Intuitionistic interpretation of deductive databases with incomplete information*

Fangqing Dong and Laks V.S. Lakshmanan

Department of Computer Science, Concordia University, Montreal, H3G 1M8, Canada

Abstract


The aim of this paper is to build the relationship between deductive databases with incomplete information and hypothetical reasoning using embedded implications. We first consider the semantics of deductive databases with incomplete information in the form of null values. We motivate query answering against a deductive database with nulls as the problem of extracting the maximal information from a (deductive) database in response to queries, and formalize this in the form of conditional answers in a (syntactic) higher-order logic. We give a fixpoint semantics to deductive databases with nulls, and examine the relationship between existing recursive query processing techniques and the proof procedure for deductive databases with nulls. We then examine hypothetical reasoning using embedded implications and develop an intuitionistic model semantics for embedded implications with integrity constraints. Finally, we illustrate by example a method for transforming embedded implications into deductive databases with nulls. This result shows that the important functionality of hypothetical reasoning can be implemented within the framework of deductive databases with null values.

1. Introduction

Deductive databases with their improved expressive power over relational databases and their underlying logical foundations, support high level declarative querying, making them suitable for many applications. Indeed, the past decade has witnessed an explosive research into deductive databases, in particular w.r.t. efficient query processing. (See e.g., Bancilhon and Ramakrishnan [5] and Ceri et al. [6] for a survey.) However, they still lack important functionalities needed for the so-called knowledge-base systems. For instance, the notion of a set of tuples retrieved from the

Correspondence to: L.V.S. Lakshmanan, Department of Computer Science, Concordia University, Montreal, H3G 1M8, Canada.

*This research was supported by grants from NSERC (Canada) and FCAR (Quebec).
database is an extremely restrictive notion of answers to queries, in the context of knowledge-base systems. We need more powerful forms of query answering mechanisms which are capable of generating plans, explanations, etc. that pertain to the situation being queried. The work reported in this paper is part of our series of efforts at extending deductive databases in this direction.

Most of the works on deductive databases have only considered a complete information model for the set of facts available for the EDB (Extensional Database) relations. For many applications available information is typically incomplete. One form of incomplete information that has been researched extensively in the context of relational databases is the well-known null values (see [1] for a survey). Of the many different types of null values, the kind most researched are the so-called “exists but unknown” type of null values. Both logical (e.g., Lipski [19], Imieliński and Lipski [16], Gallaire et al. [14], Reiter [26], Vardi [28]) and algebraic (e.g., Abiteboul et al. [1]) approaches have been investigated in the literature. The question of query processing in deductive databases in the presence of incomplete information (e.g., in the form of nulls) has received relatively little attention. Abiteboul et al. [1], Demolombe and Cerro [7], Liu [20], and Dong and Lakshmanan [8] are the representative works.

Abiteboul et al. explored the question of extracting possible facts to answer queries against incomplete information databases. They interpreted null values as variables bound by the constraints on nulls, and formalized possible answers to the queries as facts satisfiable in some models of the underlying theory of the database. Liu [20] considers incomplete information in the form of “S-constants” which are similar to marked nulls with additional information in the form of a set of possible values that the null may take.

In particular, in [8], we have extended deductive databases with the ability to generate conditional answers in the presence of incomplete information in the form of null values. We have proposed both top-down and bottom-up (based on extended magic sets transformation) query processing strategies. Based on this formalism, we have developed a methodology for the application of fault diagnosis. Furthermore, we indicated the information extracted in that manner can be applicable (1) to hypothetical query answering (see Naqvi and Rossi [25]) and (2) to answering queries in the context of design databases where specifications are often incomplete and one may want to know what would be the eventual outcomes if various design alternatives were chosen. Indeed, the idea behind possible and conditional answers can be regarded as extracting the facts which are derivable under assumptions on how null values can be replaced by constants available in the domain. Those assumptions can be supposed if they do not contradict existing knowledge on nulls. This is a certain kind of hypothetical reasoning.

Theoretical research on hypothetical reasoning has received wide attention in the database/logic programming community. Many researchers have explored the possibility of extending the power of databases and logic programming by integrating the ability of hypothetical reasoning into existing approaches. One promising approach
Deductive databases with incomplete information

for incorporating hypothetical reasoning is the use of embedded implications [3, 4, 23, 24]. McCarty [23, 24] has studied embedded implications and established their fixpoint and intuitionist model-theoretic semantics. He has also given a proof procedure for them. Intuitively, embedded implications are clauses of the form \( A \rightarrow (B \equiv C) \) which express the knowledge “infer \( A \) if \( B \) would be derivable whenever \( C \) were added to the database”. Subsequently, Bonner [3, 4] has developed McCarty’s framework into an elegant approach (called hypothetical Datalog) for hypothetical reasoning. Recently, there have been extensions to this framework. Bonner et al. [4] adds NAF to Intuitionistic logic programming, and Olivetti et al. [15] gives a top-down proof procedure for Bonner’s hypothetical Datalog. These works are different from ours: first, they have not considered using integrity constraints to eliminate unintuitive answers; and secondly, they considered only “yes/no” kind of hypothetical answers.

A related area is abductive reasoning. Indeed, recently there has been significant interest in extending the power of logic programming systems by incorporating the ability for abductive reasoning. This has resulted in the so-called abductive logic programming (see Eshghi and Kowalski [13] and Kakas and Mancarella [17, 18]). Philosophically, there might appear to be some similarities between these two paradigms. The important differences are the following. In hypothetical reasoning, embedded implications offer the possibility of “precompiling” knowledge pertaining to hypothetical reasoning (e.g. diagnostic knowledge) in the form of embedded Horn rules, whereas in abductive reasoning there is apparently no such facility. More importantly, as shown by Bonner [3], hypothetical knowledge expressed in the form of embedded implications cannot be expressed in classical logic.

To appreciate the difference between hypothetical reasoning and hypothetical query answering, we remark that in the framework proposed by Bonner [3, 4], answers to queries are restricted to a simple yes/no. Indeed, for many practical applications, the (embedded) hypotheses made use of during a proof of the query goal are at least as important (if not more!) as the yes/no answers to the (hypothetical) queries themselves. Indeed, hypotheses generated in this manner can represent important design decisions for applications such as planning and circuit design.

In this paper, we continue research on query answering against deductive databases in the presence of nulls along the line of hypothetical reasoning, in two ways. Firstly, we model deductive databases with null values in terms of embedded implications, and formalize query answering against the databases expressed in the form of embedded implications. Since Bonner’s declarative language for hypothetical reasoning cannot handle integrity constraints among predicates, we need to extend Bonner’s approach to handle integrity constraints so that information on nulls can be correctly captured, and constraints on nulls can be verified whenever it is necessary. An example is given to show the motivation and intuition behind the ideas discussed in this paper. We not only allow nulls which can be mapped to some (existing or completely new) constant, but deal with nulls which may correspond to a set of tuples of constants, subject to given constraints on nulls. We develop a fixpoint semantics for
deductive databases with nulls, which is defined by an iterative operator $T_P$ similar to that used in logic programming, augmented with a consistency checking module. Following from this result, it is easy to see that query answering against databases with nulls can be realized based on existing query processing strategies (also see [8]).

Secondly, we are interested in the prospect of incorporating hypothetical query answering capability within the framework of deductive databases (with possible extensions). The goal is to try to do this in a manner whereby existing query processing strategies developed for deductive databases can be employed for hypothetical query answering (with possible extensions). (This is to be contrasted with hypothetical Datalog.) We illustrate by example a method to transform embedded implications into databases with null values. This method implicitly suggests that extracting hypothetical answers against embedded implications is essentially similar to the question of query processing against databases with null values. It shows the possibilities of integrating the ability of hypothetical reasoning into existing approaches to deductive databases and of implementing hypothetical query processing within the framework of deductive databases. One problem that faces us is that the number of explanations/hypotheses generated for a query goal could be prohibitively large. In fact, many of these explanations could even be irrelevant or unintuitive. To solve this problem, we make use of integrity constraints as a way of eliminating irrelevant hypotheses thereby controlling the number of hypothetical answers generated.

The rest of this paper is organized as follows. Section 2 gives a motivating example to show the intuition behind the ideas and techniques developed in this paper. Section 3 introduces a syntactically higher order (but semantically first order) logic $\mathcal{L}$ as a vehicle language to express datalog programs with null values and the embedded implications. The main advantages of this higher-order logic are due to the following facts. (1) It is more expressive than the classical first-order deductive databases. As shown in Section 7, for example, hypothetical answers can be naturally expressed by higher-order predicates. (2) Many proof-theoretic features of deductive databases can be transformed to this higher-order logic. Section 4 formalizes datalog programs with null values using the higher-order logic $\mathcal{L}$. Section 5 studies a subclass of embedded implications in the presence of integrity constraints and establishes their model-theoretic and fixpoint theoretic semantics. Section 6 gives an intuitionistic interpretation for (first-order) datalog programs with null values. This interpretation provides a theoretical foundation for the relationship between query answering against datalog programs with null values and hypothetical querying answering against embedded implications. Section 7 illustrates by example a method for transforming a subclass of embedded implications into deductive databases with nulls in the logic $\mathcal{L}$. It also shows hypothetical query answering against embedded implications can be realized as conditional query answering against the transformed deductive databases with nulls. This is significant because query processing strategies recently developed (e.g. Dong and Lakshmanan [8]) for DDBs with nulls can now be used for generating hypothetical answers. This attacks the problem of realizing hypothetical query answering in the
existing framework of deductive databases by developing an algorithm which transforms embedded implications into (higher-order) deductive databases with null values. Section 8 presents the conclusions and future research.

2. Motivation

Null values have been recognized to be a convenient way of representing incomplete information of the "exists but unknown" kind in databases. Consider a file system design situation where it is desired to make use of available file organization strategies and their strengths in terms of efficiently supporting various types of queries. Suppose that information known to the database administrator (DBA) is represented in the form of the relations good_for (Strategy, Query-type) and implemented (File, Strategy), where Strategy refers to file organization strategies and the other attributes and relations have the obvious meaning. Suppose the available knowledge is represented as the following facts together with the constraint $\mathcal{C} = \{ \bot_1 \neq \bot_2 \}$. Here, $b = B^+ \text{-tree}$, $h = \text{hashing}$, $m = \text{multilist}$, $s = \text{simple}$, $r = \text{range}$, $bl = \text{boolean}$, and $f_1, f_2, f_3$ denote files.

\[
\begin{align*}
r_1 & : \text{good for}(b, s). & r_6 & : \text{implemented}(f_1, h). \\
r_2 & : \text{good for}(b, r). & r_7 & : \text{implemented}(f_2, \bot_1). \\
r_3 & : \text{good for}(m, bl). & r_8 & : \text{implemented}(f_3, \bot_2). \\
r_4 & : \text{good for}(h, s). & \mathcal{C} & : \{ \bot_1 \neq \bot_2 \}. \\
r_5 & : \text{good for}(\bot_2, r).
\end{align*}
\]

Here, $r_5$ corresponds to the DBA's knowledge that there is a strategy $\bot_2$ which is good for range queries, and this strategy could be one of the known ones, or could be something he did not encounter before (perhaps a recent invention). Also, $r_7$ and $r_8$ correspond to the facts that the access strategies for files $f_2, f_3$ have not been decided on yet, although there is a constraint to implement them with different strategies. Let $\text{supports}(F, Q)$ mean that file $F$ supports queries of type $Q$ efficiently. This can be defined as the following rule $r_9$: $\text{supports}(F, Q) \leftarrow \text{implemented}(F, S)$, $\text{good_for}(S, Q)$. Now, consider the query $Q: \leftarrow \text{supports}(F, r)$, which asks for the files supporting range queries. Mechanically resolving the given query against rule $r_9$, and resolving the second subgoal in the resulting goal against $r_2$ gives us the new goal $\leftarrow \text{implemented}(F, b)$. Under the usual least Herbrand model semantics, an attempt to unify this subgoal with $r_7$ fails, essentially because $b$ and $\bot_1$ are treated as distinct entities. However, what we really need is to be able to match the null $\bot_1$ with a (normal) constant like $b$ as long as the constraints on the null values are not violated. Thus, we

\footnote{The example that follows is an adaptation of the one in [8]. The main difference is that here query answering is motivated in terms of embedded implications, rather than conditional answers.}
need to be able to conclude something like the following:

$$r_{10}: \text{implemented}(f_2, b) \leftarrow (\text{implemented}(f_2, b) \Rightarrow \bot_1 = b),$$

to assert that $\text{implemented}(f_2, b)$ is derivable under the condition $\bot_1 = b$ if $\text{implemented}(f_2, b)$ would be whenever the null constant $\bot_1$ and the normal constant $b$ were interpreted to be the same, provided such an interpretation does not violate the constraints $\mathcal{C}$. In this case, since the constraint is not violated, we would like to be able to conclude "$\text{supports}(f_2, r)$ provided the condition $\bot_1 = b$ holds", i.e., $\text{supports}(f_2, r)$ holds in every model satisfying the constraints $\mathcal{C} \cup \{\bot_1 = b\}$ (as long as the constraints are consistent). Answers extracted in this form from a deductive database are called conditional answers to the query $Q$. The idea behind conditional answers is to extract tuples which would be answers if certain conditions held. The notion of conditional answer is formalized in Section 4 and we will eventually derive this conditional answer formally (Example 6.1).

3. A higher-order logic

In this section, we introduce a higher-order logic $\mathcal{L}$ as a vehicle language to represent deductive databases with null values as well as embedded implications. The syntax and semantics of the logic $\mathcal{L}$ are mainly adapted from Manchanda [22]. Syntactic restrictions are made in such a way that (1) the intuitive meaning of every higher-order predicate can be easily captured from the predicate symbol itself; and (2) existing top-down and bottom-up query processing strategies can be almost directly applied to these syntactic higher-order datalog programs.

3.1. Syntax and Semantics

3.1.1. Syntax

This higher-order language $\mathcal{L}$ consists of disjoint sets of infinitely many predicate and constant symbols, and infinitely many variable symbols. The variable symbols are of two types: individual variables, and predicate variables. Every predicate symbol or predicate variable is associated with a natural number $n$, called the arity of that symbol. A constant is either a normal constant, or an atomic null, or a relational null (rel-null) of arity $n$. Normal constants and atomic nulls are called individual constants of $\mathcal{L}$. We assume the vocabulary includes equality predicates $=, \neq$, and membership predicates $\in, \notin$. Equality and membership predicates are called evaluable predicates. Other predicates are called database predicates.

Throughout this paper, we denote by $p_i,q_j$ arbitrary predicate symbols, by $X_i,Y_j,Z_k$ arbitrary individual variables, by $P$ arbitrary predicate variables, by $c_i$ arbitrary normal constant, by $d_i$ arbitrary individual constant, by $\omega_i$ arbitrary rel-null constant of arity $n$, and by $A_i,B_j,C_k$ arbitrary atomic formulas. We denote by
\(\tilde{c}\) a \(n\)-tuple of normal constants, by \(\tilde{d}\) a \(n\)-tuple of individual constants, and by \(\tilde{X}\) a \(n\)-tuple of individual variables. We use \(\tilde{X} = \tilde{Y}\) as the abbreviation for \(X_1 = Y_1 \land \ldots \land X_n = Y_n\), where \(\tilde{X} = (X_1, \ldots, X_n)\) and \(\tilde{Y} = (Y_1, \ldots, Y_n)\). We denote by \(E\) a set of equality and membership conditions of the form \(\bot = d\) or \(\varepsilon(\tilde{d}, \omega)\). We also make use of \(E = \{\bot = d_1, \ldots, \varepsilon(\tilde{d}_k, \omega_k)\}\), a set of equality and membership conditions, to represent the conjunction \((\bot = d_1) \land \ldots \land \varepsilon(\tilde{d}_k, \omega_k)\) of all the conditions in \(E\). The meaning of \(E\) will be clear in the context.

"Rel-null" is a new form of null values, which capture situations corresponding to unknown sets of facts, all pertaining to the same predicate. For example, given a fact \(\text{father}(\text{john}, \bot)\), we know from common sense some one (and only one) is \text{john}'s father. If we would like to represent facts about \text{john}'s neighbors, and if we do not know exactly how many neighbors \text{john} might have, then we can represent these facts by an EDB fact of the form \(\text{neighbor}(\omega)\) with a constraint

\[\tilde{X} = \text{john} - \text{neighbor}(\omega), \quad \varepsilon(\tilde{X}, \tilde{Y}, \omega).\]

Intuitively, this EDB fact \(\text{neighbor}(\omega)\) represents a set of tuples of constants which satisfy the \text{neighbor} relation, and the associated constraint asserts that for any tuple \((\tilde{X}, \tilde{Y})\in\omega\), the first element \(\tilde{X} = \text{john}\).

**Definition 3.1.** The formulas of \(\mathcal{L}'\) are defined as follows.

1. A term is a variable, constant or predicate symbol. A first-order term is either an individual constant or variable. A higher-order term is a predicate symbol or variable.

2. An object atom is a formula of the form \(p(\omega), \varepsilon(\tilde{t}, \omega)\), or \(p(\tilde{t})\), where \(p\) is a predicate symbol of arity \(n\), \(\omega\) is a rel-null of arity \(n\), and \(\tilde{t}\) is an \(n\)-tuple of first-order terms. A higher-order atom is a formula of the form \(P(\tilde{t})\), where \(P\) is a predicate symbol or variable of arity \(n\), and \(\tilde{t}\) is an \(n\)-tuple of higher-order terms. An atomic formula (atom) is either an object atom or a higher-order atom.

3. A well-formed formula of \(\mathcal{L}'\) is either an atomic formula, or of the form \(F \land G\), \(\neg F\), \(\forall X(F)\), where \(F\) and \(G\) are well-formed formulas, and \(X\) is a variable.

Other connectives such as \(\rightarrow\), \(\lor\), \(\exists\), can be defined in the usual way.

3.1.2. **Semantics**

Higher-order formulas are interpreted in a way similar to that of model-theoretical semantics of first-order logic. First, we introduce notions of heterogeneous records and labeled relations.

**Definition 3.2.** Let \(U\) be a set of constants. A heterogeneous record of arity \(n\) w.r.t. \(U\) and a labeled relation of arity \(n\) w.r.t. \(U\) are defined as follows.

1. An ordered tuple \((d_1, \ldots, d_n)\) is a heterogeneous record of arity \(n\) w.r.t. \(U\), if every \(d_i\) is a constant of \(U\);
2. A labeled relation is a labeled set \( p \{ r_1, \ldots, r_k \} \), if the label \( p \) is a predicate symbol of arity \( n \), and all \( r_i \)'s are heterogeneous records of arity \( n \) w.r.t. \( U \); and
3. An ordered tuple \( (r_1, \ldots, r_k) \) of labeled relations is a heterogeneous record of arity \( n \) w.r.t. \( U \), if every \( r_i \) is a labeled relation w.r.t. \( U \).

We can also view a labeled relation as a set of atoms. Thus, a label relation \( p \{ r_1, \ldots, r_k \} \) can be written as the set \( \{ p(r_1), \ldots, p(r_k) \} \).

Next, we define semantic structures for Higher-Order formulas. Intuitively, we would like to interpret each individual constant as an element in individual universe, each predicate symbol as a labeled relation, and each rel-null constant as a set of records.

**Definition 3.3.** A semantic structure \( \mathcal{M} \) is a triple of \( (U, h, I) \) such that
- \( U \) is an individual universe for individual constants of \( \mathcal{L} \);
- \( 2^{U^n} \) is the set universe for rel-null constants of \( \mathcal{L} \) with arity \( n \), for every arity \( n \);
- \( h \) is a mapping function which maps every individual constant \( d \) in \( \mathcal{L} \) to an element of \( U \), and maps every rel-null \( \omega \) of arity \( n \) to an element of \( 2^{U^n} \). We denote by \( \mathcal{M}(d) \) the element of \( U \) to which \( d \) is mapped, and by \( \mathcal{M}(\omega) \) the element of \( 2^{U^n} \) to which \( \omega \) is mapped;
- \( I \) is a mapping function which maps every predicate symbol \( p \) of \( \mathcal{L} \) to a unique labeled relation \( p \{ r_1, \ldots, r_k \} \) with label \( p \) w.r.t. \( U \). \( \mathcal{M}(p) \) denotes the labeled relation \( p \{ r_1, \ldots, r_k \} \) with label \( p \).

The satisfaction of a formula can be defined using a variable assignment function \( v \).

**Definition 3.4.** Let \( \mathcal{M} = (U, h, I) \) be a semantic structure. Then a variable assignment \( v \) w.r.t. \( \mathcal{M} \) is a function on the set of all the variables of \( \mathcal{L} \) such that each individual variable is assigned an element of \( U \), each predicate variable of arity \( n \) is assigned a labeled relation of arity \( n \).

**Definition 3.5.** Let \( \mathcal{M} = (U, h, I) \) be a semantic structure, and \( v \) be a variable assignment. Then the extension \( \bar{v} \) of the variable assignment \( v \) is a mapping function on terms such that \( \bar{v} \) agrees with \( v \) on variable symbols, and with \( \mathcal{M} \) on the predicate and constant symbols.

**Definition 3.6.** Let \( \mathcal{M} = (U, h, I) \) be a semantic structure, \( v \) be a variable assignment, and \( \bar{v} \) be the extension of the variable assignment \( v \). Then the fact that the semantic structure \( \mathcal{M} \) satisfies a formula \( \psi \) with variable assignment \( v \), written as \( \mathcal{M} \models \psi[v] \), is defined as follows.
1. If \( \psi \) is an atomic formula \( P(t_1, \ldots, t_n) \), then \( \mathcal{M} \models P(t_1, \ldots, t_n)[v] \), iff \( (\bar{v}(t_1), \ldots, \bar{v}(t_n)) \in \bar{v}(P) \).
2. If \( \psi \) is an atomic formula \( p(\omega) \), then \( \mathcal{M} \models p(\omega)[v] \), iff \( p(\bar{v}(\omega)) \subseteq \bar{v}(p) \).
3. If \( \psi \) is an atomic formula \( e(t_1, \ldots, t_n, \omega) \), then \( \mathcal{M} \models e(t_1, \ldots, t_n, \omega)[v] \), iff \( (\bar{v}(t_1), \ldots, \bar{v}(t_n)) \in \bar{v}(\omega) \).
4. If \( \psi \equiv \forall X \, G \), where \( X \) is a (individual or predicate) variable, then \( \mathcal{H} \models \forall X \, G[v] \), iff \( \mathcal{H} \models G[v'] \), for every variable assignment \( v' \) over \( v \), where \( v' \) agrees with \( v \) everywhere, except possibly at \( X \).

5. If \( \psi \equiv \neg F \), then \( \mathcal{H} \models \neg F[v] \) iff \( \mathcal{H} \not\models F[v] \).

6. If \( \psi \equiv A \land B \), then \( \mathcal{H} \models (A \land B)[v] \) iff \( \mathcal{H} \models A[v] \) and \( \mathcal{H} \models B[v] \).

We say a formula \( \psi \) is satisfiable, if there exists a semantic structure \( \mathcal{H} \) such that \( \mathcal{H} \) satisfies \( \psi \) for some variable assignment.

### 3.2. Mapping to many-sorted logic

#### 3.2.1. Syntax

The higher-order language \( \mathcal{L} \) introduced in the previous section is syntactic higher-order. It can be transformed into a many-sorted logic \( \mathcal{L}_m \) with three sorts: \( \text{const} \), \( \text{rel-null} \) and \( \text{pred} \). Every individual constant \( d \) of \( \mathcal{L} \) corresponds to a constant symbol of sort \( \text{const} \) in \( \mathcal{L}_m \), every rel-null constant \( \omega \) of \( \mathcal{L} \) corresponds to a constant symbol of sort \( \text{rel-null} \) in \( \mathcal{L}_m \), and every predicate symbol of \( \mathcal{L} \) is associated a constant symbol of sort \( \text{pred} \) in \( \mathcal{L}_m \). The only predicate symbols of \( \mathcal{L}_m \) are of the form \( \varepsilon^n \), for every \( n \)-tuple of sorts. Intuitively, the predicate symbol \( \varepsilon^n \) denotes the “belongs to” relation. For simplicity, we will drop the arity superscript \( n \) from \( \varepsilon^n \).

#### 3.2.2. Semantics

**Definition 3.7.** A record of arity \( n \) with associated sorts \( (\tau_1, \ldots, \tau_n) \) is an ordered tuple \((d_1, \ldots, d_n)\) of individuals such that every \( d_i \) belongs to sort \( \tau_i \).

**Definition 3.8.** A semantic structure \( \mathcal{M} \) is a 5-tuple of \((U_c, U_\omega, U_\rho, h, I)\) such that

- \( U_c \) is an individual universe for individual constants of \( \mathcal{L}_m \) of sort \( \text{const} \);
- \( U_\omega \) is a rel-null universe for rel-null constants of \( \mathcal{L}_m \) of sort \( \text{rel-null} \);
- \( U_\rho \) is a predicate symbol universe for predicate of \( \mathcal{L}_m \) of sort \( \text{pred} \);
- \( h \) is a function which maps (1) every constant of \( \mathcal{L}_m \) of sort \( \text{const} \) to an element of \( U_c \), (2) every constant \( \omega \) of \( \mathcal{L}_m \) of sort \( \text{rel-null} \) to an element of \( U_\omega \), and (3) every constant \( \rho \) of \( \mathcal{L}_m \) of sort \( \text{pred} \) to an element of \( U_\rho \). Denote by \( \mathcal{M}(c) \) (or \( \mathcal{M}(\omega) \) or \( \mathcal{M}(\rho) \)) the element to which \( c \) (or \( \omega \) or \( \rho \)) is mapped; and
- \( I \) is a function which maps every predicate symbol \( \varepsilon^n \) of \( \mathcal{L}_m \) with a sort \( \langle s_1, \ldots, s_n \rangle \) to a subset of \( U_{s_1} \times \cdots \times U_{s_n} \).

The satisfaction of a formula can be defined using a variable assignment function \( v \).

**Definition 3.9.** Let \( \mathcal{M} \) be a semantic structure. Then a variable assignment \( v \) w.r.t. \( \mathcal{M} \) is a function on the set of all the variables of \( \mathcal{L}_m \) such that each variable of sort \( \tau \) is assigned an element of sort \( \tau \).
Definition 3.10. Let $\mathcal{M}$ be a semantic structure, and $v$ be a variable assignment. Then the extension $\bar{v}$ of $v$ is a mapping function on terms such that $\bar{v}$ agrees with $v$ on variable symbols, and with $\mathcal{M}$ on the predicate, constant, and rel-null symbols.

Definition 3.11. Let $\mathcal{M}=(U, U_0, U_1, h, I)$ be a semantic structure, $v$ be a variable assignment, and $\bar{v}$ be the extension of the variable assignment $v$. Then the fact that the semantic structure $\mathcal{M}$ satisfies a formula $\varphi$ with variable assignment $v$, written as $\mathcal{M} \models \psi[v]$, is defined as follows.

1. If $\psi$ is an atomic formula $\varepsilon(t_1, \ldots, t_n)$, then $\mathcal{M} \models \varepsilon(t_1, \ldots, t_n)[v]$, iff $(\bar{v}(t_1), \ldots, \bar{v}(t_n)) \in \varepsilon(e)$.
2. If $\psi \equiv \forall X \varphi$, where $X$ is a variable of sort $\tau$, then $\mathcal{M} \models \forall X \varphi[v]$, iff $\mathcal{M} \models \varphi[v']$, for every variable assignment $v'$, where $v'$ agrees with $v$ everywhere, except possibly at $X$.
3. If $\psi \equiv \neg \varphi$, then $\mathcal{M} \models \neg \psi[v]$ iff $\mathcal{M} \models \varphi[v]$.
4. If $\psi \equiv A \land B$, then $\mathcal{M} \models (A \land B)[v]$ iff $\mathcal{M} \models A[v]$ and $\mathcal{M} \models B[v]$.

3.2.3. Mapping the logic $\mathcal{L}$ to many-sorted logic

The formulas of $\mathcal{L}$ can be mapped to the formulas of $\mathcal{L}_m$ as follows.

- Every atom of $\mathcal{L}$ of the form $P(t_1, \ldots, t_n)$ is mapped to an atom of the form $\varepsilon(t_1, \ldots, t_n)$.
- Every atom of $\mathcal{L}$ of the form $\varepsilon(t_1, t_2)$ is mapped to the atom $\varepsilon(t_1, t_2)$ itself.
- Every atom of $\mathcal{L}$ of the form $p(\varepsilon)$, where $\varepsilon$ is a rel-null constant of arity $n$, is mapped to a formula of the form $\forall X(\varepsilon(X, \alpha) \rightarrow \varepsilon(p, \bar{X}))$.

Let $\psi$ be a formula of $\mathcal{L}$. Then we say the formula $\psi_m$ of $\mathcal{L}_m$ obtained by the transformation above from $\psi$ is many-sorted version of $\psi$.

The next theorem shows the semantic equivalence between the higher-order logic $\mathcal{L}$ and many-sorted logic $\mathcal{L}_m$. Since the higher-order logic $\mathcal{L}$ is developed based on Manchanda [22], our result extends Manchanda’s result by incorporating null values.

Theorem 3.1. Let $\psi$ be any formula of $\mathcal{L}$ and $\psi_m$ be the many-sorted version of $\psi$ in $\mathcal{L}_m$. For any structure $\mathcal{H}$ for $\mathcal{L}$, there is a corresponding structure $\mathcal{M}$ for $\mathcal{L}_m$, and vice-versa, such that for all variable assignments $v$ of $\mathcal{M}$, $\mathcal{H} \models \psi[\varphi \circ v]$ iff $\mathcal{M} \models \psi_m[v]^2$.

Proof. Let $p$ be a predicate symbol of arity $n$, and $t_i$ be a predicate or constant symbol, of $\mathcal{L}$. Then the relationship between the semantic structure $\mathcal{M}$ of $\mathcal{L}_m$ and $\mathcal{H}$ of $\mathcal{L}$ can be expressed as follows:

- $(\mathcal{M}(p), \mathcal{M}(t_1), \ldots, \mathcal{M}(t_n)) \in \mathcal{M}(e)$ iff $(\mathcal{H}(t_1), \ldots, \mathcal{H}(t_n)) \in \mathcal{H}(p)$; and
- for every re-null $\omega$, $(\mathcal{M}(t_1), \ldots, \mathcal{M}(t_n), \mathcal{M}(\alpha)) \in \mathcal{M}(e)$ iff $(\mathcal{H}(t_1), \ldots, \mathcal{H}(t_n)) \in \mathcal{H}(\alpha)$.

Let $\mathcal{H}=(U, h, I)$ be a semantic structure of $\mathcal{L}$. Define a semantic structure $\mathcal{M}=(U_c, U_0, U_1, h_m, I_m)$ of $\mathcal{L}_m$ based on $\mathcal{H}$ as follows.

\[\mathcal{M}(\varphi \circ v)[X] = \mathcal{H}(p) = h(p).\]
• $U_c = \{ \mathcal{H}(d) \mid d \text{ is an individual constant symbol}\}$;
• $U_\omega = \{ \omega \mid \omega \text{ is a rel-null in } \mathcal{L}\}$;
• $U_p = \{ p \mid p \text{ is a predicate symbol}\}$;
• Define $h_m$ as follows: $h_m(d) = \mathcal{H}(d)$, if $d$ is an individual constant. $h_m(\omega) = \omega$, if $\omega$ is a rel-null constant.
• Define $I(e)$ as follows.

1. For each heterogeneous record $(\mathcal{H}(r_1), \ldots, \mathcal{H}(r_n)) \in \mathcal{H}(p)$, generate a record for $I(e^{n+1})$ from $(p, \mathcal{H}(r_1), \ldots, \mathcal{H}(r_n))$ by replacing every $\mathcal{H}(r_i)$ by the associated label, if $r_i$ is a labeled relation.

2. For each heterogeneous record $(\mathcal{H}(r_1), \ldots, \mathcal{H}(r_n)) \in \mathcal{H}(\omega)$, generate a record for $I(e^{n+1})$ from $(\mathcal{H}(r_1), \ldots, \mathcal{H}(r_n), \omega)$ by replacing every $\mathcal{H}(r_i)$ by the associated label, if $r_i$ is a labeled relation.

In the other direction, from every semantic structure of $\mathcal{L}_m$ we can construct a semantic structure of $\mathcal{L}$. Note that (1) from the definition of $U_c$, it is easy to see $U_c = U_c$; and (2) for any variable assignment $v$ over $\mathcal{H}$, $\mathcal{H} \circ v$ is a variable assignment over $\mathcal{H}$, and for any variable assignment $v$ over $\mathcal{H}$, $\mathcal{H}^* \circ v$ is a variable assignment over $\mathcal{H}$, where $\mathcal{H}^*$ is a function which maps every element of $U$ to itself, and every labeled relation to its label. Since for each predicate symbol $p$, $\mathcal{H}$ contains a unique labeled relation with label $p$, it is easy to see there is a one to one correspondence between the variable assignments over $\mathcal{H}$ and those over $\mathcal{H}$.

Next, we prove that truth is preserved in this transformation by induction on the number of connectives of formula $\psi$ of $\mathcal{L}$.

(1) **Base case** (1.1) $\psi = P(t_1, \ldots, t_n)$, and $\psi_m = I(P, t_1, \ldots, t_n)$, where $P$ is a predicate symbol or predicate variable, $t_1, \ldots, t_n$ are terms. Let $v$ be a variable assignment w.r.t. a semantic structure $\mathcal{M}$ of $\mathcal{L}_m$. Then $\mathcal{M} \models I(P, t_1, \ldots, t_n)[v]$ i.f.f. $(\forall(P), \forall(t_1), \ldots, \forall(t_n)) \in I(e)$. Note that by construction, this can be true i.f.f. $(\mathcal{H}(\forall(t_1)), \ldots, \mathcal{H}(\forall(t_n))) \in \mathcal{H}(I(P))$. By the definition of truth in $\mathcal{H}$, this is true i.f.f. $\mathcal{H} \models I(P, t_1, \ldots, t_n)[\mathcal{H} \circ v]$.

(1.2) $\psi = p(\omega)$, and $\psi_m = \forall X (\varepsilon(X, \omega) \rightarrow \varepsilon(p, X))$, where $p$ is a predicate symbol. Then $\mathcal{M} \models \psi_m[v]$ i.f.f. for all variables $X$ and for every $(\forall(t_1), \ldots, \forall(t_n)) \in \mathcal{H}(\omega)$, $(\forall(p), \forall(t_1), \ldots, \forall(t_n)) \in I(e)$. Note that this can be true i.f.f. $(\mathcal{H}(\forall(t_1)), \ldots, \mathcal{H}(\forall(t_n))) \in \mathcal{H}(\varepsilon(p, \omega))$. By the definition of truth in $\mathcal{H}$, $\mathcal{H} \models p(\omega)[\mathcal{H} \circ v]$.

(2) **Inductive step**: Assume that the theorem is true for formulas of $\mathcal{L}$ with at most $k$ connectives. Let $\psi$ be a formula of $\mathcal{L}$ with $k + 1$ connectives. Then the following cases arise.

$\psi$ is of the form $\phi \wedge \sigma$, $\neg \phi$, or $\forall X \phi$, where $X$ is an individual variable or a predicate variable. Most of these cases are handled in a straightforward manner and we shall give the proof only for the case where $\psi$ is of the form $(\forall X) \phi$, with $X$ a predicate variable.

Clearly, $\mathcal{M} \models (\forall X) \phi[v]$ i.f.f. for any predicate symbol $p$ such that $v$ assigns $p$ to $X$, $\mathcal{M}$ satisfies $\phi'$ w.r.t. the assignment $v$, where $\phi'$ is obtained from $\phi$ by replacing every occurrence of $X$ by $p$. Suppose $\mathcal{H}$ maps $p$ to the labeled relation $r$. Then by inductive hypothesis, $\mathcal{H} \models \phi'[\mathcal{H} \circ v]$ i.f.f. $\mathcal{M} \models \phi'[v]$. □
Based on this theorem, straightforwardly, we get the following corollary.

**Corollary 3.1.** Any closed formula $\psi$ of $\mathcal{L}$ is unsatisfiable iff the many-sorted version $\psi_m$ of $\psi$ is unsatisfiable.

**Proof.** Straightforward from Theorem 3.1. ☐

These results show that the higher-order logic $\mathcal{L}$ is only syntactic higher-order, and any higher-order formula of $\mathcal{L}$ can be transformed into a semantically equivalent (many-sorted) first-order formula of $\mathcal{L}_m$. Syntactically, we can take advantage of the expressiveness of higher-order logic to support higher-order modeling. Computationally, we can transform semantic and proof-theoretical results of deductive databases into higher-order datalog programs, and use existing query answering techniques in deductive databases to develop efficient query processing strategies for higher-order datalog programs.

4. **Datalog" programs**

In this section, we formalize datalog programs with nulls. We assume the reader is familiar with the general notions of deductive databases and logic programming [21,27]. Datalog, the language of function-free Horn clauses, is the vehicle query language for deductive databases [27]. A datalog query program consists of (i) a finite set of unit clauses representing facts for the base (EDB) predicates, (ii) a finite set of Horn clause rules defining the derived (IDB) predicates, and (iii) a goal clause, representing the query. First, we introduce higher-order datalog programs. Next, we extend higher-order datalog programs to *datalog" programs*, which are datalog programs with nulls.

**Definition 4.1.** A higher-order datalog rule is a formula of the higher-order logic $\mathcal{L}$ of the form $H \leftarrow H_1, \ldots, H_k$, where $H$ and $H_i$ are (higher-order) atoms of $\mathcal{L}$. We say $H$ is the head, and $H_1, \ldots, H_k$ is the body of the rule. We also say each $H_j$ is a subgoal in the rule body. A datalog rule $H \leftarrow H_1, \ldots, H_k$, is first-order, if $H$ and $H_i$ are object atoms. A higher-order datalog program is a finite set of higher-order datalog rules. A datalog program is first-order, if every datalog rule is first order.

The effects of nulls in a database can be characterized by viewing the database as a logic theory, together with a set of axioms: Unique Name Axioms (UNA), Domain Closure Axiom (DCA), Completion Axioms (COMP), and a set of constraints on nulls. Those axioms were first proposed by Reiter [26] to formalize relational databases with nulls. Intuitively, the UNA forces the true identity of each normal constant to be fully specified. COMP ensures that any canonical model of the theory is a supported one. Integrity constraints allow the representation of partial knowledge
on nulls. We next extend these axioms to formalize DDBs with atomic nulls and rel-nulls.

**Definition 4.2.** Let $\Pi$ be a datalog program with nulls, and $D$ be the set of constants occurring in $P$. Then $\mathcal{A}$ consists of the following axioms.

1. Unique name axioms (UNA): $c_i \neq c_j$ for every distinct normal constants $c_i$ and $c_j$ of $D$, and $d_i \neq o_j$ for every pair of distinct individual constant $d_i$ and rel-null constant $o_j$ of $D$;
2. Axioms on equality and membership predicates:
   $$\forall X [X = X].$$
   $$\forall X Y [Y = X \leftarrow X = Y].$$
   $$\forall X Y Z [X = Z \leftarrow X = Y, Y = Z].$$
   $$\forall \bar{X} \bar{Y} [p(\bar{y}) \leftarrow p(\bar{x}), X = Y]$$ for every database predicate $p$.
   $$\forall \bar{X} [p(\bar{x}) \leftarrow p(\omega), e(\bar{x}, \omega)]$$ for every database predicate $p$ and rel-null $\omega$ of the same arity.

Here $p$ is a predicate symbol, $X$ and $Y$ are tuples of variables with the same arity as $p$, and $\omega$ is a rel-null. Note that for equalities of the form $X = Y$, $X$ and $Y$ have to be the variables of the same type, i.e., individual variables, or predicate variables.

3. Domain closure axiom (DCA): the axiom $\forall X [X = d_1 \lor \cdots \lor X = d_n]$, where $X$ is an individual variable, and $d_1, \ldots, d_n$ are all the individual constants occurring in the datalog program $\Pi$.

4. Completion axioms (COMP): for any predicate symbol $p$ in the program $\Pi$, defined by the rules $p(\bar{t}_i) \leftarrow A_{i1} \land \cdots \land A_{im_i}$, $i = 1, \ldots, k$, and unit clauses $p(\omega_1), \ldots, p(\omega_k)$, the following formula is the completion axiom associated with $p$

$$p(\bar{X}) \leftarrow \{ \exists \bar{Y}_1 (\bar{X} = \bar{t}_1 \land A_{11} \land \cdots \land A_{1m_1}) \lor \cdots \lor \exists \bar{Y}_k (\bar{X} = \bar{t}_k \land A_{k1} \land \cdots \land A_{km_k})$$

$$\lor e(\bar{X}, \omega_1) \lor \cdots \lor e(\bar{X}, \omega_n),$$

where $A_{ij}$'s are atoms, $\bar{t}_i$ is a tuple of first-order terms, $\bar{Y}_j$ is the tuple of variables occurring in the body of the $j$th rule but not occurring in the head, and $\omega_j$'s are rel-null constants.

We now introduce datalog programs which are intended to model deductive databases with null values.

**Definition 4.3.** A datalog program $P$ is a higher-order theory $(\Pi, \mathcal{A}, IC)$ of the logic $\mathcal{L}$, where

- $\Pi$ is a higher-order datalog program, called the datalog program associated with $P$;
• $A$ is a set of axioms consisting of the axioms on equality and membership predicates, Unique Name Axioms (UNA), Domain Closure Axiom (DCA), and Completion Axioms (COMP); and
• A set $IC$ of integrity constraints of the form $\leftarrow L_1, \ldots, L_k$, where $L_i$ is either a database atom of $\mathcal{L}$, or an evaluable predicate, or the negation of an evaluable predicate.

In the literature, a datalog program with nulls is often augmented with a set of constraints of the forms: $d_i R d_j$, where $d_i$ or $d_j$ is a null constant, and $R$ is one of evaluable predicates $=, \neq, c, \notin$. It is not hard to see that these constraints on nulls can be represented in the form of integrity constraints. For example, a constraint $\bot_1 \neq \bot_2$ can be expressed as an integrity constraint of the form

$$\leftarrow \bot_1 = \bot_2.$$ 

For better readability, in the rest of this paper, we still represent constraints on nulls in the form of $d_i R d_j$, instead of in the form of integrity constraints.

**Definition 4.4.** Let $P = (\Pi, A, IC)$ be a datalog$^o$ program. We say $P$ is first-order, if $\Pi$ and $IC$ are.

For a datalog$^o$ program to be meaningful, it needs to be consistent. This is formalized below.

**Definition 4.5.** Let $P = (\Pi, A, IC)$ be a datalog$^o$ program. Then $P$ is consistent, if the theory $\Pi \cup A \cup IC$ is logically consistent. For convenience, we say the set $IC$ of integrity constraints is consistent, whenever $P$ is consistent. Otherwise, we say $IC$ is said to be inconsistent.

A query $Q$ in $\mathcal{L}$ is a goal clause of the form $\leftarrow p(\vec{X})$, where $p$ is any database predicate and $\vec{X}$ is a $n$-tuple of free variables. Our next definition generalizes the conventional notion of answers to that of conditional answers.

**Definition 4.6.** Let $P$ be a datalog$^o$ program, and $Q \equiv \leftarrow p(\vec{X})$ be a query. Then the valid conditional answers to $Q$ against $P$ are defined as follows.

$$\|Q\|_P = \{ (\bar{r}, E) | P \models p(\bar{r}) \leftarrow E \},$$

where $\bar{r}$ is a heterogeneous record of arity $n$ and $E$ is a set of consistent equality and membership conditions. We call the conditional answers in this answer set as valid (conditional) answers.

In general, this answer set may include redundant answers. To exclude them, we need the notion of minimality. We say that a valid answer $(\bar{r}, E)$ to a query $Q$ is minimal, provided for any valid answer $(\bar{r}', E')$ of $Q$, if $P \models E \rightarrow E'$, then $P \models E \rightarrow E$. 


Conventional answers can be seen to be a special case of conditional answers where the condition set is empty.

**Definition 4.7.** Let $P$ be a datalog$^o$ program, and $Q \equiv \leftarrow p(\bar{x})$ be a query. Then the conditional answer set is defined as

$$\langle Q \rangle = \{ (\bar{r}, E) | (\bar{r}, E) \text{ is a valid answer to } Q \text{ and } E \text{ is minimal} \}.$$ 

Clearly, as defined above, the set of conditional answers to a query $Q$ is non-redundant. Note that if $P$ is a first-order datalog$^o$ program, then conditional answers to a query $Q$ against $P$ are of the form $(\bar{d}, E)$, where $\bar{d}$ is a tuple of individual constants and $E$ is a set of equality and membership conditions.

### 5. Embedded implications

In this section, first, we review embedded implications with some adaptation as proposed by McCarty [23, 24] and studied further by Bonner [3, 4] in the context of hypothetical reasoning. Then we study theories consisting of embedded implications with integrity constraints, and formalize their meaning using intuitionistic semantics. Finally, we provide a fixed-point characterization for embedded implications with integrity constraints.

#### 5.1. Embedded implications

In this section, we use the higher-order logic $L$ to express embedded implications.

**Definition 5.1.** An embedded implication is defined as follows.

1. An object atom $A$ is an embedded implication;
2. a Horn rule $A \leftarrow A_1, \ldots, A_k$ is an embedded implication, where all $A'$s are object atoms;
3. an expression of the form $A \leftarrow \exists X_1, \ldots, X_m [\psi_1, \ldots, \psi_n]$ is an embedded implication,\(^3\) where $A$ is an object atom and every $\psi_i$ is either an object atom $B$, or a Horn rule of the form $B \leftarrow C_1, \ldots, C_k$, called embedded Horn rules, and $X_1, \ldots, X_m$ are all the individual variables which occur in some $\psi_i$, but do not occur in the head $A$. For every embedded Horn rule $\psi_i$, $C_1, \ldots, C_k$ are called embedded hypotheses. For simplicity, we write an embedded implication in the form of $A \leftarrow \psi_1, \ldots, \psi_n$.

Clearly, Horn rules and $EDB$ facts are special cases of embedded implications. An embedded implication such as $(A \leftarrow (B \leftarrow C))$ expresses the knowledge: infer $A$, provided $B$ would be derivable whenever $C$ were added into the database. Both "$\leftarrow$" and

---

\(^3\)In this paper, we do not consider embedded implications where embedded Horn rules are universally quantified.
"⇒" express implications, but "⇐" is more restrictive than "⇒". The exact meaning of these two operators is given in Section 5.2.

To implement efficient hypothetical query answering, we need restrictions on hypothetical answers. Without any limitation, the number of hypothetical answers possibly generated from embedded implications could be prohibitively large, and many of them could either be irrelevant, or unintuitive. For most applications of practical interest, integrity constraints are very useful for controlling the generation of hypothetical answers. In our framework, an integrity constraint is a special case of embedded implications, which is a Horn rule of the form \( \leftarrow L_1, \ldots, L_k \), where every \( L_i \) is an object atom, or an evaluable predicate, or the negation of an evaluable predicate. Intuitively, an integrity constraint of the form above states that \( L_1, \ldots, L_k \) cannot be true simultaneously. In the rest of this paper, we will consider embedded implications with integrity constraints, and call any theory consisting of a pair of embedded implications and integrity constraints as an abductive program.

**Definition 5.2.** An abductive program \( P \) is a pair \((R, IC)\), where \( R \) is a finite set of embedded implications, and \( IC \) is a finite set of integrity constraints.

Since nulls may occur in embedded implications, to capture the correct meaning of null values, we need the following rewriting axioms on equality and membership predicates for every predicate symbol \( p \) of arity \( n \):

\[
A_1: p(\bar{Y}) \leftarrow p(\bar{Y}) \leftarrow \bar{X} = \bar{Y}, \quad \text{and} \\
A_2: p(\bar{Y}) \leftarrow p(\bar{Y}) \leftarrow \varepsilon(\bar{Y}, \omega)),
\]

for every rel-null constant \( \omega \) of arity \( n \), where \( \bar{X} \) and \( \bar{Y} \) are \( n \)-tuples of variables. These rewriting axioms assert the knowledge that \( p(\bar{Y}) \) can be inferred if \( p(\bar{Y}) \) would be derivable whenever the conditions \( \bar{X} = \bar{Y} \) (or \( \varepsilon(\bar{Y}, \omega) \)) were added into the database, as long as those conditions are consistent with existing integrity constraints. These axioms provide us a theoretical mechanism to treat equality conditions and membership conditions using rewriting axioms so that nulls can be replaced by any other constants (tuples of constants) as long as those replacements respect the constraints on nulls. We assume that the heads of embedded Horn rules cannot be evaluable predicates and embedded implications whose heads are evaluable predicates are either axioms on equality or membership predicates or EDB facts. For a set \( R \) of embedded implications, we assume that axioms on equality and membership predicates are included in \( R \), if null constants occur in it.

### 5.2. Embedded implication and intuitionistic semantics

To capture the correct meaning of embedded implications with respect to hypothetical reasoning, it turns out that they should be interpreted under intuitionistic semantics \([2, 3, 4, 23, 24]\). This section defines the intuitionistic semantics based on (a form of extended) Herbrand base of abductive programs. Since null constants
may occur in embedded implications, we extend the notion of Herbrand base to incorporate null constants.

**Definition 5.3.** Let \( P = (R, IC) \) be an abductive program. Let \( D \) be the set of all individual constants occurring in \( P \). Let \( H \) be the Herbrand base of \( R \cup IC \) w.r.t. \( D \). Then extended Herbrand base \( H^* \) of \( P \) w.r.t. \( D \) is defined as the following set:

\[
H^* = H \cup \{ p(\omega_i) | p(\omega_i) \in R \} \cup \{ \varepsilon(d, \omega) | d \in D^* \text{ and } \omega \text{ is a rel-null of arity } n \} \\
\cup \{ d_i = d_j | d_i \text{ and } d_j \text{ are individual constants of } D \}.
\]

The Herbrand instantiation of \( R (IC) \) consists of all the Herbrand instantiated rules obtained by replacing each occurrence of individual variables occurring in \( R (IC) \) by an individual constant in \( D \). The Herbrand instantiation of \( P \) is the pair of the Herbrand instantiation of \( R \) and the Herbrand instantiation of \( IC \). We use \( \mathcal{P}(H) \) to denote the power set of \( H \), and \( \mathcal{P}(H^*) \) to denote the power set of \( H^* \). A ground atom is an atom in extended Herbrand base, and a ground embedded Horn rule is a Herbrand instantiated embedded Horn rule.

We define the notions of intuitionistic structures and intuitionistic models using extended Herbrand base as follows.

**Definition 5.4.** Let \( R \) be a set of Herbrand instantiated embedded implications, \( D \) be the set of all individual constants occurring in \( R \), and \( H^* \) be the extended Herbrand base w.r.t. \( D \). An intuitionistic structure is a triple \((M, \leq, \pi)\), where \( M \) is a subset of the power set \( \mathcal{P}(H^*) \), \( \leq \) is the partial order corresponding to the usual set inclusion, and \( \pi \) is a truth assignment function such that \( \pi(A) = \{ s \in M, A \in s \} \) for every ground atom \( A \). Every element of \( M \) is called state. For any state \( s \) of \( M \), a state \( s' \) of \( M \) is a substate (superstate) of \( s \) if \( s' \leq s \) (\( s \leq s' \)); and \( s \) is minimal (maximal) w.r.t. \( M \) if every substate (superstate) of \( s \) is \( s \) itself.

For our convenience, we denote by \( M \) an intuitionistic structure without explicitly mentioning the partial order \( \leq \) and the truth-assignment function \( \pi \).

**Definition 5.5.** Let \( M \) be an intuitionistic structure. Then the (intuitionistic) satisfaction of a formula \( \psi \) at \( s \) by \( M \), denoted by \( s, M \models \psi \), is recursively defined as follows:

- \( s, M \models A \) iff \( A \in s \), for a ground atom \( A \);
- \( s, M \models \psi_1 \land \psi_2 \) iff \( s, M \models \psi_1 \), and \( s, M \models \psi_2 \);
- \( s, M \models \psi_1 \lor \psi_2 \) iff \( s, M \models \psi_1 \), or \( s, M \models \psi_2 \);
- \( s, M \models \neg \psi \) iff \( r, M \not\models \psi \), for every \( r \geq s \) in \( M \);
- \( s, M \models \psi_1 \iff \psi_2 \) iff \( r, M \models \psi_1 \), whenever \( r, M \models \psi_2 \), for every \( r \geq s \) in \( M \).

\( M \) satisfies a formula \( \psi \) iff \( s, M \models \psi \) for every state \( s \) of \( M \).
A formula of the form \( A \rightarrow B_1, \ldots, B_k \) is defined as the abbreviation for
\[
A \leftarrow B_1, \ldots, B_k = A \rightarrow B_1, \ldots, B_k \land (\neg \neg B_1 \land \cdots \land \neg \neg B_k),
\]
where \( A \) and \( B_i \) are object atoms. Intuitively, \( A \leftarrow B_1, \ldots, B_k \) expresses that \( A \rightarrow B_1, \ldots, B_k \) is true in the states which are consistent with \( B_1, \ldots, B_k \). Note that the intuitionistic interpretation for \( \neg \neg B \) is different from the classical interpretation. Indeed, \( \neg \neg B \) is equivalent to \( B \) in classical logic, while not in intuitionistic logic. Given an intuitionistic structure \( M, \neg \neg B \) is true in \( M \), if \( B \) is true in all maximal states of \( M \), while \( B \) is true in \( M \) exactly when \( B \) is true in all the states of \( M \). Later in this section, we shall make use of \( \neg \neg B \) to assert that \( B \) is a consistent hypothesis w.r.t. a set of embedded implications and a set of integrity contraints.

**Definition 5.6.** Let \( P = (R, IC) \) be an abductive program. Let \( H^* \) be the extended Herbrand base of \( P \), and \( M \) be an intuitionistic structure w.r.t. \( H^* \). Then \( M \) is an intuitionistic model of \( P \), if \( M \) satisfies all Herbrand instantiated embedded implications of \( R \) and Herbrand instantiated integrity constraints \( IC \). We say the program \( P \) is consistent, if \( P \) has intuitionistic models. Otherwise, the program \( P \) is inconsistent.

### 5.3. Hypothetical answers

In this section, we define hypothetical answers to queries against an abductive program. A hypothetical answer is of the form \( A \leftarrow B_1, \ldots, B_k \). Intuitively, hypothetical answers of this form assert that \( A \) would be true whenever the associated hypotheses \( B_1, \ldots, B_k \) were added into the program, provided \( B_1, \ldots, B_k \) are consistent with that program. Before formally defining the notion of hypothetical answers to a query \( Q \), we need to determine what hypotheses are associated with a hypothetical answer. Intuitively, we would like to define hypotheses associated with an atom \( A \) as a set of embedded hypotheses \( B_1, \ldots, B_k \) such that \( A \) can be deductively derived whenever \( B_1, \ldots, B_k \) were added into the program. First, we introduce notions of embedded hypotheses associated with atoms, and embedded hypotheses associated with embedded Horn rules.

**Definition 5.7.** Let \( R \) be a set of embedded implications, and \( R^* \) be the Herbrand instantiation of \( R \). Then a set \( H_A \) of embedded hypotheses associated with a ground atom \( A \) w.r.t. \( R \) is recursively defined as follows.

1. If \( A \in R^* \), then the empty set \( H_A = \emptyset \) is the only set of embedded hypotheses associated with \( A \) w.r.t. \( R \); and
2. If \( A \leftarrow \psi_1, \ldots, \psi_k \in R^* \), and for each \( j = 1, \ldots, k \), \( R \models \psi_j \) and \( H_{\psi_j} \) is a set of embedded hypotheses associated with \( \psi_j \) w.r.t. \( R \), then \( H_A = \cup_j H_{\psi_j} \) is a set of embedded hypotheses associated with \( A \) w.r.t. \( R \). The embedded hypotheses \( H_\psi \) associated with an embedded Horn rule \( \psi \equiv A \leftarrow B_1, \ldots, B_k \) w.r.t. \( R \) are defined as follows. If \( R \cup \{ B_1, \ldots, B_k \} \models i_A \), and \( H_A \) is a set of embedded hypotheses associated
with $A$ w.r.t. $R \cup \{B_1, \ldots, B_k\}$, then $H_\psi = H_A \cup \{B_1, \ldots, B_k\}$ is a set of embedded hypotheses associated with $\psi$ w.r.t. $R$.

Bonner [3,4] has proposed the following proof system for embedded implications (without integrity constraints).

**Definition 5.8.** Let $R$ be a set of embedded implications. Let $R^*$ be the Herbrand instantiation of $R$. Denote by $R \vdash_1 \psi$ the fact $\psi$ is derivable from $R$. Then

1. $R \vdash_1 A$, if $A \in R^*$, for any ground atom $A$;
2. $R \vdash_1 A$, if $A \leftarrow \psi_1, \ldots, \psi_k$ is an embedded implication in $R^*$, and $R \vdash_1 \psi_j$, for each $j$;
3. $R \vdash_1 A \leftarrow B_1, \ldots, B_k$, if $R \cup \{B_1, \ldots, B_k\} \vdash_1 A$, for any ground atoms $A$ and $B_j$.

For an atom or embedded Horn rule $\psi$, a derivation sequence from Bonner’s proof system for $\psi$ w.r.t. $R$ is a finite sequence $D_1, \ldots, D_k \equiv R \vdash_1 \psi$ such that every $D_j$ is derived using one of Bonner’s inference rules and results obtained before $D_j$.

It has been shown that this proof system is sound and complete [3,4], that is $R \vdash_1 \psi$ iff $R \models_1 \psi$ for any set of embedded implications $R$ and any ground embedded implication $\psi$. It is easy to see that the definition of embedded hypotheses associated with atoms or embedded Horn rules is exactly based on Bonner’s proof system. In other words, for any atom (or embedded Horn rule) $\psi$, $B_1, \ldots, B_k$ are embedded hypotheses associated with $\psi$ if and only if $\psi$ can be hypothetically derived using Bonner’s proof system and $B$'s are embedded hypotheses used in this derivation. Note that some embedded hypotheses associated with an atom (or embedded Horn rule) might be redundant, since they can be deductively derived from other embedded hypotheses. To remove redundant hypotheses from the hypotheses set, we introduce a notion of deductive derivation.

**Definition 5.9.** Let $R$ be a set of embedded implications. Let $R^*$ be the Herbrand instantiation of $R$. Let $\psi$ be a ground atom or embedded Horn rule. Then we say $\psi$ is deductively derived from $R$, denoted by $R \vdash_d \psi$, if $\psi$ satisfies the following conditions.

1. $R \vdash_d A$, if $A \in R^*$, for any ground atom $A$;
2. $R \vdash_d A$, if $A \leftarrow \psi_1, \ldots, \psi_k$ is an embedded implication in $R^*$, and $R \vdash_d \psi_j$, for each $j$;
3. $R \vdash_d A \leftarrow B_1, \ldots, B_k$, if $R \vdash_d A$, and $R \vdash_d B_j$ for each $j$, where $A$ and $B_j$ are any ground atoms.

For a ground atom or embedded Horn rule $\psi$, a $d$-derivation sequence for $\psi$ from $R$ is a finite sequence $D_1, \ldots, D_k \equiv R \vdash_d \psi$ such that every $D_j$ is deductively derived using one of the inference rules and results obtained before $D_j$.

Intuitively, $R \vdash_d A$ means that $A$ can be deductively derived from $R$ without introducing new hypotheses. Note that the last inference rule for deductive derivation is different from the third inference rule in Bonner’s proof system, since this rule says that $A \leftarrow B_1, \ldots, B_k$ can be deductively derived from $R$, only if $A$ can be deductively derived from the database $R$ where all embedded hypotheses $B_1, \ldots, B_k$ can also be deductively derived.
Using the notion of embedded hypotheses associated with an atom (or embedded Horn rule) and the deductive system, we can define hypotheses associated with an atom (or embedded Horn rule) as follows.

**Definition 5.10.** Let $R$ be a set of embedded implications. Let $\psi$ be a ground atom or embedded Horn rule, and $B_1, \ldots, B_k$ be ground atoms. Then we say $\{B_1, \ldots, B_k\}$ is a set of hypotheses associated with $\psi$ w.r.t. $R$, if $\psi$ has a set $H$ of associated embedded hypotheses such that (i) $\{B_1, \ldots, B_k\} \subseteq H$ and (ii) each $A \in H$ is deductively derived from $R \cup \{B_1, \ldots, B_k\}$.

Associated with a derivation sequence for an atom or embedded Horn rule from Bonner's proof system, several sets of hypotheses can be generated. We illustrate by an example the embedded hypotheses and hypotheses associated with atoms.

**Example 5.1.** Consider the following embedded implications

$$R = \{E \leftarrow (D \leftarrow F); A \leftarrow (B \leftarrow C); F \leftarrow G; D \leftarrow F, A; B \leftarrow C; G\}.$$  

1. **$H_C = \emptyset$:** for $C$ w.r.t. $R \cup \{C, F\}$, since $C \in R \cup \{C, F\}$.
2. **$H_B = \emptyset$:** for $B$ w.r.t. $R \cup \{C, F\}$, since $(B \leftarrow C) \in R \cup \{C, F\}$ and $H_C = \emptyset$ w.r.t. $R \cup \{C, F\}$.
3. **$H_\psi = H_B \cup \{C\} - \{C\}:**
   - For $\psi = B \leftarrow C$ w.r.t. $R \cup \{F\}$, since $R \cup \{C, F\} \models B$, and $H_B = \emptyset$ w.r.t. $R \cup \{C, F\}$.
4. **$H_A = \{C\}:**
   - For $A$ w.r.t. $R \cup \{F\}$, since $(A \leftarrow (B \leftarrow C)) \in R \cup \{F\}$ and $H_\psi = \{C\}$ w.r.t. $R \cup \{F\}$.
5. **$H_F = \emptyset$:** for $F$ w.r.t. $R \cup \{F\}$, since $F \in R \cup \{F\}$.
6. **$H_D = \{C\}:**
   - For $D$ w.r.t. $R \cup \{F\}$, since $(D \leftarrow F, A) \in R \cup \{F\}$, $H_A = \{C\}$ and $H_F = \emptyset$ w.r.t. $R \cup \{F\}$.
7. **$H_{\psi_1} = H_D \cup \{F\} = \{F, C\}:**
   - For $\psi_1 = D \leftarrow F$ w.r.t. $R$, since $R \cup \{F\} \models D$, and $H_D = \{C\}$ w.r.t. $R \cup \{F\}$.
8. **$H_E = \{F, C\}:**
   - For $E$ w.r.t. $R$, since $(E \leftarrow (D \leftarrow F)) \in R$ and $H_{\psi_1} = \{F, C\}$ w.r.t. $R$.

So, $\{C, F\}$ is a (unique) set of embedded hypotheses associated with $E$. It is trivial to see that $\{C, F\}$ is also a set of hypotheses associated with $E$. Next, we show that $F$ is deductively derived from $R \cup \{C\}$, since

1. $G$ is deductively derived from $R \cup \{C\}$, since $G \in R \cup \{C\}$; and
2. $F$ is deductively derived from $R \cup \{C\}$, since $(F \leftarrow G) \in R \cup \{C\}$ and $1$.

So, we get two sets $\{C\}$ and $\{F, C\}$ of hypotheses associated with $E$. 
Before defining hypothetical answers, we give several basic results on the notions developed above. The results are very useful in the proofs for other main theorems.

Lemma 5.1. Let $P = (R, IC)$ be an abductive program. Let $\psi$ be a ground atom or embedded Horn rule. Let $\{A_1, \ldots, A_n\}$ be a set of ground atoms. If $R \cup \{A_1, \ldots, A_n\} \vdash_d \psi$ and $R \vdash_d A_j$ for each $j$, then $R \vdash_d \psi$.

Proof. The proof is a straightforward induction on the length of $d$-derivation of $\psi$. □

Next, we define the notion of hypothetical answers to a query $Q$ using the notions developed above.

Definition 5.11. Let $P = (R, IC)$ be an abductive program. Then a query $Q$ against the program $P$ is a goal clause of the form $\leftarrow p(\bar{x})$, where $p$ is any database predicate and $\bar{x}$ is a $n$-tuple of free individual variables. A valid (hypothetical) answer to $Q$ against $P$ is defined as a ground embedded Horn rule $p(\bar{d}) \leftarrow B_1, \ldots, B_k$, $k \geq 0$ such that $B_1, \ldots, B_k$ are hypotheses associated with $p(\bar{d})$ w.r.t. $R$, and $(R \cup \{\neg \neg B_1, \ldots, \neg \neg B_k\}, IC)$ is consistent.

Since $B_1, \ldots, B_k$ are hypotheses associated with $p(\bar{d})$, it is trivial to see that $p(\bar{d})$ is deductively derived from $R \cup \{B_1, \ldots, B_k\}$. Among valid answers to the query $Q$, we are only interested in those answers whose hypotheses provide essential but non-redundant information. This intuition can be captured by the notion of minimality of hypotheses.

Definition 5.12. Let $P = (R, IC)$ be a consistent abductive program, and $Q \equiv \leftarrow p(\bar{x})$ be a query. Let $p(\bar{d}) \leftarrow B_1, \ldots, B_k$ be a valid answer to $Q$ against $P$. Then the set of hypotheses $\{B_1, \ldots, B_k\}$ is minimal, provided for any valid answer $p(\bar{d}) \leftarrow B'_1, \ldots, B'_m$ of $Q$, if $R \cup \{B_1, \ldots, B_k\} \vdash_d B'_j$ for each $j = 1, \ldots, m$, then $R \cup \{B'_1, \ldots, B'_m\} \vdash_d B_l$ for each $l = 1, \ldots, k$.

The hypothetical answers then can be defined as valid answers with minimal hypotheses.

Definition 5.13. Let $P = (R, IC)$ be an abductive program, and $Q \equiv \leftarrow p(\bar{x})$ be a query. Then $p(\bar{d}) \leftarrow B_1, \ldots, B_k$ is a hypothetical answer to $Q$ against $P$, provided $p(\bar{d}) \leftarrow B_1, \ldots, B_k$ is a valid answer to $Q$ and the set of hypotheses $\{B_1, \ldots, B_k\}$ is minimal.
Example 5.2. Consider the following embedded implications.

\[
R = \{ F \leftarrow (D \leftarrow F); A \leftarrow (B \leftarrow C); F \leftarrow G; D \leftarrow F, A; B \leftarrow C \},
\]

\[
R' = R \cup \{ G \},
\]

\[
IC = \{ \leftarrow F, C \}.
\]

Let query \( Q \) be \( \leftarrow E \). Then \( (E \leftarrow C, F) \) is the unique hypothetical answer to \( Q \) against embedded implications \( R \) (See Example 5.1). Both \( E \leftarrow C \) and \( (E \leftarrow C, F) \) are valid answers to \( Q \) against \( R' \), but the hypothesis set \( \{ C, F \} \) is not minimal. So, \( E \leftarrow C \) is the unique hypothetical answer to \( Q \) against embedded implications \( R' \). \( E \leftarrow C \) is also the unique hypothetical answer to \( Q \) against the program \((R', IC)\), but \( Q \) has no answer against the program \((R, IC)\).

5.4. Fixpoint semantics

In this section, we develop a fixed point semantics for embedded implications with integrity constraints. First, we define a transformation \( T_p \) from \( \mathcal{P}(\mathcal{H}^*) \) into \( \mathcal{P}(\mathcal{H}^*) \). This operator \( T_p \) is similar to that defined in McCarty [23, 24]. However, a straightforward definition of \( T_p \) will result in an operator that is not monotone.

**Definition 5.14.** Let \( P = (R, IC) \) be a consistent abductive program, and \( M \) be an intuitionistic structure of \( P \). Then for every state \( s \in M \), \( s \in T_p(M) \) iff the following conditions are satisfied:

- if every superstate \( s' \) of \( s (s \neq s') \) in \( M \) satisfies \( P \), then for every Herbrand instantiated embedded implication \( A \leftarrow \psi_1, \ldots, \psi_k \) of \( R \), if \( s, M \models \psi_j \) for each \( j = 1, \ldots, k \), then \( s, M \models A \);
- for every Herbrand instantiated integrity constraint \( \leftarrow L_1, \ldots, L_k \) of \( IC \), there is some \( L_j \) such that \( s, M \not\models L_j \), where \( 1 \leq j \leq k \).

In fact, the operator \( T_p \) is a deletion operator such that a state \( s \) is deleted by the operator if all its super-states satisfy \( P \), but \( s \) does not, that is, \( s \) does not satisfy an embedded implication of \( R \) or an integrity constraint of \( IC \). Note that in the first condition of the operator \( T_p \), we require that all super-states satisfy \( P \). Without this condition, we cannot generate a correct fixpoint. Using this operator \( T_p \), we can define a sequence of intuitionistic structures of \( P \).

**Definition 5.15.** Let \( P = (R, IC) \) be a consistent abductive program. Then a sequence \( M_0, M_1, \ldots \) of intuitionistic structures of \( P \) is defined as follows: (1) \( T_p \downarrow 0 = \mathcal{P}(\mathcal{H}^*) \); and (2) \( T_p \downarrow (n + 1) = T_p(T_p \downarrow n) \).

Since \( T_p \downarrow n \) is a decreasing sequence of intuitionistic structures, \( T_p \) must have a greatest fixpoint which is an intuitionistic model \( \mathcal{M} \) of \( P \). Furthermore, all hypothetical answers of \( P \) can be interpreted using this intuitionistic model \( \mathcal{M} \) in the
Deductive databases with incomplete information

following way: A \iff B_1, \ldots, B_k is a hypothetical answer against P iff there is a set \( \tilde{H} \) of embedded hypotheses associated with A such that \( \{B_1, \ldots, B_k\} \) is a subset of \( \tilde{H} \) and \( \tilde{H} \) are deductively derived from \( R \cup \{B_1, \ldots, B_k\} \). Our main theorem of this section establishes this result.

**Theorem 5.1.** Let \( P=(R, IC) \) be a consistent abductive program. Then (i) For some \( n, M_n=gfp(T_p) \), and (ii) \( g(gfp(T_p)) \) is an intuitionistic model \( \mathcal{M} \) of \( P \) with a unique minimal state.

**Proof.** (1) Since \( T_p \) is a deletion operator, it is trivial to see that \( M_0, M_1, \ldots \) is a monotonic decreasing sequence, i.e., \( M_i \supseteq M_{i+1} \), for \( i=0, 1, \ldots \). Due to the finiteness of \( \mathcal{P}(H^*) \), there must exist some finite \( n \) such that \( M_n=gfp(T_p) \). Let \( \mathcal{M} \) be the intuitionistic structure \( gfp(T_p) \). Since \( P \) is a consistent program, let us prove that \( \mathcal{M} \) is nonempty. Among all intuitionistic models of \( P \), let \( s \) be a maximal state of some intuitionistic model of \( P \) such that for any intuitionistic model \( N \) of \( P \), \( N \) does not have a distinguished state \( s^* \) which is a superstate of \( s \) (\( s \neq s^* \)). Then we can prove that for any intuitionistic structure \( M \), if \( s \in M \), then \( s \in gfp(T_p) \). Consider the following cases.

(i) \( s \) is a maximal state of \( M \). Then \( s \in gfp(T_p) \) is satisfied iff \( s \) satisfies \( P \) w.r.t. \( M \). It is true, since \( s \) is a maximal state of an intuitionistic model \( N \) of \( P \) and \( s \) is also a maximal state of \( M \). (ii) \( s \) is not a maximal state of \( M \). Then there must exist at least one super-state \( s^* \) of \( s \) in \( M \) such that \( s^* \) does not satisfy \( P \). So, \( s \) satisfies the first condition of the operator \( T_p \). For the second condition, suppose that there exists an integrity constraint \( \leftarrow L_1, \ldots, L_m \) such that \( s, M \models \leftarrow L_j \) for each \( j \). Note that \( s \) is a maximal state of an intuitionistic model \( N \) of \( P \). So, for each \( j \), if \( s, M \models \leftarrow L_j \), then \( s, N \models \leftarrow L_j \). So, \( N \) does not satisfy this integrity constraints of IC w.r.t. \( M \). Hence, \( s \in gfp(T_p) \).

Since \( \mathcal{M} \) is nonempty, obviously \( \mathcal{M} \) is an intuitionistic model of \( P \), since \( \mathcal{M} \) satisfies all Herbrand instantiated embedded implications of \( R \) and all Herbrand instantiated integrity constraints of IC.

(2) Next, prove that \( \mathcal{M} \) has a unique minimal state. Suppose that \( \mathcal{M} \) has several minimal states \( s_1, \ldots, s_m \). Let \( s \) be the intersection of \( s_1, \ldots, s_m \), and \( \mathcal{M}' \) be \( \mathcal{M} \cup \{s\} \). Then, consider the following two cases.

(2.1) For each Herbrand instantiated embedded implication \( A \iff \psi_1, \ldots, \psi_k \) of \( R \), if \( s \), \( \mathcal{M}' \models \leftarrow \psi_j \) for each \( j \), then \( s \models \psi_j \) for \( l=1, \ldots, m \) and each \( j \). So, \( A \in s_l \) for each \( l \). Hence, \( A \in s \) due to \( s \) being the intersection of \( s_1, \ldots, s_m \).

(2.2) For some Herbrand instantiated integrity constraint \( \leftarrow L_1, \ldots, L_k \) of IC, suppose that \( s, \mathcal{M}' \models \leftarrow L_j \) for each \( j \). Then \( s \models \psi_j \) for each \( l=1, \ldots, m \) and each \( j \). Then \( \mathcal{M} \) is not an intuitionistic model of \( P \). Contradiction. So, \( \mathcal{M}' \) is also an intuitionistic model of \( P \).

Furthermore, since \( s \in M_0, s \in gfp(T_p(M_0)) = M_1 \), due to (2.1) and (2.2). In the same way, we can show that \( s \in gfp(T_p(M_j)) = M_{j+1} \) for \( j=1, \ldots, n \). Hence, \( s \in \mathcal{M} \). Contradiction. \( \square \)

Theorem 5.1 generalizes the result due to McCarty [23] to the class of embedded implications with integrity constraints.
6. Intuitionistic interpretation of datalog\textsuperscript{\textcircled{a}} programs

In this section, we develop an intuitionistic fixpoint semantics for datalog programs with nulls. Our results establish a close relationship between embedded implications useful for hypothetical reasoning and deductive databases with nulls. Specifically, we show that the semantics of deductive databases with nulls can be established in a hypothetical reasoning framework.

6.1. Intuitionistic interpretation

Due to the Theorem 3.1, we know that every higher-order datalog\textsuperscript{\textcircled{a}} program can be reduced to a (many-sorted) first-order datalog\textsuperscript{\textcircled{a}} program. Without loss of generality, in this section, we only show the equivalence between first-order datalog\textsuperscript{\textcircled{a}} programs and embedded implications. The equivalence between higher-order datalog\textsuperscript{\textcircled{a}} programs and embedded implications can be obtained by transforming the higher-order datalog\textsuperscript{\textcircled{a}} programs into their many-sorted versions.

To capture the correct meaning of null values, we need equality and membership predicates to satisfy the following rewriting axioms for every n-ary predicate \( p \):

\[
A_1: p(\bar{Y}) \leftarrow (p(\bar{X}) \iff \bar{X} = \bar{Y}), \quad \text{and} \\
A_2: p(\bar{Y}) \leftarrow (p(\bar{Y}) \iff \epsilon(\bar{Y}, \omega)), \quad \text{for each rel-null constant} \ \omega.
\]

The exact meaning of these axioms has been discussed in Section 5.2.

**Definition 6.1.** Let \( P = (\Pi, \mathcal{O}, IC) \) be a first-order datalog\textsuperscript{\textcircled{a}} program. Then we say \( \mathcal{P} = (R, IC') \) is the abductive program associated with \( P \), where \( R = \Pi \cup \{ A_1, A_2 \} \cup \{ \text{axioms on equality and membership predicates} \} \), and \( IC' = IC \cup DCA \cup UNA \).

It turns out that the semantics of first-order datalog\textsuperscript{\textcircled{a}} programs can be captured using an extended form of Herbrand interpretations. The notion of extended Herbrand base and the notion of Herbrand instantiations of first-order datalog\textsuperscript{\textcircled{a}} programs are defined exactly as in Definition 5.3. The notions of extended Herbrand interpretations and Herbrand models are then defined in the usual manner.

Since in this section we are only interested in datalog\textsuperscript{\textcircled{a}} programs which satisfy Domain Closure axiom, instead of arbitrary domain \( U \), it is sufficient to consider \( U \) as the set of individual constants occurring in \( P \). It turns out that first-order datalog\textsuperscript{\textcircled{a}} programs semantically can be reduced to their Herbrand instantiations. The following lemma formally states this fact.

**Lemma 6.1.** Let \( P \) be a datalog\textsuperscript{\textcircled{a}} program, and \( P^* \) be the Herbrand instantiation of \( P \). Let \( D \) be the set of individual constants occurring in \( P \). Let \( \mathcal{F} \) be any formula of \( P \). Then for each semantic structure \( \mathcal{M} = (D, h, I) \) of \( P \), there exists a corresponding extended Herbrand interpretation \( \mathcal{M} \) of \( P^* \) which satisfies \( UNA \) and axioms on equality and
memberships, and vice versa, such that for all variable assignments \( v \), \( \mathcal{H} \models \mathcal{F}[\mathcal{H} \circ v] \) iff \( \mathcal{M} \models \mathcal{F}[v] \).

**Proof.** We define an extended Herbrand interpretation \( \mathcal{M} \) for \( P^* \) as follows.

\[
\mathcal{M} = \{ (p(\overline{a})) | h(\overline{a}) \in \mathcal{H}(p) \text{ for all predicate symbols} \}
\]

\[
\cup \{ \varepsilon(\overline{a}, \overline{\omega}) | h(\overline{a}) \in \mathcal{H}(\overline{\omega}) \text{ for all rel-nulls } \overline{\omega} \}
\]

\[
\cup \{ p(\overline{\omega}) | p(\overline{\omega}) \in R \text{ and } p(\mathcal{H}(\overline{\omega})) \subseteq \mathcal{H}(p) \}.
\]

Vice versa, for each extended Herbrand interpretation \( \mathcal{M} \) of \( P^* \) which satisfies axioms on equality predicate, we define a semantic structure \( \mathcal{N} = (D, h, I) \) as follows.

\[
h(d) = c \quad \text{if } d = c \in \mathcal{M} \text{ and } c \text{ is a normal constant;}
\]

\[
h(\bot_j) = \bot \quad \text{if } \bot_j \in \mathcal{M};
\]

\[
\mathcal{N}(\overline{\omega}) = \{ \langle \overline{a} \rangle | \varepsilon(\overline{a}, \overline{\omega}) \in \mathcal{M} \};
\]

\[
\mathcal{N}(p) = \{ p(\overline{a}) | p(\overline{a}) \in \mathcal{M} \}.
\]

Let \( p(\overline{t}) \leftarrow q_1(\overline{t}_1), \ldots, q_k(\overline{t}_k) \) be any datalog rule of \( \Pi \). Then \( \mathcal{F}[v] \) is a Herbrand instantiated datalog rule of \( P^* \). Let \( \overline{v} \) be the extension of \( v \) such that it agrees with \( v \) on individual variables, mapping individual constants to the constants themselves. Then \( \mathcal{M} \models (p(\overline{t}) \leftarrow q_1(\overline{t}_1), \ldots, q_k(\overline{t}_k))[v] \) iff \( p(\overline{v}(\overline{t})) \in \mathcal{M} \) whenever \( q_j(\overline{v}(\overline{t}_j)) \in \mathcal{M} \) for each \( j \).

Hence, \( \mathcal{H} \models (p(\overline{t}) \leftarrow q_1(\overline{t}_1), \ldots, q_k(\overline{t}_k))[\mathcal{H} \circ v] \). For the axiom \( p(\overline{X}) \leftarrow p(\overline{\omega}), \varepsilon(\overline{X}, \overline{\omega}) \).

\( \mathcal{M} \models (p(\overline{X}) \leftarrow p(\overline{\omega}), \varepsilon(\overline{X}, \overline{\omega}))[v] \) iff \( \varepsilon(\overline{X}) \in \mathcal{M} \) whenever \( p(\overline{\omega}) \in \mathcal{M} \), and \( \varepsilon(\overline{X}, \overline{\omega}) \in \mathcal{M} \). It is true iff \( \mathcal{H}(\varepsilon(\overline{X})) \in \mathcal{H}(\overline{\omega}) \) whenever \( p(\mathcal{H}(\overline{\omega})) \subseteq \mathcal{H}(p) \), and \( \mathcal{H}(\varepsilon(\overline{X}, \overline{\omega})) \in \mathcal{H}(\overline{\omega}) \). Hence, \( \mathcal{H} \models (p(\overline{X}) \leftarrow p(\overline{\omega}), \varepsilon(\overline{X}, \overline{\omega}))[\mathcal{H} \circ v] \). In the same way, it is straightforward to see for any integrity constraint and any completion axiom, the claim is correct.

Furthermore, we will show that the Herbrand instantiation \( P^* \) of a (first-order) datalog program \( P \) can be reduced further to a group of “pure” datalog programs such that the extended Herbrand models of \( P^* \) can be generated from Herbrand models of those reduced datalog programs. Those “pure” datalog programs are obtained from all sets of ground equality atoms which satisfy axioms on equality predicate.

**Definition 6.2.** Let \( E \) be any set of ground equality atoms such that \( E \) satisfies UNA and the axioms on equality predicate. Let \( P^* \) be a Herbrand instantiated (first-order) datalog program. Define a transformation on \( P^* \) w.r.t. \( E \) as follows. Replace each individual null constant \( \bot \) by a normal constant \( c \), if \( \bot = c \in E \); otherwise, replace each individual null constant \( \bot_j \) by another null constant \( \bot_i \), if \( \bot_j = \bot_i \in E \) and \( i < j \). Denote by \( \mathcal{F}_E(P^*) \) the transformed (Herbrand instantiated) program. For any extended
Herbrand model $M$ of $P^*$, denote by $\mathcal{T}_E(M)$ the transformed extended Herbrand interpretation of $P^*$ w.r.t. $E$. For any Herbrand model $N$ of the transformed program $\mathcal{T}_E(P^*)$, the extension of $N$ w.r.t. $E$, denoted by $M_E(N)$, is the following set: $M = E \cup \{ p(\bar{d}) | p(\bar{d}) \in N, \text{ and } E \text{ implies } \bar{d}' = \bar{d} \}$.

The following lemma asserts the semantic equivalence between a Herbrand instantiated datalog* program and its reduced datalog program.

Lemma 6.2. Let $P^*$ be a Herbrand instantiated datalog* program. Let $E$ be a set of ground equality atoms such that $E$ satisfies $UNA$ and the axioms on equality predicate. Then (i) for any extended Herbrand model $M$ of $P^*$ such that $E$ is the set of all equality atoms satisfied by $M$, $FE(M)$ is a Herbrand model of $FE(P^*)$; and (ii) for any Herbrand model $N$ of $FE(P^*)$, $M_E(N)$ is an extended Herbrand model of $P^*$ such that $E$ is the set of all equality atoms satisfied by $M_E(N)$.

Proof. (i) Let $r = p(\bar{d}) \leftarrow q_1(\bar{a}_1), \ldots, q_k(\bar{a}_k)$ be a datalog rule of $\mathcal{T}_E(P^*)$ and $r' = p(\bar{d}') \leftarrow q_1(\bar{a}_1'), \ldots, q_k(\bar{a}_k')$ be the datalog rule of $P^*$ such that $r = FE(r')$. Suppose that $q_j(\bar{a}_j) \in \mathcal{T}_E(M)$ for each $j$. Then there exists an atom $q_j(\bar{a}_j') \in \mathcal{T}_E(M)$ such that $q_j(\bar{a}_j') = \mathcal{T}_E(q_j(\bar{a}_j))$. So, $\{ \bar{a}_j = \bar{a}_j \}$, $\{ \bar{a}_j = \bar{a}_j^* \}$ and $\{ \bar{a}_j^* = \bar{a}_j^* \}$ are included in $E$ for each $j$ (since $E$ satisfies axioms on equality predicate). Hence, $q_j(\bar{a}_j') \in M$ for each $j$. So, $p(\bar{d}') \in M$. Hence, $p(\bar{d}) \in \mathcal{T}_E(M)$.

(ii) Let $r = p(\bar{d}) \leftarrow q_1(\bar{a}_1), \ldots, q_k(\bar{a}_k)$ be a datalog rule of $P^*$ and $r' = p(\bar{d}') \leftarrow q_1(\bar{a}_1'), \ldots, q_k(\bar{a}_k')$ be the datalog rule of $\mathcal{T}_E(P^*)$ such that $r'$ is transformed from $r$. Suppose that $q_j(\bar{a}_j) \in M_E(N)$ for each $j$. Then there exists an atom $q_j(\bar{a}_j^*) \in N$ such that $\{ \bar{a}_j = \bar{a}_j^* \} \subseteq E$. Since $\{ \bar{a}_j = \bar{a}_j \} \subseteq E$, $\{ \bar{a}_j = \bar{a}_j^* \} \subseteq E$. Hence, $\bar{a}_j$ and $\bar{a}_j^*$ are syntactically equal ground tuples. Hence, $q_j(\bar{a}_j) \in N$ for each $j$. So, $p(\bar{d}) \in N$. Hence, $p(\bar{d}) \in M_E(N)$. □

The main result of this section is that any (minimal) conditional answer to a query $Q$ against a datalog* program $P$ is a hypothetical answer to the query $Q$ against the abductive program $\mathcal{P}$ associated with $P$.

Theorem 6.1. Let $P = (H, \mathcal{A}, IC)$ be a first-order datalog* program, and $\mathcal{P} = (R, IC')$ be the abductive program associated with $P$. Then for any query $Q \equiv \leftarrow p(\bar{X})$, $(d, E)$ is a conditional answer to the query $Q$ against $P$ iff $p(\bar{d}) \Rightarrow E$ is a hypothetical answer to the query $Q$ against $\mathcal{P}$, where $E$ is a minimal set of consistent equality and membership conditions.

Proof. The proof is based on the relationship between Herbrand model characterizations of embedded implications and first-order datalog* programs. Let $P_E$ be the datalog* program $(H \cup E, \mathcal{A}, IC)$. Note that $(d, E)$ is a conditional answer against $P$ iff $P_E \models p(\bar{d})$ and $P_E$ is consistent, provided $E$ is minimal.
First, we would like to show that \( \phi(d) = 4 \) is a valid answer against \( P \cup E \) iff \( \phi(d) \) and \( P \) is consistent.

By Lemma 6.1, for all semantic structures \( \mathcal{M} \) which satisfy \( P \) with a variable assignment \( v \), \( \mathcal{M} \) satisfies \( \phi(d) \) with the variable assignment \( v \) iff for all extended Herbrand models \( \mathcal{M} \) of \( P \), \( \mathcal{M} \models \phi(d) \) and \( P \) is consistent. By Lemma 6.2, \( P \models \phi(d) \) iff \( \mathcal{F}_P(p(z)) = p(d) \), where \( E^* \) are all equality atoms which can be deductively derived from \( P \), and \( p(d') = \mathcal{F}_P(p(d)) \). Since \( \mathcal{F}_P(p(z)) \) can be viewed as a normal datalog program, \( p(d') \) belongs to the least fixed point of \( \mathcal{F}_P(p(z)) \). Note that \( p(d') \) belongs to the least fixed point of \( \mathcal{F}_P(p(z)) \) iff \( p(d) \) belongs to the least fixed point of \( P \), where the equality predicate is viewed as a normal predicate. This can be true iff there exists a derivation sequence for \( p(d) \) against \( P \cup E \) using the first two inference rules of Bonner's proof system. This can be true iff \( p(d) \) and \( E \) is a valid hypothetical answer against \( P \).

If \( p(d) \) is a valid hypothetical answer to \( Q \) against \( P \), then \( p(d) \) is a valid hypothetical answer against \( P \cup E \). Then \( (d, E) \) is a valid conditional answer of \( P \) due to the proof above.

If \( (d, E) \) is a valid answer against \( P \), then by the proof above, there exists a derivation sequence \( D_1, \ldots, D_n \) from \( R \) using Bonner's proof system such that \( p(d) \) is hypothetically derived from \( P \), and all the embedded hypotheses used in the derivation sequence can be deductively derived from \( P \cup E \). Generate a derivation sequence \( D_1, D_2, \ldots \) for \( p(d) \) against \( R \) from Bonner's proof system as follows by induction on the length of \( m \).

**Base case:** \( D_1 \equiv R \cup E \models \psi \). If \( \psi \models R \), then, \( R \models \psi \). If \( \psi \models E \), then, \( \psi \) is of the form \( d = d' \).

Then, using a rewriting axiom \( X = Y \implies (X = Y \Leftrightarrow X = Y) \), \( R \models d = d' \).

**Inductive step:** Suppose the claim is true for any atom or embedded Horn rule \( \psi \) with a \( d \)-derivation sequence for \( \psi \) of length \( \leq n \). Suppose that \( p(d) \) is deductively derived from \( R \cup E \) with a \( d \)-derivation sequence of length \( n + 1 \). Then the following cases arise.

**Case 1:** Suppose \( D_{n+1} \equiv R \cup E \models \psi_j \) is hypothetically derived using a Herbrand instantiated embedded implication \( p(d) \models \psi_1, \ldots, \psi_k \). Then each \( \psi_i \) has a derivation sequence of length \( \leq n \) from \( R \cup E \). By inductive hypothesis, \( R \models \psi_j \) for each \( j \). Hence, \( R \models p(d) \).

**Case 2:** Suppose \( D_{n+1} \equiv R \cup E \models \psi \) and \( \psi \) is a ground embedded Horn rule of the form \( A = B_1, \ldots, B_k \). Then \( B_1, \ldots, B_k \) derivation sequences of length \( \leq n \) from \( R \cup E \), respectively. By inductive hypothesis, \( R \models A \). So, \( R \cup \{B_1, \ldots, B_k\} \models A \). Hence, \( R \models \psi \).

Hence, \( p(d) \) has derivation sequence from Bonner's proof system. Furthermore, all embedded hypotheses used in the derivation sequence can be either deductively derived from \( R \cup E \) or belong to \( E \). So, \( E \) is a set of hypotheses associated with \( p(d) \). Hence, \( p(d) \) is a valid hypothetical answer.

Note that according to the proof above, all valid hypothetical answers to \( Q \) against \( P \) are also valid conditional answers to \( Q \) against \( P \). But the converse is not true, since the conditions associated with an arbitrary valid answer against \( P \) may include
redundant conditions. However, for any conditional answers, their associated condition sets are always non-redundant, and they do correspond to hypothetical answers. (see Definition 5.13).

6.2. Fixpoint semantics

We use fixpoint semantics to establish the computational relationship between existing query processing techniques and the proof procedure for deductive databases with nulls. Our fixpoint semantics is defined by an operator $T_P$ over the extended Herbrand base, which is very similar to that widely used in logic programming. With every ground atom, we associate a set of conditions to state that atom is derivable whenever the associated conditions are added into the database. Our fixpoint semantics implicitly suggests that the query processing against deductive databases with nulls can be achieved using existing query answering strategies in the domain of logic programming and databases. Indeed, Dong and Lakshmanan [8] proposed both top-down and bottom-up approaches to query answering against deductive databases with nulls. Those strategies show a justification for implementing query answering based on existing approaches. The semantics developed in this paper provides an alternative semantical foundation underlying those strategies.

The operator $T_P$ is defined over extended Herbrand base, augmented with a set of conditions associated with every ground atom. A ground atom $A$ with the associated conditions $E$ asserts the derivability of $A$ hypothetically depends on $E$. Whenever a ground atom $A$ is hypothetically derived using a rule $A \leftarrow B_1, \ldots, B_k$, we can verify if the conditions associated with all the subgoals $B_i$'s are consistent. $A$ can be a conditional answer only if all the conditions are consistent.

To define the operator $T_P$, we first introduce the notion of constrained Herbrand base.

**Definition 6.3.** Let $P$ be a first-order datalog$^*$ program, and $D$ be the set of all the constants occurring in $P$. Let $\mathcal{P}$ be the abductive program associated with $P$, and $H^*$ be the extended Herbrand base of $\mathcal{P}$. Let $\mathcal{E}$ be the power set of all the equality conditions of the form $l = d$, and all the membership conditions of the form $s(d; w)$, where $l$ is a null constant of $D$, $s$ is a rel-null of $D$, $d$ is an individual constant of $D$, and $w$ is a null constant of $D$. Then the constrained Herbrand base $H_E$ of $\mathcal{P}$ is defined to be the Cartesian product of $H^* \times E$. We refer to a pair $(A, E) \in H^* \times E$ as a constrained ground atom, and denote it as $(A | E)$ for convenience.

The operator $T_P$ is defined to be the mapping from $\mathcal{P}(H_E)$ to $\mathcal{P}(H_E)$.

**Definition 6.4.** Let $P$ be a first-order datalog$^*$ program, $\mathcal{P} = (R, IC')$ be the abductive program associated with $P$, and $R^*$ be the Herbrand instantiation of $R$. Let $H_E$ be the constrained Herbrand base of $\mathcal{P}$, and $M_E$ be an element of $\mathcal{P}(H_E)$. Then, for any ground atom $A$ associated with a set $E$ of conditions, $(A | E) \in T_P(M_E)$ iff
there is some datalog rule $A \leftarrow B_1, \ldots, B_k$ in $R^*$, and there exist constrained ground atoms $(B_j | E_j) \in M_E$, for $1 \leq j \leq k$ such that $E = E_1 \cup \cdots \cup E_k$, where $(R, IC' \cup E)$ is consistent; or

- there is some embedded implication $A \leftarrow (A \equiv E')$ in $R^*$, and some $(A | E') \in M_E$ such that $E - E' \cup E''$, where $(R, IC' \cup E)$ is consistent; or

- there exists a constrained ground atom $(A | E') \in M_E$ such that $E \subseteq E'$ and $E'$ are deductively derived from $R \cup E$.

We define a sequence of constrained interpretations of $P$ as follows:

$M_0 = \{ (\emptyset | \emptyset) \}$; where $\emptyset$ denotes the empty set;

$M_{j+1} = T_p(M_j) \cup \{ (\emptyset | \emptyset) \}$.

We have the following theorem establishing the relationship between conditional answers (hypothetical answers) and fixpoint semantics. This theorem shows that a ground atom $A$ associated with a set $E$ of conditions can be a conditional answer against the datalog" program, only if $E$ is consistent and $A$ can be derived when $E$ is added to the program.

**Theorem 6.2.** Let $P=(\Pi, \mathcal{A}, IC)$ be a first-order datalog" program, and $\mathcal{P}$ be the abductive program associated with $P$. Then (i) for some $n$, $M_n = \text{lfp}(T_p)$; (ii) for every constrained ground atom, $(A | E) \in M_n$ iff $A \equiv E$ is a valid hypothetical answer against $P$, where $A$ is a ground atom, and $E$ is a set of consistent equality and membership conditions.

**Proof.** (i) Since the constrained Herbrand base $H_E$ is finite, $M_0, M_1, \ldots$, is monotonic increasing sequence, i.e. $M_i \subseteq M_{i+1}$ for $i=0,1,\ldots$, there is a finite $n$ such that $M_n = \text{lfp}(T_p)$.

(ii) "\Rightarrow" Let $\neg \neg E$ be the set $\{ \neg \neg B | B \in E \}$. We would like to show that for any constrained atom $(A | E)$, if $(A | E) \in M_n$ for some $n$, then for each intuitionistic model $\mathcal{M}$ of $\mathcal{P}$, $\mathcal{M}$ satisfies $A$ whenever $\mathcal{M}$ satisfies $\neg \neg E$. If so, $A \equiv E$ is a valid hypothetical answer of $\mathcal{P}$, provided $E$ is consistent with $\mathcal{P}$. The proof is by induction on the iterative times $n$ of $T_p$.

**Base case:** For each constrained atom $(A | \emptyset)$ in $M_1$, $A$ is ground atom in $\Pi$. Trivially, for each intuitionistic model $\mathcal{M}$ of $\mathcal{P}$, $\mathcal{M}$ satisfies $A$.

**Inductive step:** Assume that for each constrained atom $(A | E) \in M_j (j \leq n)$, for each intuitionistic model $\mathcal{M}$ of $\mathcal{P}$, $\mathcal{M}$ satisfies $A$ whenever $\mathcal{M}$ satisfies $\neg \neg E$. Suppose that $(A | E)$ is a constrained atom such that $(A | E) \in M_{n+1}$. Consider the following cases.

**Case 1:** There is a (Herbrand instantiated) datalog rule $A \leftarrow B_1, \ldots, B_k$ of $R$ such that $(R_j | F_j) \in M_n$ for each $j$ and $F = F_1 \cup \cdots \cup F_k$ is consistent. By inductive hypothesis, for each intuitionistic model $\mathcal{M}$ of $\mathcal{P}$, $\mathcal{M}$ satisfies $B_j$ whenever $\mathcal{M}$ satisfies $\neg \neg E_j$, and $E_j$ is a set of hypotheses associated with $B_j$. Then for each intuitionistic model $\mathcal{M}$ of $\mathcal{P}$ such that $\mathcal{M}$ satisfies $\neg \neg E$, trivially $\mathcal{M}$ satisfies each $B_j$. So, $\mathcal{M}$ satisfies each $B_j$, and $E$ is a set of hypotheses associated with $A$ (since embedded hypotheses associated
with each $B_j$ are embedded hypotheses associated with $A$, and all these embedded hypotheses are deductively derived from $R \cup E$).

Case 2: There is some (Herbrand instantiated) embedded implication $A \iff (A \Leftarrow E')$ such that $(A \mid E') \in M_n$, and $E = E' \cup E''$ is consistent. By inductive hypothesis, for each intuitionistic model $M$ of $\mathcal{P}$, $M$ satisfies $A$ whenever $M$ satisfies $\neg \neg E'$. Then for each intuitionistic model $M$ of $\mathcal{P}$, $M$ satisfies $A$ whenever $M$ satisfies $E$. So, $A \Leftarrow E$ is a valid hypothetical answer against $\mathcal{P}$.

Case 3: There is some constrained atom $(A \mid E') \in M_n$ and $E$ is a subset of $E'$ such that $E'$ are deductively derived from $R \cup E$. By inductive hypothesis, for each intuitionistic model $M$ of $\mathcal{P}$, $M$ satisfies $A$ whenever $M$ satisfies $E'$. For the set of embedded hypotheses associated with $A$, they are deductively derived from $R \cup E'$. Trivially, they are deductively derived from $R \cup E$ as well, due to Lemma 5.1. Hence, $A \Leftarrow E$ is a valid hypothetical answer against $\mathcal{P}$.

"\Leftarrow" This result is proved based on the relationship between the operator $T_P$ and Bonner's proof system: $(A \mid E) \in \wp(T_P)$ iff $R \vdash A$ with associated hypotheses $E$. Let $\mathcal{P} = (\mathcal{R}, IC')$ be the abductive program associated with $P$, where $\mathcal{R} = \Pi \cup \{\text{rewriting axioms } A_1 \text{ and } A_2 \} \cup \{\text{axioms on equality and membership predicates}\}$, and $IC' = IC \cup \text{DCA} \cup \text{UNA}$.

Suppose $A \Leftarrow E$ is a valid (hypothetical) answer against $\mathcal{P}$.

$\Leftarrow$ There exists a derivation sequence $D_1, \ldots, D_k \equiv R \vdash A$ from Bonner's proof system such that (1) $E$ is a subset of the embedded hypotheses $E^*$ used at all derivation steps and $E^*$ are deductively derived from $R \cup E$; (2) for each $D_j$, (2.1) $D_j$ is of the form $R \cup E_j \vdash \psi_j$, where $E_j \subseteq E^*$ and $\psi_j$ is an atom or embedded Horn rule hypothetically derived using one of Bonner's inference rules and results hypothetically derived before $D_j$; and (2.2) every $\psi_j$ is associated with a set $E_j$ of equality and membership conditions w.r.t. $R \cup E_j$ such that $E_j$ is a subset of the embedded hypotheses $E_j$ used at all derivation steps for $D_j$ w.r.t. $R \cup E_j$, and $E_j$ are deductively derived from $R \cup E_j \cup E_j$; and (3) for each $j$, $(R \cup E_j \cup E_j, IC')$ is satisfiable (since $E_j \cup E_j$ are deductively derived from $R \cup E$).

So, the remaining issue is to prove that for each $j$

(i) If $\psi_j \equiv A_j$ is an atom, and $D_j$ is of the form $R \vdash A_j$, then $(A_j \mid E_j) \in M_n$ for some $n$; and

(ii) If $D_j \equiv R \vdash B \Rightarrow E'$ where $\psi_j \equiv B \Rightarrow E'$ is an embedded Horn rule, then $(B \mid E_j) \in M_n$ for some $n$ and $E_j \cup E'$ are deductively derived from $R \cup E$.

Proof is by induction on the length $k$ of derivation sequence for $A$.

Base case: $k = 1$. Then $D_1 = R \vdash A$ is derived using Bonner's first inference rule, where $A$ is an EDB fact of $R$. So, $E = E_1 = \phi$, and $(A \mid \emptyset) \in M_1$.

Inductive step: Assume that the results (i) and (ii) are true for any atom or embedded Horn rule $\psi$ which has a derivation sequence $D_1, \ldots, D_m \equiv R \vdash \psi$ of length $m \leq k$ from Bonner's proof system.

Suppose that $\psi$ is an atom or an embedded Horn rule such that $\psi$ has a derivation sequence $D_1, \ldots, D_k, D_{k+1} \equiv R \vdash \psi$. $E$ is a set of hypotheses associated with $\psi$ such that
the embedded hypotheses occurring at all derivation steps are deductively derived from $R \cup E$, and $(R \cup E, IC')$ is satisfiable.

Consider the following cases: (1) $\psi \equiv A$ is an atom, and hypothetically derived using Bonner's inference rule 2 with a datalog rule $A \leftarrow B_1, \ldots, B_n$. So, for every $B_j$, there is a $D_m, m \leq k$ such that $D_m \equiv R \models B_j$. By inductive hypothesis, $(B_j | E_j) \in M_{nj}$, for some $n_j$, and $E_j$ is a set of hypotheses associated with $B_j$ such that all the embedded hypotheses used at all derivation steps for $B_j$ are deductively derived from $R \cup E_j$, and $(R \cup E_j, IC')$ is satisfiable. So, $E_1 \cup \cdots \cup E_k$ is a set of hypotheses associated with $A$ such that the embedded hypotheses occurring at all derivation steps for $A$ are deductively derived from $R \cup E_1 \cup \cdots \cup E_k$, and $(R \cup E_1 \cup \cdots \cup E_k, IC')$ is satisfiable (since $E_1 \cup \cdots \cup E_k$ are deductively derived from $R \cup E$). So, by the first step of $T_p$, $(A | E_1 \cup \cdots \cup E_k) \in M_n, n = 1 + \max\{n_1, \ldots, n_k\}$. If $E = E_1 \cup \cdots \cup E_k$, then we get the result. If $E \subseteq E_1 \cup \cdots \cup E_k, (E_1 \cup \cdots \cup E_k)$ are deductively derived from $R \cup E$. So, by the third step of $T_p$, we have $(A | E) \in M_{n+1}$.  

(2) $\psi \equiv A$ is an atom, and hypothetically derived using Bonner's inference rule 2 with a rewriting axiom $A \leftarrow A \leftarrow E'$. Let $D_k$ be of the form $R \models A \leftarrow E'$. By inductive hypothesis, $(A | E') \in M_n$ for some $n$, where $E'$ are embedded hypotheses associated with $A$, such that $E \subseteq E' \cup E'$ and $E'' \cup E'$ are deductively derived from $R \cup E$. Let $E^*$ be $E'' \cup E'$. Then, by second step of $T_p$, $(A | E^*) \in M_{n+1}$ $(R \cup E^*, IC')$ is satisfiable. Next, by the third step of $T_p$, we obtain $(A | E) \in M_{n+2}$.  

(3) $\psi \equiv A \leftarrow E'$ is an embedded Horn rule, and hypothetically derived using Bonner's inference rule 3.  

(3.1) Let $D_k$ be of the form $R \cup E_k \models A$, and $A$ has associated hypotheses $E_k$. Note that equality and membership predicates occur as subgoals only in embedded Horn rules. So, $D_k \equiv R \cup E_k \models A$ iff $A$ has a derivation sequence $D'_1, \ldots, D'_m (m \leq k)$ from Bonner's proof system such that $D'_m \equiv R \models A$ with the same set of associated hypotheses $E_k$. By inductive hypotheses, $(A | E_k) \in M_n$ for some $n$, and $(R \cup E_k, IC')$ is satisfiable. So, $\psi$ is hypothetically derived using Bonner's inference rule 3, and $E_k \cup E'$ are deductively derived from $R \cup E$. \[\square\]

Theorem 6.2 generalizes the classical result due to Van Emden and Kowalski, to the class of datalog programs with nulls. Note that this fixpoint semantics makes use of an operator $T_p$ which is similar to that defined in logic programming, augmented with a consistency checking module. Several efficient consistency checking algorithms have been extensively studied in the domain of constraint logic programming. It can also be implemented based on techniques as discussed in [8]. We omit the details on consistency checking.

Example 6.1. Let us revisit the example in Section 2 and consider the query $Q \equiv \leftarrow \text{supports}(f_2, r)$. First, we introduce an embedded implication associated with every predicate $\text{good-for}, \text{implemented}$ and $\text{supports}$. Thus, we get three more embedded implications $r_{10}, r_{11}$ and $r_{12}$.  


F. Dong, L.V.S. Lakshmanan

\[
\begin{align*}
   r_1: & \quad \text{good}_\text{for}(b, s) \\
   r_2: & \quad \text{good}_\text{for}(b, r) \\
   r_3: & \quad \text{good}_\text{for}(m, bl) \\
   r_4: & \quad \text{good}_\text{for}(h, s) \\
   r_5: & \quad \text{good}_\text{for}(\bot_2, r) \\
   r_6: & \quad \text{implemented}(f_1, h) \\
   r_7: & \quad \text{implemented}(f_2, \bot_1) \\
   r_8: & \quad \text{implemented}(f_3, \bot_2) \\
   r_9: & \quad \text{supports}(F, Q) \leftarrow \text{implemented}(F, S), \text{good}_\text{for}(S, Q) \\
   r_{10}: & \quad \text{good}_\text{for}(Y_1, Y_2) \leftarrow \text{good}_\text{for}(X_1, X_2) \iff X_1 = Y_1, X_2 = Y_2 \\
   r_{11}: & \quad \text{implemented}(Y_1, Y_2) \leftarrow \text{implemented}(X_1, X_2) \iff X_1 = Y_1, X_2 = Y_2 \\
   r_{12}: & \quad \text{supports}(Y_1, Y_2) \leftarrow \text{supports}(X_1, X_2) \iff X_1 = Y_1, X_2 = Y_2 \\
   IC: & \quad \{ \bot_1 \neq \bot_2 \} \\
\end{align*}
\]

Note that the resulting program \( P \) consists of embedded implications \( r_1, \ldots, r_{12} \) as well as axioms on equality predicate, the constraint \( IC \), and the axioms \( DCA \) and \( UNA \). To the query \( Q \), we apply the constrained operator \( T_P \) to generate conditional answers:

1. \( M_0 = \{(\emptyset)\emptyset\} \)
2. \( M_1 = \{(r_1)\emptyset, \ldots, (r_8)\emptyset\} \) by applying the operator \( T_P \) to all rules;
3. \( (\text{good}_\text{for}(\bot_1, r)\{\bot_1 = b, r = r\}) \in M_2 \) by applying Step 2 to \( r_{10} \) and \( (r_2)\emptyset \);
4. \( (\text{good}_\text{for}(\bot_1, r)\{\bot_1 = b\}) \in M_2 \) by applying Step 3 to 3;
5. \( (\text{supports}(f_2, r)\{\bot_1 = b\}) \in M_3 \) by applying Step 1 to \( r_9, (r_7)\emptyset \) and 4.

So we get a conditional answer \( (\epsilon, \{\bot_1 = b\}) \) to the query, where \( \epsilon \) denotes a tuple of zero length, corresponding to the answer "yes". We next consider a recursive example.

**Example 6.2.** Consider the conditional answers to the query \( Q \equiv \leftarrow \text{ancestor}(\text{paul}, Y) \), from the following program:

\[
\begin{align*}
   r_1: & \quad \text{ancestor}(X, Y) \leftarrow \text{parent}(X, Z), \text{ancestor}(Z, Y). \\
   r_2: & \quad \text{ancestor}(X, Y) \leftarrow \text{parent}(X, Y). \\
   r_3: & \quad \text{parent}(\bot_1, \text{jim}). \\
   r_4: & \quad \text{parent}(\text{paul}, \bot_1). \\
   r_5: & \quad \text{parent}(\bot_2, \text{george}). \\
   r_6: & \quad \text{parent}(\text{john}, \text{joe}). \\
   r_7: & \quad \text{parent}(Y_1, Y_2) \leftarrow \text{parent}(X_1, X_2) \iff X_1 = Y_1, X_2 = Y_2. \\
   r_8: & \quad \text{ancestor}(Y_1, Y_2) \leftarrow \text{ancestor}(X_1, X_2) \iff X_1 = Y_1, X_2 = Y_2. \\
   IC: & \quad \{ \bot_1 \neq \bot_2, \bot_1 \neq \text{jim}, \bot_1 \neq \text{paul}, \bot_2 \neq \text{george} \}. \\
\end{align*}
\]

Let \( P \) be the resulting program consisting of rules \( r_1 \) to \( r_8 \) as well as axioms on equality predicate, and the constraint \( IC \) including the axioms \( DCA \) and \( UNA \). Then there are three conditional answers to the query \( Q: \| Q \|_P = \{ ((\text{joe}), \{ \bot_1 = \text{john} \}) \}, \)
Deductive databases with incomplete information

Here, we only show the procedure for getting the first answer to the query $Q$.

1. $M_0 = \{ (\emptyset, \emptyset) \} $; 
2. $M_1 = \{ (r_3, \emptyset), \ldots, (r_6, \emptyset) \}$ by applying the operator $T_p$ to all rules; 
3. $(parent(\_1, joe) | \_1 = john, joe = joe) \in M_2$ by applying Step 2 to $r_7$ and $(r_6, \emptyset)$; 
4. $(parent(\_1, joe) | \_1 = john) \in M_2$ by applying Step 3 to 3; 
5. $(ancestor(\_1, joe) | \_1 = john) \in M_3$ by applying Step 1 to $r_2$ and 4 
6. $(ancestor(paul, joe) | \_1 = john) \in M_4$ by applying Step 1 to $r_1, (r_4, \emptyset)$ and 5.

So, we get the conditional answer $((\_1, john), \{ \_1 = john \})$ to the query $Q$. In a similar way, we can get the other conditional answers to $Q$.

7. Hypothetical reasoning and datalog

There have been several works dealing with the semantic foundations and proof procedures of hypothetical reasoning. An interesting question from a practical viewpoint is how to realize hypothetical query answering systems effectively. We contend that the technology of deductive databases offers an attractive framework in which to realize hypothetical query answering. This section substantiates this claim by illustrating a methodology for transforming embedded implications into deductive databases with null values.

Given a set of embedded implications, the resulting database is equivalent to the embedded implications in the sense that the transformed database produces the same (hypothetical) answers to queries. We illustrate by example how embedded implications can be transformed into a datalog program.

7.1. Hypothetical reasoning via higher-order predicates

Hypothetical query answering against embedded implications (with integrity constraints) is a repeated process of reasoning through multiple databases. Each of these databases is obtained by adding hypotheses as EDB facts into existing databases as long as the resulting database is consistent. To transform embedded implications with integrity constraints into a (higher-order) datalog program, we need to represent by higher-order predicates all possible sets of hypotheses and all possible databases. Intuitively, we can represent a set of hypotheses as a labeled relation, and use the label as the identity to that set of hypotheses. Note that we have to distinguish between different sets of hypotheses by distinguished labels.

We first introduce two higher-order predicate symbols $identity$ and $labeled$ to provide a simple way for expressing and constructing all possible sets of hypotheses.
Let $P$ be an abductive program without nulls, and $D$ be the set of individual constants occurring in $P$. For every ground atom $p(\bar{c})$, we introduce a distinguished predicate symbol $q$ and use $q$ as the identity for the atom $p(\bar{c})$. This fact is represented by a higher-order predicate \textit{identity} $(p(\bar{c}), q)$. In the same way, we also use a distinguished predicate symbol $q$ as the identity for each set of ground atoms $\{C_1, \ldots, C_k\}$. Different sets of atoms are distinguished by distinct identity labels (predicate symbols). A higher-order predicate \textit{labeled} $(\{C_1, \ldots, C_k\}, q)$ is used to assert that the label $q$ is the unique identity for the set of ground atoms $\{C_1, \ldots, C_k\}$. In particular, we assume that the higher-order logic $\mathcal{L}$ has a specific predicate symbol $\emptyset$ which is used to denote empty set.

We formalize these predicates as the abbreviations of higher-order formulas as follows.

\begin{align*}
\text{identity}(p(\bar{X}), P) & \equiv \text{identity}(p, P) \land p(\bar{X}) \land P. \\
\text{labeled}(\{C_1, \ldots, C_k\}, P) & \equiv \text{identity}(C_1, P_1) \land \ldots \land \text{identity}(C_k, P_k) \\
& \land \text{labeled}_k(P_1, \ldots, P_k, P).
\end{align*}

Here, the predicate \textit{labeled}_k(P_1, \ldots, P_k, P) asserts that the predicate labeled by $P_1, \ldots, P_k$ forms a labeled relation with the label $P$. Here, $P$ represents a predicate variable of arity 0. Furthermore, the identity label $q$ for an atom $p(\bar{c})$ has to satisfy the following rules:

\begin{align*}
p(\bar{c}) & \leftarrow q. \\
q & \leftarrow p(\bar{c}). \\
\text{identity}(p, q) & \leftarrow p(\bar{c}).
\end{align*}

These rules ensure that only $q$ can be the identity for the atom $p(\bar{c})$. In the same way, every EDB fact $p(\bar{o})$ with a rel-null $\bar{o}$ is also associated with a predicate symbol $q$ as its identity label. Similarly, for any set of ground atoms $\{C_1, \ldots, C_k\}$ labeled by $q$, where each $C_i$ is labeled by $q_i$, we have the following rules and an integrity constraint associated with all the atoms and their label $q$:

\begin{align*}
\text{labeled}_k(q_1, \ldots, q_k, q) & \leftarrow \text{identity}(C_1, q_1) \land \ldots \land \text{identity}(C_k, q_k). \\
q(q_j) & \leftarrow \text{labeled}_k(q_1, \ldots, q_k, q), \text{ for each } j. \\
& \leftarrow \text{labeled}_k(q_1, \ldots, q_k, q), q(P), P \neq q_1 \land \ldots \land P \neq q_k.
\end{align*}

The first rule asserts that the set of ground atoms $\{C_1, \ldots, C_k\}$ is associated with a predicate symbol $q$ as its identity label. The second one states that if \textit{labeled}_k(P_1, \ldots, P_k, P) is satisfied, then every $P_j$ is labeled by $P$. The last integrity constraint asserts that every set of ground atoms $\{C_1, \ldots, C_k\}$ has a unique predicate symbol $q$ as its identity label. We call these rules \textit{axioms on identity labels}, and these integrity constraints \textit{integrity constraints on identity labels}. For each ground atom $p(\bar{c})$, we say $q$ is the \textit{identity label} for $p(\bar{c})$, provided $q$ and $p(\bar{c})$ satisfy the associated axioms and integrity constraints on identity labels. For each set of ground atoms
\(\{C_1, \ldots, C_k\}\), we say \(q\) is the identity label for \(\{C_1, \ldots, C_k\}\), if \(q\) and \(\{C_1, \ldots, C_k\}\) satisfy the associated axioms and integrity constraints on identity labels.

Given any two sets of ground atoms which are identified using distinguished labels \(H_1\) and \(H_2\), we make use of a predicate \(\text{union}(H_1, H_2, H)\) to assert that \(H\) is the identity label for the union of the two sets of atoms labeled by \(H_1\) and \(H_2\). Similarly, we make use of a predicate \(\text{diff}(H_1, H_2, H)\) to assert that \(H\) is the identity label for the difference of the two sets labeled by \(H_1\) and \(H_2\) (intuitively, \(H = H_1 - H_2\)). The predicate \(\text{subset}(H_1, H)\) states that the set of atoms labeled by \(H_1\) is a subset of the set of atoms labeled by \(H\) (intuitively, \(H_1 \subseteq H\)). A formal description of these predicates follows:

\[
\text{union}(H_1, H_2, H) \equiv \forall P \left[ H(P) \leftrightarrow H_1(P) \lor H_2(P) \right].
\]

\[
\text{diff}(H_1, H_2, H) \equiv \forall P \left[ H(P) \leftrightarrow H_1(P) \land \neg H_2(P) \right].
\]

\[
\text{subset}(H_1, H) \equiv \forall P \left[ H(P) \leftrightarrow H_1(P) \right].
\]

For the sake of completeness, we have specified these predicates (identity, labeled, union, diff and subset) using higher-order formulas. We remark that efficient implementation can be obtained by directly implementing these predicates, much like evaluable predicates.

To represent hypothetical reasoning, we need several more higher-order predicate symbols – hypo, holds, emh and d-derived. Intuitively, the predicate \(\text{hypo}(A)\) asserts that the atom \(A\) is a hypothesis. The predicate \(\text{holds}(A, DB, H)\) is used to state that the atom \(A\) holds w.r.t. a database (embedded implications) \(R \cup DB\) with associated hypotheses \(H\), i.e., \(R \cup DB \models A\) and \(H\) is a set of hypotheses associated with \(A\) w.r.t. \(R \cup DB\). The predicate \(\text{emh}(\text{hypo}\{C_1, \ldots, C_n\}, P)\) is used to represent that atoms \(\{C_1, \ldots, C_n\}\) is a set of embedded hypotheses labeled by \(P\). The predicate \(\text{d-derived}(DB, H^*, H)\) is used to assert that \(H\) is a subset of \(H^*\) such that embedded hypotheses \(H^*\) are deductively derived from \(R \cup DB \cup H\).

Note that these predicates are not legal forms of the higher-order logic \(\mathcal{L}\). We formalize these predicates as the abbreviations of higher-order formulas of \(\mathcal{L}\) as follows.

\[
\text{holds}(p(\bar{X}), DB, H) \equiv \text{id}(p(\bar{X}), P) \land \text{holds}(P, DB, H).
\]

\[
\text{hypo}(p(\bar{X})) \equiv \text{id}(p(\bar{X}), P) \land \text{hypo}(P).
\]

\[
\text{emh}(\text{hypo}\{C_1, \ldots, C_n\}, P) \equiv \text{hypo}(C_1), \ldots, \text{hypo}(C_n), \text{labeled}(\{C_1, \ldots, C_n\}, P).
\]

Here \(p\) is a predicate symbol, and \(\bar{X}\) is a tuple of individual variables. Note that in the definitions of the predicates holds and hypo, a new predicate variable \(P\) is introduced to represent the identity label for the atom \(p(\bar{X})\). In the following section, we will use these predicates to illustrate a method for transforming embedded implications into a higher-order datalog program with nulls. We remark the symbols used for the above predicates could be arbitrary. The suggestive names are used mainly for clarity.
7.2. Transformation process

Instead of developing a new proof system to deal with embedded implications with integrity constraints, we would like to use existing query answering techniques as an underlying framework for hypothetical query answering against embedded implications with integrity constraints. The main issue concerning this idea is how to transform embedded implications into a datalog program. Since the essential difference between reasoning with embedded implications and that present in deductive databases is the ability of deriving new Horn rules, to transform embedded implications into datalog programs, we need somehow to replace embedded Horn rules by other predicates and make sure these predicate atoms can be derivable only when the embedded Horn rules are. We use the higher-order language $\mathcal{L}'$ as a vehicle language to represent transformed embedded implications. In this paper, we illustrate the transformation procedure by an example. Complete details on this transformation can be found in [11]. We have since developed a more elegant algorithm for this transformation using HiLog as the target language for DDBs with nulls.

Example 7.1. Consider an application of circuit design. Suppose $S$ is a designed circuit with inputs $A$ and $B$ and output $D$. We are now to synthesize a new circuit using $S$ as one of the modules. The inputs are two natural numbers $A$ and $B$ and the output is another natural number $F$. Fig. 1 shows this circuit. Note that $E$ is a boolean which is true if $B \leq 3$. The switch acts as follows: $F$ equals $D$, provided $E$ is true; otherwise, it is “floating”. Suppose that a design goal is to find out all possible values of the inputs $A$ and $B$ such that $F = 14$. The design knowledge as well as system description of the circuit can be expressed using the following rules together with appropriate integrity constraints.

- **System description**

$$C(X) \leftarrow A(Y), B(Z), X = Y + Z.$$  

$$D(X) \leftarrow A(Y), C(Z), X = Y + Z.$$  

![Fig. 1. Circuit design.](image)
- Design knowledge

\[ F(X) \leftarrow \exists YZ [(D(X) \leftarrow A(Y), B(Z)), (E() \leftarrow B(Z))]. \]

(Embedded implication)

\[ E() \leftarrow B(X), X \leq 3. \]

- Integrity constraints

\[ \leftarrow A(X), A(Y), X \neq Y. \]

\[ \leftarrow B(X), B(Y), X \neq Y. \]

The embedded implication in the design knowledge expresses that: the output at \( F \) will be \( X \), provided for some appropriate values \( Y \) and \( Z \), if the inputs at \( A \) and \( B \) are set to \( Y \) to \( Z \) respectively, then the output at \( D \) would be \( X \), and that at \( E \) would be true. The integrity constraints say that each input can only assume one value at a time.

The essential difference between embedded implications and standard DDBs is the ability of deriving new Horn rules. The idea in the transformation is to replace embedded Horn rules by new predicates and ensure the corresponding atoms are deductively derivable exactly when the embedded Horn rule is derivable from the embedded implications. For the embedded implication in the example above, we can introduce two new predicates \( p_1(X) \) and \( p_2(1) \) to replace the two embedded Horn rules. This embedded implication is transformed into the following (higher order) Horn rules \( r_3, r_4 \) and \( r_5 \). The rules in system description are transformed into the (higher order) Horn rules \( r_1 \) and \( r_2 \). The other rule in design knowledge is transformed into the rule \( r_6 \).

\[ r_1: \text{holds}(C(X), DB, H) \leftarrow \text{holds}(A(Y), DB, H_1), \text{holds}(B(Z), DB, H_2), \]

\[ X = Y + Z, \text{union}(H_1, H_2, H). \]

\[ r_2: \text{holds}(D(X), DB, H) \leftarrow \text{holds}(A(Y), DB, H_1), \text{holds}(C(Z), DB, H_2), \]

\[ X = Y + Z, \text{union}(H_1, H_2, H). \]

\[ r_3: \text{holds}(F(X), DB, H) \leftarrow \text{holds}(p_1(X), DB, H_1), \text{holds}(p_2(1), DB, H_2), \]

\[ \text{union}(H_1, H_2, H). \]

\[ r_4: \text{holds}(p_1(X), DB, H) \leftarrow \text{holds}(D(X), DB_1, H_1), \text{emh(hypo \{A(Y), B(Z)\}, Emh), diff(DB_1, Emh, DB), union(H_1, Emh, H)}. \]

\[ r_5: \text{holds}(p_2(1), DB, H) \leftarrow \text{holds}(E(), DB_1, H_1), \text{emh(hypo \{B(X)\}, Emh), diff(DB_1, Emh, DB), union(H_1, Emh, H)}. \]

\[ r_6: \text{holds}(E(), DB, H) \leftarrow \text{holds}(B(X), DB, H), X \leq 3. \]

\[ r_7: \text{holds}(A(X), P, \emptyset) \leftarrow \text{hypo(A(X))), labeled(\{A(X)\}, P)}. \]

\[ r_8: \text{holds}(B(X), P, \emptyset) \leftarrow \text{hypo(B(X)), labeled(\{B(X)\}, P)}. \]

\[ r_9: \text{hypo}(A(\omega_1)). \]

\[ r_{10}: \text{hypo}(B(\omega_2)). \]

\[ r_{11}: \text{hypo}(A(X)) \leftarrow \text{hypo}(A(\omega_1)), e(X,\omega_1). \]

\[ r_{12}: \text{hypo}(B(X)) \leftarrow \text{hypo}(B(\omega_2)), e(X,\omega_2). \]

\[ A_1: \text{holds}(A, DB, H) \leftarrow \text{holds}(A, DB, H'), d-derived(DB, H', H). \]
$A_2$: \[ d\text{-derived}(DB, H^*, H) \leftarrow \text{subset}(H, H^*), \]
\[ \text{union}(H, DB, DB^*), \text{holds}(H^*, DB^*, \emptyset). \]

$A_3$: \[ \text{holds}(A, DB, H) \leftarrow \text{holds}(A, DB', H'), \text{subset}(DB', DB), \text{diff}(H', DB, H). \]

Note that the embedded hypotheses $A(Y)$ and $B(Z)$ are asserted into the resulting DDB (with nulls) as EDB facts hyp$(A(\omega_1))$ and hyp$(B(\omega_2))$ (rules $r_9$ and $r_{10}$), which state that for some constants $\omega_1$ and $\omega_2$, $A(\omega_1)$ and $B(\omega_2)$ are hypotheses. Intuitively, the rule $r_{11}$ asserts that if $A(\omega_1)$ is a hypothesis, and $X$ is a constant belonging to the rel-null $\omega_1$, then $A(X)$ is a hypothesis. The rule $r_7$ states that if $A(X)$ is a hypothesis, and $A(X)$ is labeled by $P$, then $A(X)$ holds in the database containing $A(X)$ as an EDB fact. In a similar way, we can interpret the rules $r_{12}$ and $r_8$, respectively. The remaining rules $A_1, A_2$ and $A_3$ are axioms. $A_1$ and $A_2$ establish the relationship between hypothetical reasoning and $d$-derivation as defined in Definition 5.9. The last axiom $A_3$, called inheritance axiom, asserts that if $A$ is true in a database $DB'$ with associated hypotheses $H'$, then $A$ holds in each database $DB$ which is a superset of $DB'$, with the associated hypotheses $H$ and $H'$, minus any new EDB facts in $DB$. The hypothetical answer against $F(14)$ generated in this manner is a statement of the form $F(14) \Leftarrow A(6), B(2)$, which means that $F(14)$ can be derivable, if $A(6)$ and $B(2)$ were added into the database.

Note that a (hypothetical) answer extracted using abductive approach may not be a hypothetical answer associated to our hypothetical reasoning approach, since these former answers may be deduced (abduced) using hypotheses which are not specified as embedded hypotheses. Note also that the size of datalog rules transformed from embedded implications is comparable to the size of $P$, since every newly introduced atom $B_i$ is associated with an embedded Horn rule, and each predicate hyp$(C)$ is associated with a subgoal $C$ occurring in some embedded Horn rule. The number of rewriting axioms is equal to the number of rel-null constants.

Note that in the transformation procedure introduced above, we make use of a (higher-order) rule of the form, for example,

$$\text{holds}(A, DB, H) \leftarrow \text{holds}(B_1, DB, H_1), \text{holds}(B_2, DB, H_2), \text{union}(H_1, H_2, H).$$

This rule does not quite follow the legal syntax of the logic $L$, since its head $\text{holds}(A, DB, H)$ is the abbreviation for $\text{id}(p(\overline{X}), P) \wedge \text{holds}(P, DB, H)$. To legalize this rule, we need somehow to redefine this rule in such a way that both $\text{id}(A, P)$ and $\text{holds}(P, DB, H)$ can be derived separately. Note that $\text{id}(A, P)$ can be further decomposed into three atoms. So, we view this rule as an abbreviation of the following pair of higher-order datalog rules.

$$A \leftarrow \text{holds}(B_1, DB, H_1), \text{holds}(B_2, DB, H_2), \text{union}(H_1, H_2, H).$$

$$\text{holds}(P, DB, H) \leftarrow \text{id}(A, P), \text{holds}(B_1, DB, H_1), \text{holds}(B_2, DB, H_2), \text{union}(H_1, H_2, H).$$

It is easy to see that $\text{holds}(A, DB, H)$ can be derived using the rules above and the rules associated with identity labels (see Section 7.1). Here, the predicate variable...
$P$ represents the identity label for the atom $A$. In a similar way, the rules whose heads are hypo atoms can also be rewritten into legal forms.

We have proved (see Dong and Lakshmanan [12]) that the datalog$^\theta$ program $\text{ddb}(P)$ obtained from embedded implications based on the transformation method above is query equivalent to the original embedded implications. In other words, the hypothetical answers to the queries against $P=(R, IC)$ can be obtained by deductively querying the associated datalog$^\theta$ program $\text{ddb}(P) = (\text{ddb}(R), \alpha^\theta, IC^\theta)$ and extracting conditional answers. We can make use of this methodology to develop a hypothetical query answering strategy against transformed datalog$^\theta$ programs. The approach developed in [10] is a hypothetical answering algorithm based on this method. Indeed, we can view higher-order predicates holds and hypo as structured data, and other higher-order predicates such as deduced and subset, as meta-information instructing the proof-procedure to limit the search space for hypothetical answers.

8. Conclusion

We have proposed an intuitionistic semantics for deductive databases in the presence of incomplete information. We motivated the problem of querying deductive databases containing null values as extracting conditional answers against the database. Null values are treated as mapping functions to match themselves to normal constants, where the mappings respect given constraints on nulls. We developed a fixpoint semantics for deductive databases with nulls. Our results not only characterized the semantics of conditional answers, but also established a computationally close relationship between existing query processing techniques and the proof procedure for the deductive databases containing nulls. Furthermore, we also illustrated a method to transform embedded implications with integrity constraints (with some restrictions) into query-equivalent deductive databases with nulls. This result shows the possibilities of implementing hypothetical reasoning within the existing framework of deductive databases, and of achieving hypothetical query answering based on existing techniques for query processing.

References


