shall adopt the convention of using the predicate iir to indicate or derive an IIR between Datalog constants. Hence these will have the form:

iir(a,b).

iir(X,Y).

iir(a,X).

iir(X,a).

where a and b are Datalog constants, and X and Y are Datalog variables.

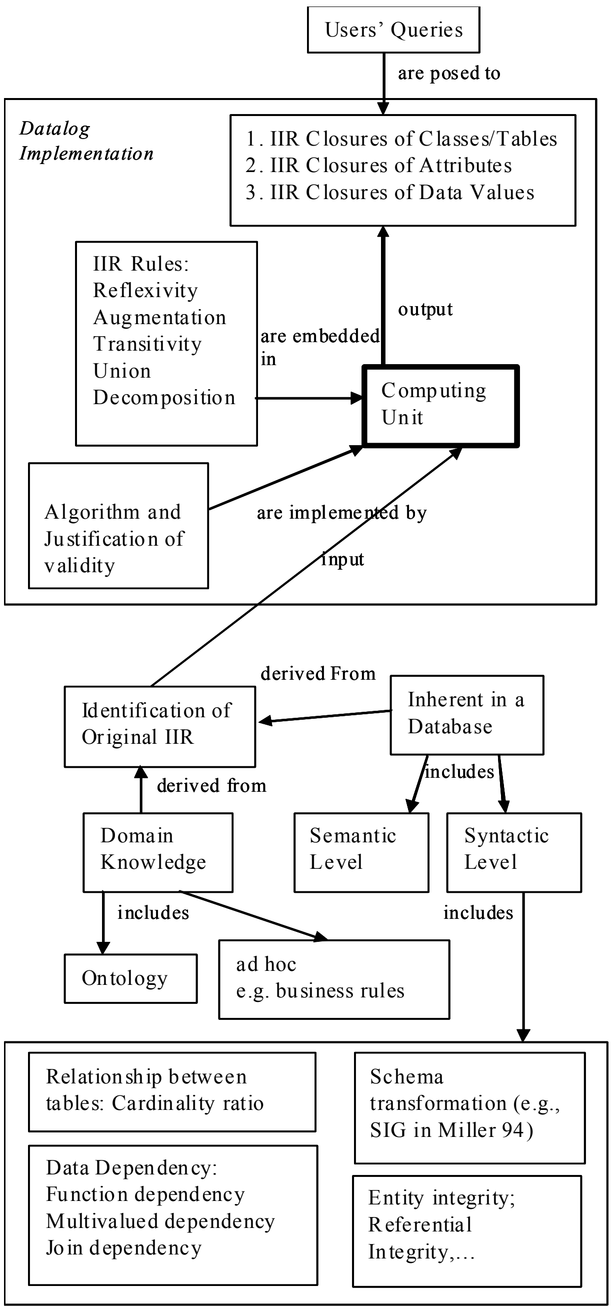
Users' Queries
are posed to
Datalog Implementation
1. IIR Closures of Classes/Tables
2. IIR Closures of Attributes
3. IIR Closures of Data Values
IIR Rules: Reflexivity Augmentation Transitivity Union Decomposition
output
are embedded in
Computing Unit
Algorithm and Justification of validity
are implemented by
input
derived From
Identification of Original IIR
Inherent in a Database
includes
derived from
Domain Knowledge
Semantic Level
Syntactic Level
includes
includes
Ontology
ad hoc e.g. business rules
Relationship between tables: Cardinality ratio
Schema transformation (e.g., SIG in Miller 94)
Data Dependency: Function dependency Multivalued dependency Join dependency
Entity integrity; Referential Integrity,...

Figure 2. A system for reasoning about information content of data in a database

3) Products of Random Events

If we have IIR($A \cap C \cap D$,*B*), then $A \cap C \cap D$ represents the product of random events *A*, *C* and *D* containing the information content of random event B. This product may be represented in Datalog in the following manner:

product(p*ACD*,a,c,d).

where we have adopted the following conventions, a, c, d and p*ACD* are Datalog constants. We assume *a* ≡ *A*, *c* ≡ *C*, and $d \equiv D$, and the single Datalog fact product($pACD$, a, c, d)is equivalent to

$$pACD=A \cup C \cup D.$$

4) Sums of Random Events

If we have $L=P \cup Q \cup A$, then random event L represents the sum of random events P, Q and A. That is, we use L to refer to a random event where at least one of P, Q and A is extant. We choose to represent the sum in Datalog in the following manner:

```
sum(l,p).
sum(l,q).
sum(l,a).
```

where we have adopted the following conventions l, p, q and a are Datalog constants, and we assume that l $\equiv L$, p $\equiv P$, q $\equiv Q$ and a $\equiv A$. The three facts above may be considered as $L=P \cup Q \cup A$. The reason we have adopted this convention is that it allows the construction of sums containing two or more than two random events, and allows their eventual evaluation as binary relations, between any two random events. As Datalog is just the evaluation of a series of Horn clauses, then such structures are evaluated in conjunction for validity. This is exactly the effect we desire as random events that may happen, may be considered as unions of the state space for those random events, and thus may be treated as conjunctions happening across all random event closures. This also makes the coding in Datalog considerably simpler and more elegant.

5.1.2 IIR Inference Rules

1) "*Sum*" *Rule*

In Subsection 3.4, the "Sum" rule was given: If $Y = X_1 \cup X_2 \ldots \cup X_n$, then IIR($X_i$, Y) for i = 1,…, n. With the convention above, to code the "Sum" in Datalog, we use the rule:

```
iir(X, Y):-sum(Y, X).
```

This indicates that the information content of the sum Y is contained in any of the sum's members, denoted X. So given

```
sum(l, p).
sum(l, q).
sum(l, a).
```

if we run the query 'iir(X,l)?', we get the response:

```
{
  iir(a,l),
  iir(p,l),
  iir(q,l)
}
Info: 3 tuples computed.
```

This indicates iir(a,l), iir(p,l) and iir(q,l), respectively.

2) "*Product*" *Rule*

In Subsection 3.4, the "Product" rule was given: If $X = X_1 \cap X_2 \ldots \cap X_n$, $Y = X_i$ for i = 1, …, n, then IIR(X,Y). With the convention given earlier for products, in

```
product(pEG,e,g).
```

e, and g are Datalog constants, pEG represents the product of random events E and G, pEG = E∩G

The "Product" rule may now be represented by the Datalog rule as follows:

```
iir(P,X):-product(P,X,A).
iir(P,X):-product(P,A,X).
```

This depicts, that any product P, has an IIR with any member of that product. The variable A represents a place holder, indicating to Datalog that for any matching predicates, then this variable is not to be returned in the query. In answer to the query, iir(pEG,X), asking what is in the information content of product of random event, E and G, Datalog returns the following:

```
{
  iir(pEG,e),
  iir(pEG,g)
}
Info: 2 tuples computed.
```

These indicate IIR($E \cap G$,E) and IIR($E \cap G$,G) respectively.

In general, to represent the product rule for a product consisting of n random events, then an additional n rules are required to show the product rule for a set of random events.

3) *Transitivity*

Assuming that we have IIR(C,A), IIR(A,B) which may be represented by the following Datalog facts:

```
iir(c,a).
iir(a,b).
```

where we have adopted the following conventions, c, a, and b are Datalog constants, X, Y and Z are Datalog variables. We assume c $\equiv C$, a $\equiv A$, and b $\equiv B$. The rule required in Datalog to represent transitivity may now be coded in Datalog as the following:

```
iir(X,Y):-iir(X,Z),iir(Z,Y).
```

and iir(X,Y)represents an IIR between two random events X and Y. This rule states that if any random event contains in its information content a second random event, which in turns contains in its own information content a third random event, then the first has the third in its information content. In answer to the query, iir(c,X), asking what is in the information content of random event, C, Datalog returns the following:

```
{
  iir(c,a),
  iir(c,b)
}
Info: 2 tuples computed.
```

These indicate IIR(C,A) and IIR(C,B) respectively, and the latter is arrived at due to Transitivity.

4) *Union*

In Subsection 3.4, the Union was given: If IIR(*X*, *Y*), IIR(*X*, *Z*), then IIR(*X*, *Y*∩*Z*).

We now need to represent the relationship between these random events in Datalog, which may be done with the following facts:

```
iir(a,b).
iir(a,c).
product(pCB,c,b).
```

c, b, and a are Datalog constants. We assume c ≡ *C*, b ≡ *B*, and a ≡ *A*, and pCB represent the products of the random events *C* and *B*, such that *pCB=C∩B*.

We now need to create a Datalog rule which will link the product of random events *C* and *B* to random event *A*.

```
iir(X,P):-product(P,Y,Z),iir(X,Y),product(P,Y,Z),iir(X
,Z).
```

This will return all products of random events that have contain a random event with an existing IIR with the first argument.

In answer to the query, iir(a,X), asking which random events and products of random events are in the information content of random event *A*, Datalog returns the following:

```
{
  iir(a,b),
  iir(a,c),
  iir(a,pCB)
}
Info: 3 tuples computed.
```

These indicate IIR(*A*,*B*), IIR(*A*,*C*) and IIR(*A*,*B*∩*C*) respectively, and IIR(*A*,*B*∩*C*) is arrived at due to the Union rule.

5) *Augmentation*

Augmentation is a little more involved. Let us assume *W*=*X*∩*Y*∩*Z*, M = X∩Y, and IIR(*A*,*B*). We code these in Datalog as the following facts:

```
iir(a,b).
product(m,x,y).
product(w,x,y,z).
```

where w, x, y, z, a and b are Datalog constants.

We assume w ≡ *W*, x ≡ *X*, y ≡ *Y*, z ≡ *Z*, a ≡ *A*, b ≡ *B*, and iir(a,b)represents *IIR*(*A*,*B*), product(m,x,y) represents M = X∩Y, product(w,x,y,z) represents *W*=*X*∩*Y*∩*Z*. Then according to Augmentation, if *IIR*(*A*,*B*), M is a subset of W, then IIR(A∩W, B∩M). In general, to implement Augmentation we must use the following Datalog rules:

```
iir(P,W1∩Y):-iir(X,Y),product(W,W1,W2,W3),produ
ct(P,X,W).
iir(P,W1∩Y):-iir(X,Y),product(W,W1,W2,W3),produ
ct(P,W,X).
```

For W2 and W3 we would have a similar pair of Datalog rules.

If there are further products containing differing numbers of random events, then augmentation rules must be created for these as well. In general there will be two additional rules for each defined product of *n* members.

The two rules above effectively gives IIR from the product *M* to a product between random events *W* and *Y*. We need some further rules to show these as binary relations, to allow the further uncovering of available IIR. This is allowable as we have no other iir predicates with three arguments, so only those relationships, arising from the representations of random events being involved in augmentation will be evaluated. So to derive binary relationships the additional rules required are:

```
iir(P,W):-iir(P,W∩Y).
iir(P,Y):-iir(P,W∩Y).
```

This states, that the first argument, *i.e.* a random event, contains the information content, of another random event, if that latter random event is in a product derived from the augmentation rules above. There are two instances of the rule to allow both parts of the product to be uncovered. In answer to the query, iir(m,X), asking which random events are contained in the information content of random event *M*, Datalog returns the following:

```
{
  iir(m,b),
  iir(m,w)
}
Info: 2 tuples computed.
```

Indicating IIR(*M*,*B*) and IIR(*M*,*W*), respectively. Note that the product *W* will be further decomposed by the product and decomposition rules.

6) *Decomposition*

According to Decomposition, if IIR(D,E∩G), then IIR(D, E) and IIR(D, G). This may now be coded in Datalog as follows.

```
iir(d,pEG).
product(pEG,e,g).
```

where d, e, g and pEG are Datalog constants. We assume d ≡ *D*, e ≡ *E*, and g ≡ *G*, product(pEG,e,g) is equivalent to *pEG=E∩G*. To decompose this product we must use the Datalog rules:

```
iir(X,Y):-product(P,Y,A),iir(X,P).
iir(X,Y):-product(P,A,Y),iir(X,P).
```

This states, that the first argument, *i.e.*, a random event, contains the information content, of another random event, if that latter random event is in a product which is in the information content of the random event, represented by the first argument. There are two instances of the rule to allow for unordered evaluation. In answer to the query, iir(d,X), asking which random events are contained in the information content of random event *D*, Datalog returns the following:

```
{
  iir(d,e),
  iir(d,g),
  iir(d,pEG)
}
Info: 3 tuples computed.
```

These indicate IIR(*D,A*), IIR(*D,G*) and IIR(*D,E*∩*G*), respectively. Note that the first two are created due to Decomposition.

We will now consider two examples herein. Firstly we shall consider a notional group of IIR and determine whether we can elaborate the closure, *i.e.*, all the pertinent (*i.e.*, logically implied) IIR arising from a set of specified IIR. Secondly we will consider a more real world example of a student database.

5.2 An Example of IIR Closures

Example 1

For our first example, we assume that the following IIR are given:

F={*IIR*(*A*∩*B*,*C*), *IIR*(*C*,*A*),
IIR(*B*∩*C*,*D*),
IIR(*A*∩*C*∩*D*,*B*),
IIR(*D*,*E*∩*G*),
IIR(*B*∩*E*,*C*),
IIR(*C*∩*G*,*B*∩*D*),
IIR(*C*∩*E*,*A*∩*G*) }

In addition, we assume following random events (Note that some of them are products/sums of some others):

W=*X*∩*Y*∩*Z*
M=*A*∩*W*,
L=*P*∪*Q*∪*A*,
T=*B*∩*D*∩*W*

In which as said before IIR(*X*,*Y*) is a simplified version of *I*(*X*)∋*Y*. Each upper-case letter stands for a random event in a notional database, extant at given spatial and temporal coordinates. That is, each random event is an entity in a database containing a specific set of attribute values. Additionally we adopt the convention that any intersection of random events, such as *A*∩*B* implies that both random events must have, or should have occurred concurrently, which results in a product of random events. Lastly the union of random events indicates that at least one of any of the random events in the union takes place, which results in a sum of random events.

Supposing we wanted to know the IIR closures of all combinations of the database entities (*i.e.*, random events) based on the above given IIR. It has been proved by Xu *et al*, 2008 that for the above IIR, the IIR inference rules given in Subsection 3.4 are sound and complete to derive all IIR that are logically implied by a given set of IIR.

We are now in a position to code the example, specified above. Here is the listing of the code.

1) Example Code

```
% Purpose of this program is to try and
% generate the IIR closure for
% specific set of random events given
% the original IIR

% facts
% =====
% Here is the IIR we wish to represent
% in Datalog

% 1. IIR(A∩B,C)
% 2. IIR(C,A)
% 3. IIR(B∩C,D)
% 4. IIR(A∩C∩D,B)
% 5. IIR(D,E∩G)
% 6. IIR(B∩E,C)
% 7. IIR(C∩G,B∩D),
% 8. IIR(C∩E,A∩G)
% 9. W = X∩Y∩Z
% 10. M = W∩A
% 11. L = P∪Q∪A

% To find the closure BDW+

%12. T = B∩D∩W

% The number of Datalog constants we
% employ are:
% t, a, b, c, d, e, g, l, m, n, w, x,
% y, z, pAB,
% pBC, pACD, pEG, pBE, pCG, pBD, pCE,
% pAG
% 23 constants in total

% 1. IIR(A∩B,C)
product(pAB,a,b).
iir(pAB,c).

% 2. IIR(C,A)
iir(c,a).

% 3. IIR(B∩C,D)
product(pBC,b,c).
iir(pBC,d).

% 4. IIR(A∩C∩D,B)
product(pACD,a,c,d).
iir(pACD,b).

% 5. IIR(D,E∩G)
product(pEG,e,g).
iir(d,pEG).
% 6. IIR(B∩E,C)
product(pBE,b,e).
```

```
iir(pBE,c).

% 7. IIR(C∩G,B∩D),
product(pCG,c,g).
product(pBD,b,d).
iir(pCG,pBD).

% 8. IIR(C∩E,A∩G)
product(pCE,c,e).
product(pAG,a,g).
iir(pCE,pAG).

% 9. W = X∩Y∩Z
product(w,x,y,z).
% 10. M = W∩A
product(m,w,a).

% 11. L = P∪Q∪A
sum(l,p).
sum(l,q).
sum(l,a).

% The final rule expresses the product
% BDW+. This allows us
% to find the closure for these three
% random events.

%12. T = B∩D∩W
product(t,b,d,w).

% rules
% =====

% product

% for a product of 2 members

iir(P,X):-product(P,X,A).
iir(P,X):-product(P,A,X).

% for a product of 3 members

iir(P,X):-product(P,X,A,B).
iir(P,X):-product(P,A,X,B).
iir(P,X):-product(P,A,B,X).

% Note: variables A and B are place
% holders in the above
% predicates - we are not interested
% in their content.

% We need no further product rules as
% there are no products
% containing more than 3 random
% events.

% transitivity

iir(X,Z):-iir(X,Y),iir(Y,Z).

% union

iir(X,P):-product(P,Y,Z),iir(X,Y),product(P,Y,Z),iir(X,Z).

% augmentation

% We have products of 2 and 3 members
% so need two sets
% of augmentation rules.
% each of these requiring 2 (n +2)
% rules where n is the number in
% the product and two sets allows
% any ordering.

iir(P,W):-product(P,X,W),iir(X,Y),product(W,W1,W2,W3).
iir(P,W):-product(P,W,X),iir(X,Y),product(W,W1,W2,W3).
iir(P,W1):-product(P,X,W),iir(X,Y),product(W,W1,W2,W3).
iir(P,W1):-product(P,W,X),iir(X,Y),product(W,W1,W2,W3).
iir(P,W2):-product(P,X,W),iir(X,Y),product(W,W1,W2,W3).
iir(P,W2):-product(P,W,X),iir(X,Y),product(W,W1,W2,W3).
iir(P,W3):-product(P,X,W),iir(X,Y),product(W,W1,W2,W3).
iir(P,W3):-product(P,W,X),iir(X,Y),product(W,W1,W2,W3).

iir(P,Y):-iir(X,Y),product(P,W,X),product(W,W1,W2,W3).
iir(P,Y):-iir(X,Y),product(P,X,W),product(W,W1,W2,W3).

iir(P,W):-product(P,X,W),iir(X,Y),product(W,W1,W2).
iir(P,W):-product(P,W,X),iir(X,Y),product(W,W1,W2).
iir(P,W1):-product(P,X,W),iir(X,Y),product(W,W1,W2).
iir(P,W1):-product(P,W,X),iir(X,Y),product(W,W1,W2).
iir(P,W2):-product(P,X,W),iir(X,Y),product(W,W1,W2).
iir(P,W2):-product(P,W,X),iir(X,Y),product(W,W1,W2).

iir(P,Y):-iir(X,Y),product(P,W,X),product(W,W1,W2).
iir(P,Y):-iir(X,Y),product(P,X,W),product(W,W1,W2).


% sum

% rules (Note the variables A and B are
% place holders).
% this is a sum of 3 members.

iir(X,Y):-sum(Y,X).
```

We will now attempt to generate the closure for a specific product of $B \cap D \cap W$. We have represented this product by assigning this product to random event *T*. This generates the following set of tuples, in response to the query **iir(t,X)**:

```
{
  iir(t,a),
  iir(t,b),
  iir(t,c),
  iir(t,d),
  iir(t,e),
  iir(t,g),
  iir(t,l),
  iir(t,m),
  iir(t,pAB),
  iir(t,pAG),
  iir(t,pBC),
  iir(t,pBD),
  iir(t,pBE),
  iir(t,pCE),
  iir(t,pCG),
  iir(t,pEG),
  iir(t,w),
  iir(t,x),
  iir(t,y),
  iir(t,z)
}
Info: 20 tuples computed.
```

That is, all defined random events and their various products are in the information content of the product $B \cap D \cap W$, as expected, except for the additional members of the sum *L*.

5.3 What We Learnt from This Example

This example shows that we can use Datalog to implement all the inference rules that we developed for carrying out deduction on IIR. Moreover, this example also demonstrates that IIR closures can be computed by using Datalog. In the section that follows, we give the algorithm for computing IIR closures.

5.4 An Algorithm for Computing IIR Closures

We now give an algorithm for uncovering logically implied IIR as pseudo logic below.

```
Select random events
Create a product of the random events (as these events
may be considered to have occurred simultaneously).

LOOP until
        OR any iteration gives the same product as before
        OR all available random events are included
        OR no further random events can be obtained for
        the product
        OR any remaining IIR do not contain any subset
        of the current product
                IF any random event of the product has
                an IIR transitive relation with further
                random events
                        Use the union rule to add these
                        further random events to the
                        product of random events
                END-IF
                IF any single random event and any
                sub-product of the product may be used
                in the IIR augmentation rule
                        Use the augmentation rule to
                        add each member of the
                        product to the product of random events.
                END-IF
                IF any random event of the product
                belongs to a union of random events
                        Use the sum rule and the union
                        rule to add the sum to the
                        product of random events
                END-IF
END-LOOP
```

The result product of random events is the closure of the set of original random events that was selected at the beginning of the algorithm.

Note the algorithm implicitly uses the decomposition rule, and product rule, when utilizing sub-products of the resultant random event product string.

5.5 An Example of Querying a Database Using IIR Closures

Example 2

The next example is taken from Wu and Feng [17], but we use our Datalog system to complete the job. This example is concerned with how to derive IIR upon an example of a real-world database. We show how our Datalog system accomplishes this task. Functional dependencies between attributes constitute a basis for IIR between values of attributes. The reasoning behind this is that an instance, *i.e.*, a tuple of an entity (represented by a *relation*) such as *student*, *class* and *enrolment* can be seen as a collection of particulars of some random events, and moreover the random events may be involved in certain IIR. Some IIR are captured by functional dependencies between attributes. For example, attribute *student id* functionally determines attributes *surname*, *major*, *level* (or *year*) and *age*. Attribute value "*student id* being 100" *is a* random event, so is "*surname* being Smith", and moreover, the latter is in the information content of the former, which can be denoted as IIR("*student id* being 100", "*surname* being Smith"). This IIR is underpinned by the aforementioned functional dependency.

We now show how this example is coded up in Datalog as follows:

```
% Facts

student(100,smith,history,gr,25).
student(150,parks,geology,so,21).
student(200,baker,finance,gr,24).
student(250,glass,history,sn,19).
student(300,baker,geology,sn,20).
student(350,rosso,finance,jr,18).
student(400,bryan,geology,sr,22).

class(ba200,tth9,sc110).
class(bd445,mwf3,sc213).
class(bf410,mwf8,sc213).
class(cs150,mwf3,ea304).
class(cs250,mwf1,eb210).

enrollment(100,bd445).
enrollment(150,ba200).
enrollment(200,bd445).
enrollment(200,cs250).
enrollment(300,cs150).
enrollment(400,ba200).
enrollment(400,bf410).
enrollment(400,cs250).
enrollment(450,ba200).

businessRule(history,swimming).
businessRule(geology,diving).
businessRule(finance,basketball).

% rules
iir(X,Y):-student(A,B,X,C,D),businessRule(X,Y).

% functional dependencies

iir(X,Y):-student(X,B,C,D,E),enrollment(X,Y).
iir(X,Y,Z):-enrollment(A,X),class(X,Y,Z).

result(A,B,C,D,E,F,G,H,I):-
student(A,B,C,D,E),
iir(A,F),
iir(F,G,H),
iir(C,I).
```

If the program is run with the following query:
result(A,B,C,D,E,F,G,H,I).
It gives the following results:

```
{
result(100,smith,history,gr,25,bd445,mwf3,sc213,swimming),
result(150,parks,geology,so,21,ba200,tth9,sc110,diving),
result(200,baker,finance,gr,24,bd445,mwf3,sc213,basketball),
result(200,baker,finance,gr,24,cs250,mwf1,eb210,basketball),
result(300,baker,geology,sn,20,cs150,mwf3,ea304,diving),
result(400,bryan,geology,sr,22,ba200,tth9,sc110,diving),
result(400,bryan,geology,sr,22,bf410,mwf8,sc213,diving),
result(400,bryan,geology,sr,22,cs250,mwf1,eb210,diving)
}
Info: 8 tuples computed.
```

These are IIR closures arrived at of "Student ID being 100", "Student ID being 150", etc. respectively, which constitute the "information base" (see **Figure 1**) for queries. If a query is looking for "all the students that like diving", by checking the IIR closures, we get students Parks, Baker and Bryan.

5.6 What We Learnt from This Example

This example demonstrates that our approach works on real world situations. We code tuples in a database as Datalog "facts", and identify part of original IIR from functional dependencies between attributes of a relation, which are then used in coding Datalog "rules". Then IIR closures are computed, which serve as the information base for answering queries. Our Datalog system works exactly as expected.

6. Contributions of This Work

We observe that this work makes the following contributions to the field of databases. Fist of all, a justifiable approach to defining the notion of the "information content" of data with mathematical rigor was developed. This approach appears superior to intuitive approaches that we have seen thus far in the literature that is based on equating data with information.

Secondly, we have shown that reasoning about information content of data (rather than data *per se*) is possible, and this can be achieved by identifying a set of sound inference rules, and then these rules are implemented with a logic based system, *i.e.*, Datalog.

Thirdly, we also have demonstrated that information content based reasoning can go beyond "direct match" that conventional query answering employs. The former reveals information that the latter cannot find.

Therefore, in summary, we have constructed and tested an innovative approach to a fundamental problem in the field of databases, namely the information that data in a database carry. This may be seen constituting some significant value in further developing database theory and we also have shown that this is also applicable to real world problems.

7. Conclusions

In this paper, we have proposed a novel approach to the information content of data in a database. We gave a set of basic concepts and described an experimental system that makes use of this notion. A number of examples were used to test our system. With information sources outside a database imported into the system, the information content of a random event (data values) within the database expanded dramatically. Users could make the most of the information content of data by posing queries. Thus, more information can be discovered than conventional queries. The increase of random events' closures is based on the boost in original IIR and the inference capability using IIR. Identification of original IIR rules could be hard due to wide range of sources outside database. However, once original IIR have been identified and then integrated into the computing unit of our system, the system provides a powerful engine for users to query a database. Our experiment shows that with the IIR inference capability hidden information within database can be discovered with the increase of original IIR derived from database itself and external sources.

With IIR rules, we discussed the relation of information content inclusion between random events. Such a relation at a higher level, *i.e.*, that between random variables requires more work. How the relations on different levels are connected also deserves further investigation. The process of identifying original IIR was done manually, for which a semi-automated technique making use of meta-data to suit the need of a user is desirable and looks feasible. Moreover, how to approach and inference about the information content of data that are stored in independent and yet inter-operating databases should be investigated.

In summary, our work thus far seems to have shown that the information that a database can potentially provide is definable by using the notion of *information content inclusion relation* (IIR). Furthermore, the inference rules for formally reasoning about such a relation enables the development of a seemingly elegant way, by means of *IIR closure*, of identifying the information content of data in a database, which serves as a basis for answering queries.

8. Acknowledgments

This work is partly sponsored by the a grant for Distributed Information Systems Research from the Carnegie Trust for Universities of Scotland, 2007, a grant for research on Semantic Interoperability between Distributed Digital Museums from the Carnegie Trust for Universities of Scotland, 2009, and a PhD studentship of the University of the West of Scotland, UK.

REFERENCES

[1] T. Connolly and C. Begg, "Database systems: A practical approach to design, implementation, and manage- ment," Pearson Education, 2004.

[2] Y. Feng, "Database foundation," Hua Zhong Institute of Technology Press, 1986.

[3] J. Mingers, "Information and meaning: Foundations for an intersubjective account," Information Systems Journal, Vol. 5, No. 4, pp. 285–306, 1995.

[4] F. Dretske, "Knowledge and the flow of information," CSLI Publications, Stanford, 1999.

[5] S. Wang, C. Lin, and J. Feng, "A hermeneutic approach to the notion of information in IS," Proceedings of the 7th WSEAS International Conference on Simulation, Modeling and Optimization, pp. 400–404, 2007.

[6] J. Feng and M. Crowe, "The notion of 'Classes of a Path' in ER Schemas," Proceedings of 3rd East European Conference on Advances in Databases and Information Systems, Springer, Berlin, 1999.

[7] J. Barwise and J. Seligman, "Information flow: The logic of distributed systems," Cambridge University Press, 1997.

[8] H. Xu and J. Feng, "Towards a definition of the 'information bearing capability' of a conceptual data schema," Systems Theory and Practice in the Knowledge Age, Kluwer Academic/Plenum Publishers, New York, 2002.

[9] R. Hull, "Relative information capacity of simple relational database schemata," SIAM Journal of Computing, Vol. 15, No. 3, pp. 856–886, 1984.

[10] R. J. Miller, Y. E. Ioannidis, and R. Ramakrishnan, "The use of information capacity in schema integration and translation," Proceedings of the 19th International Conference on Very Large Data Bases, Dublin, Ireland, pp. 120–133, 1993.

[11] R. J. Miller, Y. E. Ioannidis, and R. Ramakrishnan, "Schema equivalence in heterogeneous Systems: Bridging theory and practice," Information Systems, Vol. 19, No. 1, pp. 3–31, 1994.

[12] L. Floridi, "Is semantic information meaningful data?" Philosophy and Phenomenological Research, Vol. 70, No. 2, pp. 351–370, 2005.

[13] K. Xu, J. Feng, and M. Crowe, "Defining the notion of 'information content' and reasoning about it in a database," Knowledge and Information Systems, Vol. 18, No. 1, 2009.

[14] J. Feng, "The 'information content' problem of a conceptual data schema," Systemist, Vol. 20, No. 4, pp. 221-233, November 1998.

[15] A. Shimojima, “On the efficacy of representation,” Ph.D. Thesis, Indiana University, 1996.

[16] Y. Wang and J. Feng J, “FCA assisted IF channel construction towards formulating conceptual data modeling,” WSEAS Transactions on Systems, Vol. 6, No. 6, pp. 1159–1167, June 2007.

[17] E. Eessaar, “Guidelines about usage of the complex data types in a database,” WSEAS Transactions on Information Science and Applications, Vol. 3, No. 4, pp. 712–719, April 2006.

[18] S. Urman, R. Hardman, and M. McLaughlin, “Oracle database 10G PL/SQL programming,” McGraw-Hill/Osborne, 2004.

[19] X. Wu and J. Feng, “A framework and implementation of information content reasoning in a database,” WSEAS Transactions on Information Science and Applications, Vol. 6, No. 4, pp. 579–588, April 2009.