I.J. Information Technology and Computer Science, **2012, 1, 50-63**
Published Online February 2012 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2012.01.07

Using Logic Programming to Represent Information Content Inclusion Relations

Doug Salt
Database and Semantic Web Group, School of Computing, The University of the West of Scotland, Paisley, Scotland
Email: douglas.salt@uws.ac.uk

Junkang Feng
Database and Semantic Web Group, School of Computing, The University of the West of Scotland, Paisley, Scotland
Email: junkang.feng@uws.ac.uk

Abstract— **Datalog is a widely recognised language for a certain class of deductive databases. Information Content Inclusion Relation (IIR) formulates a general, information theoretic relationship between: data constructs; between data constructs and real world objects, and between real world objects. IIR is particularly concerned with the information that data carry. It would therefore seem desirable to find out whether IIR and reasoning based on IIR may be implemented by using *'safe'* Datalog. We present and prove the following theorem:**

Any database system that can be modelled using IIR can be represented as a 'safe' Datalog program.}

This paper explores the nature of the relationship between the two frameworks for representing domains of application, in order that such representations of IIR by `safe' Datalog can then be used as a tool for the analysis of any site that can be approached with the notion of information content, and in particular any given database, and hence how a database works may be approached in terms of information content of events.

Index Terms— **Databases, Information theory, models and principles, Information technology, Programming languages.**

1. Introduction

When a database is queried, the assumption is that we obtain information from it. The query can only be answered by a precise match between the query and the data within a database [1]. Thus, we have a match on the syntactic nature of information [2], but have no definite relationship with the semantic level of the information, inasmuch as such a query only tells us about the symbols in the database. It may possibly tell us something about the domain of application, but this is by no means certain. For instance, a database may contain entities, and a grid reference has a link to an entity that indicates that at that grid reference, it is raining, say as a current measure of precipitation. To a cognitive agent, this also includes the information that the ground at the grid-reference is wet. We cannot currently query the database asking if the ground at that grid-reference is wet, we may only query whether precipitation occurs at this grid reference. This is an example of 'other' information, besides that of the primary meaning of what is carried in the database.

We take a 'semantic externalist' view of information [3] and we assert that data within a database may carry such information [4], and information inclusion content relations (IIR) can be used to describe the relationship between these pieces of information that the data in a database may carry [5], and the domain of application. IIR is domain independent, and is a model of the informational relationship between components of a site/system. We developed the notion of IIR [5] with a main assumption that information is generated by a reduction in uncertainty. This follows [6] and [7] and is further refined in [5] and [8] That information is generated by an occurrence of an event, and this event may tell us truly of some other event [5].

Datalog are programs, consisting of a subset of Prolog syntax, which are used to define rules and facts declaratively, and in turn is used to derive new facts from the database of such facts [9]. 'Safe' Datalog is Datalog that has only positive sub-goals and is 'safe', that is, its variables are limited to finite ranges [10, p. 67]. 'Safe' Datalog may always be represented as an existentially unqualified, first-order Horn clauses [10] and because its first order variables are limited to finite sets and as such are necessarily are a subset of first order predicate logic (FOL).

The idea is that a finite probability space can be modelled using a Datalog database, and by invoking a particular query this acts as selecting the occurrence of a particular event in the probability space, upon which the concept of IIR is constructed. Using the declarative deduction of Datalog, we can then model the inference laws for IIR (see section 2, and determine what the consequences of that event are, or as we term it the 'closure' for that event. That is, which other events that an occurrence of a single event can tell us truly. Thus, if the above relationship is mathematically proven, then we have a rigorous tool for modelling IIR in all situations.

Received November 9, 2011; revised January 17, 2012; accepted January 30, 2012.
Corresponding author: Douglas Salt.

In particular, it is hoped that this will provide a tool to model both the logic of a database, and the domain of application in informational terms. With such a tool, we may be able to uncover reasoning at a distance for such representation systems [11], [12, Ch. 20].

The structure of this paper is as follows. We give a brief definitions of IIR, how we believe that IIRs interact in a database, and then a brief description of 'safe' Datalog. This will allow us to define the terms used in the above proposition. We then give a theoretical justification of the above theorem. This is where the main contribution that our work makes is justified. We illustrate the above theorem with some examples of IIR relationships represented with 'safe' Datalog. Finally we draw conclusions and propose further work to complement and extend the findings of this paper.

2 Definition of Information Content Inclusion Relations

2.1 IIR and IIR Inference Rules

The following arguments are based on [5]. In this paper, they managed to derive, and prove domain-independent rules for the manipulation of information flow based on probability theory. We take the most of the following definitions from this paper.

Following the terminology of [6, ch. 1] we define the term of 'random selection process' ((selection process) for short), as a set of conditions and outcomes. For instance, the roll of a die would be a random selection process, whose outcomes are in the set {1, 2, 3, 4, 5, 6}, and the conditions would be that the die is fair (equal probability of rolling any number).

Definition 2.1. Let s be some *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 an event if $E \ni X$ and there is a probability of X, i.e. $P(X)$.

Definition 2.2. Let s be some selection process under a set C of conditions, O the set of possible outcomes of s, E the power set of O and let P: $E \rightarrow [0,1]$ be the probability measure, such that: $P(O) = 1$;

$\forall X_i \subseteq E$ for $i = 1, \ldots, n$ if X_i is a countable collection of pairwise disjoint sets then $P(\sqcup X_i) = \Sigma P(X_i)$ where '$\sqcup$' denotes the disjoint union; then the triple (O,E,P) is the *probability space*.

Definition 2.3. Let s be a selection process under a set C of conditions, X_i an event concerning s, x_i an instance of s, x_i is a *particular* of X_i if x_i is in state Ω, written $\Omega = \text{state}(X_i)$ and $X_i \ni \Omega$.

Definition 2.4. Let s be a selection process the result of which is reduction of possibilities, and therefore be an information source, and k prior knowledge about s;

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

Let s's being F be an 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, if and only if the conditional probability of s's being F given r is 1 (and less than 1given k alone).

This last definition makes more explicit the definition of information content in [6, p. 65], whereas unlike in the original formulation, and as [5] point out, Dretske deals with information content at a type level whereas information is carried only by particulars. The prior knowledge in this case is not easily quantified, but represents an amendment to the state space of the source, represented by the receivers prior knowledge (see [7] for more details on this concept). For example consider the three cup game, in which a pea is under one of three cups on a table. One of the cups has already been shown to the people already present not to have the pea under it, and the cup has been replaced on the table. Some further participants now arrive after this event. To those already present they know the pea to be under only one of two cups, whereas the new arrival only knows that the pea is under any of the three cups. So the state space is altered by prior knowledge. Such prior knowledge represents the relativisation of information content.

Definition 2.5. 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.

An example of this would be: 'if it is raining on my head at 1500 on the 2nd March 2011', then this includes the information content that 'my hair is wet at 1500 on the 2nd March 2011'.

Definition 2.6. Let X and Y be an 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.

Extending the last example, this gives 'if it is raining on my head' then this includes the information content 'my hair is wet.' Notice it is the particulars that actually carry the information, but IIR allows us to predict the behaviour of the information content of such particulars at type level.

Definition 2.7. Let X be an event, the *information content of X*, denoted $I(X)$, is the set of events with each of which X has an information content inclusion relation.

Sufficient conditions for $I(X) \ni Y$ are:

- Both X and Y are events, namely they could be contingently true, or contingently untrue, but are neither necessarily true nor necessarily false. Mathematically $P(X) \neq 1$ and $P(X) \neq 0$ and $P(Y) \neq 1$ and $P(Y) \neq 0$
- Whenever X is true Y is true. That is $P(Y|X) = 1$, in other words $X \subset Y$.

Given the above definitions [5] went on to prove the following inference rules for events, which are sound and complete.

Theorem 2.8 (Sum). *If* $Y = X_1 \cup X_2 \cup \ldots \cup X_n$, *then* $I(X_i) \ni Y$ *for* $i = 1, \ldots, n$.

Theorem 2.9 (Product). *If* $X = X_1 \cap X_2 \cap \ldots \cap X_n$, $Y = X_i$ *for* $i = 1 \ldots, n$ *then* $I(X) \ni Y$.

Theorem 2.10 (Transitivity). *If I(X) ∋ Y,I(Y) ∋ Z then I(X) ∋ Z.*

Theorem 2.11 (Union). *If I(X) ∋ Y,I(X) ∋ Z then I(X) ∋ Y ∩ Z.*

Theorem 2.12 (Augmentation). *If W = W_1 ∩ W_2 ∩… ∩ W_n, Z is the product of a subset of $W_1,W_2,…,W_n$, I(X) ∋ Y then i(W ∩ X) ∋ Z ∩ Y .*

Theorem 2.13 (Decomposition). *If I(X) ∋ Y ∩ Z then I(X) ∋ Y,I(X) ∋ Z.*

In the above theorems $X_1,X_n,Y,Z,W,W_1,W_2,W_n$ are all events in *E*.

Although these rules are sound and complete, it should be noted that [5] have already mentioned that these rules are not all independent of one another. They mention that theorem 2.13 may be proved with the use of theorems 2.8 and 2.9.

Furthermore we find that (theorem 2.12) may be proved as follows:

Proof

$W = W_1 \cap W_2 \cap \ldots \cap W_n$	Premise	1
$I(X) \ni Y$	Premise	2
$W \cap X$	Assumption	3
Z is a factor of *W*	Premise	4
$I(W) \ni Z$	Theorem 2.8 applied to 4	5
$I(W \cap X) \ni Z \cap Y$	Theorem 2.10 applied to 2,3 and 5	

■

Therefore, in any discussion of the inference rules for IIR, it is sufficient only to consider IIR itself, reflexivity and inference rules, i.e., from theorem 2.8 - theorem 2.11 inclusive.

2.2 IIR underlying 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 [12, 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 show how IIR support a database and also may be seen as explaining how a database may provide the user with information. That is, IIR may formulate an information theoretic view of databases.

2.2.1 Types of IIR and their sources

In this section we show how IIR support a database. A database can be involved with two different types of events: those that are internal to the database which could be termed *database events*, and those that are in the real world, which will be called *application domain events*. Consequently, there are four types of IIR shown in Table 1 which summarise the types of IIR and their sources.

Table 1 Types of IIR and their sources (based on [5])

Information Inclusion Relation: I(X) ∋ Y	Sources
X, Y : both database events	Syntactic relations between data constructs and data values
X: a database event; Y : an application domain event	Semantic values and information content of data
X: an application domain event; Y : a database event	Rules and processes of database design and database operations
X, Y : both application domain events	Relations between real world objects, Business rules

2.2.2 How IIR work for a database

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

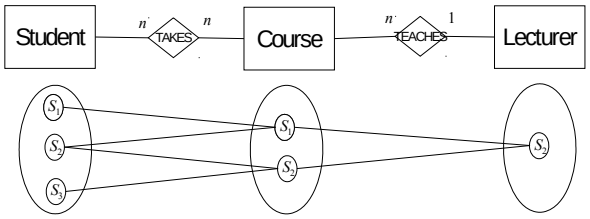

Figure 1: A Path in an ER schema

The following discussions are based on the ER diagram in Fig. 1.

IIR as shown in row 1 of Table 1 are purely the result of the syntactic characteristics of a database such as rules for the structure and data integrity of a database. That is, they arise as a direct result of the nomic constraints [11] within a database. To illustrate, consider the path shown in Figure 1, let α_1 be the connection between node entity Student and node entity Course, α_2 entity Course and entity Lecturer, and α_3 entity Student and entity Lecturer, then there is a nomic constraint α_1, $\alpha_2 \vdash \alpha_3$, which means that α_1 and α_2 in conjunction entail α_3. Such a constraint captures information flow [12, p. 29], and consequently there is an IIR between some combination of α_1 and α_2 having an information content inclusion relation with α_3 (In fact, in our notation, this may be represented by: $I(\alpha_1 \cap \alpha_2) \ni \alpha_3$ - see section 2).

IIR now provides a framework for reasoning over a database in order to derive information. For example, let β_1 be a real world event that a student takes a course, β_2 the course is taught by a lecturer, and β_3 a student is taught by a lecturer. To obtain that student s_1 is taught by lecturer t_1 (which is a particular of β_3), either the following IIR may be used:

- $I(\alpha_1 \cap \alpha_2) \ni \alpha_3, I(\alpha_3) \ni \beta_3$, so through Transitivity (i.e., theorem 2.10), we derive β_3, or alternatively,
- $I(\alpha_1) \ni \beta_1, I(\alpha_2) \ni \beta_2$, through Augmentation (i.e., theorem 2.12) and Union (i.e., theorem 2.11) we derive $\beta_1 \cap \beta_2$, and from this, we derive $I(\beta_1 \cap \beta_2) \ni \beta_3$, resulting in β_3.

The above chains of reasoning show how the type of IIR of row 2 of Table 1 work, namely we need an IIR from α_i to β_i (among others) for the reasoning to work. Thus, IIR from a database event to an application domain event pertains as to how data is used within the database to derive information. This kind of IIR may also involve so called semantic values of representations

[11], which are concerned with an application domain. For instance in the above relationship, at the syntactic level, the intersection of α_1 and α_2 (i.e., the product of the two events) has α_3 in its information content from which a cognitive agent can derive the information in the application domain that a particular lecturer teaches a student.

The IIR from application domain event to a database event is involved in database design. It would remain unclear as to whether an entity Student should be placed in an ER schema until the students are identified in the application domain for which the database is designed. A further example could be constraints placed on a relation, which will not be obtained until some relation between objects in the application domain is captured. This type of IIR is described in row 3 of Table 1.

The IIR between application domain events could be concerned with requirements analysis and query writing, etc. Say for instance we have a business rule of 'if a student takes a course that a lecturer teaches, then the lecturer teaches the student' , 'a student takes a course and the course is taught by a teacher' is an event, denoted say *X*, and 'the student takes the course' is another event, say *Y* . Thus the information content of *Y* is already in *X*. Furthermore, due to this IIR, we need only embody (carry) *X* by using data and not *Y* , as *Y* can be derived from *X*. This elaborates on row 4 of Table 1. In addition, this level of IIR contains constraints of the real world. For instance, a particular lecturer teaches a particular student, generally carries such information, such as the lecturer is qualified to teach the course, or we have a particular by whence such an IIR would make sense, such as this is not before the lecturer or student was born.

As can be seen from an earlier part of this section there are several inference rules available for deriving further IIR from a given set of IIR, the most important of these being the transitive property of IIR. It is these inference rules that we wish to model with Datalog.

2.2.3 Information and meaning of a data construct

It is our belief that meaning is often incorrectly taken as synonymous with information that a particular datum may carry. Such ambiguity, we feel, hampers the analysis of information when related to a database. In this section we will attempt to make a clear distinction between the two.

Let *f* be some relation between objects in the application domain, which is either true or false, but which cannot always be true, or always false, that is, it is *contingently* true. Let *e* be a particular data construct, such as a node or a path in a graph. Using the above example in Figure 1, then *e* would be the nodes s_1, c_1 and an edge between them, labelled, *takes*.

If f can be comprehended by some 'semantic rule' (terminology borrowed from [11]) without any additional inference (ibid., p. 21) from *e* then *f* is in the *primary meaning* of e [13]. Again, to illustrate with the above example. The primary meaning of e is that student s_1 takes the course c_1. If we assume the accepted rules for following the meanings of ER diagrams is like that in Figure 1, then the data construct shown therein has a 'type' of primary meaning that a student takes a course, and a lecturer teaches a course as opposed to the actual meaning of an individual data construct.

If under certain conditions on both data and the part of the application domain with which the data are concerned, such as the structure and constraints of a data schema, on top of what can be comprehended directly from the data, that is *f* can be derived from *e*, then this is beyond the primary meaning and f is part of the *implied meaning* of *e* (ibid.). For example, the data constructs in Figure 1 are capable of giving the meaning that a student is taught by a lecturer if there exists a business rule that 'if student takes a course and the course is taught by a lecturer then the lecturer teaches that student'. This implies that the meaning of such a construction of entity, relation, entity is given by its type. Types are modelled by the data schema, and types arise from what Dretske terms concepts [6, p. 214]. Hence any data instance that fit these data constructs inherit the meaning of their corresponding types. Meaning arises from these concepts and thus concepts give meaning to these instances (cf. [6, p. 222]). Some relevant interpretation rule in the application domain is then applied to such data constructs thereby giving them meaning. Using the ideas above, then the meanings of a data construct does not necessarily need to be contained in its information content. We follow the idea that information must be contingently true [4], but this does not necessarily apply to meaning. Say f is part of the meaning of e, only if it is also a particular of some application domain event, say Y with which a database 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 [12, p. 10] required. The meaning of a data construct may happen to be part of its information content, however such conditions for meaning to be part of the information are neither necessary, nor sufficient.

3 Purpose of Investigation

The aim of this paper, is to discover whether there is any kind of linkage between the two frameworks described, and furthermore what the nature of this linkage or correspondence is. We believe that there must be some kind of linkage. On the basis that 'safe' Datalog is a subset of First Order Predicate Logic (FOL) then Datalog can successfully represent certain aspects of an application domain. That is, at one level, it is a representation system. There must exist constraints between the database and the domain of application it is trying to model. However, this representation, when looked at from the purely physical viewpoint is at best just variances in electrical charge within the internals of some computing device [14, p. 156]. We define these electrical variances with values of 0 or 1, and in turn use those symbols to represent other symbols. Hence a computing device is merely a contrivance for manipulating symbols (ibid.), and hence computing

devices do not exhibit cognition. Eventually these symbols can be used to represent aspects of the real world. It is this linkage via these constraints to the real world by this accumulation and refinement of such symbols that we believe information content can arise. IIR is a proven way of modelling an application domain in terms of information content. As we have two frameworks, both of which can represent an aspect of the real world, then it seems logical that there must be some kind of correspondence between them. This paper is part of a series which tries to uncover this assumed correspondence.

'Safe' Datalog also represents, as stated before a particular subset of FOL. FOL, and in particular one of the two inference rules of FOL [15, p.86]; the *modus ponens*, underlies the vast majority of computing [14, p. 156]. By proving such a correspondence then we must be close to determining why computing devices are so useful in modelling the real world, and what, therefore is the informational justification of the modus ponens, that is a possible informational theoretical justification of FOL

We note there are other implementations used when analysing databases in terms of IIR. We already know we can model IIR using Prolog [8] and Oracle PL/SQL [16], [1], [17] and [18]. This would seem to support the above claim that there is some link between the fundamentals of computing and an information theoretic representation of such systems.

Additionally as seen in the above papers, we can model IIR using existing computing frameworks. It would therefore, be useful to possess yet another form of representing such relationships, which has been formally proven to be able to represent any given set of IIR. This would allow further investigation of IS in terms of IIR using a proven tool.

Finally, IIR itself is a framework that describes information content of a given state of affairs and relationship therein. As we are able to successfully model IIR, and IIR is not necessarily tied to an IS, and furthermore we can model such IIR with several implications, it would seem that IIR itself must represent some kind of generalised framework, with which to undertake informational analysis of virtually any situation that can be represented in probabilistic terms.

4 Brief Definition of 'safe' Datalog

Although Datalog has a relatively simple syntax, in this paper we will only be considering 'Safe' Datalog, that is Datalog that does not contain any negative sub-goals and is safe [10, pp. 67-68] [19]. One of the reasons for this, is that there are many implementations of Datalog available, which extend Datalog to include negative sub-goals and disjunction [10, p. 119] [20]. Throughout this paper we make use of a limited set of the functionality of the Datalog Educational System [21] .We therefore attempt to define, briefly, the Datalog we wish to use to model IIR. We now define the syntax of the Datalog we in which we are interested.

Datalog is a subset of Horn programs, which are suitable for application to databases. Datalog's semantics are based on FOL [22]. Note, all following logic program definitions are derived from [10] and [19]. Essentially Datalog predicate logic consists of a subset of *formulae*. A formula is a set of predicates called *atoms*, or *atomic formulae*. These are of the form:

$p(a_1,\ldots,a_n)$

where p is the predicate name, a_1, a_2, …, a_n are constants or variables, known as *terms* and n is termed the *arity* of the predicate. The collection of predicates, constants and variables is referred to as the *basis*. In particular the basis we are considering contains no function symbols. A formula consists of series of atoms connected using standard logical connectives. It can be shown that any formula can be converted to the *clausal form* [9, pp. 734-735]. So if $p_1, p_2, \ldots, p_n, q_1, \ldots, q_m$ are a series of atoms, then a clausal form of the formulae is:

$\neg p_1 \vee \neg p_2 \vee \ldots \vee \neg p_n \vee q_1 \vee \ldots \vee q_n$ where all atoms are assumed to be universally qualified. Each qualified atom in the clause is referred to as a *sub-goal*. Note this structure can be transformed using standard FOL axioms, and may give rise to *Horn clauses* (or *definite clauses*, and *general Horn clauses* which are a subset of the clausal form.

A clause $p_1 \vee \ldots p_n \leftarrow q_1 \vee \ldots \vee q_m$ is called a:

- *Horn clause* if n is less or equal to one and every sub-goal is an atom;
- *general Horn clause* if n is greater than 1 and every sub-goal is an atom.

Definition 4.1. A *positive logic program*, P, is a set of Horn clauses.

Definition 4.2. r be a general Horn clause, a variable V of r is called *limited* if

- V appears in an ordinary positive sub-goal, or
- V is equated to a constant c in a built-in sub-goal, e.g. $V = c$ or $c = V$, or
- V is equated to a limited variable W in a built-in sub-goal, e.g. $V = W$ or $W = V$.

r is called *safe* if each variable of r is limited. A positive logic program is *safe* if all it clauses are safe.

Definition 4.3. A positive logic program is called a *'safe' Datalog program* if

- its basis does not contain any function symbols, and
- it is safe.

4.1 'Safe' Datalog syntax

What follows is a précis of the descriptions found in [9], [19] and [10].

The following conventions are adopted to indicate syntax:

- [ ] indicates optional;
- … indicates one or more, but finite;
- | indicates 'either …or …'.

If none of the above are present, then this indicates that the object is mandatory.

We now need to detail the following structures:

- constants;
- variables;
- terms;

- built-in predicates;
- conditions;
- facts;
- rules;
- programs.

4.1.1 constants

These are of the form:

lower case letter[any alphanumeric character | underscore _]…| Number

So examples of constants are:

```
a
x
constant
cOnstant
cONSTANT
c12939
1
-1
2E10
1.5
1.314E-34
```

Such constants will be denoted as lower case single letters, a, b, c,… or a_n, b_n, c_n, …for $n \in \mathbb{N}$ in subsequent definitions.

4.1.2 variables

These are of the form:

upper_case_letter| underscore _ [[any alphanumeric character | underscore _]…

So examples of variables are:

```
X
Y

_
_aVar
Variable
VARIABLE
VariaBle
V123
```

Variables will be denoted as upper-case letters A, B, C, …or A_n, B_n, C_n for $n \in \mathbb{N}$ in the following definitions.

4.1.3 terms

Terms may be either variables or constants. This will be denoted as emboldened lower case a, b, c,… or a_n, b_n, c_n,… for $n \in \mathbb{N}$ in the following definitions.

4.1.4 built-in predicates

Because we are only using 'safe' Datalog, then the only predicate available is the equality. This takes the form:

```
a = b
```

Examples of these are

```
doug = 10
X=Y
2=3
X=doug
size=doug
```

Conditions will be denoted as emboldened upper case A, B, C, …or A_n, B_n, C_n, for n $\in \mathbb{N}$ in subsequent definitions.

4.1.5 predicates

These are of the form:

$p(a_1,\ldots,a_n)$ where $n \geq 0$.

Examples of predicates are:

```
ancestor(titus, A).
student(doug).
supervises(doug, dr_feng).
supervises(shuang, Who).
```

Predicates will be denoted as lower case single letters p, q, r,… or p_n, q_n, r_n,… for $n \in \mathbb{N}$ in subsequent definitions. If p, or p_i is stated without accompanying terms, then the terms are assumed, that is p is shorthand for $p(a_1,\ldots,a_n)$.

4.1.6 facts

These are of the form.

$g(a_1,\ldots,a_n)$ for $n \geq 1$.

Examples of facts are:

```
true.
employee(doug).
ancestor(titus,doug).
```

Notice that facts are a specialisation of predicates. Indeed facts are known as ground predicates.

Facts will be denoted as lower case single letters g, h, i,… or g_i, h_i, i_i,… for $i \in \mathbb{N}$ in subsequent definitions. If g, or g_n is stated without accompanying terms, then the terms are assumed, that is g is shorthand for $g(a_1,\ldots,a_n)$.

Also notice in the first example the predicate without arguments invariably returns false (i.e. no tuples computed), and is not much use for querying. We do not consider it within this paper.

4.1.7 rules

Rules are of the form:

p:-p_1,p_2,…,p_n,P_1,…,P_m where $m + n \geq 1$, and p must start with a lower case letter.

The right hand side of the rule, is often termed the rule body. Examples of rules are:

```
superior(A,B):-supervises(A,B).
superior(A,B):-supervises(A,X),
     superior(X,B).
smallSize(X):-size(X,Y),X=8.
```

The collection of predicates and conditions (known as sub-goals) in the rule body is conjunctive, i.e. all sub-goals must be satisfied for the rule to be true. A rule may contain itself in the rule body, That is, rules may be recursively defined.

Rules will be denoted as lower case emboldened single letters r, s, t,… or r_n, s_n, t_n,… for $n \in \mathbb{N}$ in subsequent definitions. If r, or r_n is stated without accompanying terms, or associated predicates, then the terms and predicates are assumed, that is r is shorthand for $r(a_1,\ldots,a_l)$: -$p_1(A_1,\ldots,A_m)$,$p_2(b_1,\ldots,b_n)$,…,P_1,,P_2,… for l, m, $n \geq 0$.

Ground rules, i.e. rules with only zero parameters or constants in the rule head, are considered to be facts, and the rule body is not evaluated.

4.1.8 Programs

A collection of rules and facts is referred to as Datalog program. Identically named facts may appear in a Datalog program many times, and this collection of facts is referred to as the Datalog database to which rules are applied, in order to return tuples that satisfy the query entered. Order is important, Datalog will evaluate facts in the order they appear in the program. Similarly the same rule name may appear multiple times (as well as appearing in its own rule body). Again Datalog will evaluate rules in the order they appear. All such occurrences of the rule will be evaluated in an attempt to return tuples.

4.1.9 Querying in Datalog

A program normally resides in a simple text file. To perform a query in Datalog, the Datalog program must be 'consulted', i.e. loaded up into the Datalog environment. Queries are then performed within the Datalog environment. These are just predicates with usually, at least one variable, and are entered interactively in the Datalog environment.

Examples of queries are:

```
ancestor(doug, A).
employee(Anybody).
```

Datalog then returns all matching tuples of the predicate.

5 Theoretical justification of the transformation of IIR to Datalog.

We believe that it is vital to obtain theoretical guidance and justification on our investigation. The basic idea is to utilise the proven approaches found in [10] and [23] of creating an interpretation of a given set of predicates and thence trying to produce a model, i.e. an interpretation which gives answers corresponding to the truth of that set of predicates. In order to do this we strictly define the components that we are working with: the probability space, and the possible set of IIR which will arise from that probability space (but are, as yet, unknown), we then create an interpretation. Using the properties of IIR we construct predicates within our interpretation which reflect these properties. We then go on to show that such predicates may always be constructed if we have the prior conditions set out below. Finally once these structure have been created, we use the fact that 'safe' Datalog is itself an implementation of a subset of FOL (that is, without any negative sub-goals) to prove that a given set of IIR may always be represented by a suitable Datalog program.

Within the bounds of a database we are dealing purely with data. Such databases are alethically neutral [4], that is, not all the semantic value [11] of the instances of data in a database is information about the application domain it is modelling. However, there is an informational relationship between data as row 1 of table 1 illustrates, and thus we can use IIR to reason about this data. To this end, we treat a value of a data construct (i.e., an attribute or a set of attributes) being of a certain value in an instance of a notional database as an event. This is along the same line of [24] [25], [26]. For event B to be in the information content of event A, that is, for there to be an IIR between these two events, denoted I(A) ∋ B, then we must have, by definition:

$P(B|A) = 1$

That is, event B is certain if we have event A. So this may be stated: given event A, then we have the information that event B exists. In terms of an instance of a database, then if we have an instance of event A at a given time, it is certain that we have an instance of event B. This means there is always a link between entity A and entity B.

Definition 5.1. If we have the probability space O,E,P , then there exists a *complete set* of IIR and the relationships between those IIR due to IIR rules, denoted *IIR*. *IIR* includes all IIR that are logically implied by a given set of IIR, due to IIR rules being complete and sound (see [5]).The relationships are determined by reflexivity, IIR inductance theorems 2.8 - 2.11 and simple IIR relations. These rules would be represented as the IIR relation ∋ between set relations.

Definition 5.2. An *evaluated probability space*, W is the tuple $((O,E,P),IIR)$ where (O,E,P) is the probability space as mentioned above and *IIR* is defined as above.

Definition 5.3. A *universe* or *domain of discourse*, D is the tuple (T,R) where T are all the terms of D, and R, are all the rules in D, for manipulating members of T.

Definition 5.4. An *interpretation*, J of domain of discourse D, where U is a non-empty set of elements, called the *universe* (or *domain*) of interpretation, such that:

- For each member T of D, in J, an assignment to element in U, i.e. , $J{:}T\rightarrow U$.
- For each member, R of D, the assignment of an n-arity Horn predicate q in J, being the assignment of a mapping in U^n into true, false (or, equivalently, a relation on U^n)

We now create an interpretation J, for some evaluated probability space $W=(O,E,P),IIR)$ with domain of discourse, E, giving $J{:}E\rightarrow U$.

Definition 5.5. For any event e ∈ E there exists an *closure* for e, denoted e^+, for which $\forall x \in e^+$, I(e)∋x. This closure is created by the repeated application of theorems 2.8 - 2.11.

Definition 5.6. For any two events e_1 and e_2 linked in any way by being members of the same closure then they are said to be *related*. Note such events will appear in some kind of relation in the complete IIR set, *IIR*.

Definition 5.7. A set of terms, T for that particular interpretation of an IIR rule are said to be *linked* if all the terms, representing those events, are related, and this set is denoted T^{IIR}.

Definition 5.8. A *linked predicate*, denoted *iir* is constructed recursively from predicates with linked terms, thus:

$$iir(T) \leftarrow p_1(T_1) \wedge \ldots \wedge p_m(T_m), m \geq 0, T \subseteq T^{IIR}, T_1 \cup \ldots \cup T_m = T^{IIR}$$

where $p_i(T_i)$ is either a fact, or a further linked predicate such that $T_i \subseteq T^{IIR}$, and the arity(T), arity(T_i) ≥ 2

Note the above does not impose order (as these are sets), so the above example could represent, for example, if T, in the above were $T=\{a,b\}$, then this predicate could be either be $p(a,b)$ or $p(b,a)$, both of which could represent I(J^1)(a))∋J^1(b) or I(J^1)(b))∋J^1(a).

Note that with the $iir(T)$, then the set T may be a selection of any of the terms in T^{IIR}.

Definition 5.9. The IIR predicate rule set is said to be *consistent* if and only if $\forall a_m, a_n \in U$ then I($J(a_m)$) ∋ $J(a_n) \Leftrightarrow a_m, a_n \in T^{IIR}$.

Note this could consist of only one IIR predicate, a series of IIR predicates, a single rule, or rules that call other rules, or any combination of the above to complete the IIR predicate set.

Lemma 5.10. *A consistent IIR linked predicate rule set may always be constructed.*

Proof.

Given $e_n \in e_1^+$, then as theorems (2.8) - (2.11), are complete there must exist a path between events e_1 and e_n such that $I(e_1) \ni e_2$, $I(e_2) \ni e_3,\ldots,I(e_{n-1}) \ni e_n$, $I(e_i) \in IIR$.

There may be many such paths. We select the one with lowest order.

Firstly we construct the relationship for e_1 to e_n. This is simply:

$closure_1(T_1) \leftarrow iir_1(T_2)$ where $\{J(e_1),J(e_2)\} \subset T_1 \subseteq T_2 \subseteq T_1^{IIR}$, T_1^{IIR} is the set of linked terms for linked predicate involving e_1 and e_2.

We now construct the remainder of the predicate, inductively, from e_1 to e_p, where $2 < p \leq n$. So we have:

$$closure_p(T_p) \leftarrow closure_{p-1}(T_{p-1}) \wedge iir_p(T_1)$$

where: $J(e_1) \in T_{p-1}$, $J(e_p) \in T_1$, $\{J(e_1),J(e_m)\} \subset T_p \subseteq T_1 \subseteq T_m^{IIR}$ and $T_{p-1} \subseteq T_{p-1}^{IIR}$.

There are two cases we must examine to prove the above constructed predicate is consistent. These are:

- The predicate evaluates to true and there is no closure
- The predicate evaluates to false and there is a closure

Considering the first, that the predicate evaluates to true and there is no closure. By definition, this means that there must exist a sub-goal which is true which contains terms that are not linked, representing events that are not related. However this is a contradiction, as all sub-goals have been constructed from linked terms.

More formally:

If we have constructed $closure_n(T_n)$, where $\{J(e_n),J(e_1)\} \subset T_p$, for $e_n \in e_1^+$, then there exists p, such that $closure_p(T_p) \leftarrow closure_{p-1}(T_{p-1}) \wedge iir_p(T_1)$, $1 < p \leq n$ where some eventual fact of $iir_p(T_1)$ is not linked. However, this is a contradiction as by definition as terms in $iir_p(T_1)$ must be linked, that is for any two terms in T_1, then the event representation of these terms are linked by definition. Additionally since $iir_p(T) \leftarrow predicate_1(T_1) \wedge\ldots\wedge predicate_m(T_m)$, $m \geq 0$ and in particular $T \subseteq T_p^{IIR}, T_1 \cup\ldots\cup T_m = T_p^{IIR}$, then all such terms must be present.

Now considering, that the predicate evaluates to false and there is a closure. This implies there are related events which are not represented by linked terms. However, by definition all linked terms must be present. This is a contradiction, and hence no predicate can evaluate to false if there is a closure.

Again, more formally: since we have the events $e_n \in e_1^+$ and the assumption is that these are linked, then we must have $I(e_1) \ni e_2$, $I(e_2) \ni e_3,\ldots I(e_{n-1}) \ni e_n$. But the assumption is that the predicate is untrue, this means there must be a missing fact in $iir_p(T) \leftarrow predicate_1(T_1) \wedge\ldots\wedge predicate_m(T_m)$, $m \geq 0$, $1 < p \leq n$, but by definition the set T_p^{IIR} must contain all linked terms for that predicate, and hence have a representation for related events - this is a contradiction, so there can never be a false constructed predicate when there is a closure.

■

Lemma 5.11. *A set of consistent IIR predicate rules is always safe.*

Proof.

Replacing all terms $J(e_i)$ in linked iir predicates by V_i then all iir linked predicates are Horn clauses, so by definition are general Horn clauses.

- V_i only appears in an ordinary positive sub-goal, and
- There are no constants in J.

■

Definition 5.12. An interpretation J, which upon evaluation makes all rules in the complete IIR rule set, IIR evaluate correctly is called a *representation of* W . That is $\forall u_i, \underline{u_j} \in U$ then $I(J^I(u_i) \ni J^I(u_j)$ must be true.

Lemma 5.13. *A interpretation J is a representation for W , if and only if the predicate representation of the IIR rule set is consistent.*

Proof.

The IIR predicate rule set is consistent, thus as we can see from lemma (5.10) when for any iir predicate rule is true for J it is true for the representation W. When any iir predicate rule is false for J, it is false for W , hence J is a model for W.

■

Lemma 5.14. *An interpretation J of W that has a consistent IIR predicate rule set is a Horn program.*

Proof.

From (5.11) we see that all iir linked predicates of an interpretation J of W are Horn clauses, and from definition (4.1) we see that the set of iir linked predicates is therefore a Horn program.

■

Definition 5.15. An interpretation J that makes true all rules in P is called a *model* for P .

It should be noted that any positive program P has been shown to have a model [19, p. 193].

Definition 5.16. A model M for a program S is said to be its least model, denoted $lms(S)$ if $M' \supseteq M$ for every model M' of S. Because S is a safe 'safe' Datalog program, then there always exists a least model, $lms(S)$ for S (see [10] and [19]). This least model consists only of series of facts, i.e. predicates of arity one or above.

Theorem 5.17. *Any database system that can be modelled using IIR can be represented as a 'safe' Datalog program.*

Proof.

Lemma (5.13) shows we can construct a representation of W within interpretation J if and only if the iir predicate rule set is consistent. Lemma (5.10) shows we can construct this consistent rule set. Lemma (5.14) means that J is a Horn program, and hence may be represented as a program in 'safe' Datalog.

■

5.1 How to construct IIR from 'safe' Datalog

We note that this is only an existence theorem, and does not actually show how to construct a given Datalog database to represent the given set of IIR. To do this we must prove each of the four inductance theorems 2.8 - 2.8 can be represented in 'safe' Datalog

We now show one possible way such a set may be constructed.

5.1.1 Sum

This refers to theorem 2.8

If $Y = X_1 \cup X_2 \cup \ldots \cup X_n$, then $I(X_i) \ni Y$ for $i = 1, \ldots, n$.

One way to do this in Datalog is the following:

```
iir(X,S) : -sum(S,X,X1,…,Xn).
iir(X,S) : -sum(S,X1,X,…,Xn).
□ □
iir(X,S) : -sum(S,X,…,Xn,X).
```

Where `S` is a fact (or derived fact) , representing a sum of arity $n + 1$, and the first parameter of the fact names the sum.

Proof.

Let $Y_1, X_1, \ldots X_n$ be events is E. We now represent the union of $X_1, \ldots, X_n$ as the Datalog fact: `sum(s,x1,…,xn)`. The sum is called `s`. We now construct the sum inference rule 2.8 as above. To prove that we have indeed created a sum, then we must have we must have an IIR relationship between $X_1, \ldots, X_n$.

■

We note immediately, that the naming of the sum need not occur in the first argument of the fact, so from this method, we have another n - 1 possible representations of the same set of IIR. Additionally, and possibly more neatly

5.1.2 Product

This refers to theorem 2.9:

If $X = X_1 \cap X_2 \cap \ldots \cap X_n$, $Y = X_i$ for $i = 1\ldots, n$ then $I(X) \ni Y$.

An example of representing the product would be:

```
iir(P,X) : -product(P,X,X1,…,Xn).
iir(P,X) : -product(P,X1,X,…,Xn).


iir(P,X) : -product(P,X1,…,Xn,X).
```

Where `P` is a fact (or derived fact) , representing an intersection of arity $n + 1$, and the first parameter of the fact names the product.

Proof.

We note immediately, that the naming of the product need not occur as the first argument of the fact, so from even from this method, we have another n- 1 possible representations of the same set of IIR.

■

5.1.3 Transitivity

This refers to theorem 2.10

If $I(X) \ni Y$, $I(Y) \ni Z$ then $I(X) \ni Z$.

This may be represented by the 'safe' Datalog program:

$$P_T = \begin{cases} rule(x,y). \\ rule(y,z). \\ rule(X,Z) :- rule(X,Y), rule(Y,Z). \end{cases}$$

Proof.

Let X, Y and Z be events in a probability space E and let $I(X) \ni Y$ and $I(Y) \ni Z$. By the transitivity theorem, 2.10, then $I(X) \ni Z$. Thus the domain of discourse, by definition 5.3 is the tuple $D = (\{X,Y,Z\}, \{I(X) \ni Y, I(Y) \ni Z, I(X) \ni Z\}$. Now let P_T be an interpretation of D, such that $P_T{:}X \rightarrow x$, $P_T{:}Y \rightarrow y$, $P_T{:}Z \rightarrow z$.

By definition 5.5, the closures for X,Y and Z are:

- $Z^+ = \{Z\}$
- $Y^+ = \{Y,Z\}$
- $X^+ = \{X,Y,Z\}$

We now construct the following linked predicates, as per definition 5.8. As consistent rule sets need to be built recursively, (as explained in 5.10 from the smallest set of linked terms, then we take the lowest order set of the closure and begin with that. We can ignore Z^+, as this cannot produce any rules of arity 2 or above. Thus we now have Y^+, which we can select a consistent linked predicate, this being any subset of

We select a single fact from this set: $fact_y(y,z)$, and use this to build the next consistent linked predicate.

We have already chosen to select $fact_x(x,y)$. Given this then, the least model (as in definition 5.16) produced for the program must be something like:

```
{fact(x,y).,fact(x,z).,fact(y,z).}
```

Although again, there are other least models which will equally satisfy our requirements.

The next linked predicate must be a subset of the following (noting that this is 'safe' Datalog and all arguments are limited)

We also note we have named all facts and sub-goals the same. This need not necessarily be the case. So the actual set from which we can choose our model is somewhat greater than that presented above. We are perforce, constrained somewhat by the union of the previous iteration to find the consistent linked predicate, (but not by a great deal), so after rename the facts and rules, we now choose:

```
{iir(y,z).,iir(y,z).,
iir(X,Z):-iir(X,Y),iir(Y,Z).}
```

We note that this predicate rule set is consistent as per definition 5.9, as for every $I(P_T^{-1}(x_i)) \ni P_T^{-1}(x_j) \forall x_i,x_j \in \{x,y,z\}$, hence we have a representation of D as per definition 5.12.

Which gives us the program as above.

■

This rule, is why we believe is why 'safe' Datalog is so suitable for representing IIR. The recursive nature of 'safe' Datalog is amply able in representing the transitive nature of IIR.

5.1.4 Union

This refers to theorem 2.11

If $I(X) \ni Y$, $I(X) \ni Z$ then $I(X) \ni Y \cap Z$.

One way to do this is by doing the following:

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

6 Examples of IIR converted to Datalog

6.1 Closure example

The examples that follow are from [17], but have been slightly modified to exemplify all the IIR inference rules, as in [27], and thus demonstrate how to derive an IIR closure using a Datalog.

It should be noted that [17] uses a modified version of the IIR inference rules. These are equivalent, although the omission of the sum and product, loses some of the generality associated with IIR. The modified versions of the inference rule may be presented as follows.

If X, Y and Z are separate and individual events, in a state space, as defined in section 2 the forms of IIR may be rewritten thus, imitating in the form of the well-known Armstrong's Axioms for functional dependencies in relational databases [5], [28]:

1. **Reflexivity**: $X \rightarrow X \Rightarrow X \rightarrow X$, or using previous notation: I(X) ∋ X.
2. **Augmentation**: $X \rightarrow Y \Rightarrow X \cap Z \rightarrow Y \cap Z$.
3. **Transitivity**: $X \rightarrow Y, Y \rightarrow Z \Rightarrow X \rightarrow Y$.
4. **Union**: $X \rightarrow Y, X \rightarrow Z \Rightarrow X \rightarrow Y \cap Z$.
5. **Decomposition**: $X \rightarrow Y \cap Z \Rightarrow X \rightarrow Y,\ X \rightarrow Y$.

For the purposes of this paper, we use the original versions of IIR mentioned in section 2. That is, we use definition, (2.7) and theorems (2.8 - 2.11). Bearing this in mind, the example we wish to code may be represented by the following set of IIR:

I($A \cap B$) ∋ C
I(C) ∋ A
I($B \cap C$) ∋ D
I($A \cap C \cap D$) ∋ B
I(D) ∋ $E \cap G$
I($B \cap E$) ∋ C
I($C \cap G$) ∋ $B \cap D$
I($C \cap E$) ∋ $A \cap G$
$W = X \cap Y \cap Z$
$M = W \cap A$
$L = P \cup Q \cup A$
$T = B \cap D \cap W$

All the above literals represent events in a probability space as defined above. The reason we use them is because each conjunction or disjunction of an event needs to be named in order to be manipulated by the 'safe' Datalog program.

For the purposes of this paper, we now demonstrate how the constructions in section 5 may be used to construct a combination of a straight-forward IIR and the product rule.

For the example I($A \cap B$) ∋ C we need to represent that the product $A \cap B$ has in its information content C. If we assume the product `pAB` = $A \cap B$, then the original relationship may be rewritten I(`pAB`) ∋ C. Thus we have the linked events `pAB` and C Representing these as a linked predicate gives us

```
iir(pAB, c).
```

We now need to represent the product inference rule as a rule. To do this we must show that the product `pAB` has in its information content A or C. This could be represented as two further predicates

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

If we now represent transitivity as the following, which again maintains the linked events as linked predicates, using:

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

The evaluation of the rule leads to new facts:

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

Which is correct. Or, alternatively the product inference rule could be represented by the rule:

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

The rule, when run in conjunction with the representation of the transitivity rule gives more flexibility as not all the examples of the product then have to be represented. Additionally the statement of the inference rules will lead to other IIR relationships which may not be immediately aware to the person coding the representation. Notice also, the 2nd representation, returns exactly the same results as the former representation, maintaining the linked events as linked predicates.

Bearing this kind of construction in mind, we now take the relationships listed above, and originally used in [27] and code them in Datalog.

```
% facts
% =====

% I(A ∩ B)-> C
product(pAB,a,b).
iir(a,c).
% I(B ∩ C) -> D
product(pBC,b,c).
iir(pBC,d).
% I(A ∩ C ∩ D) -> B
product(pACD, a, c, d).
iir(pACD,b).
% I(D) -> E ∩ G
product(pEG, e, g).
iir(d,pEG).
% I(B ∩ E) -> C
product(pBE, b, e).
iir(pBE, c).
% I(C ∩ G) -> (B ∩ D)
product(pCG, c, g).
product(pBD, b, d).
iir(pCG, pBD).
% I(C ∩ E) -> (A ∩ G)
product(pCE, c, e).
product(pAG, a, g).
iir(pCE,pAG).
% W = X ∩ Y ∩ Z
product(w,x,y,z).
% M = W ∩ A
product(m,w,a).
% L = P ∪ Q ∪ A
sum(l,p,q,a).
% T = B ∩ D ∩ W
product(t,b,d,w).
% Rules
% =====
% Sum...
```

```
% ...for a sum of three parts.
iir(X,S):-sum(S,X,A,B).
iir(X,S):-sum(S,A,X,B).
iir(X,S):-sum(S,A,B,X).
% 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).
% transitivity
iir(X,Y):-iir(X,Z),iir(Z,Y).
% union
iir(X,P):-product(P,Y,Z),iir(X,Y),
     product(P,Y,Z),iir(X,Z).
```

If the program is run with the following query:

```
iir(t,X).
```

It gives the following uncovered IIR (amongst many others involving all the sums and intersections) :

```
{
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.
```

This is identical to the closure generated within [27], although in this paper we have used a different representation this time.

6.2 Student database example

This is the 2nd example from [17] and demonstrates how to derive and implement IIR upon an example of a real-world database. The functional dependencies, used as IIR are encoded as IIR. The reasoning behind this is that each instance of each of the entities below: student; class; and enrolment represent types of events. The instantiation of any of them represents an event, i.e., the instantiated entity of student has the corresponding nested information of student id, surname, major, level (or year) and age. Each instantiation in the database, must obviously have been extant at a given time, place (for instance when we created the database in the Datalog system), so a single instantiation of each entity represents an event. Likewise for the entities class and enrolment. The functional dependencies between instances of these entities may, therefore be modelled as information content belonging to these events. This makes the modelling, using Datalog quite straightforward. The only IIR we need to model are the transitivity property (2.10) and the IIR relationship itself.

So, this may be 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.
```

This is the result as expected, agreeing with the results in [17].

7 Conclusions

We believe that we have proved and demonstrated the following theorem:

Theorem: Any database system that can be modelled using IIR can be represented as a 'safe' Datalog program.

The above theorem is an existence theorem and does not impose much structure on the creation of mappings between IIR to 'safe' Datalog, the positioning of terms and variables in heads of rules or facts are not determined. This leads to a very large number of possible interpretations of IIR in 'safe' Datalog. Is it that such representations are logically equivalent? That is, they might possibly be ways of saying the same thing. Are these sets of statements when converted to FOL, logically equivalent? Indeed, although the theorem above has been proved, this does not guarantee that the methods suggested above are the only ways to encode a given set of IIR as 'safe' Datalog. We see from [27] that the sum rule (2.8) can be modelled as a Datalog disjunction. That is, the individual parts of the sum are modelled as separate rules, and the disjunctive evaluation, which gives the desired effect. Seemingly the crucial component to such representations (and the proof that they are possible), appears to be the transitivity rule (2.10). This is the mathematical formulation of Dretske's Xerox principle [6, pp. 57-58], and seems very similar to the propositional logic's and first or predicate logic's modus-ponens. It appears that this allows the assemblage of linked iir, via linked predicates. It would appear, that given a probability space as defined above, then what we have is some kind of informational content justification of the modus-ponens.

The examples shown in section 6 above have been tried in three sorts of inference engines. Firstly one based on Oracle PL/SQL [8] [29], then against a 'normal' inference mechanism of Prolog [8] , and then finally in this paper, 'safe' Datalog - all leading to the same results. However none of these attempts have been proven to formally represent a site of information content in terms of IIR. We now have a tool, which we know, formally can represent any set of IIR. Thus by modelling a site of information content, and then by invoking a Datalog query, we are effectively modelling the occurrence of that event, and by use of 'safe' Datalog can derive the consequences of that event, in terms of all the other events the occurrence of the original event tells us truly of other events.

We suggest that now we have such a relationship established between 'safe' Datalog and IIR, then 'safe' Datalog itself may be used to perform further, revealing analyses of information systems (IS) that can be represented as probabilistic situations, i.e., an information source [6, p. 4]. Probability spaces have been explored by [25], [26]and [24] as a means to investigate a database. Our efforts attempt to complement such analyses by using the tools developed in [5] as a means to understand the behaviours of information within a site of information content, e.g. a database, but by representing the information within a source of information as a probability space, and hence one of the aims of this paper has been to start to discover whether there is any kind of linkage between the IIR framework and 'safe' Datalog. IIR is a proven way of modelling an application domain in terms of information content. As we have two frameworks, both of which can represent an aspect of the real world, then it seems logical that there must be some kind of correspondence between them. This paper is part of a series which tries to uncover this assumed correspondence.

Additionally 'safe' Datalog, being a subset of FOL implements one of the two inference rules of FOL [15]; the modus ponens,. This underlies Datalog deduction, and indeed the vast majority of modern computing [14, p. 156]. By proving such any correspondence between Datalog and IIR, then we may be close to determining why computing devices are so useful in modelling information content, and by extension, the real world. This may give an information-theoretic justification of the modus ponens, and thus a possible informational theoretical justification of some section of FOL - especially given that all deductions from FOL are certain, and thus by many definitions of information [30, pp. 94-109], must as a consequence contain no information whatsoever.

We note there are other implementations of IIR used when analysing databases in terms of IIR. We already know we can model IIR using Prolog [8] and Oracle PL/SQL [16], [1], [17] and [18]. This would seem to support the above claim that there is some link between the fundamentals of computing and the modelling of such systems using information content. As seen in the above papers, we can model IIR using existing computing frameworks. It would therefore, be useful to possess yet another form of representing such relationships, which has been formally proven to be able to represent any given set of IIR. This would allow further investigation of IS in terms of IIR using a proven tool. It has already been shown in [27] that for some specific examples of a given set of IIR and its associated probability space, then 'safe' Datalog, can indeed be used to model those IIR.

Furthermore, 'safe' Datalog has three kinds of semantics [19, pp. 192-197] which are all equivalent, these being:

- Model-Theoretic;
- Proof-Theoretic;
- Least Model Theoretic.

Because of the transitive nature of IIR (see section 2.7 - this implies some form of recursion), it should be noted that in both [8] and [17] when attempts were made to model IIR using Prolog, difficulties were encountered. Prolog uses a method known as SLD resolution [10] which utilises a top-down proof-theoretic strategy to evaluate Prolog Programs. However, SLD can create trees which are infinite, i.e. do not terminate. Because of the cyclic nature of IIR closures, it becomes apparent why simple definitions of recursive IIR in Prolog do not work (they produce infinite SLD trees). 'safe' Datalog has been proven to have a least model [19, pp. 192-197] ,

and thus using the underlying theorems described in section 5 to model IIR, then simple IIR recursive models can be established, as the least model is always finite, given a 'safe' Datalog program. Because of this 'safe' Datalog seems an ideal way of representing a given set of IIR. Because IIR is domain independent, then using 'safe' Datalog to model a given set of IIR would allow us to model the more general situations that IIR covers, and investigate the relationships between the levels of IIR in a database shown in table 1.

In summary:

- we can model the transitive nature of IIR using recursion in Datalog;
- we have the three semantics available for modelled IIR, namely model, fixpoint and proof theoretic semantics, which allows the operational semantics of least fixpoint to be used in generating IIR closures;
- there are possibly many interpretations of IIR using Datalog. These are not term order dependent in the IIR eventual predicate rule set;
- 'safe' Datalog can be used as a tool for the investigation and modelling of any information source.

Finally, IIR itself is a framework that describes information content of a given state of affairs and relationship therein. As we are able to successfully model IIR, and IIR is not necessarily tied to an IS, and furthermore we can model such IIR with several implementations (as shown above), it would seem that IIR itself must represent some kind of generalised framework, with which to undertake informational analysis of virtually any situation that can be represented in probabilistic terms.

8 Future Work

We have shown that there is some kind of transformation from IIR to 'safe' Datalog. In future papers we propose to show that the reverse is also true; that is, there is some kind of transformation from 'safe' Datalog to IIR.

Additionally why are there so many 'safe' Datalog representations of IIR rule sets available? Does this reflect the fact that FOL (of which 'safe' Datalog is a subset) is not quite correct when describing informational relationships? To determine the number of interpretations, and why this should be the case may be the subject of future work. Indeed, it would appear that such a paradigm as first-order logic is not stringent enough to be isomorphic with information-theoretic views of domains of discourse. This would agree with Devlin's claim that first-order logic is insufficient to describe informational relations [31]. The multiplicity of representations and the reason for this needs further investigation. Also, there is no proof in the above, that this is the only method by which IIR can be represented in 'safe' Datalog. We believe, that because IIR is a more general framework, and thus, in some ways, possibly a specialised, or, on the contrary a more generalised version of FOL. If so, then there must be a specific morphism between a subset or super-set of First Order Categorical Logic to IIR which we believe is a category, with events as objects, IIR as arrows, composition as IIR under transitivity, with reflexivity as the identity [32].

Because of this formally proven linkage between IIR and 'safe' Datalog, we know we can represent any information source, that is any situation that can be presented as a state space, in 'safe' Datalog. Furthermore such a linkage suggests that there must be a more formal link between 'safe' Datalog and IIR than that explored in the previous parts of this paper. Lastly we suggest that IIR itself is not limited to an IS. In fact it is a general purpose description of any suitably constrained domain of discourse. As there appear to be many ways of representing such a relationship, listed above, then there is some evidence to show such IS representations must be some kind of subset of the IIR framework. What this subset is, and what parts of, say for instance, 'safe' Datalog contains, is also the subject for future work. In particular, as we have now successfully proved that 'safe' Datalog can be used to represent a given set of IIR. This now allows us to investigate each of the four levels of IIR in a database, as in table 1, and we should now be able to cast some light on the nature of such databases, which at heart are representation systems, after [11]. The source representation system is the database itself, and the target of the representation system is the domain of application. Furthermore what is the nature of this linkage or correspondence? 'Safe' Datalog can successfully represent certain aspects of an application domain. That is, at one level, it is such a representation system, and there must exist constraints between the database and the domain of application it is trying to model. However, the source of this representation, e.g. a database resident on some computing device, when, as mentioned previously, looked at from the purely physical viewpoint is at best just variances in electrical potential within the internals of that computing device [14, p. 156].Eventually these symbols can be used to represent aspects of the real world. We define these differences of electrical potential with values of 0 or 1, and in turn use groups of these symbols to represent other symbols. Hence a computing device is merely a contrivance for manipulating symbols (ibid), and consequently computing devices do not exhibit cognition. It is this linkage via these constraints to the real world by this accumulation and refinement of such symbols that we believe information content can arise. We intend to use Datalog to model such area of information content, and uncover reasoning at a distance for such representation systems.

References

[1] X. Wu and J. Feng, "A framework and implementation of information content reasoning in a database," WSEAS Trans. Info. Sci. and App., vol. 6, no. 4, pp. 579–588, 2009.

[2] J. Feng, Y. Wang, and S. Wang, "Causes and types of connection traps in data schemata," WSEA Transactions on

Information Science and Applications, vol. 4, pp. 1303 –1311, July 2007.
[3] L. Floridi, "Philosophical conceptions of information," in Formal Theories of Information: From Shannon to Semantic Information Theory and General Concepts of Information, pp. 13–53, Springer-Verlag, 2009.
[4] L. Floridi, "Is information meaningful data?," Philosophy and Phenomenological Research, vol. 70, no. 2, pp. 351–370, 2005.
[5] 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, pp. 29–59, Jan. 2009.
[6] F. I. Dretske, Knowledge and the Flow of Information. Cambridge University Press, new edition ed., May 1999.
[7] W. Hu and J. Feng, "Considering norms and signs within an information Source-Bearer-Receiver (S-B-R) framework," in Virtual, Distributed and Flexible Organisations, pp. 183–184, Springer Netherlands, 2005.
[8] K. Xu, The Notion of "Information Content of Data" for Databases. Doctoral thesis, University of the West of Scotland, July 2009.
[9] R. Elmasri and S. B. Navathe, Fundamentals of Database Systems. Benjamin-Cummings Publishing Company, Subs of Addison Wesley Longman, Inc, 2nd revised edition ed., Feb. 1994.
[10] M. Dahr, Deductive Databases. International Thomson Computer Press, Dec. 1996.
[11] A. Shimojima, On the Efficacy of Representation. PhD thesis, Indiana University, May 1996.
[12] J. Barwise and J. Seligman, Information Flow: The Logic of Distributed Systems. Cambridge University Press, July 1997.
[13] H. Xu and J. Feng, "Towards a definition of the "information bearing capability" of a conceptual data schema," in Systems Theory and Practice in the Knowledge Age, pp. 431–438, Springer, 1st ed., June 2002.
[14] K. Devlin, Goodbye Descartes: End of Logic and the Search for a New Cosmology of the Mind. John Wiley & Sons, new edition ed., Mar. 1998.
[15] P. J. Cameron, Sets, Logic and Categories (Springer Undergraduate Mathematics). Springer, illustrated edition ed., Jan. 1999.
[16] S. Zhu, Reasoning about the information content of data: The use of ontology. PhD thesis, University of the West of Scotland, Aug. 2009.
[17] X. Wu, An Architecture of System of 'Information Content' Reasoning in Databases. Masters dissertation, University of the West of Scotland, Dec. 2008.
[18] S. Zhu and J. Feng, "Using an ontology to help reason about the information content of data," Journal of Software Engineering and Applications, vol. 3, pp. 629–643, July 2010.
[19] C. Zaniolo, S. Ceri, C. Faloutsos, R. T. Snodgrass, V. Subrahmanian, and R. Zicari, Advanced Database Systems. Morgan Kaufmann, May 1997. Part III.
[20] T. Eiter, G. Gottlob, and H. Mannila, "Disjunctive datalog," ACM Trans. Database Syst., vol. 22, no. 3, pp. 364–418, 1997.
[21] F. Sáenz-Pérez, "Datalog educational system v1.6.2 user's manual," 2004. Technical Report SIP 139-04.
[22] S. Abiteboul, R. Hull, and V. Vianu, Foundations of Databases. Addison Wesley, facsimile ed., Feb. 1995.
[23] Das, Deductive Databases and Logic Programming. Addison Wesley, July 1992.
[24] M. Arenas and L. Libkin, "An information-theoretic approach to normal forms for relational and XML data," J. ACM, vol. 52, pp. 246–283, Mar. 2005.
[25] T. T. Lee, "An Information-Theoretic analysis of relational databases part i: Data dependencies and information metric," IEEE Trans. Softw. Eng., vol. 13, no. 10, pp. 1049–1061, 1987.
[26] T. T. Lee, "An Information-Theoretic analysis of relational databases part II: information structures of database schemas," IEEE Trans. Softw. Eng., vol. 13, no. 10, pp. 1062–1072, 1987.
[27] J. Feng and D. Salt, "Information content inclusion relation and its use in database queries," Journal of Software Engineering and Applications, vol. 3, pp. 255–267, Mar. 2010.
[28] T. M. Connolly and C. E. Begg, Database Systems: A Practical Approach to Design, Implementation and Management. Addison Wesley, 5 ed., Apr. 2009.
[29] H. Xu and J. Feng, "The "how" aspect of information bearing capability of a conceptual schema at the path level," in 7th Annual Conference of the UK Academy for Information Systems, UKAIS, vol. 9, (Leeds), pp. 209–215, 2002.
[30] M. Bremer and D. Cohnitz, Information and Information Flow: An Introduction. ontos verlag, July 2004.
[31] K. Devlin, Logic and Information. Cambridge University Press, Sept. 1995.
[32] B. C. Pierce, Basic Category Theory for Computer Scientists. MIT Press, Sept. 1991.

Douglas Salt: BSc Hons was born in the UK and graduated with the Open University. Before graduating he spent 25 years in the commercial computing industry, working as a developer through to project lead. He has a wide experience of many database and development paradigms. He is currently undertaking a PhD at the University of the West of Scotland.

Junkang Feng:, BSc, MPhil, PhD was born in Shanghai China and studied at the Shanghai High School in Shanghai and then graduated from the Institute of Military Engineering of the People's Liberation Army (PLA), the Chinese armed forces, majoring in Guided Missiles Engineering. In China, he lectured at the National University of Defence Technology of the PLA and then worked as a Research Engineer in a research institute of the Shanghai Academy of Spaceflight Technology. In the UK, he received his MPhil from the University of Portsmouth and PhD from the University of the West of Scotland (the UWS) both in Computer Science. He worked as a Research Assistant at the University of Manchester, a Part-time Lecturer at the University of Portsmouth, and a Lecturer at the University of the West of Scotland. Currently Dr Feng is a Senior Lecturer and the Leader of the Database Research Group of UWS. He is also a Visiting Professor of Donghua University and Beijing Union University in China. Dr Feng's research interests include qualitative information and information flow theories, distributed information systems and database theory and systems, and has published widely in journals, conferences and books in these fields, altogether 195 items thus far. Dr Feng has completed or been involved in 9 externally (some from EU and China) funded research projects. His work was submitted to RAE (Research Assessment Exercise) in UK. He was Convener of Central Scotland of UK Academy of Information Systems. He reviewed funding applications for the British Council, research manuscripts for the Journals of Knowledge and Information Systems, Information Systems Frontier, Systemist and various WSEAS Transactions, and conferences of UKAIS and ICISO. He has supervised 3 PhD programmes (1 as 2nd supervisor) and 33 MSc dissertation projects to successful completion. He currently supervises 8 PhD and 2 MSc students for their thesis/dissertation research.