Knowl Inf Syst (2009) 18:29–59
DOI 10.1007/s10115-008-0129-3

REGULAR PAPER

Defining the notion of 'Information Content' and reasoning about it in a database

Kaibo Xu · Junkang Feng · Malcolm Crowe

Received: 5 February 2007 / Revised: 23 December 2007 / Accepted: 19 January 2008 /
Published online: 11 March 2008

Abstract The problem of 'information content' of an information system appears elusive. In the field of databases, the information content of a database has been taken as the instance of a database. We argue that this view misses two fundamental points. One is a convincing conception of the phenomenon concerning information in databases, especially a properly defined notion of 'information content'. The other is a framework for reasoning about information content. In this paper, we suggest a modification of the well known definition of 'information content' given by Dretske(Knowledge and the flow of information,1981). We then define what we call the 'information content inclusion' relation (IIR for short) between two random events. We present a set of inference rules for reasoning about information content, which we call the IIR Rules. Then we explore how these ideas and the rules may be used in a database setting to look at databases and to derive otherwise hidden information by deriving new relations from a given set of IIR. A prototype is presented, which shows how the idea of IIR-Reasoning might be exploited in a database setting including the relationship between real world events and database values.

Keywords Information content · Reasoning · Data semantics · Semantic information theory · Inference rules

K. Xu (✉) · J. Feng
e-Business Research Institute, Business College,
Beijing Union University, Beijing, China
e-mail: kaibo.xu@bcbuu.edu.cn; kaibo.xu@uws.ac.uk

J. Feng
e-mail: junkang.feng@uws.ac.uk

K. Xu · J. Feng · M. Crowe
Database Research Group School of Computing,
University of the West of Scotland, Paisley, UK
e-mail: malcolm.crowe@uws.ac.uk

1 Introduction

Information systems are constructed for storing and providing information. And yet, it would appear that the notion of 'information content' of an information system is elusive. In the field of databases, the information content of a database 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 [23,32,33]. We argue that this view misses two fundamental points. One is a convincing conception of 'information content'. 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 [5, p.10], that is, must relate to a contingent truth [18] while for data there is no such requirement. The other is a framework for reasoning about information content to reveal hidden information. In addressing this problem, our purpose is twofold. One is that we want to reveal what machinery actually already exists with a database from the perspective of reasoning about information content. The other is to look at the relationships between information content, database structure and business rules, and thus discover how tacit business knowledge can then be explicitly expressed and analysed.

We proceed to report our work in the rest of the paper as follows. We define the notion of 'information content' in Sect. 2. One seemingly unusual notion we put forward is the 'particulars' of a random event, for which we give Definition 3 with an example in Sect. 2, and we give further examples in Sect. 4. In Sect. 3, we define the relation concerning information content between two states of affairs called 'information content inclusion' relation (IIR for short) and present a set of inference rules for reasoning about IIR. In Sect. 4, we address the problem of how IIR reasoning might be exploited in a database setting by describing the main components of a prototype. We would draw reader's attention to the alignment between real world events and database values in that section, which is one of the crucial points for IIR to underpin a database. Then we give a brief comparison between our work and related works that are concerned with applying the classic information theory (i.e., Theory of Shannon's [37]) to database problems in Sect. 5. Finally, we make concluding remarks in Sect. 6.

2 The notion of 'Information Content'

The origin of the work presented here was an attempt to apply the *semantic theory of information* presented by [13] to information systems design. One of the most fundamental notions that would be needed within this endeavour is that of 'information content' of a sign, a random event, and in the most general terms, a state of affairs, and to be able to reason about it. Let us consider the following list:

Example 1 That there is smoke carries the information that there is a fire.

Example 2 That he is awarded a grade 'A' for his Programming course contains the information that Jack Brown has gained 70% or above for that course.

Dretske[13, p.45] defines the notion of the 'information content' of a state of affairs as follows:

> *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.*

Following Dretske, we take information as in the form of 'de re', rather than 'de dicto', that is, in the form of '*a*'s being *F* carries the information that *b* is *G*'. Dretske [13] establishes the following definition:

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*).

In this definition, k stands for prior knowledge about information source s. Dretske's approach, which we will extend for our purposes, is based upon the notion of probability [5, pp.14–18], which is concerned with characterising random events, we first give a definition of *random event*:

Definition 1 Let s be a selection process under a set C of conditions, O the set of possible outcomes of s, which are called states, and E the power set of O, X is a *random event* if $E \ni X$ and there is a probability of X, i.e., $\mathrm{P}(X)$.

In our analysis the notion of 'probability distribution' will be relevant, which is concerned with a *probability space*.

Definition 2 Let s be a selection process under a set C of conditions, O the set of possible outcomes of s, E the power set of O and $E \ni X_i$ for $i = 1, \ldots, n$, P_s is the *probability space* of the random events X_i for $i = 1, \ldots, n$ if $\mathrm{P}_s = \{\mathrm{P}(X_1), \mathrm{P}(X_2), \ldots, \mathrm{P}(X_n)\}$ and $\Sigma\mathrm{P}(X_i) = 1$.

Dretske's definition quoted above sounds plausible. However, there are a few points to note about it. First of all, it is particulars, i.e., individual things in the world that carry information [5, p.27]. It is a particular map that carries information about a particular mountain. It is the particular grading event that he is awarded a grade 'A' for his Programming course that contains the information that Jack Brown has gained 70% or above for that course. It is a particular smoke that carries information that there is a particular fire. So the two examples above work on different levels—the first is concerned with a relation (which can be said as an informational one) between two types of random events, namely smoke and fire taking place, and the second a similar informational relation between two particulars of random events, Jack Brown has an 'A' for 'Programming' and he scores 70% or more for it. We now give a definition to the term *particular* of a *random event* below.

Definition 3 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 $\Omega = \mathrm{state}(X_i)$, and $X \ni \Omega$.

The term *particular* was borrowed from Barwise and Seligman [5, p.27]. It could be misleading if being taken to mean something specific. By *particular*, we mean an individual occurrence of a random event. For example, s could be concerned with data values going into an attribute, say, the Emp_Name column of a table; X_i is a data value in the Emp_Name column at a time t, which happens to be 'tony_wu'; the state of X_i, i.e., $\mathrm{state}(X_i)$ = 'a value in Emp_Name column being tony_wu', which is Ω; X is the disjunction of two states, namely, Ω and say, Γ = 'a value in Emp_Name column being shirley_wu'. Then, X_i is a particular of X.

Secondly, probability theory concentrates on the level of types. That a map happens to be of a specific type is a random event, and therefore has a probability. Thus, Dretske's definition is concerned with types, and there is no concept of particulars in his theory as Barwise and Seligman [5, p.26] correctly point out.

But there is a twist, and this is the third point that we are making, namely, that the very capability of a particular to carry information comes from their belonging to types. It is the aforementioned informational relation between types, which reflects some regularities in the world, that is the basis for information to flow, i.e., for one thing to carry information about another. Therefore, only particulars can carry information whereas their types determine what sort of information can be carried.

Finally, information, which has to be carried by particulars rather than types as argued above, comes in the form of types [5, p.27] and not anything else. That is, it is something that is concerned with a type, i.e., something that is common for all the instances of the type, i.e., particulars of a random event (which is represented by the type), and not those that are unique to a specific particular that can be carried. The map's being a certain type of maps carries information that the mountain is of some certain type of mountains [5]. The content of the information itself is what the type of the mountain could tell us, e.g, a mountain with the height of 1,000 ft above sea level, not anything unique to the mountain, such as its specific shape.

Therefore, it would seem appropriate that the above definition of 'the information content of a state of affairs' by Dretske [13] be modified as follows.

Definition 4 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 5 When a particular r_i carries the information that a particular s_j exists we will say that the *information content* of r_i includes s_j, or in other words, s_j is in the *information content* of r_i.

The notion of Shannon's entropy can be used to measure the amount of information associated with a random variable, which models a collection of messages but not an individual message. To talk about the content of an individual message, we have to use the notion of random event as above definitions show.

3 'Information content inclusion' relation

Closely following the previous section, given two random events, say X and Y, there might be a special type of relation between them, i.e., 'the particulars of random event Y are in the information content of the particulars of random event X'. For brevity, we will also call such a relation 'random event Y is in the information content of random event X'. We suggested calling such a relation an 'information content inclusion relation' (IIR) [16]. Interestingly, it happens that this term also appears in the literature, e.g., in her manuscript Duží [14], points out that information content inclusion relations (in relation to attributes) are of partial order.

Definition 6 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.

A random event may have an information content inclusion relation (IIR) with more than one other random event. Every one of the latter provides the former with its set of particulars,

[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.

the whole collection of which is 'what a suitably placed observer could learn by consulting' the particulars of the former. Therefore this is the information content of the former. That is to say, the information content of a random event is the set of random events with which the former has an information content inclusion relation.

Definition 7 Let X be a random event, the *information content* of X, denoted $\mathrm{I}(X)$, is the set of random events with each of which X has an information content inclusion relation.

Therefore, $\mathrm{I}(X) \ni Y$ is a valid expression, which denotes that random event Y is in the information content of random event X through the particulars of random event Y being in the information content of the particulars of random event X (For the notion of 'information content', see Definitions 4 and 5 above). For the sake of the completeness of the definition, we allow $\mathrm{I}(X) \ni X$, which is a trivial case of $\mathrm{I}(X) \ni Y$, when X and Y are not distinct. Note that in this paper we concern ourselves with the 'information content inclusion' relation as just defined only between random events (and their particulars), not any other things. This is because we observe that this random event based approach to information in databases is helpful.

Given a set of IIR, such as the above $\mathrm{I}(X) \ni Y$, generally there are other IIR that are nested within (i.e., logically implied by) them. Sometimes, we wish to derive such implied IIR logically from those that we have already known somehow. 'Logically' here means that we use domain independent inference only and do not use any of what we call 'domain dependent knowledge'. One example of domain dependent knowledge is a business rule that 'if a supplier supplies a part and the part is used by a project, then the supplier supplies the project'. To this end, in the next section, we present and prove a set of domain independent inference rules, which can be called IIR Rules.

3.1 Inference rules for IIR

Let $\mathrm{P}(Y|X)$ be the probability of Y under the condition X; Let $\mathrm{P}(Y)$ be the probability of Y without the condition X. We are now in the position to present a set of inference rules for logically reasoning about information content. The proofs to be presented below contain some comments on important steps: the comments are not formally part of the proof. We borrow some terms from the well known Armstrong's axioms [3] for functional dependency. We will discuss the differences between IIR and functional dependency shortly.

Note that the inference rules to be presented below are not for the identification of the original (i.e., before applying the IIR Rules) set of IIR except the trivial ones through the Sum or Product rules shown shortly. Original non-trivial IIR have to be identified by directly using the definition of information content that we gave earlier. In addition, other theories such as Information Flow [5] and Formal Concept Analysis [42] can also be used for the identification of IIR.

To facilitate the reader to understand the proofs of the rules, we re-iterate the following.

What is meant by 'I(X) ∋ Y'?

It means that *a suitably placed observer could learn that Y (particulars of Y) by consulting the particulars of X.*

A sufficient condition for '$\mathrm{I}(X) \ni Y$':

1) Both X and Y are random events, namely they could be contingently true and contingently untrue, but are neither necessarily true nor necessarily untrue. Mathematically, $\mathrm{P}(X) \neq 1$ and $\mathrm{P}(X) \neq 0$, and $\mathrm{P}(Y) \neq 1$ and $\mathrm{P}(Y) \neq 0$.
2) Whenever X is true, Y is always true. That is, $\mathrm{P}(Y|X) = 1$. In other words, $X \subset Y$.

Inference rule 0

'Sum': If $Y = X_1 \cup X_2 \cdots \cup X_n$, then $\mathrm{I}(X_i) \ni Y$ for $i = 1, \ldots,$ 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.

Proof

Assume X_i	Assumption	1
(Comment: assume that X_i is true for i = 1,…,n.)		
$Y = X_1 \cup X_2 \cdots \cup X_n$	Premise	2
Y	1 and 2	3
$\mathrm{P}(Y \mid X_i) = 1$	1 and 3	4
(Comment: If X_i is true, Y is always true. Thus Y's probability is 1 given X_i.)		
$\mathrm{P}(Y) \neq 1$	Premise	5
(Comment: X, Y, W and Z are assumed random events.)		
$\mathrm{I}(X_i) \ni Y$	4 and 5	
(Comment: this is because $\mathrm{P}(Y) \neq 1$ and $\mathrm{P}(Y \mid X) = 1$.)		

□

We will use the following example states of affairs[2] to explain the IIR inference rules:

(ψ_1) Someone a holds a colored (green, red, blue or yellow) ball in her hand.
(ψ_2) Someone a holds a blue ball in her hand.
(ψ_3) Someone a utters the words 'she', 'is' and 'insane' while pointing to Jane.
(ψ_4) a utters the word 'she' while pointing to Jane.
(ψ_5) Jane is insane at some moment t.
(ψ_6) Jane's behaviour is odd at t.
(ψ_7) ψ_4 and ψ_5 above combined to form that someone a utters the word 'she' while pointing to Jane at time t and Jane is insane at t.
(ψ_8) Someone a is a PhD student and part time lecturer.
(ψ_9) a is a PhD student.
(ψ_{10}) a is awarded a grade 'A' for her/his Programming course.
(ψ_{11}) a gains 70% or above for her/his Programming course.
(ψ_{12}) the product of above ψ_8 and ψ_{10}.
(ψ_{13}) the product of above ψ_9 and ψ_{11}.
(ψ_{14}) a attends the Programming course.
(ψ_{15}) the product of ψ_{11} and ψ_{14} above.

Example 3 The ψ_1 above is a disjunction of a few others including the ψ_2 above. Thus according to the rule, ψ_1 is in the information content of ψ_2.

Inference rule 1

'Product': If $X = X_1 \cap X_2 \cdots \cap X_n$, $Y = X_i$ for $i = 1, \ldots, n$, then $\mathrm{I}(X) \ni Y$

This rule says that if a random event X is the conjunction of a number of random events, then any of the latter is in the information content of the former. A trivial case is where X and Y above are not distinct.

[2] These states of affairs may be used by more than one example, and they retain their identifications throughout the examples that use them.

Proof

Assume X	Assumption	1
(Comment: Assume that X is true.)		
$Y = X_i$	premise	2
(Comment: This is for $i = 1, \ldots,$ n as usual.)		
Y	1 and 2	3
$\mathrm{P}(Y\|X) = 1$	1 and 3	4
(Comment: If X is true, Y is always true. Thus Y's probability is 1 given X.)		
$\mathrm{P}(Y) \neq 1$	Premise	5
(Comment: X, Y, W and Z are assumed random events.)		
$\mathrm{I}(X) \ni Y$	4 and 5	
(Comment: this is because $\mathrm{P}(Y) \neq 1$ and $\mathrm{P}(Y\|X) = 1$.)		

□

Example 4 (ψ_3) Someone a utters the words 'she', 'is' and 'insane' while pointing to Jane.
(ψ_4) a utters the word 'she' while pointing to Jane.

The ψ_3 above is a conjunction of a few others including the ψ_4 above, so according to the rule, ψ_4 is in the information content of ψ_3.

Inference rule 2
Transitivity: If $\mathrm{I}(X) \ni Y$, $\mathrm{I}(Y) \ni Z$ then $\mathrm{I}(X) \ni Z$
This rule says that if the information content of a random event X includes another random event Y, and the information content of Y includes yet another random event Z, then the information content of X includes Z.

Proof

$\mathrm{I}(X) \ni Y$	Premise	1
(Comment: Information content of X includes Y, which is given.)		
$\mathrm{P}(Y\|X) = 1$	Definition and 1	2
(Comment: Y's probability, under the condition X, is 1 by definition.)		
$\mathrm{I}(Y) \ni Z$	Premise	3
(Comment: Information content of Y includes Z, which is given.)		
$\mathrm{P}(Z\|Y) = 1$	Definition and 3	4
(Comment: Z's probability, under the condition Y, is 1 by definition.)		
Assume X	Assumption	5
(Comment: Assume that X is true.)		
Y	2 and 5	6
(Comment: Y is true.)		
Z	4 and 6	7
(Comment: Z is true.)		
$\mathrm{P}(Z\|X) = 1$	5, 7	8
(Comment: If X is true, then Z's probability is 1.)		
$\mathrm{P}(Z) \neq 1$	premise	9
$\mathrm{I}(X) \ni Z$	8 and 9	

□

Example 5 (ψ_3) Someone a utters the words 'she', 'is' and 'insane' while pointing to Jane.
(ψ_5) Jane is insane at some moment t.
(ψ_6) Jane's behaviour is odd at t.

Given that ψ_5 above is in the information content of ψ_3 (which would be the case if when ψ_3 is true, ψ_5 is true, and if ψ_3 is not certainly true, the state of ψ_5 is uncertain), and ψ_6 above is in the information content of ψ_5, according to this rule, we get that ψ_6 is also in the information content of ψ_3. That is to say, by observing ψ_3, one could learn ψ_6.

Inference rule 3

Union: If $\mathrm{I}(X) \ni Y$, $\mathrm{I}(X) \ni Z$ then $\mathrm{I}(X) \ni Y \cap Z$

This rule says that if the information content of a random event X includes another two random events Y and Z respectively, then the information content of X includes random event $Y \cap Z$ that is the product of Y and Z. 'Union' here indicates that whenever event X happens, both event Y and event Z happen.

Proof

$\mathrm{I}(X) \ni Y$	Premise	1
$\mathrm{P}(Y\|X) = 1$	Definition and 1	2
$X \subset Y$	2	3
(Comment: This follows probability theory.)		
$\mathrm{I}(X) \ni Z$	Premise	4
$\mathrm{P}(Z\|\ \ X) = 1$	Definition and 4	5
$X \subset Z$	5	6
(Comment: This follows probability theory.)		
$X \subset Y \cap Z$	3 and 6	7
$\mathrm{P}((Y \cap Z)\|X) = 1$	7	8
(Comment: This follows probability theory.)		
$Y \cap Z$ is a random event	Premise	9
$\mathrm{P}(Y \cap Z) \neq 1$	9	10
$\mathrm{I}(X) \ni Y \cap Z$	8 and 10	

□

Example 6 (ψ_3) Someone a utters the words 'she', 'is' and 'insane' while pointing to Jane.
(ψ_4) a utters the word 'she' while pointing to Jane.
(ψ_5) Jane is insane at some moment t.
(ψ_7) ψ_4 and ψ_5 above combined to form that someone a utters the word 'she' while pointing to Jane at time t and Jane is insane at t.

ψ_4 and ψ_5 above are in the information content of ψ_3 above, respectively. Following this rule, the product of ψ_4 and ψ_5 namely ψ_7 is also in the information content of ψ_3. That is to say, by observing ψ_3, one could learn ψ_7.

Inference rule 4

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

This rule says that if $W = W_1 \cap W_2 \cdots \cap W_n$, random event Z is the product of a subset of $\{W_1, W_2, \ldots, W_n\}$, and the information content of random event X includes random event

Y, then the information content of the random event $W \cap X$ formed by the product of W and X includes the random event $Z \cap Y$ formed by the product of Z and Y.

Proof

Given W	Assumption	1
Z is the product of a subset of $\{W_1, W_2, \ldots, W_n\}$	Premise	2
$\mathrm{I}(W) \ni Z$	Rule 1	3
Given X	Assumption	4
$W \cap X$	Assumption	5
W is a factor of $W \cap X$	5	6
$\mathrm{I}(W \cap X) \ni W$	Rule 1	7
(Comment: Information content of $W \cap X$ includes W by Rule 1.)		
$\mathrm{I}(W \cap X) \ni Z$	Rule 2 applied to 7 then 3	8
$\mathrm{I}(X) \ni Y$	Premise	9
X is a factor of $W \cap X$	5	10
$\mathrm{I}(W \cap X) \ni X$	Rule 1	11
$\mathrm{I}(W \cap X) \ni Y$	Rule 2 applied to 11 then 9	12
$\mathrm{I}(W \cap X) \ni Z \cap Y$	Rule 3 applied to 8, 12	

□

Example 7 (ψ_8) Someone a is a PhD student and part time lecturer.
(ψ_9) a is a PhD student.
(ψ_{10}) a is awarded a grade 'A' for her/his Programming course.
(ψ_{11}) a gains 70% or above for her/his Programming course.
(ψ_{12}) the product of above ψ_8 and ψ_{10}.
(ψ_{13}) the product of above ψ_9 and ψ_{11}.

In the above, ψ_9 is a factor of ψ_8, and ψ_{11} is in the information content of ψ_{10}. Following the rule discussed here, by consulting ψ_{12}, one could learn ψ_{13}.

Inference rule 5

Decomposition: If $\mathrm{I}(X) \ni Y \cap Z$ then $\mathrm{I}(X) \ni Y$, $\mathrm{I}(X) \ni Z$

This rule means that if the information content of random event X includes random event $Y \cap Z$ that is the product of random event Y and random event Z, then Y and Z, as separate random events, are in the information content of X, respectively.

Proof

$\mathrm{I}(X) \ni Y \cap Z$	Premise	1
$\mathrm{P}((Y \cap Z)\|X) = 1$	1	2
(Comment: By the definition of 'information content'.)		
$X \subset Y \cap Z$	2	3
(Comment: This follows probability theory.)		
$X \subset Y$	3	4
$X \subset Z$	3	5
$\mathrm{P}(Y\|X) = 1$	4	6
$\mathrm{P}(Z\|X) = 1$	5	7
(Comment: This follows probability theory.)		
Y, Z are random events	Premise	8

$\mathrm{P}(Y) \neq 1$	8	9
$\mathrm{P}(Z) \neq 1$	8	10
$\mathrm{I}(X) \ni Y$	6 and 9	
$\mathrm{I}(X) \ni Z$	7 and 10	

□

Example 8 (ψ_{10}) a is awarded a grade 'A' for her/his Programming course.
(ψ_{11}) a gains 70% or above for her/his Programming course.
(ψ_{14}) a attends the Programming course.
(ψ_{15}) the product of ψ_{11} and ψ_{14} above.

In the above, if it is given that ψ_{15} is in the information content of ψ_{10}, then following the rule discussed here, by consulting ψ_{10}, one could learn ψ_{11} and ψ_{14} respectively.

3.1.1 Independence of the IIR rules

Similar to Armstrong's inference rules on functional dependencies between attributes of a relation [3] not all the inference rules are independent, and rules 1, 3 and 5 above can be derived from the other three rules. Here is the proof for the Decomposition rule.

Proof

$\mathrm{I}(X) \ni Y \cap Z$	Premise	1
$Y \cap Z$	Assumption	2
Y is a factor of $Y \cap Z$	2	3
$\mathrm{I}(Y \cap Z) \ni Y$	Rule 1	4
Z a factor of $Y \cap Z$	2	5
$\mathrm{I}(Y \cap Z) \ni Z$	Rule 1	6
$\mathrm{I}(X) \ni Y$	Rule 2 applied to 1, 4	
$\mathrm{I}(X) \ni Z$	Rule 2 applied to 1, 6	

□

The Union rule can also be proved from other rules: we leave this to the reader.

3.1.2 The completeness of IIR inference rules

For the IIR inference rules to be complete, IIR that is logically implied by a set F of known IIR must be deducible from F by using the rules. That a set of IIR, say F, logically implies an IIR, say IIR_i, means that whenever F is true IIR_i is true.

The proof of the completeness of IIR inference rules is based upon the idea of 'constructive proof'. We take the approach similar to that of the 'standard' proof of the completeness of Armstrong's rules for functional dependencies [3]. We want to show that any IIR that cannot be deduced from F by using the IIR inference rules is not logically implied by F, i.e., it is not the case that whenever F is true the IIR is true. To this end, we only need to find an instance of a set of random events under consideration against which F is true and the IIR is not.

Let $R(X, Y, Z, \ldots)$ be a set of random events under consideration, F be a set of IIR that hold on R. Suppose $\mathrm{I}(X) \ni Y$ cannot be deduced from F by the IIR inference rules. Consider an instance of R, say r, with two tuples as follows that capture how the random events fare at two different occasions (Table 1).

Table 1 A relation r with two tuples

	X^+				$U - X^+$			
t_1	T	T	···	T	T	T	···	T
t_2	T	T	···	T	F	F	···	F

In the table above, X^+ is the closure of X with respect to F, namely within X^+, for any $X_i (i = 1, 2 \ldots \text{n})$, $\text{I}(X) \ni X_i$ can be deduced from F by the inference rules; 'T' indicates that a random event happens to be true (i.e., to occur), and 'F' false (i.e., not to occur); and U is the set of random events in R.

The instance r of R has the same truth value 'T' in the first tuple. In the second, it has a 'T' for all random events within X^+ and 'F' for the rest, i.e., $U - X^+$.

Firstly let us show that all IIR in F are satisfied by r.

Assume a relation $\text{I}(V) \ni W$ in F, we want to deduce that $\text{I}(V) \ni W$ holds on r. There are only two possible different situations in terms of whether V is in X^+ or not. These will now be examined in turn.

• If $V \subseteq X^+$	then $\text{I}(X) \ni V$	definition of X^+	1
	$\text{I}(V) \ni W$ in F	assumption	2
	$\text{I}(X) \ni W$	1, 2 and Transitivity	3
	$W \subseteq X^+$	definition of X^+	4
In r			
	When V is true	assumption	5
	V is in t_1 and t_2	definition of r	6
	W is true	definition of r	7
	$\text{P}(W\|V) = 1$ is not violated	5 and 7	8
	$\text{I}(V) \ni W$ holds on r	8 and the Definition	9
• If $V \not\subset X^+$ (note that W can be either in X^+ or in $U - X^+$), then in r			
	V is in $U - X^+$	assumption	10
	When V is true	assumption	11
	V can only be in t_1 and not in t_2	definition of r	12
	W is true	definition of r	13
	Comment: as only t_1 needs to be considered		
	$\text{P}(W\|V) = 1$ is not violated	11 and 13	14
	$\text{I}(V) \ni W$ holds on r	14 and the Definition	15
	We have now shown that all IIR in F are satisfied by r.		
	We will now show that $\text{I}(X) \ni Y$ does not hold on r:		
	$\text{I}(X) \ni Y$ cannot be deduced by the IIR inference rules with respect to F	premise	16
	$Y \not\subset X^+$ (i.e., $Y \subseteq R - X^+$)	definition of X^+	17
	$X \subseteq X^+$	definition of X^+	18

In r

When X is true	assumption	19
X is in t_1 and in t_2	Definition of r	20
Y is either true (if int_1) or false (if in t_2)	Definition of r	21
$P(Y\|X) \neq 1$	19 and 21	22
So $I(X) \ni Y$ does not hold on r.		

Up to this point we have shown that all IIR of F hold on r but $I(X) \ni Y$ does not. So the instance r that we set out to find has been found. We can now conclude that any $I(X) \ni Y$ that cannot be deduced by the IIR inference rules from F is not logically imply by F. In other words, any IIR that is logically implied by F can be deduced from F by using the IIR rules. This proves that the IIR inference rules are complete.

Even though we have taken the same approach to proving the completeness of the IIR rules as that for the completeness of Armstrong's rules, there are a few fundamental differences between functional dependencies and IIR as shown in Table 2.

3.2 IIR underpin a database

The notion of 'information content' of a state of affairs is essentially the same as that of 'information flow' in the sense that information is carried by a state of affairs in order to flow. We agree with Barwise and Seligman [5, p.4], '*Once one reflects on the idea of information flowing, it can be seen to flow everywhere—not just in computers and along telephone wires but in every human gesture and fluctuation of the natural world. Information flow is necessary for life*.' In this section we explore how IIR underpin a database.

3.2.1 Types of IIR and their sources

A database system is involved with two types of random events: those that are within the database per se, which may be called *database random events*, and those that are in the real world, which could be called *real world random events*. Consequently, there are four types of IIR. The following table summaries the types of IIR and their sources (Table 3).

Table 2 The differences between functional dependencies and IIR

	Functional Dependencies	IIR
Objects concerned	Attributes in a relation	Events—members of power set of outcomes of a selection process
Characterisation of objects concerned (1)	Both random and certain ones are covered	Random
Characterisation of objects concerned (2)	Within a DB	DB and the real world—altogether four types (see Section 3.2)
What is based on	Syntactic characterisation	Syntactic, Semantic, Norms, Business rules…
Veridicality	N/A	The veridicality of event X is a necessary condition for X to be qualified as information being carried

Table 3 Types of IIR and their sources

Information inclusion relation: Information content of X includes Y	Sources
X, Y: both database random events	Syntactic relations between data constructs and data values
X: a database random event; Y: a real world random event	Semantic values and information content of data
X: a real world random event; Y: a database random event	Rules and processes of database design and database operations
X, Y: both real world random events	Relations between real world objects, Business rules

3.2.2 How IIR work for a database

We observe that constructing and using a database to carry and convey information involve all the above four types of IIR.

The IIR between real world random events (Row 4 of Table 1) is concerned with requirements analysis and query writing, among others. For example, in a business rule of 'if a supplier supplies a part and the part is used by a project, then the supplier supplies the project', 'a supplier supplies a part and the part is used by a project' is a random event, denoted say X, and 'the supplier supplies the project' is another random event, say Y. The rule embeds $\mathrm{I}(X) \ni Y$. Furthermore, due to this IIR, we need only embody (carry) X by using data and not Y, as Y can be derived from X. Consequently to query about Y should be implemented by querying X.

The IIR from a real world random event to a database random event (Row 3 of Table 1) at least partly underpins database design. It remains uncertain whether an entity Supplier should be placed in an ER schema until suppliers are identified in the application domain for which the database is designed. Another example would be ad hoc constraints placed on a relation being specified, which would not be certain until some relevant relation between real-world objects is captured.

Row 2 of Table 1, i.e., IIR from a database random event to a real world random event, is concerned with how to interpret data in order to obtain information. For example, in Fig. 1, the connection between node s_1 and node j_1 may convey the information that supplier s_1 supplies project j_1. This is possible only because this connection is a particular of the database random event that entity *Supplier* and entity *Project* is connected, and supplier s_1 supplies project j_1 is a particular of the real world random event that a supplier supplies a project, and there is an IIR from the former to a latter.

Finally, the IIR between database random events indicated by Row 1 of Table 1 appears to be the least understood of all. Such IIR are purely determined by the syntactic characteristics of a database. More precisely, they are fully determined by the *nomic constraints* [12, p.81] of a database. For example, in the path shown in Fig. 1, let σ_1 be the connection between node entity *Supplier* and node entity *Part*, σ_2 entity *Part* and entity *Project*, and σ_3 entity *Supplier* and entity *Project*, there is a nomic constraint $\sigma_1, \sigma_2 + \sigma_3$, which means that σ_1 and σ_2 conjunctively entails σ_3. A constraint captures what information flows [5, p.29], and therefore there is an IIR: $\mathrm{I}(\sigma_1 \cap \sigma_2) \ni \sigma_3$.

We observe that IIR now provides a framework for reasoning about a database in order to obtain information. For example, let θ_1 be a real world random event that a supplier supplies

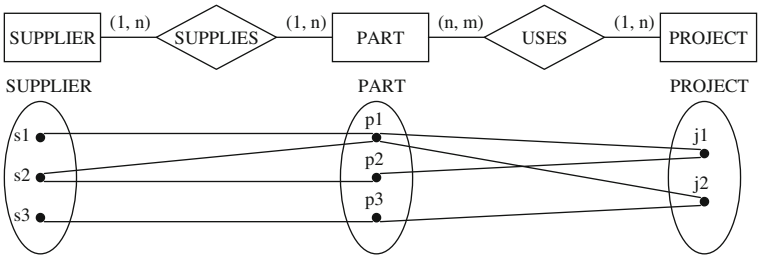

Fig. 1 A path in an ER schema

a part, θ_2 a part is used by a project, and θ_3 a supplier supplies a project. To obtain supplier s_1 supplies project j_1 (which is a particular of θ_3), either of the two chains of IIR is used:

1) $I(\sigma_1 \cap \sigma_2) \ni \sigma_3$, $I(\sigma_3) \ni \theta_3$, through transitivity, we get θ_3.
2) $I(\sigma_1) \ni \theta_1$, $I(\sigma_2) \ni \theta_2$, through augmentation and union, we get $\theta_1 \cap \theta_2$, then apply $I(\theta_1 \cap \theta_2) \ni \theta_3$, we get θ_3.

This example is simple and straightforward, but we hope that the reader is convinced that the ideas presented in this paper so far represent a promising start of a theorization of the notion of information content, and that its use in databases can lead to a sophisticated formulation of the elusive notion of the information content of a database [6].

3.2.3 Meaning versus information content of a data construct

Information that data bear and convey is often confused with and taken as the meaning that data may have, which we observe is the main obstacle that hampers a scientific study of information for databases. We clarify the difference between them in this section.

> Let f be a relation between objects in the real world, which may or may not be true.
> Let e be a data construct, which can be defined as a node or a path in a graph. For example, e could be, in Fig. 1, the nodes s_1, p_1 and the connection between them, which is termed 'supplies'.

If f can be 'read off' directly by following some interpretation rule (called 'semantic rule' by Shimojima 1996) without any effort such as reasoning [38, p.21] from e then f is in the *primary meaning* of e [43]. For example, the primary meaning of e is that suppler s_1 supplies part p_1. Following the rule for interpreting an ER diagram in the notations shown in Fig. 1, the data constructs shown in this diagram have a 'type' of primary meaning that a supplier supplies a part, and a project uses a part as compared with the meaning of an individual data construct.

If under certain conditions on both data and the part of the real world with which the data are concerned, such as the structure and constraints of a data schema, on top of what can be read off directly from the data, f can be derived from e, that is beyond the primary meaning, then f is part of the *implied meaning* of e [43]. For example, the data constructs in Fig. 1 are capable of giving the meaning that a supplier supplies a project if there exists a business rule that 'if a supplier supplies a part and the part is used by a project, then the supplier supplies the project'.

Note that the meaning of an instance, such as the individual entities and links between them shown in the lower half of Fig. 1, of a data construct is given by its type. Types are

captured by the data schema, and types are concepts [13, p.214]. Data instances that follow the schema inherit the meaning of their respective types. This is due to concepts being capable of giving meaning to their instances [13, p.222]. Some relevant interpretation rule is then applied to data constructs whereby meanings of them are produced.

The possible meanings of a data construct are not necessarily part of its *information content*. Information must be contingently true [18], but meaning does not. Suppose that f is part of the *meaning* of e, only if it is also a particular of some real world random event say Y with which a database random event say X, of which e is a particular, has an IIR, does f qualify as part of the information that e bears and conveys. The IIR would make sure of the veridicality [5, p. 10] required. The meaning of a data construct may happen to be part of its information content. But this is only accidental, not essential.

4 Exploiting IIR-reasoning in a database setting

4.1 The architecture of a standalone system

To explore how the ideas of IIR and the rules for reasoning about IIR presented above may be implemented and made use of for a database, we have developed a prototype of a standalone system, which works with a database and a number of other elements as shown below. The architecture of the prototype is shown in Fig. 2.

As shown in Fig. 2, we propose that a reasoning process about IIR consist of two major steps: Clause Conversion and Reasoning. In the Clause Conversion step, we convert the ontology, the database, relevant business rules and IIR inference rules into Prolog clauses, which become either 'facts' or 'rules'. These generated Prolog clauses are then reasoned about by the Prolog Inference Engine[3] whereby hidden and nested information is derived.

IIR-Reasoning provides us with a mechanism exploiting the notion of information content inclusion relationship in a database setting, which helps reveal information that is carried by the data in a database and yet is hidden in the sense that it is normally not accessible by queries by using standard query languages such as SQL.

Notice that in the literature, most database-to-Prolog conversions [1,30,31,36] are more concerned with the Prolog-based representation of individual data set than the Prolog-based representation of IIR between them. Their approaches focus on the direct projection between database elements and Prolog predicates. Our conversion looks at the IIR between database elements and converts the IIR to Prolog predicates. This procedure is guided by ontology and users' perspective.

4.2 Prolog clauses conversion

4.2.1 Converting ontologies

Ontologies are useful because they encourage standardization of the terms used to represent knowledge about an application domain [24] or refer to a wide range of formal representations from taxonomies and hierarchical terminology vocabularies to detailed logical theories describing a domain [35]. Therefore, adding ontologies to our approach is useful for deriving more new facts, namely new IIRs. For the first step, there are many known Ontology-to-*Prolog* translators, which can deal with different ontology representation

[3] It is included in Visual Prolog that is downloadable from http://www.visual-prolog.com.

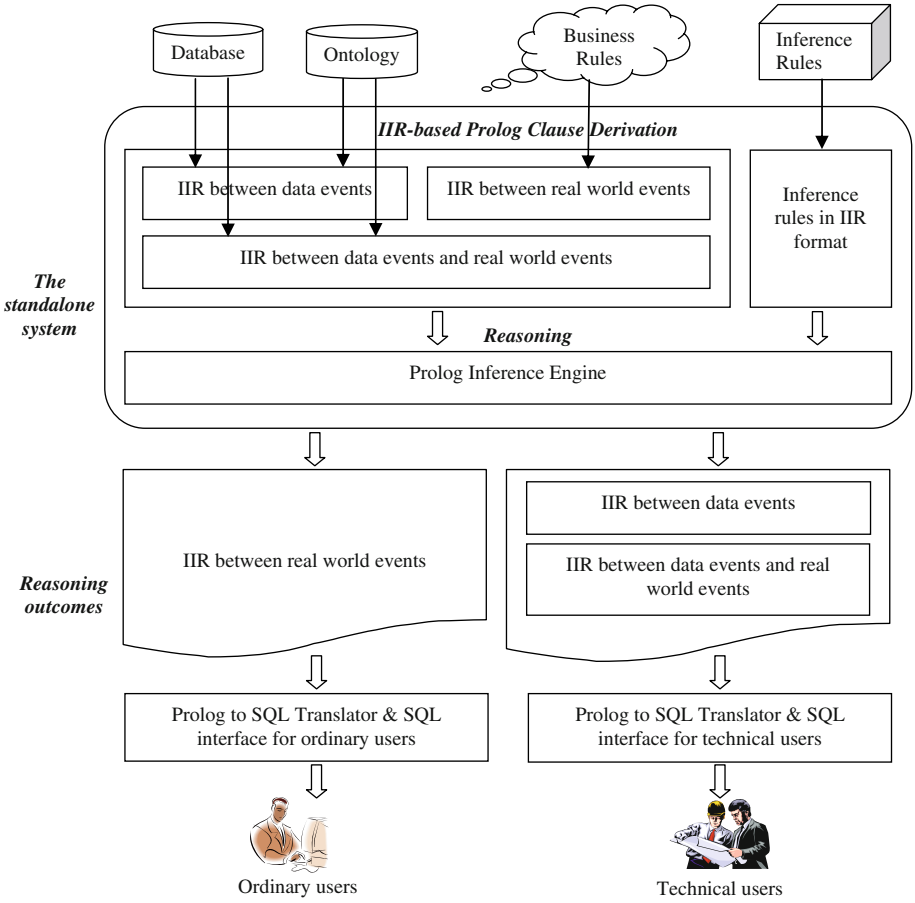

Fig. 2 The architecture of a standalone IIR-based system

languages such as *RDF* [26], *Ontolingua* [15] and *Protégé* [20]. For example, there exist *RDF*-to-*Prolog*, *Ontolingua*-to-*Prolog* and *Protégé*-to-*Prolog* translators. In our implementation, we developed an in-house conversion tool that partially translates the employee ontology written in *OWL* [40]. In the literature, some *Ontology*-to-*Prolog* translators (for instance, the translator used in *IF-Map* [25]) ignores constructs such as documentation slots, template slots and own slots used in *KIF* [19] or *Ontolingua*. The reason is that the absence of these constructs from the translated codes does not invalidate their meanings [25]. In our architecture, these constructs are still useful for deriving hidden and nested information, and therefore they must be captured by our system by representing them as IIR. This is legitimate because IIR is suitable for representing the information inclusion relation between two objects whatever forms these two objects have adopted and wherever they exist in the ontology.

4.2.2 Converting databases

The database conversion is divided into two parts: automatic conversion and manual conversion. Automatic database conversion translates the basic database elements, for example,

instances, into Prolog fact clauses. In the prototype, this automatic conversion is done explicitly: a production version would perform this step only in principle, and would integrate the Prolog reasoner with the database. Firstly, we query the databases by *SQL*. Secondly, we translate the result of a query, i.e., a data set into Prolog fact clauses in the form of iir(x, y), which implements I(x) ∋ y and denotes that the information content of particular x includes another particular y. In such an expression, x and y are particulars of random events in the database, namely particulars of the types of individual data constructs or a collection of individual data constructs, each of which happens to meet some conditions. In the Manual conversion step the system administrator converts those relevant facts selected from a database according to the user's perspective [39].

The notion of random event in databases can be defined as:

A random event is a set of outcomes A that denotes the occurrence of a set of values in a certain data construct in a database with the probability P.

For example, a set of values $\{v_0, v_1, v_2\}$ happens to appear in attributes StudentID, Grade, Course in table CourseResult respectively, and the probability for this to happen is (say) P_0. This is a random event (say) A_0 in the database.

The notion of *a particular of a database random event* can be defined as:

An individual occurrence of a set of values in a data construct, for which the appearance of the set of values in the data construct has been defined as a random event in a database.

For example, that the set of values {'001', 'A', 'Programming' } happens to appear in a tuple in table CourseResult at a certain time t_i and location l_i, which is part of a certain *database state*, is a particular of the above database random event A_0.

Here is an example of automatic conversion. Suppose we have a table Employee (Emp_no, Emp_name, Emp_age, Emp_gender, Emp_tel), which contains some personnel information (Table 4).

After querying this table, we convert the results into Prolog fact clauses in the form of IIR(x, y) as follows:

```
iir(Employee, Emp_no_is(0001)).
iir(Employee, Emp_name_is(0001, Jack Smith)).
iir(Employee, Emp_age_is(0001, 29)).
iir(Employee, Emp_gender_is(0001, male).
. . .. . .
```

Table 4 A table Employee(Emp_no, Emp_name, Emp_age, Emp_gender, Emp_tel)

Employee				
Emp_no	Emp_name	Emp_age	Emp_gender	Emp_tel
0001	Jack Smith	29	male	01065940656
0002	Morag Black	24	female	01081196443
0003	David Brown	60	male	02159832889
. . .	. . .	. . .	. . .	. . .

4.2.3 *Converting particulars of 'Relevant' database random events using the notion of 'Context-awareness'*

In the process of manual conversion, someone could generate a fact clause like this:

$$\text{IIR}(\text{Employee}, \text{tel\_starts\_with}(0001, 010)).$$

This illustrates a problem about translating particulars of relevant database random events and their *granularity*. This is because in addition to atomic data values and 'normal' data constructs such as 'entity', 'tuples', and combinations of these that are useful from the user's perspective, infinitely many other (redundant) combinations of data constructs can give rise to IIR. For example, a particular of a relevant database random event formulated by IIR (Employee, (Emp_no_is(0001), Emp_name_is(tony_wu), Emp_tel_is(0001, 01087534245))) means that:

There is a tuple in table *Employee*, whose information content includes:
A data value *0001* appears in the Emp_no column of the tuple.
A data value *tony_wu* appears in the Emp_name column,
A data value *01087534245* appears in the Emp_tel column.

Our solution to this problem is to follow the idea of *context-awareness*[12]. It is within the *context* of user requirements that information and/or services are relevant to the user, and therefore should be considered. That is, the process of converting database random events is *context-aware*. In the literature, there are approaches to dealing with context-aware behaviours by representing *user information context* with different tools such as ontologies [39] or by offering a framework for the development of context-aware applications [12].

In [39], the user interacts with the ontology by selecting concept nodes that are relevant to their information needs and by deselecting concept nodes that are not relevant. Concept nodes are selected where they contribute to information content (what can be learned from them), whereas concept nodes that are redundant are deselected, thus selection of concept nodes is positive evidence whereas de-selection is a kind of negative evidence. The user context is represented as a pair of elements: $c_i =< P, N >$ where P is a term vector for an element that represents positive evidence and N negative evidence. Operations *min* and *max* are used to represent the *meet* (Greatest Lower Bound) and the *join* (Least Upper Bound) operations and induce a concept lattice whose elements represent various combinations of concepts in the original concept hierarchy. The *min* and *max* operation is the extension of set intersection and set union operation to vectors, respectively.

$$P = n_1 \wedge n_2 \wedge \cdots \wedge n_k, N = n_1 \vee n_2 \vee \cdots \vee n_k$$

The nodes that are involved in selection and de-selection may not be in the ontology. New nodes can be added into this concept lattice by user's interactions. This approach shows a way of representing the user information context as an extension of the concept hierarchy that is maintained and updated incrementally, based on user's interactions with concepts in the ontology.

Figure 3 shows an example following the idea of [39] on representing the user information context for the above database example. To find out what the user information context is, an original ontology of the relevant domain is used. Users can select and deselect the nodes in it to show what they are relevant or unnecessary (respectively). The selected nodes will be treated as positive evidence by *min* operations whereas the deselected nodes will be treated as negative evidence by *max* operations. For the example of Table 2, assume that the user is interested in the telephone number starting with '010' or '021'.

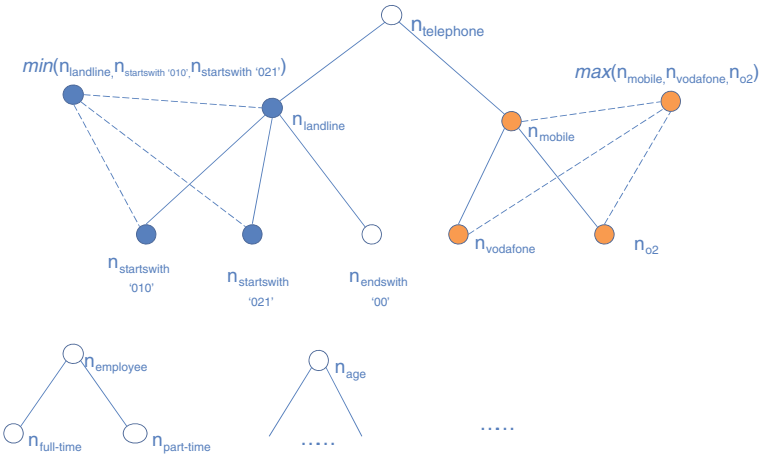

Fig. 3 An example on the selection and de-selection of nodes

Then we derive facts according to the context and if needed using additional business rules to derive more information through reasoning. A user may be interested in various nodes in the original ontology such as position, age of an employee. As shown below, an information context can be described as $c =< P, N >$. $P = \min(m_1, m_2, \ldots, m_k)$. $N = \max(n_1, n_2, \ldots, n_k)$, where $m_1 \ldots m_k$ are user-selected concept nodes and $n_1, \ldots, n_k$ are user-deselected concept nodes.

In order to derive the fact clause within the user context $c =< P, N >$, the following condition has to be satisfied. That is, assume X_i, Y_i are individual concepts of the particulars x, y in a clause iir(x, y),

$$\text{The clause iir}(x, y) \text{ is within } c \text{ if and only if } P \ni X_i, Y_i \text{ and } N \not\ni \{X_i, Y_i\}.$$

With our running example, only facts related to 'landline' and telephone numbers starting with '010' or '021' are taken from the database and translated into IIR clauses.

4.2.4 Converting business rules

Similar to ontologies and databases, business rules and inference rules can also be converted into Prolog clauses. For example, a business rule 'A person lives in Beijing if her/his telephone number starts with "010"' can be described as IIR that the information content of a real world random event 'the telephone number of A starts with "010" includes another real world random event 'A lives in Beijing', namely:

$$\text{iir}(\text{tel\_starts\_with}(\text{A}, 010), \ \text{livesinBeijing}(\text{A})).$$

If there were a database random event that code '010' appears in attribute Telephone, then there would be an IIR between this and tel_starts_with(A, 010). This shows the link between a data random event and a real world random event through IIR. Through such a link, information about real world random event (which is what really matters) is conveyed to the user by means of some database random event(s).

Another example of business rules is 'A person A is retired if her/his age exceeds sixty', which can be written as a Prolog rule clause:

$$\text{iir(age\_exceeds(A), is\_retired(A))}.$$

Note that in the above two IIR expressions, the operands are random events, not their particulars. That is to say, for the sake of implementing information content inclusion relations (i.e., IIR), we use iir(X, Y) to represent $\mathrm{I}(X) \ni Y$, the meaning of which was given in Sect. 3.1.

4.2.5 Converting inference rules

Following the same idea the inference rules can be translated as:

Sum Rule:

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

Product Rule:

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

Augmentation Rule:

iir(A, B) :- product((W, X), A), product((Z, Y), B), product((C, Z), W), iir(X, Y).

Union Rule:
iir(X, S) :- product((Y, Z), S), iir(X, Y), iir(X, Z).

Decomposition:
iir(X, A) :- product((A, B), S), iir(X, S).
iir(X, B) :- product((A, B), S), iir(X, S).

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

'sum' and 'product' are two predicates. 'sum((A, X), Y)' means that Y is the random event which is formed by the disjunction of A and X. 'product((W, X), A)' means that A is the random event which is formed by the product of W and X.

As we mentioned in Sect. 3, three of the six rules are independent and 'basic' rules in that they cannot be derived from the rest of the rules. In our experimentation, we found that there is no difference between using three basic rules and using all six rules.

Part of the pseudocode for the conversion algorithm is as follows:

```
//– – – – – – – – – – – –Database conversion– – – – – – – – – – – –
For each table T in a database
  //Extract IIR among grids in each table
  For each possible set of tuples t_i in T
    For each possible set of attributes a_i in t_i
      Extract IIR between a_i and t_i, namely iir(t_i, a_i).
    End for
    For each possible pair of attribute sets a_i and a_j in t_i
```

Extract information content inclusion relationship between a_i and a_j, namely iir(a_i, a_j).
End for
End for
End for
For each possible pair of attributes a_i and a_j between two tables t_m and t_n in a database
Extract IIR between a_i and a_j, namely iir(a_i, a_j).
End for
//– – – – – – – – – – – –Ontologies– – – – – – – – – – – – conversion
For each possible pair of objects (i.e., nodes, instances) o_i and o_j among ontologies
Extract IIR between o_i and o_j, namely iir(o_i, o_j).
End for

4.3 Inference engine

In our implementation, we use *PIE* (Prolog Inference Engine) as the inference engine. It uses the standard Prolog syntax and releases the most popular set of predicates.

Here is an example of our implementation. Assume that t is a table in a database. In t there are two employee records e_1 and e_2 whose telephone number starts with '010'. Following the informational relation captured by the above IIR expression, we get:

iir(t, telstarts(e_1, 010)). 1
iir(t, telstarts(e_2, 010)). 2

1 and 2 are two IIR between two particulars of random events, which shows the information content of a data item t includes another data item telstarts(e_1, 010) (telstarts(e_2, 010) respectively).

Assume that there is also a business rule that 'A person lives in Beijing if her/his telephone number starts with "010"', which links a real world random event (i.e., a person lives in Beijing) and a database random event (i.e., telephone number recorded starting with '010'). Better still, there should be another IIR linking the database random event that the data value in attribute Telephone Number starts with '010' and the real world random event that someone's telephone number starts with '010'.

To test our example, we raise a *goal* (a term in Prolog) of 'Who lives in Beijing?' The converted clauses (including the Transitivity rule) are as follows:

1. iir(telstarts(A, 010), livesinBJ(A)). 3
2. iir(X, Z) :- iir(X, Y), iir(Y, Z). 4

The 3 above is an IIR between two real world random events that represents the above business rule and 4 the IIR Transitivity inference rule.

To make these clauses acceptable by our inference engine and enable them to be terminated by the inference engine, we have to slightly change them to:

iir(t, telstarts(e_1, 010)).
iir(t, telstarts(e_2, 010)).
iir(telstarts(A, 010), livesinBJ(A)).
startiir(X, Y) :- iir(X, Y).
startiir(X, Z) :- iir(X, Y), iir(Y, Z).

Through inference, the system gives the following answers to our question (i.e., the goal):

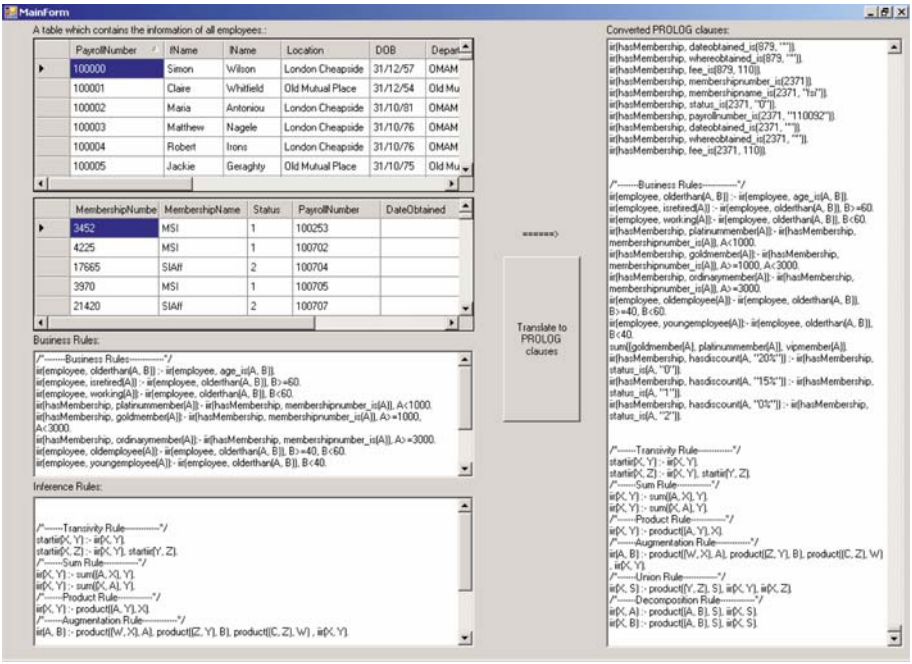

Fig. 4 A prototype written in C#

startiir(t, livesinBJ(A))
$A = e_1$
$A = e_2$
2 Solutions

4.4 A Prototype based upon C#

We developed a prototype for Prolog clause conversion, which is based upon *C#*. In the example, it derives facts and rules from different sources and converts them into the form of IIR (see Fig. 4). After having copied the derived Prolog clauses to the Prolog Inference Engine, we can consult them in the engine and derive more information by queries that cannot be answered without our inference mechanism (see Figs. 5, 6). We tested the prototype against a number of real world databases such as a small-scale employee database (Access-based) and a medium-scale finance database (Oracle-based). These real world databases contain various data types, attributes and relationships. The IIR in the databases in which the user is interested were extracted by the prototype. In the prototype, some slight changes are made to the Transitivity rule to suit the needs of Prolog programming. The test shows that the inference procedure works as expected.

As shown in the example above, a query 'who are retiring?' gets no answer without importing other rules. After adding a business rule 'A data construct X tells us that an employee A is retiring if her/his age B equals or is greater than 60'. We obtained from the prototype that 315 employees in total were retiring by raising a query 'iir(employee, isretired(A))' in *PIE*.

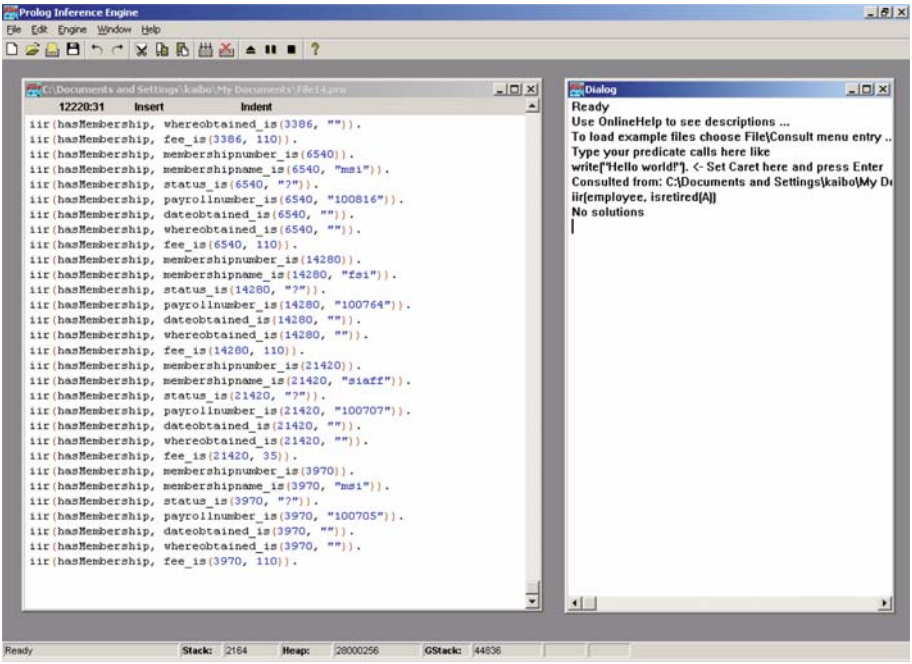

Fig. 5 Reasoning on facts only

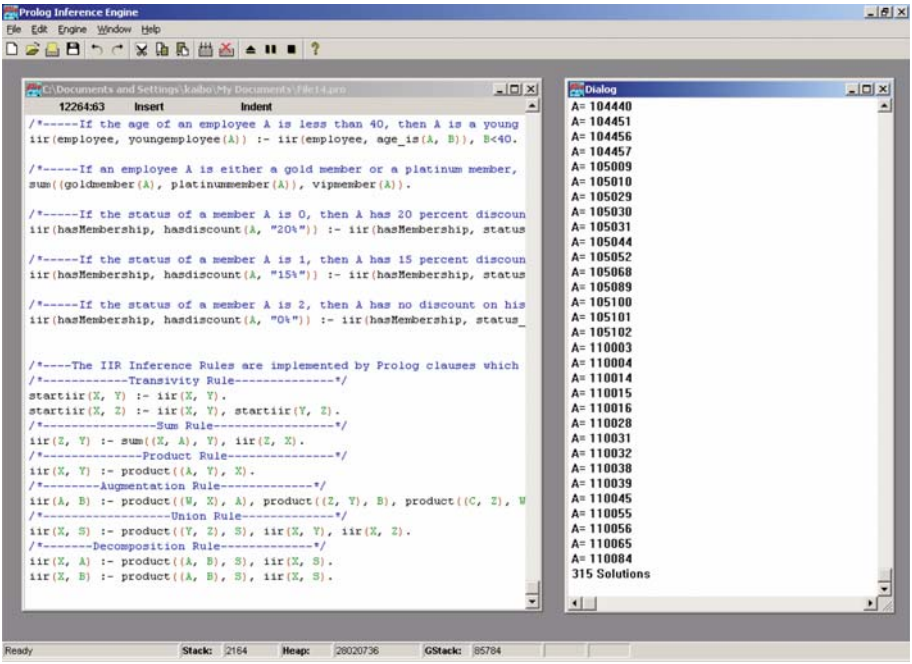

Fig. 6 Reasoning with both facts and additional inference rules

The test results match what we expect of the system. For example, assume that a user were provided the extra knowledge that an employee is retired if her/his age equals or is greater than 60, then the user could execute a SQL query 'SELECT Employees.* FROM Employees WHERE 2007 - YEAR(DOB) >= 60', which would give the same 315 results. Note that our system does not require the user to have this extra knowledge, as this knowledge has already been captured as a business rule in the system.

Another example is as follows:

If we want to find out who in the Marketing Department has become a VIP member and whether she/he wants to keep her/his VIP membership, then we submit to Prolog Inference Engine a query: iir(employee, dept_is(A, 'Marketing')), iir(hasmembership, wannakeepvip(A)).

To answer this query, a number of business rules are used, which are:

/*– – – – – Business Rule 1 – – – – –*/

/*– – – – – If a member has a membership number which is greater than 1000 and less than 3000, then this member is a 'gold member'. – – – – –*/

iir(hasMembership, goldmember(A)) :- iir(hasMembership, membershipnumber_is(A)), A>=1000, A<3000.

/*– – – – – Business Rule 2 – – – – –*/

/*– – – – – If an employee A is either a 'gold member' or a 'platinum member', then A is a 'VIP member'. – – – – –*/

sum((goldmember(A), platinummember(A)), vipmember(A)).

/*– – – – – Business Rule 3 – – – – –*/

/*– – – – – If an employee A is a VIP member, then A has 200 pounds voucher on her/his training. – – – – –*/

iir(hasMembership, hasvoucher(A, '200 pounds')) :- iir(hasMembership, vipmember(A)).

/*– – – – – Business Rule 4 – – – – –*/

/*– – – – – If a member A has a 200 pounds voucher, then she/he must attend a general training course. – – – – –*/

iir(teachesworkshoprun, mustattend(PAYROLLNUMBER, 'general')) :- iir(hasMembership, hasvoucher(A, '200 pounds')), iir(hasMembership, payrollnumber_is(A, PAYROLLNUMBER)).

/*– – – – – Business Rule 5 – – – – –*/

/*– – – – – If her/his identity is recorded in table teachesworkshoprun and in the record the value for attribute workshop is 'general', then an employee has attended a general training course. – – – – –*/

iir(teachesworkshoprun, attended(PAYROLLNUMBER, 'general')) :- iir(teachesworkshoprun, workshopname_is(PAYROLLNUMBER, 'general')).

/*– – – – – Business Rule 6 – – – – –*/

/*– – – – – If she/he must attend and has attended a general training course, then an employee wants to keep her/his VIP membership. – – – – –*/

iir(hasMembership, wannakeepVIP(PAYROLLNUMBER)) :- iir(teachesworkshoprun, mustattend(PAYROLLNUMBER, 'general')), iir(teachesworkshoprun, attended(PAYROLLNUMBER, 'general')).

Our test shows that the reasoning procedure for the above query is consistent with what we expect in terms of how facts are used and how rules fire. For the above example, the trace monitor of *PIE* shows the reasoning procedure as follows (we added some comments):

Query: iir(employee, dept_is(A, 'Marketing')), iir(hasmembership, wannakeepvip(A))

. . .. . .

/*– – – Match A to check whether iir(employee, dept_is(A, 'Marketing')) can be satisfied. – – –*/

Trace: >> CALL: iir(employee, dept_is(_,Marketing))
Trace: >> RETURN: iir(employee, dept_is(110092,Marketing))
/*– – – A = 110092 is found. – – –*/
/*– – – Check iir(hasmembership, wannakeepvip(A)) – – –*/
Trace: >> CALL: iir(hasmembership, wannakeepvip(110092))
/*– – – Check according to Business Rule 6 – – –*/
Trace: >> CALL: iir(teachesworkshoprun, mustattend(110092, general))
/*– – – Check according to Business Rule 4 – – –*/
Trace: >> CALL: iir(hasmembership, hasvoucher(_,200 pounds))
/*– – – Check according to Business Rule 3 – – –*/
Trace: >> CALL: iir(hasmembership, vipmember(_))
/*– – – Check according to Business Rule 2 – – –*/
/*– – – Check according to Sum rule iir(Z, Y) :- sum((X, A), Y), iir(Z, X). – – –*/
/*– – – Check sum((X, A), Y) – – –*/
Trace: >> CALL: sum(_ , _, vipmember(_))
/*– – – sum((X, A), Y) is matched – – –*/
Trace: >> RETURN: sum(goldmember(_), platinummember(_), vipmember(_))
/*– – – Check iir(Z, X) when X = goldmember(_), Z = hasmembership – – –*/
Trace: >> CALL: iir(hasmembership, goldmember(_))
Trace: >> CALL: iir(hasmembership, membershipnumber_is(_))
Trace: >> RETURN: iir(hasmembership, membershipnumber_is(2371))
Trace: >> CALL: 2371 > = 1000
Trace: >> RETURN: 2371 > = 1000
Trace: >> CALL: 2371 < 3000
Trace: >> RETURN: 2371 < 3000
/*– – – iir(Z, X) is matched – – –*/
Trace: >> RETURN: iir(hasmembership, goldmember(2371))
/*– – – Backtrack– – –*/
Trace: >> RETURN: iir(hasmembership, vipmember(2371))
Trace: >> RETURN: iir(hasmembership, hasvoucher(2371,200 pounds))
Trace: >> CALL: iir(hasmembership, payrollnumber_is(2371,110092))
Trace: >> RETURN: iir(hasmembership, payrollnumber_is(2371,110092))
Trace: >> RETURN: iir(teachesworkshoprun, mustattend(110092,general))
Trace: >> CALL: iir(teachesworkshoprun, attended(110092,general))
Trace: >> CALL: iir(teachesworkshoprun, workshopname_is(110092,general))
Trace: >> RETURN: iir(teachesworkshoprun,workshopname_is(110092,general))
Trace: >> RETURN: iir(teachesworkshoprun, attended(110092,general))
Trace: >> RETURN: iir(hasmembership, wannakeepvip(110092))
/*– – – Got the results – – –*/
A = 110092
. . .. . .

Our test results show that the inference rules and business rules help the prototype answer many types of queries that are otherwise impossible. This is because the system derives new IIR from a given set of IIR, which enables hidden information from the database to be revealed.

Through prototyping, we find that the reasoning procedure takes a long time (a few minutes) when a large number of data facts are involved or derived. The translation procedure could be complex. A possible solution to this problem of efficiency is to only derive the data facts that are relevant to the user. There can be redundancies and conflicts in new data facts

derived or reasoned about. A further problem is concerned with the matching order of Prolog clauses. All these will be further investigated in the near future.

4.5 Future work

With the development of the *Semantic Web*, many *RDF*-based tools are becoming available. *RDF* includes the notion of entailment, and has inference rules similar to *Prolog*. Recent work in *SWI-Prolog* [41] includes *RDF* parsers and an interface to *SPARQL* servers. Our group has also developed a novel database architecture, *Pyrrho* DBMS [9], with its own semantic web capabilities, but which also presents database data directly to the SWI-Prolog inference engine, thus avoiding the step discussed in the previous section of importing database data as Prolog assertions.

The production version of the current work envisages combined use of these tools, to realise a part of Tim Berners-Lee's vision of the Web [7] as an almost limitless semantic graph, with many constructs, including relational databases, providing inferable elements of a virtual graph.

In this vision, a semantic query takes its points of reference (similarly to the context-aware notion above) from the terms and vocabulary used in the query, and the namespaces referred to in the current environment. It then uses inference, based on explicit and implicit semantic rules, to navigate through both local and remote repositories of semantic data. Semantic servers supply *RDF* triples on request from inference graphs whose members never need all to be instantiated, and databases supply semantic triples based on their contents, but instantiated only as required by the currently executing query.

We treat the IIR as a minimum atomic unit to be used in an inference engine because this kind of relation under '*Semantic Information Theory*' is easy and acceptable for people to understand and use.

5 Comparisons with related work

Several approaches concerning the 'information content' of databases have been reported in the literature. As we mentioned in Sect. 1, one of the approaches is the theory of relative information capacity (RIC) [23]. [32,33] redefine the notions of *absolute* and *internal dominance* and put forward the Schema Intension Graph (SIG) data model. This RIC- or SIG-based approach is widely accepted and used for measuring the correctness of data schema transformation. The limitation of this approach has been discussed in Sect. 1.

Another approach is based upon Formal Language Theory [29]. The information content of a hypernode database is formalised as the language generated by the context-free grammar, which is defined by Formal Language Theory. This approach is on the data content level because its main point is to execute accurate data value matching when evaluating the equivalence of information content.

To tackle other information content problems such as information content measure [2], information dependency [10], information preserving [27] and information structures [28], Shannon's notion of *entropy* has been used. This notion was widely used in the analysis of the quantity aspect of information such as information relevance [8], the analysis of Neural Network Learning (Ng et al. 2000), cluster analysis [21] and many other areas. [2] define a measure of information content of elements in a database with respect to a set of constraints in terms of Shannon's mathematical information theory [37]. Their basic idea is to use the notion of *entropy* to measure how much information we can gain if the value of an assumed

lost position gets restored. In this way, only the quantity aspect of information is considered. A similar example is in [10]. The notion of *Information Dependency measure* (*InD measure*) characterizes the uncertainty remaining about the values for a set of attributes N when the values for another set of attributes M are known. Their measure is based upon the calculation of entropy also.

It appears that Shannon's entropy is recognized as the most important notion of his theory, and information-theoretic seems to mean 'entropy-theoretic'. But [13] points out that Shannon's entropy is concerned with *a collection of* messages and the *amount* of information involved, and it is not capable of addressing the information *content* of a *single* message. In other words, entropy can be used to measure the quantity of information associated with a random variable, but neither the quantity nor the content of information that a random event carries.

Our notion of 'information content' is based upon [13], which we extended for our purposes by including [5] idea of duality of 'token' and 'type'; it is on individual messages (individual data instances), rather than on many messages (for example, all data instances such as all tuples in a database under study) with which others' works are in fact concerned. They look at the amount of information associated with a set of attributes and measure it by using entropy. Their basic idea would seem to use this concept to formulate data dependencies such as functional, multivalued, and acyclic join dependencies [27,28] by identifying conditions in terms of entropy that are equivalent to the definitions of data dependencies.

In the database setting, individual messages with which our notion of information content is concerned could be various. For example, a set of attributes having a certain set of values is a random event, an instance of which is a particular of it. Similarly individual tuples are particulars of the random events that the whole set of attributes of a relation have various sets of values respectively. IIR formulates the situation where the realisation of one random event can tell us truly about the existence of another. For example,

1) In iir(Employee, Emp_no_is(0001)), 'Employee' is a random event in that a table selected happens to be Employee among other tables, which could possibly be selected. 'Emp_no_is(0001)' is also a random event in that a set of tuples whose employee number happens to be '0001' among other possible tuples.
2) In iir(tel_starts_with(A, 010), livesinBeijing(A)), 'tel_starts_with(A, 010)' is a random event in that a set of tuples whose telephone number happens to start with '010' among other possible tuples. 'livesinBeijing(A)' is a random event in that a person happens to live in Beijing.

An attribute in a relation is, in our notion, an information source, and the average amount of information associated with it, which is called *self-information* cited by [27], is the entropy. The amount of information that an attribute carries about another is termed *mutual-information*, which in our terminology extended from [13], is the average amount of information that is the information that a carrier carries about an information source.

Our focus is on the content of information that is carried by individual messages (individual data instances), not only the (average) amount of information that many messages carry, which is the concern of others' work. This definition is closely related to Shannon's information-theoretic terms, especially 'entropy', in that entropy is the weighted average amount of information (called 'surprisal') that individual messages carry. Surprisal quantifies the amount of information that a message carries, and therefore places constraints on what information (i.e., the information content) a message can carry. Our definition of the information content of a state of affairs satisfies the requirements on both 'amount' and 'content' of information.

Therefore, our notion may be seen as extending others' work that applies information theory to database problems into a fundamentally different dimension in terms of the following:

- Not only the quantity (amount) of information that data carry, but also the content of information that data carry are addressed.
- Not only the statistical characterisation of data's carrying information, but also how individual piece of data carries information are addressed.
- Above all though, what we are concerned with is ultimately the provision of an account for the phenomenon that data in a database carry and convey information to the user, which includes the relationship between data and what the data can tell us truly about the real world. We do not concern ourselves with dependencies between data constructs for their own sake. One example of those works is how 'information-lossless' decomposition of a relation into several sub-relations can be achieved by preserving mutual information between data.
- To this end, we look at IIR between data constructs, which may be seen stemming from one of the inclusion-exclusion identities put forward by [27], namely $\mathrm{H}(Y|X) = 0$ (i.e., Entropy of Y given X), which means that once X is known, Y carries no uncertainty. That is, X carries all the information about Y. Our extensions to this notion are: (1) we look at individual data values, rather than all data values in X put together. For example, we take $X = a$ as a random event and look at what its information content might be; (2) we look at how one individual data value carries information about another data value, and moreover we look at how one individual data value carries information about individual objects in the real world, and vice versa. We envisage that existing work on information-theoretic relationships between data constructs can be incorporated into our overall framework for looking at the phenomenon of 'information' within databases.
- In order to ground our endeavour on sound theories, not only Shannon's ideas, but also ideas in the program of semantic information and information flow, which may be seen started from Dretske [13] and includes Barwise and perry [4], Devlin (1991), and Barwise and Seligman [5], are drawn on and Dretske's ideas are extended.

The main extension to Dretske's notion of 'information content' of a state of affairs that is reported in the paper is as follows: apart from these known entropy-based measurements, Barwise and Seligman put forward a mathematical model called 'information flow' as an account for the laws that govern how and what information flows within a distributed system. In this paper, we extend (which we believe is necessary) Dretske's notion of information content by including the ideas of the duality between 'types' (one kind of which is random events, which we address in this paper) and 'tokens' (one kind of which is particulars of random events, which we address in this paper) that is put forward by Barwise and Seligman [5].

6 Conclusions

In this paper, we have investigated how the notion of information content of a message, a state of affairs, or a data construct may be used to explore the information carried by data in a database. We have proposed to define the notion of 'information content' by following the approach represented by Dretske's [13] semantic theory of information and complementing it by drawing on the ideas of the duality of 'tokens' (particulars) and 'types' (random events) put forward by Barwise and Seligman [5]. We have thus extended the classic definition of information content by Dretske [13] by incorporating the role that particulars play and consequently the role that the random events (types) play in it. We hope that this makes the notion

more accurate and more comprehensive than Dretske's. Based upon this extended definition of information content, we have defined the 'information content inclusion relation' between two random events, and then formulated a set of inference rules for reasoning logically about such relations.

Then we showed how these ideas and rules might be applied within a database setting including distinguishing the information content from the linguistic meaning of a data construct. We described how reasoning about information content might be exploited with a database by experimenting with a prototype, which includes, among other things, the architecture of the prototype, the use of the idea of 'context-aware' systems and the exploitation of relevant ontologies and business rules. We showed that our approach is a general one. Similar to Armstrong rules [3], inference rules are domain-independent, which can be applied to application domains. No interpretation of the rules that are specific to an application domain is needed. Original sets of IIR (i.e., those derived from database facts and business rules) are domain-dependent, the identification of which would require domain knowledge.

Our main conclusions through this work are: the program of semantic information theory represented mainly by the work of Dretske [13], Barwise and Seligman[5], and Devlin [11] is insightful and helpful in addressing the problem of information content and how it fares in information systems and databases. Based on these sources, our work offers the machinery for a systematic approach to the problem, which can bring benefits in identifying previously inexplicit business information in databases.

Acknowledgments This work is partly sponsored by the Chinese BMEC (Beijing Municipal Education Commission) under the grant number KM200311417153 for the Project of 'The Research on Information Bearing Capability'. The second author is also supported by a Senior Visiting Fellowship from the Royal Academy of Engineering, UK, 2005 and a grant for Distributed Information Systems Research from the Carnegie Trust for Universities of Scotland, 2007.

References

1. Amble T (1987) Logic programming and knowledge engineering. Addison Wesley, Reading
2. Arenas M, Libkin L (2005) An information-theoretic approach to normal forms for relational and XML data. J ACM 52:246–283
3. Armstrong WW (1974) Dependency structures of database relationships. In: Proceedings of IFIP 74, North- Holland Pub. Co., Amsterdam, pp 580–583
4. Barwise J, Perry J (1983) Situations and attitudes. MIT Press, Cambridge
5. Barwise J, Seligman J (1997) Information flow: the logic of distributed systems. Cambridge University Press, London. ISBN 0-521-58386-1
6. Batini C, Ceri S, Navathe SB (1992) Conceptual database design—an eitity–relationship approach. The Benjamin/Cummings Publishing Company, Inc., USA
7. Berners-Lee T (1998) What the semantic web can represent, http://www.w3.org/DesignIssues/RDFnot.html. Accessed Jan 2007
8. Chachoua M, Pacholczyk D (2002) Qualitative reasoning under ignorance and information-relevant extraction. Knowl Inf Syst 4:483–506
9. Crowe MK (2007) The Pyrrho database management system, computing and information systems technical reports, 38, University of Paisley, UK. http://www.pyrrhodb.com
10. Dalkilic MM, Roberston EL (2000) Information dependencies.In: Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems, Dallas, Texas, USA pp 245–253
11. Devlin K (1991) Logic and information. Cambridge University Press, London
12. Dey AK, Abowd GD (2000) Towards a better understanding of context and context-awareness. In the Workshop on The What, Who, Where, When, and How of Context-Awareness, as part of the 2000 Conference on Human Factors in Computing Systems (CHI 2000), The Hague, The Netherlands, 3 April 2000.
13. Dretske FI (1981) Knowledge and the flow of information. Blackwell, Oxford

14. Duží M (2001) Logical foundations of conceptual modelling. In: VŠB-TU Ostrava.
15. Farquhar A, Fikes R, Rice J (1997) The ontolingua server: a tool for collaborative ontology construction. Int J Hum Comput Stud 46(6):707–727
16. Feng J (1998) The 'Information Content' problem of a conceptual data schema, SYSTEMIST, vol 20, No.4, pp 221–233, ISSN:0961-8309
17. Feng J, Crowe M (1999) The notion of 'Classes of a Path' in ER schemas. In: Proceedings of third East European Conference on advances in databases and information systems, ADBIS'99. Springer, Berlin.
18. Floridi L (2005) Is semantic information meaningful data. Philos Phenomenol Res 70(2):351–370
19. Genesereth MR, Fikes RE (1998) Knowledge interchange format (KIF). Draft proposed American National Standard, NCITS. T2/98-004
20. Grosso W, Eriksson H, Fergerson R, Gennari J, Musen M (1999) Knowledge modelling at the millennium (the design and evolution of Protégé-2000). In: 12th Workshop on knowledge acquisition, modeling and management, Banff, Alberta, Canada.
21. Hu T, Sung S (2006) Finding centroid clusterings with entropy-based criteria. Knowl Inf Syst 10(4):505–514
22. Hu W, Feng J (2002) Some considerations for a semantic analysis of conceptual data schemata. In: Ragsdell E et al (eds) Systems theory and practice in the knowledge age. Kluwer Academic/Plenum Publishers, New York. ISBN 0-306-47247-3
23. Hull R (1986) Relative information capacity of simple relational database schemata. SIAM J Comput 15(3):856–886
24. Jurisica I, Mylopoulos J, Yu E (2004) Ontologies for knowledge management: an information systems perspective. Knowl Inf Syst 6:380–401
25. Kalfoglou Y, Schorlemmer M (2003) IF-Map: an ontology mapping method based on information flow theory. J Data Semant I. Lecture Notes in Computer Science 2800, pp 98–127
26. Lassila O, Swick R (1999) Resource description framework (RDF) model and syntax specification. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/. W3C Recommendation.
27. Lee TT (1987a) An information-theoretic analysis of relational databases—part I: data dependencies and information metric. IEEE Trans Softw Eng 13(10):1049–1061
28. Lee TT (1987b) An information-theoretic analysis of relational databases—part II: information structure of database schemas. IEEE Trans Softw Eng 13(10):1061–1072
29. Levene M (1998) On the information content of semi-structured databases. Acta Cybern 13:257–275
30. Li D (1984) A Prolog database system. Research Studies Press, New York
31. Lucas R (1988) Database applications using Prolog. Halsted Press, New York
32. Miller RJ, Ioannidis YE, Ramakrishnan R(1993) The use of information capacity in schema integration and translation. In: Proceedings of the International Conference on very large data bases, Dublin, Ireland, pp 120–133
33. Miller RJ, Ioannidis YE, Ramakrishnan R (1994) Schema equivalence in heterogeneous systems, bridging theory and practice. Inf Syst 19(1):3–31
34. Ng G, Chan K, Erdogan S, Singh h (2000) Neural network learning using entropy cycle. Knowl Inf Syst 2:53–72
35. Noy N, Klein M (2004) Ontology evolution: not the same as schema evolution. Knowl Inf Syst 6:428–440
36. Nurcan S, Kouloumdjian J (1991) An advanced knowledge base management system based on the integration of logic programming and relational databases. In: Proceedings of IEEE, CompEuro '91, Bologna, Ital, pp 740–744
37. Shannon CE (1948) A mathematical theory of communication. Bell Sys Tech J 27:379–423,623–656
38. Shimojima A (1996) On the efficacy of representation. Ph.D. Thesis, The Department of Philosophy, Indiana University
39. Sieg A, Mobasher B, Burke R, Prabu G, Lytinen S (2005) Representing user information context with ontologies. In: Proceedings of the 3rd International Conference on universal access in human–computer interaction, Las Vegas, NV
40. Smith MK, Welty C, McGuinness DL (2004) OWL web ontology language guide. http://www.w3.org/TR/2004/REC-owl-guide-20040210/. W3C Recommendation
41. Wielemaker J (2007) SWI-Prolog semantic web library, HCS, University of Amsterdam. http://www.swi-prolog.org. Accessed on Jan 2007
42. Wille R (1997) Introduction to formal concept analysis. In: Negrini G (ed), Modelli e modellizzazione. Models and modelling. Consiglio Nazionale delle Ricerche, Instituto di Studi sulli Ricerca e Documentazione Scientifica, Roma, 39-51
43. Xu H, Feng J (2002) Towards a definition of the 'Information Bearing Capability' of a Conceptual Data Schema. In: Systems theory and practice in the knowledge age. Ragsdell E et al (eds) Kluwer Academic/Plenum Publishers, New York. ISBN 0-306-47247-3

Author Biographies

Kaibo Xu is a Ph.D. candidate in the School of Computing at the University of the West of Scotland (the UWS). He received his B.Sc. in Computer Science from the Beijing University of Chemical Technology, China and his M.Sc. in Information Systems Development from the UWS. He works as a Lecturer at the Business College of Beijing Union University, China. His interests include semantic information theories and systems, database theory and systems.

Junkang Feng graduated from the Institute of Military Engineering of the People's Liberation Army, China. He received his M.Phil. from the University of Portsmouth, UK and Ph.D. from the University of the West of Scotland (the UWS) UK both in Information Systems. He worked as a Research Associate in the Department of Computer Science at the University of Manchester, UK before becoming a Lecturer and then Senior Lecturer at the UWS. He currently heads the Database Research Group of the UWS. His interests include qualitative information and information flow theories, distributed information systems and database theory.

Malcolm Crowe changed from an early career in Mathematics (B.A.Mod. Dublin 1969 and D.Phil. Oxford 1978), gaining a Chair in Computing at Paisley College in 1985, and is now a senior academic in the School of Computing at the University of the West of Scotland. He has published public-domain software tools for the .NET framework: for programming languages, network management and database management. His interests extend from semantic information systems to web technology, and his current project is in open graphic editing. His most recent book (with John Atkinson) is on Interdisciplinary Research (Wiley 2006).