Combining Multiple Knowledge Bases

Chitta Baral, Sarit Kraus, and Jack Minker, *Fellow, IEEE*

***Abstract*— Knowledge present in multiple knowledge bases might need to be reviewed to make decisions based on the combined knowledge. We define the concept of combining knowledge present in a set of knowledge bases and present algorithms to maximally combine them so that the combination is consistent with respect to the integrity constraints associated with the knowledge bases. For this we define the concept of maximality and prove that the algorithms presented combine the knowledge bases to generate a maximal theory. We also discuss the relationships between combining multiple knowledge bases and the view update problem.**

***Index Terms*— Consistent, integrity constraints, knowledge bases, logic programs, maximally combining theories.**

I. Introduction

THE main concern in this paper is to combine knowledge present in multiple knowledge base systems into a single knowledge base. A knowledge based system can be considered an extension of a deductive database in that it permits function symbols as part of the theory. We consider alternative knowledge bases that deal with the same subject matter. Each knowledge base consists of a set normal clauses and a set of integrity constraints. The set of integrity constraints (IC) is assumed to be the same for all knowledge bases, but the sets of normal clauses may differ. We assume that each knowledge base is consistent with respect to the integrity constraints when considered alone.

While combining multiple knowledge bases we have to ensure that the combination is consistent with respect to the integrity constraints. Such a problem might arise in a large company with branches overseas. Each branch manages its own knowledge base. A consistent combination of the knowledge bases is required while trying to make a decision about the overall company.

In Section II, we discuss problems that may arise in combining alternative knowledge bases. Basic definitions are presented in Section III. In Section IV, we present algorithms to combine multiple knowledge bases. In Section V, we compare this problem to the view update problem in databases [7], [6], [24].

Manuscript received September 14, 1989; revised March 21, 1990. This work was supported by the National Science Foundation under Grant IRI-86-09170 and the Army Research Office under Grant DAAL-03-88-K0087.

C. Baral is with the Department of Computer Science, University of Maryland, College Park, MD 20742.

S. Kraus and J. Minker are with the Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742.

IEEE Log 9144308.

II. Combining Knowledge Bases

We assume the knowledge bases to be logic programs with integrity constraints associated with the program. A logic program consists of a finite set of clauses of the form

$$A_1, \cdots, A_n \leftarrow B_1, \cdots B_m \qquad n \geq 1,\, m \geq 0$$

where the expression on the left-hand side of the implication is a disjunction of atoms and the expression on the right-hand side is a conjunction of literals. When n is 1, we call it a *normal logic program* and we call a normal logic program, a *Horn logic program* when the literals in the right-hand side are all atoms. When n is greater than 1, we call it a disjunctive logic program.

A. Combining Logic Programs

Suppose we consider the problem of combining several logic programs without integrity constraints. In order to combine the logic programs one may take their union, i.e., all information available from all of the logic programs are combined into a single program. It is easy to see that this union is consistent.

Theorem 1: If $T_1, \cdots, T_k$ are general Horn logic programs then $T_1 \cup \cdots \cup T_k$ is consistent. □

Consider two logic programs T_1 and T_2 that contradict each other, i.e., using one we can derive $P(a)$ while in the other we may derive $\neg P(a)$. (Note: By "derive" we refer to an SLDNF derivation [13].) As shown in Theorem 1, $T_1 \cup T_2$ is consistent, and therefore even in such a case it is reasonable to combine them by taking their union. In the union of the two theories we might derive $P(a)$ or we might derive $\neg P(a)$. This is because the minimal model of $T_1 \cup T_2$ is not necessarily the union of the minimal models of P_1 and P_2, due to nonmonotonicity of the semantics. We explain this by the following example.

Example 2.1: Consider the following logic programs P_1 and P_2:

P_1 :
  abnormal(tweety)
  bird(tweety)
  bird(charli)
P_2 :
  abnormal(ostrich)
  bird(tweety)
  flys(X) $\leftarrow$ *bird(X)*, $\neg$*abnormal(X)*

We can derive $\neg$*flys(tweety)* from P_1 and *flys(tweety)* from P_2. But from $P_1 \cup P_2$ we can derive $\neg$*flys(tweety)*. On the other hand, from both P_1 and P_2 we can derive $\neg$*flys(charli)* but from $P_1 \cup P_2$ we can derive *flys(charli)*. □

B. Handling Integrity Constraints

In real world applications, integrity constraints restrict a knowledge base to a particular meaning [16]. In the presence of integrity constraints, the union of a set of logic programs can violate an integrity constraint, even though each individual logic program does not violate it. Hence, in the presence of integrity constraints, the union of logic programs may not always be the correct way to combine the theories. For example if we have *fatherof(ken,nick)* in T_1, where *fatherof(X, Y)* denotes that *Y* is the father of *X*, and *fatherof(ken,george)* in T_2, and we take the union of the two theories, then although the union is consistent, it violates an integrity constraint that characterizes fatherhood, which states that, "one person cannot have two fathers," even though both T_1 and T_2 individually satisfy the integrity constraint. To combine theories T_1 and T_2, we can choose one of them and include it in the combination, we can partially favor one theory and combine them as *fatherof(ken,nick)*$\leftarrow \neg$*fatherof(ken,george)* or we can combine them as *fatherof(ken,nick)*$\vee$ *fatherof(ken,george)*. In the last situation we do not prefer any theory, and give the user the freedom to choose the appropriate model from the various models suggested by the combined theory. A normal clause representation of the last approach would be the unstratified clauses

$$fatherof(ken, nick) \leftarrow \neg fatherof(ken, george)$$

$$fatherof(ken, george) \leftarrow \neg fatherof(ken, nick)$$

Semantics of such unstratified logic programs, have been proposed in [3], [26], [22], and [10]. We do not explore this possibility in this paper.

C. Possible Solutions

As noted in the previous section there are different options available to combine alternative theories. The following are the major possible alternatives.

1) An Oracle exists which knows everything. Whenever a contradiction between theories exists, a decision as to what to include in the combination is determined according to the Oracle, which may support one of the theories or neither.
2) A partial order exists between all possible pairs of <*theory, concept*>. In case of a contradiction between two theories relative to a specific concept, in the combination we may always select data from the preferable theory with respect to this concept. For example, <*cardiologist, cancer*> $\leq$ <*oncologist, cancer*>. That is, the cancer specialist (called "oncologist") knows at least as much as the cardiologist about cancer and so we might trust his beliefs about cancer rather than the cardiologist's view. A syntactic method based on this is discussed in Section IV-C.
3) In the extreme case we might define a fact to be *unknown* when the facts in the two knowledge bases contradict one another. We want to remark that to define a concept as *unknown* is different than not to include it in the combination. If we do not include it in the knowledge base, in logic programs we will be able to derive the concept's negation. In order to be able to implement such an approach one needs to move to three-valued logic [22], [3], [8], [9] or to protected circumscription [17]–[19].
4) A maximal amount of consistent information could be combined from alternative theories. In case of a contradiction, the information could be combined to make the knowledge base consistent by converting it into a disjunctive knowledge base. This knowledge base may be presented to the user and he might choose among the disjunctive facts.

Creating an Oracle, as suggested in the first approach, is almost impossible, especially while dealing with distributed knowledge bases. There are some technical problems, but there are also some essential problems, such as who knows the truths to supply to the Oracle. We may not have the priority information available so that the second approach might not be possible. When we define a concept as *unknown* we lose information that existed in the original knowledge bases; so the third option may be questionable.

In this paper, we take the last approach, where we maximally combine the set of knowledge bases subject to consistency with the integrity constraints. We also note that there is another option possible, where the user decides what is to be done.

III. Basic Definitions

Recently, several different semantics have been given to logic programs [26], [22], [10], [3]. In the case of Horn logic programs without negation, it is well known [25], [2] that there exists a unique minimal model. This model is used as the meaning of the program. In the case of stratified general Horn programs, there is a single perfect model [23], [1] and this model is used as the meaning of the program. In the case of stratified disjunctive logic programs we use its set of perfect models as its meaning. We call the set of perfect models of a stratified disjunctive program P, as *MINSET*(P). In this paper, we only consider stratified programs and further assume that the union of the theories to be combined is also stratified.

Definition 3.1: Given a program P, *HERB*(P) denotes the (possibly infinite) set of clauses which are ground instances of program clauses in P. □

The definition of perfect model is based on a partial order between minimal models. The partial order between minimal models is based on a partial order between elements of the Herbrand base of the program P, dictated by the position of literals in HERB(P). We now define this partial order formally.

Definition 3.2: Definition of $<$ *and* $\leq$, from [21]: The $<$ *and* $\leq$ relation between atoms of the Herbrand base of a program P, are defined based on their position in HERB(P).

1) $C < B$, if $\neg B$ is a negative literal in the body of a clause in HERB(P), with C in the head.
2) $C \leq B$, if B is a positive literal in the body of a clause in HERB(P), with C in its head.

3) $C \leq B$ and $B \leq C$, if B and C are in the head of the same clause in HERB(P).
4) $A \leq B, B \leq C \Rightarrow A \leq C$.
5) $A \leq B, B < C \Rightarrow A < C$.
6) $A < B \Rightarrow A \leq B$.
7) Nothing else satisfies $<$ or $\leq$. □

Since, we are considering only stratified programs, the relation $<$ is a partial order.

Definition 3.3: Relation between minimal models [21]: Let M and N be two distinct models. We say $N \ll M$ (N is preferable to M) if, $\forall A$(a ground atom) in $N - M$, $\exists$ a ground atom B in $M - N$, such that $A < B$. □

Example 3.1: Consider the program

$A \leftarrow \neg B$

$C \leftarrow \neg D$.

It has four minimal models $\{A, C\}$, $\{A, D\}$, $\{B, C\}$, and $\{B, D\}$.

Using the definition for the relation $<$, we have $A < B$ and $C < D$

Consider the models $\{A, C\}$ and $\{A, D\}$. Since $C < D$, hence we have $\{A, C\} \ll \{A, D\}$. □

Definition 3.4: Definition of *MINSET*(*P*): The MINSET(P) of a program P is a set of minimal models of P, such that

1) $\forall x$ (x is a minimal model of $P \Rightarrow \exists y$, $y \in$ MINSET(P) and $y \ll x$).
2) $\forall x, y$ ($x, y \in$ MINSET(P) and $x \neq y \Rightarrow x \not\ll y, y \not\ll x$). □

Example 3.2: In the last example we have $\{A, C\} \ll \{A, D\}$, $\{A, C\} \ll \{B, C\}$, $\{A, C\} \ll \{B, D\}$. Hence, MINSET(P) = $\{ \{A, C\} \}$. □

The syntax of integrity constraints we use is similar to that used by Sadri and Kowalski [12] and Chakravarthy, Grant, and Minker [4]. Integrity constraints are of the form $\leftarrow L_1, \cdots, L_m$, $m > 0$, where the L_i are literals and all variables are assumed to be universally quantified in front of the constraint in which they occur. Such clauses are called *denials*. When L_i are restricted to be only atoms we call it a *positive integrity constraint*. Denials have to be range restricted, that is, any variable that occurs in a negative literal of the constraint must also have an occurrence in a positive literal of the constraint. Sadri and Kowalski [12] show that integrity constraints of this form are general enough to represent any closed first-order formula using the transformations suggested in [12] and [14].

There are several different definitions of integrity constraint satisfaction in a database or knowledge base [12]. The two main definitions are the theoremhood definition and the consistency definition. Let P be a program and *Sem*(*P*) be the semantics of the program. According to the *theoremhood definition*, program P satisfies an integrity constraint I iff I is true in Sem(P). According to the *consistency definition* P satisfies I iff Sem(P) $\cup$ I is consistent. The two approaches are the same when the program is complete. A program P is said to be *complete* with respect to a semantics, Sem(P), when for any formula W, either W is *true* in Sem(P) or W is *false* in Sem(P). Most papers that deal with the theoremhood approach use *Comp*(*P*), Clark's completion of a program [5], as Sem(P).

We use Sem(P) as the perfect model of the program, if the program is a normal program and, as we are dealing with stratified programs, they have a unique perfect model. When the program is a disjunctive program we use the set of minimal models defined as MINSET(P) as Sem(P). In the case of normal programs, P is complete with respect to Sem(P), and hence there is no difference between the consistency and the theoremhood definition with respect to our definition of Sem(P). In the case of disjunctive programs we use the consistency definition.

Definition 3.5 Consistency: A theory T, where T may be disjunctive, is said to be *consistent* with respect to a set of integrity constraints IC, iff every minimal model of T that is present in MINSET(P) satisfies the integrity constraints. □

Definition 3.6: The $\leq$ relationship between theories: Let T_1 and T_2 be two theories (possibly disjunctive). We say $T_1 \leq T_2$ iff $\forall x : x \in$ MINSET(T_1), $\exists y : y \in$ MINSET(T_2) and $x \subseteq y$. □

Definition 3.7 Maximality: A theory T_i is said to be *maximal* among a set of theories $\{T_1 \cdots T_n\}$, iff there does not exist j, such that $1 \leq i, j \leq$ n and $T_i \leq T_j$. □

Definition 3.8 Correctness: A theory T is said to be *correct* with respect to theories $T_1, \cdots, T_k$, if $T \leq T_1 \cup \cdots \cup T_k$. □

Definition 3.9 Combination of Theories: Let $T_1, \cdots, T_k$ be a set of theories and IC be a set of integrity constraints, where each T_i satisfies IC. The combination function C maps from a set of theories and a set of integrity constraints into a theory. It should satisfy the following four properties.

1) *(identity)* $C(T, \text{IC}) = T$.
2) *(commutativity)* $C(T_1, \cdots, T_{i-1}, T_i, \cdots, T_j, T_{j+1}, \cdots, T_k, \text{IC}) = C(T_1, \cdots, T_{i-1}, T_j, \cdots, T_i, T_{j+1}, \cdots, T_k, \text{IC})$.
3) *(consistency)* $C(T_1, \cdots, T_k, \text{IC})$ is consistent with respect to IC.
4) (correctness) $C(T_1, \cdots, T_k, \text{IC})$ is correct with respect to the theories $T_1, \cdots, T_k$.

Another useful property is associativity, which is defined as follows. $C(T_1, \cdots, T_j, C(T_{j+1}, \cdots, T_k, \text{IC}), \text{IC}) =_{mm} C(T_1, \cdots, T_j C(T_{j+1}, \cdots, T_k), \text{IC}, \text{IC}) =_{mm} C(T_1, \cdots, T_k, \text{IC})$, where $P =_{mm} Q$ means MINSET(P) = MINSET(Q). □

Our goal is to combine a set of theories such that $C(T_1, \cdots, T_k, \text{ IC})$ is maximal among all consistent and correct combinations of $T_1, \cdots, T_k$. The intuition behind the maximality property is that we would like the combination of the theories to include as much information as possible from all theories. The intuition behind the correctness is that the combination will not include new information that does not have a basis in the union. Consider the following example.

Example 3.3: Consider two theories $T_1 = \{A\}$ and $T_2 = \{B\}$ and the integrity constraint $\leftarrow A, B$. In the absence of any priority information, there are three consistent and correct combinations of T_1 and T_2.

1) The combination whose only minimal model is $\{A\}$.
2) The combination whose only minimal model is $\{B\}$.

3) The combination that has two minimal models $\{A\}$ and $\{B\}$.

By definition of maximality the third combination is the maximal combination. Our goal is to develop algorithms to combine theories maximally. □

Theorem 2: Let $T_1, \cdots, T_k$ be a set of theories and IC be a set of positive integrity constraints. There exists a maximal combination of $T_1, \cdots, T_k$. All such combinations have the same set of minimal models.

Proof: Consider the set of minimal models in MINSET($T_1 \cup \cdots \cup T_k$). Let them be $m_1, \cdots, m_s$. Divide these minimal models so as to obtain a set of maximal models, such that none of the maximal models violate any integrity constraints in IC. For example, if $m_i = \{A_1 \cdots A_n\}$ is one of the original minimal models, and $\leftarrow A_1 \cdots A_r$ where $r \leq n$ is an integrity constraint in IC, then m is divided to the set of maximal models:
$\{A_1, \cdots, A_{r-1}, A_{r+1} \cdots A_n\}$, $\{A_1, \cdots, A_{r-2}, A_r, A_{r+1} \cdots A_n\} \cdots \{A_2, \cdots, A_{r+1} \cdots A_n\}$. After dividing the original set of minimal models so as to be consistent with respect to all the integrity constraints, suppose we obtain $M = m'_1, \cdots m'_j$ as the set of maximal models. We claim that any theory which has its minimal models as M, is a maximal combination of $T_1, \cdots, T_k$.

Suppose this is false, i.e., $\exists M'$: $M \leq M'$, and M' is a combination of $T_1, \cdots, T_k$. Since, $M \leq M'$, $\exists x, y : x \in M$ and $y \in M'$ and $x \subset y$. Since M' is correct (a property of all combinations), $\exists i \; 1 \leq i \leq s : y \subseteq m_i$. Since, $x \subset y$, $x \subseteq m_i$, i.e., x is obtained from m_i, while dividing m_i to make it consistent with respect to the integrity constraints. Since the division creates a set of maximal consistent models, $x \subset y$ implies y is not consistent with respect to the integrity constraints. Hence, M' is not consistent with respect to the integrity constraints and hence it is not a combination. This contradicts our initial assumption that M' is a combination of $T_1, \cdots, T_k$, and hence M is the maximal combination of $T_1, \cdots, T_k$. □

IV. Combining a Set of Belief Systems

In this section we give algorithms to maximally combine a set of normal logic programs. First we give algorithms to combine theories that contain only facts, then we allow rules without negation in their body and finally we consider normal logic programs.

Throughout the rest of the paper we assume that the SLDNF-proof tree when an integrity constraint is considered as a query, with respect to the union of the theories, is finite. A sufficient restriction for having finite SLDNF proof trees is that the theory be function-free and hierarchical. By hierarchical, we mean that the theory does not have any recursion. We also assume that integrity constraints do not have negation in their body.

A. Combining Theories Consisting Only of Facts

Our initial assumption is that *theories consist only of facts*. Consider a simple case where T_1 consists of $P(a)$ and T_2 consists of $P(b)$, and the integrity constraint is $\leftarrow P(X), P(Y), X \neq Y$. In the absence of any information about preferences we can combine these two theories, without the combination (T) violating the integrity constraint and without preferring one theory over the other, by placing only $P(a) \vee P(b)$ in the combination T, of the two theories. The theory T has two minimal models: one is $\{P(a)\}$ and the other is $\{P(b)\}$.

Before presenting our algorithm we first show, through an example, how a combination is achieved.

Example 4.1: Let $T_1 \overset{\text{def}}{=} \{P(a); P(b)\}$,$T_2 \overset{\text{def}}{=} \{P(c)\}$, and the integrity constraint be $\leftarrow P(a), P(b), P(c)$. The combined theory has to satisfy the logically equivalent integrity constraint, $\neg P(a) \vee \neg P(b) \vee \neg P(c)$. To be maximal, we want the least number of atoms to be false in the combination. In other words, we want the least number of negative literals to be true. We rename the negative literals in the clauses obtained by transferring atoms to the left in each of the integrity constraints. We rename $\neg P(a)$ as $P'(a)$ and others in a similar manner. After renaming we obtain the clause $P'(a) \vee P'(b) \vee P'(c)$. This clause has three minimal models, $\{\{P'(a)\}, \{P'(b)\}, \{P'(c)\}\}$. The minimal model $\{P'(a)\}$ means $P'(a)$ is *true*, and others are *false*; that means $\neg P(a)$ is *true* and all other negative literals are *false*, which means $P(a)$ is *false*, and all other atoms are *true*. Each minimal model of the renamed clauses corresponds to a maximal model.

Hence, we take the integrity constraints and form clauses by transferring the atoms to the left. Next we rename the atoms in the clauses. We find the set of minimal models for the renamed clauses. Since each minimal model corresponds to a maximal model, we collect the set of these maximal models. We then construct a theory whose minimal models are the set of maximal models. The theory constructed is the maximally combined theory.

In this example, the minimal models of the renamed theory are $\{\{P'(a)\}, \{P'(b)\}, \{P'(c)\}\}$. The maximal model corresponding to $\{P'(a)\}$ is $\{P(b), P(c)\}$; the maximal model corresponding to $\{P'(b)\}$ is $\{P(a), P(c)\}$ and the maximal model corresponding to $\{P'(c)\}$ is $\{P(a), P(b)\}$. The theory, whose minimal models are $\{\{P(b), P(c)\}, \{P(a), P(c)\}, \{P(a), P(b)\}\}$ is $P(a) \vee P(b)$, $P(a) \vee P(c)$, $P(b) \vee P(c)$, and is the combined theory which is consistent with respect to the integrity constraints. □

It should be noted that the combined theory is disjunctive in the above example. An answer to the query $\leftarrow P(X)$ with respect to the combined theory is $P(a) \vee P(b)$, while the answer to the query $\leftarrow P(a)$ is "unknown."

We now give an algorithm to combine a set of theories consisting only of facts, such that the resultant theory is correct with respect to the original theories, consistent with respect to the integrity constraints and is a maximal combination.

Algorithm 4.1 (To combine theories consisting only of facts):
INPUT:
1. A set of k Horn theories $\{T_1 \cdots T_k\}$ which have to be combined, where each T_i consists only of facts.
2. A set of s integrity constraints, IC $= \{\text{IC}_1, \cdots, \text{IC}_s\}$ where each integrity constraint is satisfied by each of the theories.
OUTPUT:
A theory T, which is the maximal combination of the input set

of theories, such that T is also satisfied by each of the integrity constraints. The combined theory T can be disjunctive.

STEP 1: [Find S, the set of instances of integrity constraints that violate the union of the theories.]

```
$S = \varnothing$;
For $i = 1$ to $s$ do
begin
    Let IC$_i$ be of the form $\leftarrow P_1, \cdots, P_k$, where $P_1, \cdots, P_k$ are atoms.
    Solve IC$_i$ with respect to $T_1 \cup \cdots \cup T_k$.
    (Note that, in the beginning of this section we assume this to be decidable.) If there are no solutions then continue with the next $i$, else:
    Let $\theta_1 \cdots \theta_{l_i}$ be all the ground answer substitutions when the IC$_i$ is satisfied as a query to the union of the knowledge bases, meaning that IC$_i$ violates the union of the knowledge bases
    $S = S \cup \{\leftarrow (P_1, \cdots P_k)\theta_1; \cdots; \leftarrow (P_1, \cdots P_k)\theta_{l_i}\}$
end
```

S is the set of instantiated integrity constraints which violates the combined theory $T_1 \cup \cdots \cup T_k$. If S is empty, then the combined theory $T_1 \cup \cdots \cup T_k$ is consistent with respect to the integrity constraints and the algorithm terminates.

If S is not empty, continue with STEP 2.

STEP 2: [Obtain the set of minimal models of the transformed and renamed clauses in S and their corresponding maximal model.]

Let S be $\{C_1, \cdots, C_n\}$, where each C_i $(1 \le i \le n)$ is defined by $\leftarrow C_{i1}, C_{i2}, \cdots, C_{ik_i}$.

Let $\{A_1, \cdots A_p\}$ be all ground atoms present in S. We call this set the *interfering facts.*

The set of facts that is present in $T_1 \cup \cdots \cup T_k$ but not in $\{A_1, \cdots A_p\}$ are called *noninterfering facts.*

Construct the set of clauses $\{\neg C_{11} \vee \neg C_{12} \vee \cdots \neg C_{1k_1}, \cdots, \neg C_{n1} \vee \neg C_{n2} \vee \cdots \neg C_{nk_n}\}$ by transferring the atoms in the instantiated integrity constraints to the left of the arrow.

Rename the negative literals $\neg C_{ij}$ by C'_{ij} in the above set of clauses.

Find all minimal models of the renamed set of clauses given by $\{C'_{11} \vee C'_{12} \vee \cdots C'_{1k_1}, \cdots, C'_{n1} \vee C'_{n2} \vee \cdots C'_{nk_n}\}$

```
Let the minimal models be $m'_1, \cdots, m'_l$
For $i = 1$ to l do
    $m^*_i = m'_i$ with$C'_{ij}$ replaced by$C_{ij}$.
```

STEP 3: [Obtain the theory whose only minimal models are the maximal models found in STEP 2.]

```
For $i = 1$ to $l$ do
    $m_i = \{A_1, \cdots A_p\} - m^*_i$
```

Construct a theory T (possibly disjunctive) whose only minimal models are $m_1 \cdots m_l$ by doing the following.

$T = \{x : x = fact(A_1 \vee A_2 \cdots \vee A_l)$, where for all $i \le l$, A_i is in $m_i\}$, where *fact* is a function that removes all but one occurrence of repeated atoms in a disjunction.

STEP 4: [Obtain the combined theory, by augmenting the noninterfering facts.]

Add all the atoms in $T_1 \cup T_2 \cup \cdots \cup T_k$ that do not appear in $\{A_1, \cdots A_p\}$ from STEP 2 to T, and call it $C(T_1, \cdots, T_k, \text{IC})$. □

The intuition behind this algorithm is as follows; in order for the combination of the theories to be consistent with the integrity constraints, at least one negative literal of each clausal form of the integrity constraint has to be true in each minimal model of the combination. On the other hand, in order to make the combination maximal one needs to minimize those negations.

We demonstrate this algorithm by some examples.

Example 4.2: We first show Example 4.1 with respect to this algorithm.

Let $T_1 \overset{\text{def}}{=} \{P(a); P(b)\}$, $T_2 \overset{\text{def}}{=} \{P(c)\}$, and the integrity constraint be $\leftarrow P(a), P(b), P(c)$.

STEP 1: We find that the integrity constraint $\leftarrow P(a), P(b), P(c)$ violates the union of the theories.

STEP 2: We transfer the atoms in the integrity constraint to the left and rename the negative literals. The resulting clause is $\{P'(a) \vee P'(b) \vee P'(c)\}$. We find the minimal models of $P'(a) \vee P'(b) \vee P'(c)$ to be $\{\{P'(a)\}, \{P'(b)\}, \{P'(c)\}\}$.

STEP 3: For all minimal models found in STEP 2 the corresponding maximal models are $\{\{P(b), P(c)\}, \{P(a), P(c)\}, \{P(a), P(b)\}\}$. This is because for the minimal model $\{P'(a)\}$ in STEP 2 the corresponding maximal model $\{P(b), P(c)\}$. This is further explained in Example 4.1. We find the theory, whose minimal models are $\{\{P(b), P(c)\}, \{P(a), P(c)\}, \{P(a), P(b)\}\}$ to be $P(a) \vee$ P(b), $P(a) \vee P(c)$, $P(b) \vee P(c)$.

STEP 4: The combined theory is the theory found in STEP 3. □

Example 4.3: Let $T_1 \overset{\text{def}}{=} \{P(a)\}$, $T_2 \overset{\text{def}}{=} \{P(b)\}$, $T_3 \overset{\text{def}}{=} \{P(c)\}$, and the integrity constraint be $\leftarrow P(X), P(Y), X \neq Y$.

STEP 1: After solving the constraint we find $\leftarrow P(a), P(b); \leftarrow P(a), P(c)$ and $\leftarrow P(b), P(c)$ to be the instantiated integrity constraints that violate the union of the theories.

STEP 2: We find the minimal models of $\{P'(a) \vee P'(b), P'(a) \vee P'(c), P'(b) \vee P'(c)\}$ to be $\{\{P'(a), P'(b)\}, \{P'(a), P'(c)\}, \{P'(b), P'(c)\}\}$.

STEP 3: We find the theory, whose minimal models are $\{\{P(c)\}, \{P(b)\}, \{P(a)\}\}$ to be $P(a) \vee P(b) \vee P(c)$.

STEP 4: The combined theory is the theory found in STEP 3. □

The following proves the correctness of our algorithm.

Theorem 3: $C(T_1, \cdots T_k, \text{IC})$, as given by Algorithm 4.1 is a maximal combination of $T_1 \cdots T_k$.

Proof:

a) $C(T_1, \cdots, T_k, \text{IC})$ is correct because all atoms in its clauses are satisfied by $T_1 \cup \cdots \cup T_k$.

b) $C(T_1, \cdots, T_k, \text{IC})$ is consistent because all its minimal models, by virtue of its construction in STEP 3, do not violate the integrity constraints.

c) Let C be all the facts in $T_1 \cup \cdots \cup T_k$. Let $B = \{A_1, \cdots, A_p\}$ be the ground atoms present in S in STEP 2. Let $A = C - B$.

From STEP 1 of the algorithm, we find that the atoms in B cannot be all *true* in the same model of $C(T_1 \cup \cdots \cup T_k, \text{IC})$, because they violate the integrity constraints.

In STEP 2 of the algorithm we find a set of minimal sets, where each minimal set has the minimal number of atoms

false for the integrity constraints to be satisfied.
Thus, each minimal set of *false* atoms is equivalent to a maximal set of *true* atoms that can be *true*, while still not violating the integrity constraints.
Thus, in STEP 3 of the algorithm we split B into a set $\{m_1, \cdots, m_l\}$ of maximal models, and we construct T, which has $\{m_1, \cdots, m_l\}$ as the only minimal models.
In STEP 4 we have $C(T_1 \cup \cdots \cup T_k, \text{IC}) = T \cup A$.
Now assume that our combination is not maximal,
i.e., $\exists T' : C(T_1 \cup \cdots \cup T_k, \text{IC}) \leq T' \leq T_1 \cup \cdots \cup T_k$.
$\Longrightarrow \exists$ a model m of $c : m \subset m'$ and m' is a model of T'.
$\Longrightarrow m = A + m_i$, where $1 \leq i \leq l$.
$\Longrightarrow m_i \subset m' - A$
$\Longrightarrow$ But since m_i is one of the maximal set (by construction in STEP 3), the above cannot be true.
$\Longrightarrow$ Such a T' cannot exist and hence our combination is maximal. □

B. Combining Theories with Rules Without Negation in their Body

We now extend Algorithm 4.1 to the case where we have rules in the theories. We do not allow negative literals in the body of the rules.

Algorithm 4.2: (To combine theories with Horn rules).

INPUT:
1. A set of k Horn theories $\{T_1 \cdots T_k\}$ which have to be combined. (Note that Horn theories do not have negative literals in the body of their rules.)
2. A set of s integrity constraints, $\text{IC} = \{\text{IC}_1, \cdots, \text{IC}_s\}$ where each integrity constraint is satisfied by each of the theories.

OUTPUT:
A theory T, which is the maximal combination of the input set of theories, such that T is also satisfied by each of the integrity constraints. The combined theory T can be disjunctive.

ASSUMPTIONS:
Each of the theories in $T_1 \cdots T_k$ is Horn and consists of facts and rules without negation in their body.

STEP 1: [Find S' the set of instances of integrity constraints that violate the union of the theories and for each such instance find the set of interfering rules.]

```
S' := ∅;
For i = 1 to s do
begin
    Let IC_i be of the form ← P_1, ···, P_k, where P_1, ···, P_k are atoms.
    Solve IC_i with respect to T_1 ∪ ··· ∪ T_k.
    (Note that we assume this to be decidable, in the beginning of the section.) If there are no solutions continue with the next i, else:
    Let θ_1, ···, θ_{l_i} be the ground answer substitutions for the IC_i; and for each ground answer substitution θ_j,
    Let R_j = {R_{j1}, ···, R_{jk}} be the set of rules, which were used to obtain the ground answer substitution θ_j, where their heads unify with one of the P_i's.
    Let R_{ji} be the rule whose head unifies with P_i.
    S' := S' ∪ {<← (P_1, ···, P_k)θ_1, R_1, θ_1 >, ···, <← (P_1, ···, P_k)θ_{l_i}, R_{l_i}, θ_{l_i} >}.
end
```

S' contains the instantiated integrity constraints that are violated by $T_1 \cup \cdots \cup T_k$ and a set of rules for each, that has to be restricted.
The set of rules present in S' is called *interfering rules.*
The set of rules present in $T_1 \cup \cdots \cup T_k$ but not in S' are called *noninterfering rules.*
If S is empty, then the combined theory $T_1 \cup \cdots \cup T_k$ is consistent with respect to the integrity constraints and the algorithm terminates. If S is not empty, continue with STEP 2.

STEP 2: [Find the set of minimal models of the transformed and renamed clauses in S and their corresponding maximal models.]
S := the first element of each tuple of S'. (S contains the set of instantiated integrity constraints of S' in step 1). Do STEP 2 of Algorithm 4.1.

STEP 3: [Find the theory whose only minimal models are the maximal models found in STEP 2
Do STEP 3 of Algorithm 4.1 and let the theory generated be T.]

STEP 4: [Find the resultant theory by augmenting the noninterfering facts.]
$T := T \cup$ all facts in $T_1 \cup \cdots \cup T_k$ that do not appear in $\{A_1, \cdots A_p\}$ (see Algorithm 4.1).

STEP 5: [Find the resultant theory, by augmenting the restricted version of the interfering rules.]

```
function Restrict(R: a set of rules, θ_i: a variable substitution): a set of rules;
begin
    Restrict(R, θ_i) := ∅
    If θ_i is empty then STOP
    else
    For all rules R_i in R do
        Consider the variables in the head of R_i and the substitution given by θ_i. Let X_{i1}, ···, X_{ik_i} be variables in the head of R_i and t_{i1}, ···, t_{ik_i} be the corresponding variable substitutions.
        Let R_i be head ← tail.
        Restrict(R,θ_i) := Restrict(R,θ_i) ∪
                {head ← tail, X_{i1} ≠ t_{i1}; ···; head
                ← tail, X_{ik_i} ≠ t_{ik_i}}
end
```

{This function restricts rules with respect to a variable substitution.}
Obtain from S' the pairs $< R, \theta >$ and call it *ruleset.*
The ruleset is $\{< R_1, \theta_1 >, \cdots, < R_t, \theta_t >\}$.
Let $R = \{R_1, \cdots, R_t\}$. We call this set of rules the *interfering rules.*
Since there might be multiple instances of the same rule in R, for each rule r in R do $\text{Restrict}(\text{Restrict}(\cdots \text{Restrict}(r, \theta_{r_1}) \cdots \theta_{r_{s-1}})\theta_{r_s})$ where $\theta_{r_1} \cdots \theta_{r_s}$ are the various variable substitutions associated with the rule r, in ruleset.
$T := T \cup$ the collection of the above restricted rules.

STEP 6: Find the combined theory, by augmenting the noninterfering rules. $T := T \cup$ all the rules in $T_1 \cup \cdots \cup T_k$ that do not appear in R.
$C(T_1 \cup \cdots \cup T_k, \text{IC}) := T$. □

One may ask why we restrict the rules and do not, for example, remove the facts that are used in order to solve the integrity constraints. This is because we want to gain maximality. Taking out the facts will restrict the combined theory. On the other hand, by restricting the rules, and minimizing the negations of the facts of the integrity constraints, and leaving as much information as possible from the original theories, we obtain a maximally combined theory consistent with the integrity constraints.

The following example illustrates the above algorithm.

Example 4.4: Consider the integrity constraint $\leftarrow P(X), P(Y), X \neq Y$, and the theories T_1 and T_2.

T_1	T_2
$P(Y) \leftarrow Q(Y)$	$P(X) \leftarrow R(X)$
$R(a)$	$Q(b)$.

Step 1: When we solve the integrity constraint as a query we find that it is violated when $x = a$ and $y = b$. So the instantiated integrity constraint is $\leftarrow P(a), P(b)$ and the two rules that are used are $P(X) \leftarrow R(X)$ and $P(Y) \leftarrow Q(Y)$. In the representation given in STEP 1 of Algorithm 4.2 we obtain $S' = \{<\leftarrow P(a), P(b), \{P(X) \leftarrow R(X), P(Y) \leftarrow Q(Y)\}, \{X/a, Y/b\}>\}$.
Step 2 and 3 gives us $T = P(a) \vee P(b)$.
In Step 4 we do not have any noninterfering facts to be added.
In Step 5 we add the restricted rules $P(Y) \leftarrow Q(Y), Y \neq b$ and $P(X) \leftarrow R(X), X \neq a$ to T. After we add the facts in $T_1 \cup T_2$ that are absent in T, we get $C(T_1, T_2, \text{IC})$ as

$P(a) \vee P(b)$
$R(a)$
$Q(b)$
$P(Y) \leftarrow Q(Y), Y \neq b$
$P(X) \leftarrow R(X), X \neq a$.

The above combined theory has the minimal models $\{P(a), R(a), Q(b)\}$ and $\{P(b), R(a), Q(b)\}$. The union of the original theories has the minimal model $\{P(a), P(b), R(a), Q(b)\}$ and since the instantiated integrity constraint is $\leftarrow P(a), P(b)$, we observe that indeed this algorithm gives a maximal and consistent combination of the theories. □

Theorem 4: $C(T_1, \cdots, T_k, \text{IC})$ as given by Algorithm 4.2 is a maximal combination of $T_1, \cdots, T_k$.

Proof:

a) $C(T_1, \cdots, T_k, \text{IC})$ is consistent because, for all satisfiable instances of integrity constraints $\leftarrow (P_1, \cdots, P_k)\theta$ we restrict the rules (in STEP 5) which expand $P_1\theta, \cdots, P_k\theta$, and add a set of disjunctive facts in STEP 3, to make sure they are not satisfied by the resulting combination.

b) $C(T_1, \cdots, T_k, \text{IC})$ is correct because, besides restricting the rule, the disjunctive fact we add has all its minimal models as a subset of $T_1 \cup \cdots \cup T_k$.

c) *(maximality)*
Let C be the minimal model of $T_1 \cup \cdots \cup T_k$. (Note that $T_1 \cup \cdots \cup T_k$ is a Horn theory.)
Let $B = \{A_1, \cdots, A_k\}$ be the ground atoms present in S in STEP 2.
In STEP 5 by restricting the rules we make sure that atoms in B cannot be proven using those rules. Also it is clear that atoms in $C - B$ belong to all models in $\text{MINSET}(C(T_1, \cdots, T_k, \text{IC})$.
Let $A = B - C$
Similar to Algorithm 4.1 we split B into a set $\{m_1, \cdots, m_l\}$ of maximal models and construct T in STEP 3 which has $\{m_1, \cdots, m_l\}$ as the only minimal models. The rest of the proof is the same as in the proof of Theorem 3. □

C. A Syntactic Approach

Several syntactic methods have been suggested to compile integrity constraints into a deductive database [11], [4]. In these approaches, the integrity constraints are embedded as part of the deductive rules and the integrity constraints are no longer needed. Normal clauses result in the theory combined with the integrity constraints. It would then appear that multiple knowledge bases might be combined syntactically by compiling the integrity constraints to their union using the methods suggested in [11] and [4]. We show by a counterexample that this approach does not provide the desired result. We first describe Kowalski and Sadri's method [11] and then give counterexamples to show why such a syntactic approach cannot be used to combine knowledge bases. However, applying Algorithm 4.2 to the union of the theories provides the desired result.

Kowalski and Sadri in [11] give a method to compile integrity constraints inside a theory. They assume that one atom in every integrity constraint is more preferable than the rest of the atoms. They call it the retractable atom of the theory.

```
*Algorithm 4.3 (Kowalski–Sadri algorithm)*
INPUT: A theory $T$, and a set of s integrity constraints, IC = { $\text{IC}_1, \cdots, \text{IC}_s$ } where for each integrity constraint a retractable atom is specified.
OUTPUT: Revised theory that satisfies all the integrity constraints in IC.
MAIN STEP: For $i$ =1 to $s$, Call Eliminate($T$, $\text{IC}_i$).
*Procedure Eliminate*($T$: Theory, IC: an integrity constraint with its retractable atom specified);
{Comments: $T$ is the input theory and also the revised theory that is output.}
begin
If $\leftarrow A(t)$,Conj is the integrity constraint, where $A(t)$ is retractable, to eliminate it by compiling it into the rules, replace each deductive rule in $T$ of the form $A(t') \leftarrow \text{Conj}'$ by
1. $A(t')\theta \leftarrow \text{Conj}'\theta, \neg\text{Conj}\theta$, where $\theta$ is the mgu of $A(t)$ and $A(t')$.
2. $A(t') \leftarrow \text{Conj}', t \neq t'$, where $t'$ is an instant of $t$.
end □
```

The following example explains this technique.

Example 4.5: Let the theory be $P(X) \leftarrow Q(X)$ and the integrity constraints be

$\text{IC}_1 : \leftarrow P(Y), R(Y)$
$\text{IC}_2 : \leftarrow P(b), S(c)$.

When we eliminate IC_1, the retractable atom $P(Y)$ in IC_1 unifies with the head of the rule $P(X) \leftarrow Q(X)$ and after elimination we obtain the rule $P(X) \leftarrow Q(X), \neg R(X)$, using STEP 1 of the procedure Eliminate. When we eliminate IC_2, we have

$P(b) \leftarrow Q(b), \neg R(b), \neg S(c)$, by STEP 1 of the method and
$P(X) \leftarrow Q(X), \neg R(X), X \neq b$, by STEP 2 of the method.
□

If we are given a preference relation between atoms through retractable atoms in each integrity constraint, we can use Kowalski and Sadri's syntactic method. We initially assumed that we are not given any such information. In that case, one may think that to compile the integrity constraint $\leftarrow P(a), Q(b), R(c)$ when the retractable atom is not given, we compile three integrity constraints $\leftarrow P(a), Q(b), R(c)$, $\leftarrow Q(b), P(a), R(c)$, and $\leftarrow R(c), Q(b), P(a)$ where the first atom in each is the retractable atom. We show by a counterexample that such a syntactic method of compiling each integrity constraint a multiple number of times, each time assuming a different atom as retractable does not necessarily provide a consistent combination.

Example 4.6: Consider combining four theories each consisting of a single fact, A; B; C; D, respectively, and the set of integrity constraints IC = $\{\leftarrow B, D\ ;\ \leftarrow A, B\ ;\ \leftarrow A, C\}$. After eliminating all versions of the integrity constraints from the union of the theories, we obtain $P = \{A \leftarrow \neg B, \neg C;\ B \leftarrow \neg D, \neg A;\ C \leftarrow \neg A;\ D \leftarrow \neg B\}$. One of the minimal models in MINSET(P), $\{A, B\}$, violates the integrity constraint. Hence, the syntactic approach does not provide the desired result.

Using Algorithm 4.2 we generate the theory $\{A \vee C; D \vee C; D \vee B\}$ whose minimal models are $\{A, D\}$, $\{C, D\}$, and $\{B, C\}$ and they all agree with the integrity constraint. □

D. Incremental Combination of Theories

In the previous sections we gave algorithms to maximally combine a set of theories, which do not have negation in the body of the rules of the theory, so that the combination of the theories is correct and consistent. Consider the case when we have combined the theories $T_1, \cdots, T_k$ to obtain a possibly disjunctive theory T, and we obtain another set of theories $T_{k+1}, \cdots, T_n$ to be combined with the initial set of theories. We would like to combine the partially combined theory T with the new set of theories so as to obtain an equivalent theory to the one we would have obtained if we had started with the theories $T_1, \cdots, T_n$. But now we cannot use the combining algorithm given in the previous sections. They can only be used to combine Horn theories without negation; and in this case T could be a disjunctive theory. We consider this problem in the following subsection.

1) Integrity Constraints and Disjunctive Theories: A disjunctive theory has multiple minimal models. As defined before, we say a disjunctive theory satisfies an integrity constraint if it is satisfied in all minimal models of the disjunctive theory. To combine theories we have to first determine whether or not the naive union of the theories violates the integrity constraints. Note that although the algorithms in previous sections generated a disjunctive theory, they made sure that the integrity constraints were satisfied.

Definition 4.1 Semantic Definition: A disjunctive theory is said to violate an integrity constraint, iff it is violated in some minimal model. □

Let IC : $\leftarrow P_1, \cdots, P_k$ be an integrity constraint and T be a theory. If T is a Horn theory then T violates IC if $\leftarrow P_1, \cdots, P_k$ has a solution with respect to T, i.e., $T \Rightarrow \exists(P_1, \cdots, P_k)$. But if T has more than one minimal model or it is a disjunctive theory this is not the case. If T is a disjunctive theory $\leftarrow P_1, \cdots, P_k$ will have a solution with respect to T iff $\leftarrow P_1, \cdots, P_k$ has a solution in *all minimal models* of T; while $\leftarrow P_1, \cdots, P_k$ violates T iff $\leftarrow P_1, \cdots, P_k$ has a solution in *at least one minimal model* of T. It may therefore be seen that, determining if an IC is violated by a disjunctive theory is not trivial.

Definition 4.2 Syntactic Definition: A query $\leftarrow Q$ is true in at least one minimal model of a theory T iff $\exists K$, where K is clause (possibly nil) and $T \vdash Q \vee K$ and $T \nvdash K$. □

The proof of the equivalence of the syntactic and semantic definition of a disjunctive theory violating an integrity constraint is similar to the proof of the equivalence of the syntactic and semantic definition of Minker's GCWA [15]. The algorithm to determine if a disjunctive theory violates an integrity constraint can easily be constructed using the above syntactic definition and Minker and Rajasekar's [20] algorithm to solve a negative ground query with respect to a disjunctive theory.

But even after finding all instances of the integrity constraints that violate the disjunctive theory, we do not know which particular minimal model is violated. Without knowing the particular minimal model violated, it is difficult to subdivide those particular minimal models violated, into a set of maximal nonviolating models. This is not a problem in the case of Horn theories as there is only a single minimal model.

The following algorithm explains semantically what we mean by a combination of disjunctive theories. In this algorithm we start with the set of minimal models of the union of the theories, and start dividing these minimal models such that we get a set of maximal models, such that none of these maximal models violate the integrity constraints. We call this algorithm a semantic algorithm because we do not have an algorithm to find which integrity constraints are violated by which minimal model. With the hope that such an algorithm can be found later, we present the following algorithm. We also hope the following algorithm will be used as a guideline to a more practical algorithm.

Algorithm 4.4 (Semantic combination of disjunctive theories)

```
INPUT: A set $S = \{T_1, \cdots, T_k\}$ of $k$ disjunctive theories and
IC = $\{\mathrm{IC}_1, \cdots, \mathrm{IC}_n\}$, a set of $n$ integrity constraints.
OUTPUT: A combined theory of the $k$ disjunctive theories,
such that it satisfies all the integrity constraints and is maximal.
STEP 1: Find all minimal models of $T_1 \cup \cdots \cup T_k$.
Let the set of minimal models be $m = \{m_1, \cdots, m_t\}$.
STEP 2: $t_1 := t$;
For $i$ =1 to $n$ (there are $n$ integrity constraints) Do
Begin
newm := $\emptyset$;
    For $j$ =1 to $t_i$ DO
    Begin
    Solve IC$_i$ with respect to $m_j$.
    Divide $m_j$ into a set of minimal models as follows.
    Apply Algorithm 4.1 to ($m_j$,IC$_i$) up to STEP 3 and let
    $m_j^*$ := be the set of minimal models .
    newm := newm $\cup$ $m_j^*$.
    end
```

m := newm
t_{i+1} := number of elements in newm.
end
STEP 3:
Let $m = \{m_1, \cdots, m_{t_{n+1}}\}$;
Remove all models from m which are subsets of some other model and let the resultant set be m'. Construct a theory whose only minimal models are the models in m'. □

We illustrate the above algorithm by the following example.

Example 4.7: Consider the combined disjunctive theory (T):
$A \vee B \leftarrow C$
$C \vee D$
E
F
which is inconsistent with respect to the integrity constraint $\leftarrow B, C$.
The minimal models of the theory are $m_1 = \{D, E, F\}$, $m_2 = \{C, B, E, F\}$ and $m_3 = \{C, A, E, F\}$. m_2 violates the integrity constraint and therefore in STEP 2 of the algorithm it is split to $m_{21} = \{C, E, F\}$ and $m_{22} = \{B, E, F\}$, so that each of them satisfy the integrity constraint. In STEP 3 of the algorithm we have $m = \{m_1, m_3, m_{21}, m_{22}\}$. Since $m_{21} \subset m_3$, and only the maximal sets should be present in m, we remove m_{21} from m. Hence, the consistent combined theory should have the minimal models as $\{m_1, m_3, m_{22}\}$. □

Theorem 5: Algorithm 4.4 generates a maximal combination of theories.

Proof: (Using Theorem 2 and Theorem 3): By Theorem 3, STEP 2 of Algorithm 4.4 divides minimal models to a maximal set of minimal models. Hence, by Theorem 2, Algorithm 4.4 generates a maximal combination. □

2) Incremental Combination Using Partial Backtracking: Let $S = \{T_1, \cdots, T_k, \cdots, T_n\}$ be a set of Horn theories, IC be a set of integrity constraints, and $T = C(T_1, \cdots, T_k, \text{IC})$. T could be a disjunctive theory. We assume that we are given the first k theories of S and subsequently, we need to combine the rest of the theories in S. If T is a disjunctive theory, we cannot use the previous algorithms. We now provide a method to combine the remaining theories with the previously combined theories. Our method takes advantage of the fact that T is a maximal and consistent combination of Horn theories. By virtue of the combining algorithm of Horn theories the clauses in T are of two major types. The first type is the set of disjunctive facts and the second type is the restricted rules and the unaffected rules and facts. We take the atoms in the disjunctive facts and combine these atoms with the remaining clauses of the theory T, to obtain a new theory T'. The theory T' is addition equivalent to the union of the original theories $T_1, \cdots, T_k$, where addition equivalence of Horn theories is defined as follows:

Definition 4.3: We say two Horn theories T_1 and T_2 are *addition equivalent* to each other (denoted by $T_1 \equiv T_2$) iff $\forall S$, where S is a set of Horn clauses (possibly empty), the minimal model of $T_1 \cup S$ is same as the minimal model of $T_2 \cup S$ and vice versa. □

Example 4.8: Consider the theory T:
$P(X) \leftarrow Q(X)$
$Q(a)$
$Q(b)$.
A theory which is addition equivalent to this theory is
$P(X) \leftarrow Q(X), X \neq a, X \neq b$
$P(a)$
$P(b)$
$Q(a)$
$Q(b)$.
The theory $T' = P(a); P(b); Q(b); Q(a)$ is not addition equivalent to T, even though their minimal models are same. This is because $P(c)$ is in the minimal model of $T \cup \{Q(c)\}$, while it is not in the minimal model of $T' \cup \{Q(c)\}$. □

Algorithm 4.5: The incremental combination algorithm
INPUT: $T_1, \cdots, T_n$ are Horn theories. $T = C(T_1, \cdots, T_k)$; and $T_{k+1}, \cdots, T_n$ are to be combined with T. IC is the set of integrity constraints satisfied by each T_i, $i = 1 \cdots n$ and T.
OUTPUT: The combined theory of T and $T_{k+1}, \cdots, T_n$.
STEP 1: By virtue of construction of T using Algorithm 4.2, $T = \text{Gen}_3 \cup \text{Gen}_4 \cup \text{Gen}_5$ where Gen_i is the theory added in the STEP i of Algorithm 4.2. Gen_3 consists of a set of disjunctive clauses. Gen_4 consists of the noninterfering facts and Gen_5 consists of the restricted rules and the noninterfering rules.
STEP 2: FACTS := The set of atoms present in the clauses of Gen_3. Let $T' := FACTS \cup \text{Gen}_4 \cup \text{Gen}_5$. By virtue of the construction of T using Algorithm 4.2 and as proved in Theorem 6, $T' \equiv T_1 \cup \cdots \cup T_k$ Now use Algorithm 4.2 to combine $T', T_{k+1}, \cdots, T_n$. □

Theorem 6: In STEP 2 of Algorithm 4.5, $T' \equiv T_1 \cup \cdots \cup T_k$.

Proof: The difference between T' and $T_1 \cup \cdots \cup T_k$ is that some clause (C) like:
$P(X_1, \cdots, X_n) \leftarrow \text{Body}$,
in $T_1 \cup \cdots \cup T_k$ is replaced by the set of clauses:
$C_1 : P(a_{11}, \cdots, a_{1n})$,
$\vdots$
$C_m : P(a_{m1}, \cdots, a_{mn})$
and
$C_{m+1} : P(X_1, \cdots, X_n) \leftarrow \text{Body}, X_{11} \neq a_{11}, \cdots, X_{1n} \neq a_{1n}, \cdots, X_{mn} \neq a_{mn}$ in T'.
Since for all such cases $C \equiv C_1 \cup \cdots \cup C_{m+1}$,
i.e., for any set of clauses S, $C \cup S$ has the same minimal model as $C_1 \cup \cdots \cup C_{m+1} \cup S$; we have $T' \equiv T_1 \cup \cdots \cup T_k$. □

Theorem 7: If $T_1' \equiv T_2'$ then $C(T_1', T_1, \cdots, T_k)$ and $C(T_2', T_1, \cdots, T_k)$ have the same minimal models.

Proof: Since $T_1' \equiv T_2'$, $T_1' \cup T_1 \cup \cdots T_k$ and $T_2' \cup T_1 \cup \cdots T_k$ have the same minimal models. Hence, the same integrity constraints violate them and since the combination is a maximal combination and the set of minimal models of the combinations are unique, the maximal combination of the above two sets of the theories have the same minimal models. □

Theorem 8: Associativity of the combination: $C(T', T_{k+1}, \cdots, T_n)$ has the same minimal models as $C(T_1, \cdots, T_k, \cdots, T_n)$.

Proof: From the previous two theorems. □

In the above algorithm we decompose T into a Horn theory T', strongly equivalent to the initial theory component of T. But this does not mean that we are starting all over again. That is because any integrity constraint that violates $T_1 \cup \cdots \cup T_k$ will also violate T', but we can determine its violation with respect to T' very easily as we only use the facts (not any rules) when we try to solve it as a query. Hence, we call our algorithm a partial backtracking algorithm. The following example explains the incremental combination algorithm.

Example 4.9: Let $T_1 \stackrel{\text{def}}{=} \{P(X) \leftarrow Q(X); Q(X) \leftarrow R(X); R(a)\}$, $T_2 \stackrel{\text{def}}{=} \{P(b)\}$, $T_3 \stackrel{\text{def}}{=} \{P(c)\}$, and the integrity constraint be $\leftarrow P(X), P(Y),\ X \neq Y$.
If we combine T_1 and T_2 first we obtain $C(T_1, T_2) = \{P(X) \leftarrow Q(X), X \neq a; Q(X) \leftarrow R(X); R(a); P(a) \vee P(b)\}$.
$T' = \{P(X) \leftarrow Q(X), X \neq a; Q(X) \leftarrow R(X); R(a); P(a); P(b)\}$.
$C(T', T_3) = \{P(X) \leftarrow Q(X), X \neq a; Q(X) \leftarrow R(X); R(a); P(a) \vee P(b) \vee P(c)\}$.
If we had combined all of the theories together as in Example 4.3 we would have obtained $C(T_1, T_2, T_3) = \{P(X) \leftarrow Q(X), X \neq a; Q(X) \leftarrow R(X); R(a); P(a) \vee P(b) \vee P(c)\}$ which is the same as $C(T', T_3)$.
Notice that when we solve the integrity constraint $\leftarrow P(X), P(Y),\ X \neq Y$ with respect to $T' \cup T_3$ we do not use any rules in T' while when we solve the integrity constraint $\leftarrow P(X), P(Y),\ X \neq Y$ with respect to $T_1 \cup T_2 \cup T_3$ we use the rules in T_1. □

E. Normal Theories

We now allow negative literals in the body of rules of a theory. In general such theories have multiple minimal models. As explained in the previous subsection, it is extremely difficult to combine theories that have multiple minimal models. We shall assume that our theories are stratified. In that case, such a theory has a single preferred model [21] among its minimal models, called the perfect model. If we combine theories, $T_1, \cdots, T_k$ all of which are stratified, it does not necessarily mean that $T = T_1 \cup \cdots \cup T_k$ is also stratified. If T is not stratified it is again very difficult to restrict it with the integrity constraints. We further assume that T is stratified. If T is stratified then we need an algorithm similar to Algorithm 4.2 to combine the theories appropriately. Since T has negated atoms in the body of the rules, the combined theory might also have negated atoms in the body of its rules.

The combined theory, which will be a disjunctive theory with negated atoms in the body of its rules, will have multiple minimal models. But, as in stratified Horn theories, where we consider only the perfect model among all minimal models, here we want to consider a subset of the set of minimal models, i.e., the set of minimal models present in MINSET(P) (defined in Definition 3.4).

We now give an example and show why Algorithm 4.2 is not sufficiently powerful to combine normal theories.

Example 4.10: Consider the following theories.

T_1 :	T_2 :
$P(X) \leftarrow \neg Q(X)$	$R(a)$
$Q(a)$.	

Let the integrity constraints be $\leftarrow P(a), R(a)$ and $\leftarrow Q(a), R(a)$.
Step 1 of Algorithm 4.2 gives us $S = < \leftarrow Q(a), R(a), \{Q(a), R(a)\}, \{\} >$.
STEPS 2 and 3 give $T = Q(a) \vee R(a)$.
STEP 5 adds the rule as it is.
The combined theory is
$Q(a) \vee R(a)$
$P(X) \leftarrow \neg Q(X)$.
The integrity constraint $\leftarrow P(a), R(a)$ is violated in one minimal model of the combined theory even though it did not violate the original union of the two theories. This is because before the combination, $Q(a)$ was *true* and $P(a)$ was *false* in all minimal models, but after the combination $Q(a)$ became *false* in some minimal models and $P(a)$ became *true* in those minimal models. Also, the new combination is not correct according to Definition 3.8. That is because $\{P(a), R(a)\}$, a minimal model of the combination is not a subset of any minimal model of the union of the original theorems.

In order to solve these problems one needs to restrict rules with negative literals in their body that are added as disjunctions to the combined theory. In this example, since $Q(a)$ is added as a disjunction we would like to restrict rules with $\neg Q(X)$ in their bodies. Hence, we restrict $P(X) \leftarrow \neg Q(X)$ as $P(X) \leftarrow \neg Q(X), X \neq a$ and the combined theory becomes
$Q(a) \vee R(a)$
$P(X) \leftarrow \neg Q(X), X \neq a$
which does not violate any of the integrity constraints. □

We now give the combination algorithm that uses Algorithm 4.2

Algorithm 4.6: The General Combination Algorithm

```
INPUT:
1. A set of $k$ normal theories $\{T_1 \cdots T_k\}$ which have to be combined. Each of the $T_i$ is stratified. The union of the $T_i$'s is also stratified.
2. A set of $s$ integrity constraints, IC = $\{\text{IC}_1, \cdots, \text{IC}_s\}$ where each integrity constraint is satisfied by each of the theories.
OUTPUT: A theory $T$, which is the maximal combination of the input set of theories, such that $T$ is also satisfied by each of the integrity constraints. The combined theory $T$ can be disjunctive.
STEP 1 to STEP 5 are same as in Algorithm 4.2.
STEP 6: For each rule $R$ in the union of the theories, that has a negative literal in its body, say $\neg P(X_1, \cdots, X_l)$, if there exists a substitution $\theta$ such $P(X_1, \cdots, X_l)\theta$ is a member of $\{A_1, \cdots, A_p\}$ defined in STEP 2 of Algorithm 4.2, then add Restrict($R$, $\theta$) to the combined theory. (Restrict is defined in Algorithm 4.2.)
end
STEP 7: $T := T \cup$ all the rules in $T_1 \cup \cdots \cup T_k$ that do not appear in $R$ (defined in STEP 5 of Algorithm 4.2) and are not restricted in STEP 6 of this algorithm.
$C(T_1 \cup \cdots \cup T_k, \text{IC}) := T$. □
```

Theorem 9: Let $T_1 \cdots T_k$ be stratified Horn theories that consist of general Horn clauses and let $T_1 \cup \cdots \cup T_k$ also be stratified. Then $C(T_1, \cdots, T_k, \text{IC})$, obtained by Algorithm 4.6 is a maximal combination with respect to $T_1 \cup \cdots \cup T_k$.

Proof: Consistency and Correctness: In Step 1 to Step 5 of the algorithm, the resulting theory is made consistent with the integrity constraints. In Step 6, by restricting the rules with negative literals in their body that are added as disjunctions to the combined theory we have correctness. Steps 6 and 7 do not affect the fact that the resulting theory is made consistent with respect to the integrity constraints.
Maximality: The only difference between Algorithm 4.6 and Algorithm 4.2 is that in the former we restrict the rules with negative literals in their body that are added as disjunctions to the combined theory. A particular ground instance of the head of these rules is not present in any minimal model of $T_1 \cup \cdots \cup T_k$, using the unrestricted form of the rule. Therefore, by restricting the rule, we do not loose maximality. □

In the case of theories with negated atoms in the body of its rules, the incremental combination of theories is not effective. The reason is that the combination is not associative. The following example illustrates what might happen.

Example 4.11: Consider the theories

T_1	T_2	T_3
$P(X) \leftarrow \neg Q(X)$	$R(a)$	$Q(a)$

Let the integrity constraint be $\leftarrow P(a), R(a)$.
$C(T_1, T_3) = \{P(X) \leftarrow \neg Q(X); Q(a)\}$.
$C(C(T_1, T_3), T_2) = \{P(X) \leftarrow \neg Q(X) Q(a); R(a);\}$.
$C(T_1, T_2) = \{P(X) \leftarrow \neg Q(X), X \neq a;\ P(a) \vee R(a)\}$
$C(C(T_1, T_2), T_3) = C(T_1, T_2) \cup \{Q(a)\}$. □

The nonmonotonic nature of negation is the reason the combination is not associative. One way to deal with this is to consider the preference relation dictated by the syntax of the program. In the above example, when we combine T_1 and T_2, its union has a perfect model $\{P(a), R(a)\}$, which violates the integrity constraint. Since $R(a)$ is a fact and $P(a)$ is obtained using default reasoning in the union of T_1 and T_2, we might assume that $R(a)$ is preferable to $P(a)$. In that case, we can eliminate $P(a)$ to make the combination consistent. To eliminate $P(a)$, we restrict the rule with $P(X)$ in the head by adding $X \neq a$ to the body. We then obtain the combined theory to be $\{P(X) \leftarrow \neg Q(X), X \neq a;\ R(a)\}$. But this approach is not general enough. If we have $R(X) \leftarrow \neg Q(X)$ instead of $R(a)$ in T_2, then we can obtain both $P(a)$ and $R(a)$ using default reasoning and we cannot decide how to choose between $P(a)$ and $R(a)$.

V. Combining Theories and the View Update Problem

The problem of view update in databases has been discussed by Fagin *et al.* [7], [6], Rossi and Naqvi [24], and others. They deal with the ambiguity that arises from translating a view update to an update over the underlying base relations. Consider the database $D = \{P(X) \leftarrow b_1(X), P(X) \leftarrow b_2(X)\}$, where $P(X)$ can only be defined intensionally. When $P(a)$ is to be inserted to this database, by only changing base facts (the basic requirement of the view update problem) there are two possible updated theories $T_1 = D \cup \{b_1(a)\}$ and $T_2 = D \cup \{b_2(a)\}$. Fagin *et al.* want the update to reflect both these theories and to have the updated theory a disjunctive theory which has multiple models (in this case two), each reflecting a possible updated theory. Rossi and Naqvi [24] give algorithms to deal with a general sequence of updates in an efficient way (both time and space). In the above example, the updated theory obtained by Rossi and Naqvi [24] is $\{P(X) \leftarrow b_1(X); P(X) \leftarrow b_2(X); b_1(a) \vee b_2(a)\}$, which is a disjunctive theory. In this paper, we dealt with the problem of combining a set of theories represented by logic programs, where we do not prefer any one theory over the others.

In the view update problem, rules are considered to be universal and hence unchangeable. In the case of combining theories, rules are part of individual theories. Since they lose their individuality when we combine them, we permit restrictions to them in the modified theory.

The insertion problem in view updates as discussed in [24] is different than combining the facts to be inserted with the original theory. In the case of combining multiple knowledge bases, the combined theory has to be consistent with respect to a set of integrity constraints.

Fagin *et al.* in section 3 of [7] consider the case of updating databases in the presence of integrity constraints. Their approach requires giving priorities to sentences of the theory. In the case of combining theories, we are concerned with the situation where priorities are not available.

If the integrity constraints are considered part of the original theory and facts are inserted, then in the updated theory the inserted facts have to be *true*. In the case of combining multiple theories, the combination is affected by the integrity constraints and the facts to be combined may become disjuncts in the combined theory.

On the other hand, the integrity constraints could be considered as deletions followed by insertions. In that case, a deletion is forced after any insertion. We deal with this case later, when we discuss deletions.

Even in the absence of integrity constraints, while combining theories we can modify rules. This is not allowed during insertion in the view update problem. The following example shows the differences in the two approaches:

Example 5.1: Let the initial theory T be
$P(X) \leftarrow b_1(X)$
$P(X) \leftarrow b_2(X)$
$b_1(a)$.

If we want to insert the fact $P(b)$ to the theory T we obtain T_1 to be
$P(X) \leftarrow b_1(X)$
$P(X) \leftarrow b_2(X)$
$b_1(a)$
$b_1(b) \vee b_2(b)$.

If we want to accomplish the insertion of fact $P(b)$ by combining T with the theory $\{P(b)\}$ we obtain the theory T_2 to be
$P(X) \leftarrow b_1(X)$
$P(X) \leftarrow b_2(X)$
$b_1(a)$
$P(b)$.

Note that T_1 is not correct according to our definition of correctness as $b_1(b)$ is present in a minimal model of T_1, even though it is absent in the minimal model of $T \cup \{P(b)\}$. □

The deletion in the view update problem is similar to making a theory consistent with respect to an integrity constraint.

When we combine a set of theories we essentially try to make the union of the theories consistent with respect to the integrity constraints. The deletion of $P(a) \wedge Q(a)$ from a theory T is equivalent to making T consistent with respect to the integrity constraint $\leftarrow P(a), Q(a)$. The updated theory T_1 in the view update problem neither changes nor deletes rules. When we make a theory consistent with respect to an integrity constraint we might change the rules. Let this theory be called T_2. Because of this T_1 is not necessarily maximal, while T_2 is always maximal. The following example illustrates this situation.

Example 5.2: Let the original theory T be

$P(X) \leftarrow b_1(X)$
$P(X) \leftarrow b_2(X)$
$b_1(a)$
$b_2(a)$.

If we want to delete $P(a)$ from T, the updated theory T_1 is

$P(X) \leftarrow b_1(X)$
$P(X) \leftarrow b_2(X)$
$\neg b_1(a)$
$\neg b_2(a)$.

If we treat deleting $P(a)$ as making T consistent with respect to the integrity constraint $\leftarrow P(a)$ we obtain the theory T_2 to be

$P(X) \leftarrow b_1(X), X \neq a$
$P(X) \leftarrow b_2(X), X \neq a$
$b_1(a)$
$b_2(a)$.

Since $b_1(a)$ and $b_2(a)$ are in the minimal model of T_2 and not in the minimal model of T_1, T_1 is not maximal. □

VI. Conclusion and Further Work

In this paper, we presented algorithms to combine knowledge contained in Horn theories. We did so in such a manner as to achieve a maximal theory. We defined the concept of a maximal theory, and we proved the maximality of our algorithms. We gave methods to combine different positive theories written in the same language such that the combination does not violate the integrity constraints and yet they are a maximal combination. We defined what it means for an integrity constraint to violate a disjunctive knowledge base and defined the notion of strong equivalence of logic programs, and used them in our algorithms and proofs. We also gave algorithms to combine them incrementally. Finally, we gave an algorithm to combine normal theories (i.e., theories with negation in their body) maximally, when the theories and their union are stratified.

We used a a restricted syntax for integrity constraints. We are presently working on generalizing the approach given in this paper to general integrity constraints.

Although we have given a semantic algorithm to combine disjunctive theories, a more practical algorithm is desirable. Practical algorithms to combine normal theories that are not stratified are also needed.

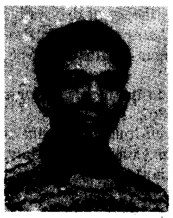
Acknowledgment

We wish to express our appreciation to the National Science Foundation for their support of our work under Grant IRI-86-09170 and the Army Research Office under Grant DAAL-03-88-K0087. We would also like to thank V. S. Subrahmanian for insightful comments on an earlier draft of the paper.

References

[1] K. R. Apt, H. A. Blair, and A. Walker, "Toward a theory of declarative knowledge," in *Foundations of Deductive Databases and Logic Programming*, J. Minker, Ed. Los Altos, CA: Morgan-Kaufmann, 1988, pp. 89–148.
[2] K. R. Apt and M. H. van Emden, "Contributions to the theory of logic programming," *J. ACM*, vol. 29, no. 3, pp. 841–862, 1982.
[3] C. Baral, J. Lobo, and J. Minker, "Generalized well-founded semantics for logic programs," Tech. Rep. CS-TR-2330, Dep. Comput. Sci., Univ. of Maryland, College Park, MD 20742, 1989. A shorter version appears in *Proc. CADE '90*.
[4] U. S. Chakravarthy, J. Grant, and J. Minker, "Foundations of semantic query optimization for deductive databases," in *Proc. Workshop Foundations Deductive Databases and Logic Programming*, J. Minker, Ed., Washington, DC, Aug. 18-22, 1986, pp. 67–101.
[5] K. L. Clark, "Negation as failure," in *Logic and Data Bases*, H. Gallaire and J. Minker, Eds. New York: Plenum, 1978, pp. 293–322.
[6] R. Fagin, G. Kuper, J. Ullman, and M. Vardi, "Updating logical databases," in *Advances in Computing Research, Vol. 3,* 1986, pp. 1–18.
[7] R. Fagin, J. D. Ullman, and M. Y. Vardi, "On the semantics of updates in databases," in *ACM SIGACT/SIGMOD Symp. Principles Database Syst.*, 1983, pp. 352–365.
[8] M. Fitting, "A Kripke-Kleene, Semantics for logic programs," *J. Logic Programming*, vol. 3, pp. 93–114, 1986.
[9] M. Fitting and M. Ben-Jacob, "Stratified and three-valued logic programming semantics," in *Proc. 5th Int. Conf. Symp. Logic Programming*, R. A. Kowalski and K. A. Bowen, Eds., Seattle, WA, Aug. 15-19, 1988, pp. 1054–1069.
[10] M. Gelfond and V. Lifschitz, "The stable model semantics for logic programming," in *Proc. 5th Int. Conf. Symp. Logic Programming*, R. A. Kowalski and K. A. Bowen, Eds., Seattle, WA, Aug. 15-19, 1988, pp. 1070–1080.
[11] R. Kowalski and F. Sadri, "Knowledge representation without integrity constraints, draft manuscript, Dec. 1988.
[12] ——, "An application of general purpose theorem proving to database integrity," in *Proc. Workshop Foundations Deductive Databases Logic Programming*, J. Minker, Ed., Washington, DC, Aug. 18–22, 1986, pp. 477–517.
[13] J. W. Lloyd, *Foundations of Logic Programming*, second ed. Berlin, Germany: Springer-Verlag, 1987.
[14] J. W. Lloyd and R. W. Topor, "Making Prolog more expressive," *J. Logic Programming*, vol. 1, no. 3, pp. 225–240, Oct. 1984.
[15] J. Minker, "On indefinite databases and the closed world assumption," in *Lecture Notes in Computer Science 138*. Berlin, Germany: Springer-Verlag, 1982, pp. 292–308.
[16] J. Minker and J. Grant, "Integrity constraints in knowledge based systems," in *Knowledge Engineering, Vol. II, Applications*, H. Adeli, Ed. New York: McGraw-Hill, 1990, pp. 1–25. Also CS-TR-2223 of Univ. of Maryland, College Park.
[17] J. Minker and D. Perlis, "Computing protected circumscription," *J. Logic Programming*, vol 2, no. 4, pp. 235–249, Dec. 1985.
[18] ——, "Applications of protected circumscription" in *Lecture Notes on Computer Science, Vol. 170, 7th Conf. Automat. Deduction*, May 1984, pp. 414–425.
[19] ——, "Protected circumscription," in *Proc. Workshop Non-Monotonic Reasoning*, Oct. 17–19, 1984, pp. 337–343.
[20] J. Minker and A. Rajasekar, "Procedural interpretation of non-Horn logic programs," in *Proc. 9th Int Conf. Automat. Deduction*, E. Lusk and R. Overbeek, Eds., Argonne, IL, May 23-26, 1988, pp. 278–293.
[21] T. C. Przymusinski, "On the declarative semantics of deductive databases and logic programming," in *Foundations of Deductive Databases and Logic Programming*, J. Minker, Ed. Los Altos, CA: Morgan-Kaufmann, 1988, pp. 193–216.
[22] ——, "Every logic program has a natural stratification and an iterated fixed point model," in *Proc. 8th ACM SIGACT-SIGMOD-SIGART Symp Principle Database Syst.*, 1989, pp. 11–21.
[23] ——, "Perfect model semantics," in *Proc. 5th Int. Conf. and Symp. Logic Programming*, R. A. Kowalski and K. A. Bowen, Eds., Seattle, WA, Aug. 15–19, 1988, pp. 1081–1096.
[24] F. Rossi and S. Naqvi, "Contributions to the view update problem," in *Proc. Int. Conf. Logic Programming Lisbon*, 1989, pp. 398–415.

[25] M. H. van Emden and R. A. Kowalski, "The semantics of predicate logic as a programming language," *J. ACM*, vol. 23, no. 4, pp. 733–742, 1976.
[26] A. Van Gelder, K. Ross, and J. S. Schlipf, "Unfounded sets and well-founded semantics for general logic programs," in *Proc. 7th Symp Principles Database Syst.*, 1988, pp. 221–230.

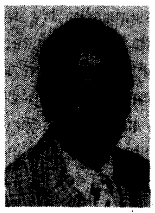


Chitta Baral received the B.Tech. (hons) degree in computer science and engineering from the Indian Institute of Technology, Kharagpur, in 1987, and the M.S. degree in computer science from the University of Maryland, College Park, in 1990

He is currently a graduate student pursuing the Ph.D. degree in computer science at the University of Maryland. He will join the University of Texas at El Paso in the Fall of 1991. His current research interests include nonmonotonic reasoning, knowledge representation, logic programming, databases, and data structures.

Sarit Kraus received the B.Sc. degree in mathematics and computer science (with distinction), the M.Sc. degree (with distinction), and Ph.D. degree in computer science from the Hebrew University, Jerusalem, Israel, in 1982, 1983, and 1989, respectively.

She was a Visiting Assistant Professor at the Institute for Advanced Computer Studies and the Department of Computer Science, University of Maryland, College Park in 1989–1990. She joined the Hebrew University in the Fall of 1990. Her current research interests include automated negotiations, programs capable of intelligent interaction with other agents, logic programming, planning, and nonmonotonic logics.



Jack Minker (M'76–SM'80–F'91) received the B.A. degree (cum laude with honors) in mathematics from Brooklyn College in 1949, the M.S. degree in mathematics from the University of Wisconsin, in 1950, and the Ph.D. degree in mathematics from the University of Pennsylvania, in 1959.

He is a Professor of Computer Science in the Department of Computer Science and the Institute for Advanced Computer Studies.

Dr. Minker serves on the Editorial Boards of a number of journals such as the *Journal of Logic Programming*, IEEE EXPERT, and *Information System Journal*. He also served as Chairman of the Advisory Committee on Computing to the National Science Foundation (1979–1982). In 1985, he received the Association for Computing Machinery's Outstanding Contribution Award for his work in human rights. He was also elected Fellow of the American Association for the Advancement of Science based on his work in artificial intelligence, database theory, and his efforts on behalf of human rights. He is also a fellow of the AAAI.