Defaults in Open World Relational Databases

Navin Viswanath
Department of Computer Science,
Georgia State University,
Atlanta, GA 30302
nviswanath1@student.gsu.edu

Rajshekhar Sunderraman
Department of Computer Science,
Georgia State University,
Atlanta, GA 30302
raj@cs.gsu.edu

Abstract

The Open World Assumption(OWA) has been found to be necessary in a number of applications. In this paper, we study the issue of incompleteness in an open world database. We define an extension of the relational model which has two forms of negation - the explicit negation, in which certain atoms are known to be false, and a default negation which is a form of non-monotonic negation for unknown atoms in the relation. We define operators for this extended relational model. We show that this model is a generalization of the relational model in the sense that we obtain some intuitive answers in the negative component in addition to the answers obtained in the relational model.

1 Introduction

Normally, relational databases adopt the Closed World Assumption(CWA) of Reiter [12]. Roughly, the CWA says that given a query Q (a first order sentence) on a database DB, the answers to the query are the logical consequences of DB and Q. Also, we assume as false those atoms for which no proof exists. However, there are certain domains of application where the CWA does not suffice. A typical example is a biological database where it may not be appropriate to assume certain atoms as being false. It then becomes necessary to study what qualifies as an answer to a query. In the case where the database is complete, there is no difficulty. The answers to a query are those for which a proof exists given DB. But when the database is incomplete, query answering is not that straightforward. First order logic can no longer be used and certain non monotonic forms of reasoning have to be adopted. The major forms of non monotonic reasoning are Reiter's CWA [12], Reiter's default logic [11], McCarthy's circumscription [8] and Moore's autoepistemic logic [9]. Relational databases normally adopt the CWA. The reason is that the number of negative facts to be stored become prohibitively large and storing them explicitly is not feasible. But this becomes necessary in certain domains of application and when the knowledge is incomplete, a default form of negation must be used. Logical entailment by itself is limited in application when the knowledge is incomplete. But in common sense reasoning, in practice, we do reason about things that we are not completely aware of. A typical example of such a form of reasoning is the statement "*birds fly*".i.e., in general we tend to assume that all birds fly unless we have strong enough reasons to believe otherwise. Consider a particular bird, say Tweety. We would normally assume that Tweety flies as long as we have no reason to believe otherwise. The pattern of reasoning followed here is "*in the absence of information to the contrary ...*". This form of reasoning is nonmonotonic because if we were to subsequently acquire information to the contrary, then we would have to

L. Magdalena, M. Ojeda-Aciego, J.L. Verdegay (eds): Proceedings of IPMU'08, pp. 212–219
Torremolinos (Málaga), June 22–27, 2008

retract our original beliefs. For example, if we were to discover at a later point in time that Tweety is in fact an ostrich, then we would have to retract our earlier belief that Tweety flies. This problem has been studied in detail from the logic programming and deductive database aspect in [5],[6], [7] and [10]. However, this problem has not been studied extensively from the open world relational database viewpoint. In an open world database, some atoms are explicitly defined and we can make some common sense assumptions about some others. The problem that we address here is to decide about which atoms we can make assumptions within the realm of some of the operators of the relational algebra. The aim of this paper is to define an extension to the relational model and an algebra where two forms of negation are used - an explicit negation and a nonmonotonic form of negation. We show that this model is a generalization of the relational model in the sense that we obtain some intuitive answers in the negative component in addition to the answers obtained in the relational model.

The rest of the paper is organized as follows. Section 2 introduces the notion of default negation in relational databases. Section 3 formally defines default relations and notions of operator generalizations. Section 4 presents the algebraic operators on default relations. Section 5 explains the intuition behind the default conclusions. Section 6 presents an example of query evaluation on default relations. Finally, Section 7 contains concluding remarks and directions for future work.

2 Default Negation in a Relational Database

In relational databases that adopt the CWA, we store only the true facts and other facts are implicitly assumed to be false. In an open world setting, a relation is a pair $< R^+, R^- >$, where R^+ is the positive component which stores facts that are *known to be true* of the relation R, and R^- is the negative component which stores facts that are *known to be false* in R. Thus, unlike the CWA where we implicitly assume facts not stored to be false, we do not make such an assumption in the open world setting. Facts that we believe to be false are only the ones stored in R^-. Such a model is described in [1],[2]. Bagai and Sunderraman in [1] and [2] define an algebra for their paraconsistent relational data model which has these two components. However, apart from the facts that are known to be false, we also want to be able to make default assumptions about certain facts when the relation is incomplete. The model described in [1] uses the four-valued logic of Belnap [3] and assigns the default truth value of *unknown* to the missing facts in the relation. Some facts may also appear in both R^+ and R^- thus making the relation R inconsistent. Such facts are assigned the fourth truth value *overdetermined*.

In this paper, apart from the facts that are known to be true and those that are known to be false in a relation R, we adopt a form of nonmonotonic negation so that some of the unknown facts can be *assumed to be false*. Notice that this is a form of closed world reasoning in an open world setting. It is necessary when the database is incomplete. The form of nonmonotonic reasoning that we will adopt here is closely related to the one described in [13]. The algebra that we define with two kinds of negation is a generalization of the paraconsistent algebra of [1] and [2]. Apart from the explicit positive and negative components of the answers obtained in that model, we extend it to produce more intuitive negative answers that we conclude to be *false by default*. The basic idea is that we define to be *false by default* certain atoms, as yet unknown, whose addition to the corresponding relation would not have changed the positive consequences of the result of applying a relational operation. It must be noted, however that adding them to the R^- component may change the negative consequences of the relational operation.

3 Default Relations

In this section, we construct a set theoretic formulation of our model. In this model, some

tuples are *known* to hold a certain underlying predicate, some are *known not* to hold the predicate and some others are *not known* to hold the predicate.

Let a relation scheme Σ be a finite set of attribute names, where for any attribute name $A \in \Sigma$, dom(A) is a non-empty domain of values for A. A tuple on Σ is a map $t : \Sigma \rightarrow \cup_{A\in\Sigma} dom(A)$, such that $t(A) \in dom(A)$, for each $A \in \Sigma$. Let $\tau(\Sigma)$ denote the set of all tuples on Σ.

Definition 1 *An ordinary relation on scheme Σ is any subset of $\tau(\Sigma)$. We let $\mathcal{O}(\Sigma)$ denote the set of all ordinary relations on Σ.*

Definition 2 *A default relation on scheme Σ is a triple $< R_e^+, R_e^-, R_d^- >$ where R_e^+,R_e^- and R_d^- are any subsets of $\tau(\Sigma)$. We let $\mathcal{D}(\Sigma)$ denote the set of all default relations on Σ.*

Here, the subscript e denotes explicit and the subscript d denotes default. The superscripts + and - denote true and false respectively. Hence the three components are explicitly true, explicitly false and default false respectively.

Intuitively, R_e^+ may be considered as the set of tuples for which R is known to be true, R_e^- is the set of tuples for which R is known to be false and R_d^- is the set of tuples for which R is not known to be true and hence can be assumed to be false by default.

We denote by $\bar{R}$ the set of tuples on scheme Σ that have not been assigned truth values. Thus $\bar{R} = \tau(\Sigma) - (R_e^+ \cup R_e^- \cup R_d^-)$. We say that a tuple t is *unknown in* R if $t \in \bar{R}$.

Definition 3 *A default relation R on scheme Σ is said to be complete if $R_e^+ \cup R_e^- = \tau(\Sigma)$. R is said to be a consistent default relation if $R_e^+ \cap R_e^- = \emptyset$ and $R_e^+ \cap R_d^- = \emptyset$. If R is both consistent and complete, it is said to be total.*

It should be observed that the (positive parts of) total relations are essentially ordinary relations. We make this relationship explicit by defining an operator $\lambda_\Sigma(R) = R_e^+$ where R is a total relation on Σ.

Default relations are a generalization of ordinary relations in the sense that for each ordinary relation there is a default relation with the same information content. We adopt the notions of generalizations discussed in [1].

An *n-ary* operator on ordinary relations with signature $< \Sigma_1, \ldots, \Sigma_{n+1} >$ is a function $\Theta : \mathcal{O}(\Sigma_1) \times \ldots \times \mathcal{O}(\Sigma_n) \rightarrow \mathcal{O}(\Sigma_{n+1})$ where $\Sigma_1, \ldots, \Sigma_{n+1}$ are any schemes. Similarly, an *n-ary* operator on default relations with signature $< \Sigma_1, \ldots, \Sigma_{n+1} >$ is a function $\Psi : \mathcal{D}(\Sigma_1) \times \ldots \times \mathcal{D}(\Sigma_n) \rightarrow \mathcal{D}(\Sigma_{n+1})$.

Definition 4 *An operator Ψ on default relations with signature $< \Sigma_1, \ldots, \Sigma_{n+1} >$ is totality preserving if for any total relations $R_1, \ldots, R_n$ on schemes $\Sigma_1, \ldots, \Sigma_n$ respectively, $\Psi(R_1, \ldots, R_n)$ is also total.*

We associate with a consistent default relation R the set of all relations obtainable from R by throwing in the missing tuples. The completion of a consistent default relation R is given by,

$$comps_\Sigma(R) = \{Q \in \mathcal{O}(\Sigma) \mid R_e^+ \subseteq Q \subseteq \tau(\Sigma) - (R_e^- \cup R_d^-)\}$$

For any operator $\Theta : \mathcal{O}(\Sigma_1) \times \ldots \times \mathcal{O}(\Sigma_n) \rightarrow \mathcal{O}(\Sigma_{n+1})$ on ordinary relations, we let $\Gamma(\Theta) : 2^{\mathcal{O}(\Sigma_1)} \times \ldots \times 2^{\mathcal{O}(\Sigma_n)} \rightarrow 2^{\mathcal{O}(\Sigma_{n+1})}$ be a map on sets of ordinary relations defined as follows: For any sets $M_1, \ldots, M_n$ of ordinary relations on schemes $\Sigma_1, \ldots, \Sigma_n$ respectively,

$$\Gamma(\Theta)(M_1, \ldots, M_n) = \{\Theta(R_1, \ldots, R_n) \mid R_i \in M_i, \forall i, 1 \leq i \leq n\}.$$

Definition 5 *An operator Ψ on default relations with signature $< \Sigma_1, \ldots, \Sigma_{n+1} >$ is consistency preserving if for any consistent default relations $R_1, \ldots, R_n$ on schemes $\Sigma_1, \ldots, \Sigma_n$ respectively, $\Psi(R_1, \ldots, R_n)$ is also a consistent default relation.*

Definition 6 *A consistency preserving operator Ψ on default relations with signature $< \Sigma_1, \ldots, \Sigma_{n+1} >$ is a strong generalization of an operator Θ on ordinary relations with the same signature, if for any consistent relations $R_1, \ldots, R_n$ on schemes $\Sigma_1, \ldots, \Sigma_n$ respectively, we have*

$$comps_{\Sigma_{n+1}}(\Psi(R_1, \ldots, R_n)) = \Gamma(\Theta)(comps_{\Sigma_1}(R_1), \ldots, comps_{\Sigma_n}(R_n)).$$

4 Algebraic Operators on Default Relations

In this section, we present generalizations of each of the algebraic operators on ordinary relations. To reflect generalization, a dot is placed over the ordinary relational operator to obtain the corresponding default relation operator. The operators defined here are extensions of the operators defined in the paraconsistent data model in [1] and [2]. We also state theorems on strong generalization for each of the operators. The proofs are avoided due to space constraints.

Definition 7 *Let R and S be default relations on scheme Σ. The union of R and S, denoted $R \dot{\cup} S$, is a default relation on scheme Σ, given by,*

$$\begin{aligned}(R\dot{\cup}S)_e^+ &= R_e^+ \cup S_e^+ \\ (R\dot{\cup}S)_e^- &= R_e^- \cap S_e^- \\ (R\dot{\cup}S)_d^- &= R_d^- \cap S_d^-\end{aligned}$$

The union operation may be understood as follows: The tuples in the union of R and S are those that possess either the property R or the property S, which is simply the union of the tuples in R_e^+ and S_e^+ . Similarly, the explicit negation of the union is the tuples which have neither property. They are exactly the tuples in $R_e^- \cap S_e^-$. The tuples *not known to possess* property R or S are those that are not known to possess either - which is exactly the set $R_d^- \cap S_d^-$. Among the unknown tuples in $\bar{R}$ and $\bar{S}$, any of those, if added to either of the original relations, would be present in the union as well. Hence none of them can be negated by default.

Theorem 4.1 *The operator $\dot{\cup}$ on default relations is a strong generalization of the operator $\cup$ on ordinary relations.*

Definition 8 *Let R and S be default relations on scheme Σ. The intersection of R and S, denoted $R \dot{\cap} S$, is a default relation on scheme Σ, given by,*

$$\begin{aligned}(R\dot{\cap}S)_e^+ &= R_e^+ \cap S_e^+ \\ (R\dot{\cap}S)_e^- &= R_e^- \cup S_e^- \\ (R\dot{\cap}S)_d^- &= R_d^- \cup S_d^- \cup (\bar{R} \cap \bar{S})\end{aligned}$$

For the intersection operation, the positive component of the intersection will contain exactly those tuples which possess both properties R and S. These are the tuples in $R_e^+ \cap S_e^+$. The tuples in the explicit negative component are those for which it is not the case that they possess properties R and S. i.e. those tuples that either do not possess R or do not possess S. These are the tuples in $R_e^- \cup S_e^-$. The default negative tuples are those tuples that are not known to possess R and S. They include the tuples in $R_d^- \cup S_d^-$. Apart from these, any tuple in $\bar{R}$ which does not appear in S_e^+ will not appear in the intersection even if it were added to R. It will appear in the explicit negative component of the intersection if it was present in S_e^- . Thus we are interested only in tuples that appear in $\bar{R} \cap \bar{S}$. Notice that this holds only if these tuples were to be added separately in R or S. For if any tuple in $\bar{R} \cap \bar{S}$ were to be added to both R and S simultaneously, this tuple would appear in the intersection as well.

Theorem 4.2 *The operator $\dot{\cap}$ on default relations is a strong generalization of the operator $\cap$ on ordinary relations.*

Definition 9 *Let R and S be default relations on scheme Σ. The difference of R and S, denoted $R \dot{-} S$, is a default relation on scheme Σ, given by,*

$$\begin{aligned}(R\dot{-}S)_e^+ &= R_e^+ \cap S_e^- \\ (R\dot{-}S)_e^- &= R_e^- \cup S_e^+ \\ (R\dot{-}S)_d^- &= R_d^- \cup (\bar{R} \cap \bar{S}) \cup (\bar{S} - R_e^-)\end{aligned}$$

The tuples in the difference of R and S are those that are in R and not in S. i.e., in $R_e^+ \cap S_e^-$. The tuples that are known not to be present in the difference are exactly those that are not in R or in S - the tuples in $R_e^- \cup S_e^+$. Any tuple not known to be in R can be assumed to not be in the difference - this is the set R_d^- . Any tuple in $\bar{R} \cap \bar{S}$ would not affect the result of the difference if it were to be added to R or S. Hence they can be assumed to be false by default. Also, the tuples in $\bar{S}$ would not affect the difference even if they were to be added to S. However, some of them already appear in the explicit negative component because they are present in R_e^- .

Thus the tuples from $\bar{S}$ that can be assumed to false in the difference are those in $\bar{S} - R_e^-$.

Theorem 4.3 *The operator $\dot{-}$ on default relations is a strong generalization of the operator $-$ on ordinary relations.*

If Σ and Δ are relation schemes such that $\Delta \subseteq \Sigma$, then for any tuple $t \in \tau(\Delta)$ we let t^Σ denote the set $\{t' \in \tau(\Sigma) \mid t'(A) = t(A), \text{ for all } A \in \Delta\}$ of all extensions of t. We extend this notion for any $T \subseteq \tau(\Delta)$ by defining $T^\Sigma = \bigcup_{t \in T} t^\Sigma$.

Definition 10 *Let R be a default relation on scheme Σ and let F be any formula involving attribute names in Σ, constant symbols (denoting values in the attribute domains), the equality symbol $=$, the negation symbol $\neg$, and the connectives $\wedge$ and $\vee$. Then, the selection of F by R, denoted $\dot{\sigma}_F(R)$, is a default relation on scheme Σ, given by*

$$\begin{array}{lcl} \dot{\sigma}_F(R)_e^+ & = & \sigma_F(R_e^+) \\ \dot{\sigma}_F(R)_e^- & = & R_e^- \cup \sigma_{\neg F}(\tau(\Sigma)) \\ \dot{\sigma}_F(R)_d^- & = & R_d^- \end{array}$$

The positive component of the selection consists of exactly those tuples in R_e^+ that satisfy F. i.e. the tuples that possess property R and satisfy the formula F. The explicitly negated component of a selection includes the set of all tuples in R_e^- since they do not possess property R. Also, tuples in $\tau(\Sigma)$ that do not satisfy F are also explicitly negated. The tuples in R_d^- can be assumed to be false in the selection since they are not known to possess property R.

Theorem 4.4 *The operator $\dot{\sigma}$ on default relations is a strong generalization of the operator σ on ordinary relations.*

Definition 11 *Let R be a default relation on scheme Σ, and $\Delta \subseteq \Sigma$. Then, the projection of R onto Δ, denoted $\dot{\pi}_\Delta(R)$, is a default relation on Δ, given by*

$$\begin{array}{lcl} \dot{\pi}_\Delta(R)_e^+ & = & \pi_\Delta(R_e^+) \\ \dot{\pi}_\Delta(R)_e^- & = & \{t \in \tau(\Delta) \mid t^\Sigma \subseteq R_e^-\} \\ \dot{\pi}_\Delta(R)_d^- & = & \{t \in \tau(\Delta) \mid t^\Sigma \subseteq (R_d^- \cup R_e^-)\} \\ & & -\dot{\pi}_\Delta(R)_e^- \end{array}$$

The positive component of the projection is the projection of tuples in R_e^+ . The explicitly negated component of the projection is those tuples in Δ all of whose extensions are explicitly negated in R. Similarly, the tuples that are unknown in Δ all of whose extensions are in R_d^- can be assumed to be false in the projection. Apart from this, there may be tuples unknown in Δ some of whose extensions are in R_e^- and the others in R_d^- . These tuples can also be assumed to be false by default.

Theorem 4.5 *The operator $\dot{\pi}$ on default relations is a strong generalization of the operator π on ordinary relations.*

Definition 12 *Let R and S be default relations on scheme Σ and Δ respectively. Then, the natural join of R and S, denoted $R \dot{\bowtie} S$, is a default relation on scheme $\Sigma \cup \Delta$, given by*

$$\begin{array}{lcl} (R \dot{\bowtie} S)_e^+ & = & R_e^+ \bowtie S_e^+ \\ (R \dot{\bowtie} S)_e^- & = & (R_e^-)^{\Sigma \cup \Delta} \cup (S_e^-)^{\Sigma \cup \Delta} \\ (R \dot{\bowtie} S)_d^- & = & (R_d^-)^{\Sigma \cup \Delta} \cup (S_d^-)^{\Sigma \cup \Delta} \cup \\ & & \{t^{\Sigma \cup \Delta} \mid (t \in \bar{R} \wedge \{t\} \bowtie S_e^+ = \emptyset) \vee \\ & & (t \in \bar{S} \wedge R_e^+ \bowtie \{t\} = \emptyset)\} \end{array}$$

The positive component of the join is simply the natural join of the positive components of the corresponding relations. The explicitly negated component of the join consists of all extensions of the tuples in R_e^- and S_e^- since these are already not true in R and S respectively. The default negative component consists of all extensions of R_d^- and S_d^- . This component will also contain extensions of tuples in $\bar{R}$ that do not join with any tuple in S_e^+ . Similarly, we can also add all tuples from $\bar{S}$ that do not join with any tuple in R_e^+ .

Theorem 4.6 *The operator $\dot{\bowtie}$ on default relations is a strong generalization of the operator $\bowtie$ on ordinary relations.*

5 Intuition

The aim of this section is to explain the intuition behind how the default conclusions are made for each of the relational operators. Since the semantics of each of the relational operators is clear, we know what kind of inferences can be made at least as far as the

positive conclusions are concerned. For example, we know that the union of two relations is exactly those set of tuples that are known to have either property. In order to derive default conclusions, we are motivated by two reasons - one, we want to minimize the extent of a relation. This is the idea behind nonmonotonic reasoning methods like circumscription [8]. Thus we attempt to derive default negative conclusions in order to minimize the result of an algebraic operation. The question that then arises is on what basis do we minimize? As mentioned earlier, since the semantics of the relational operators are clear, an interesting approach would be to treat this as the definite result of applying the particular relational operator and try to minimize the relation as much as possible while maintaining consistency without introducing any change in the definite answers.This is the second motivation for deriving default conclusions. This approach leads to the obvious question - why not negate every unknown fact as in the CWA? The reason that this approach is unsatisfactory is that we want to differentiate between two kinds of negation in an open world database. The explicit negation component is the set of tuples whose falsity has been constructively established. The default negation component is the set of tuples whose falsity can be assumed. The need for distinguishing between these two forms of negation has been studied extensively from the logic programming perspective. In particular, PROLOG's negation operator *not* is a nonmonotonic form of negation based on the negation as failure rule due to Clark [4].

Most default assumptions are made on the inference "in the absence of any information to the contrary, assume ...". Our attempt here is to find a formalization of this principle in relational databases. For the relational operators, since our effort is to minimize the resulting relation, we assume that a tuple is not in the result of a relational operation unless we have good enough reasons to believe otherwise. Since both the explicit positive and negative components of the result of a relational operation are defined as functions of the corresponding components of the input relations, for each tuple that is unknown in the input relations, we assume the tuple to be in the relation and then compute the result. If there is no change in the result, then we conclude that the tuple can be negated by default in the result. For the purpose of illustration, consider a default relation R and the selection of R by a formula F. The positive component of the selection is defined in terms of R_e^+ . Among the tuples that are unknown in R, there are some for which F holds and others for which it does not. Consider the tuples for which F does not hold. Even if they were to be in R, the result of selection would be the same since F does not hold for them. Hence these are the only tuples in $\bar{R}$ that can be negated by default. Notice here that this form of negation is stronger than the CWA since the CWA dictates that all tuples for which R does not hold are assumed to be false. Since we are dealing with the open world assumption we need a stronger notion of negation.

6 Query Example

In this section, we present an example of query evaluation on default relations. Consider the database shown in Fig 1. This is an instance of a hospital database with two relations `Patient(pname,symptom)` and `Disease(dname,symptom)` which records patient and disease names and their corresponding symptoms. Assume the following domains for each of the attributes:

dom(pname)	=	{Tom, Ann, Jack}
dom(symptom)	=	{Forgetfulness, Nausea, Sneezing, Headache}
dom(dname)	=	{Cold, Alzheimer's, Jaundice}

Since this is a toy example, we will make the simplifying assumption that if a patient shows symptoms of a disease then he suffers from the disease although this may not be the case in reality. For each relation in the database, in the table representing the relation, we use a horizontal line to differentiate between the components R_e^+ , R_e^- and R_d^- in that order. An empty component is denoted using $\emptyset$. In this instance, we assume that the R_d^- component

is empty for both the `Patient` and `Disease` relations. Consider the following query to the database: `Which patients suffer from Alzheimer's disease?` The query can be

Patient	
pname	symptom
Tom	Forgetfulness
Jack	Headache
Tom	Nausea
Jack	Nausea
Jack	Forgetfulness
Ann	Forgetfulness
Ann	Sneezing
Ann	Headache
Ann	Nausea
$\emptyset$	

Disease	
dname	symptom
Cold	Headache
Alzheimer's	Forgetfulness
Jaundice	Nausea
Alzheimer's	Headache
Jaundice	Forgetfulness
$\emptyset$	

Figure 1: An instance of a hospital database

expressed in relational algebra as follows:
Temp(pname,symptom,dname)=
$\dot{\sigma}_{<dname='Alzheimer's'>}(Patient \dot{\bowtie} Disease)$
Answer(pname)=$\dot{\pi}_{<pname>}(Temp)$

Fig. 3 shows the tables **Temp** and **Answer**.

In the above example, we obtain Tom as an explicit positive answer, Ann as an explicit negative answer and Jack as a default negative answer. The tuples highlighted in bold font are the answers that are obtained with the default relations in addition to the others that may be obtained with the paraconsistent model. Notice that with the ordinary relational model we would have obtained Tom as the only answer to the above query, with the paraconsistent data model [1] we would have obtained Tom and Ann as positive and negative answers respectively, but we would not have been able to arrive at any conclusions about Jack. However, with default relations, we are able to obtain Jack as a default false answer. We first investigate why Jack does not appear in the explicit negative component of the answer even though headache is not a symptom of Alzheimer's disease. To ob-

Temp		
pname	symptom	dname
Tom	Forgetfulness	Alzheimer's
Jack	Headache	Cold
Tom	Nausea	Alzheimer's
Tom	Nausea	Cold
Tom	Nausea	Jaundice
Jack	Nausea	Alzheimer's
Jack	Nausea	Cold
Jack	Nausea	Jaundice
Jack	Forgetfulness	Alzheimer's
Jack	Forgetfulness	Jaundice
Jack	Forgetfulness	Cold
Ann	Forgetfulness	Alzheimer's
Ann	Forgetfulness	Jaundice
Ann	Forgetfulness	Cold
Ann	Nausea	Alzheimer's
Ann	Nausea	Cold
Ann	Nausea	Jaundice
Ann	Headache	Alzheimer's
Ann	Headache	Cold
Ann	Headache	Jaundice
Ann	Sneezing	Alzheimer's
Ann	Sneezing	Jaundice
Ann	Sneezing	Cold
Jack	Headache	Alzheimer's
Tom	Headache	Alzheimer's
Tom	Forgetfulness	Jaundice
Tom	**Sneezing**	**Alzheimer's**
Tom	**Sneezing**	**Jaundice**
Tom	**Sneezing**	**Cold**
Jack	**Sneezing**	**Cold**
Jack	**Sneezing**	**Alzheimer's**
Jack	**Sneezing**	**Jaundice**
Tom	**Forgetfulness**	**Cold**
Tom	**Headache**	**Cold**
Tom	**Headache**	**Jaundice**
Jack	**Headache**	**Jaundice**

Answer
pname
Tom
Ann
Jack

Figure 2: The result of the query

tain Jack as an answer in the explicit negative component, we require that all tuples involving Jack after the join and selection appear in the negative component. Since there is no information on whether Jack shows symptoms of sneezing, no information regarding this will appear in the explicit components of the join. However, the tuples `<Jack,Sneezing,Cold>` and `<Jack,Sneezing,Jaundice>` will appear in the explicit negative component as a result of selection in the paraconsistent model. The tuple `<Jack,Sneezing,Alzheimer's>` will still not appear since the selection involves Alzheimer's disease. But in the default model, since sneezing is not mentioned as a symptom of any disease, it can be assumed to be false by default(adding any such tuple will not change the result of the join). Now after the selection, we have complete information about Jack since the tuple `<Jack,Sneezing,Alzheimer's>` appears in the false by default component and hence it appears in this component in the projection.

7 Conclusion and Future Work

In this paper we have introduced a data model based on the relational model. We show that default relations are a generalization of the relational data model. This extension allows us to arrive at certain default conclusions apart from the positive conclusions we obtain from the relational model and the positive and negative conclusions obtained from the paraconsistent relational data model. We have shown that the default relations allow us to arrive at more intuitive answers by way of an example.

In the future, we aim to extend this model to include other forms of incompleteness such as disjunctions.

References

[1] Rajiv Bagai and Rajshekhar Sunderraman. A paraconsistent relational data model. *International Journal of Computer Mathematics*, 55(3), 1995.

[2] Rajiv Bagai and Rajshekhar Sunderraman. Bottom-up computation of the fitting model for general deductive databases. *Journal of Intelligent Information Systems*, 6(1):59–75, 1996.

[3] N. D. Belnap. A useful four-valued logic. In G. Eppstein and J. M. Dunn, editors, *Modern Uses of Many-valued Logic*, pages 8–37. Reidel, Dordrecht, 1977.

[4] K. L. Clark. Negation as failure. In M. L. Ginsberg, editor, *Readings in Nonmonotonic Reasoning*, pages 311–325. Kaufmann, Los Altos, CA, 1987.

[5] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3/4):365–386, 1991.

[6] John Grant and V. S. Subrahmanian. Reasoning in inconsistent knowledge bases. *IEEE Trans. Knowl. Data Eng.*, 7(1):177–189, 1995.

[7] J.Grant and V.S. Subrahmanian. Applications of paraconsistency in data and knowledge bases. *Synthese*, 125:121–132, 2000.

[8] John McCarthy. Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[9] Robert C. Moore. Semantical considerations on nonmonotonic logic. *Artif. Intell.*, 25(1):75–94, 1985.

[10] Shamim A Naqvi. Negation as failure for first-order queries. In *PODS '86*, pages 114–122. ACM Press, 1986.

[11] R. Reiter. A logic for default reasoning. In *Readings in nonmonotonic reasoning*, pages 68–93. Morgan Kaufmann, 1987.

[12] R. Reiter. On closed world data bases. In *Readings in nonmonotonic reasoning*, pages 300–310. Morgan Kaufmann, 1987.

[13] Marek A. Suchenek and Rajshekhar Sunderraman. On reasoning from closed world databases with disjunctive views. In *LPNMR*, pages 132–149, 1990.