Amalgamating Knowledge Bases*

V.S.Subrahmanian†

Abstract

Integrating knowledge from multiple sources is an important aspect of automated reasoning systems. When different knowledge bases are used to store knowledge provided by multiple sources, we are faced with the problem of integrating multiple knowledge bases: under these circumstances, we are also confronted with the prospect of inconsistency. In this paper, we present a uniform theoretical framework, based on *annotated logics*, for amalgamating multiple knowledge bases when these knowledge bases (possibly) contain inconsistencies, uncertainties, and non-monotonic modes of negation. We will show that annotated logics may be used, with some modifications, to *mediate* between different knowledge bases. The multiple knowledge bases are amalgamated by a transformation of the individual knowledge bases into new annotated logic programs, together with the addition of a new axiom scheme. We will characterize the declarative semantics of such amalgamated knowledge bases, and study how the semantics of the amalgam is related to the semantics of the individual knowledge bases being combined.

1 Introduction

During recent years, there has been an increase not only in the automation of reasoning tasks, but also in the awareness that complex reasoning tasks need to be automated. Until recently, developments in the theory and practice of databases have proceeded largely on the assumption that users will consult a single appropriate knowledge base when they require information. This assumption is valid in many situations where knowledge/data is stored in a single knowledge base.

Complex reasoning tasks, on the other hand, may need to integrate information from a multitude of different sources. These sources may be databases (relational or otherwise), knowledge bases (of logical or other form), sensors (radar, heat sensors, etc.) and plant monitoring devices.

*We are grateful to the Army Research Office for supporting this work under Grant Number DAAL-03-92-G-0225 and to the National Science Foundation for their support under Grant Number IRI-9109755.

†Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, U.S.A. Electronic Mail: vs@cs.umd.edu

In a recent NSF workshop on identifying the "*technology pull* factors that will serve as forcing functions for advanced database technology" [40, p. 111], Silberschatz, Stonebraker and Ullman observed that "cooperation among different organizations on common scientific, engineering, and commercial problems will require large-scale, heterogeneous, distributed databases." They further emphasize ([40, p. 118]) "the necessity of logically integrating databases from multiple organizations into what appears to be a single database."

The primary aim of this paper is to develop a formal logical foundation for integrating multiple *deductive* databases into a single amalgamated database. Our framework is rich enough to integrate databases that contain:

- *non-monotonic modes of negation* (interpreted in terms of *both* the stable model semantics[19] and the well-founded semantics [44]),
- *uncertainty* in the form of possibilistic logic and fuzzy logic,
- *simple temporal information* such as that used to denote at what time points and/or time intervals certain statements are true, as well as
- *inconsistencies* in the sense of classical logic.

The key technical device used to perform such reasoning is the framework of multivalued *annotated logics* studied in [6, 8, 22, 23, 33, 25, 26, 27, 28, 29, 31, 36, 37, 38, 39, 41, 43]. As part of our formal theoretical framework, we will develop and study:

- *Model Theory for Local Databases:* By a local database, we mean a database that is being integrated with other (also local) databases. The model-theoretic framework will be based on the semantics of generalized annotated programs (GAPs for short) developed in [30]. GAPs are logic programs interpreted over complete lattices of truth values. The principal model-theoretic technical advance on [30] is the development of the notions of stable model and well-founded model for GAPs, thus enabling uncertainty, inconsistency and temporal data to be handled within non-monotonic local databases. This is related to work by Fitting [17] on stable model semantics for bilattices.
- *Amalgamation Theory:* Figure 1 shows how multiple knowledge bases interact with a supervisory database[1] (supervisory KBs are called *mediators* in [40, p.119]) that specifies, amongst other things, how local KBs are to combined. The language within which supervisory DBs are expressed must have the ability to reason about, and manipulate local knowledge bases. The key difference is that the supervisor/mediator will be expressed as a GAP over a different language, and that this supervisory GAP will be constructed from the local databases in a specific way by the inclusion of a specific axiom scheme. We will develop a formal model theoretical basis for combining multiple local databases (expressed as GAPs) using the conflict resolution strategies specified in a supervisory KB.
- *Fixpoint Theory:* We will develop a formal fixpoint theoretic treatment of amalgamated knowledge bases.
- *Linking Local and Amalgamated Knowledge Bases:* We will develop theorems that link the meaning of local knowledge bases with the semantics of the amalgam. In particular, we will show how the stable (resp. minimal) models of local KBs are inter-related to stable (resp. minimal) models of the amalgamated KB. Similar results will be proved for well-founded semantics.

[1] Until Definition 3.10, we will use the phrase "supervisory database" informally to mean the database that specifies how to combine information from multiple knowledge bases.

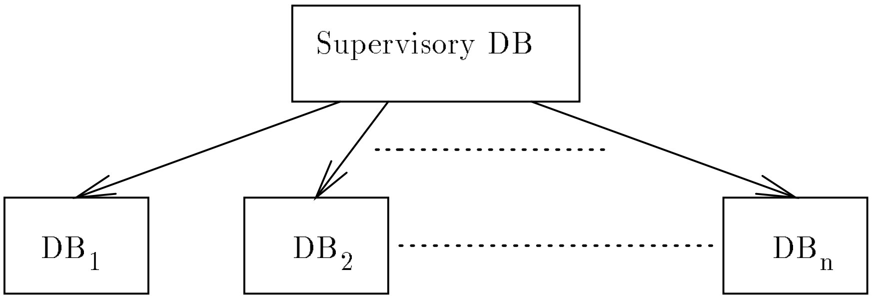


Figure 1: Interaction between Supervisory and Local Databases

2 Motivating Examples

In this section, we present (informally) three examples that motivate our work. The point of these examples is to present three situations where conflict resolution is (1) necessary and (2) has an "intuitively correct" solution. These examples motivate the need to develop a formal language for specifying conflict resolution axioms. As the paper proceeds, we will use these examples to illustrate the technical definitions.

2.1 Target Recognition Example

An autonomous vehicle in a hostile environment (e.g. a military battlefield) may need to identify objects. This is typically done as follows: using a collection of sensors, information about the object (e.g. its surface temperature, speed, structure, appearance, etc.) is gathered. For each sensor (actually, a sensor may itself be composed of hundreds of "smaller" sensors), we have a set of rules describing what the object may be, based on the properties detected by the sensor. These rules constitute a knowledge base. The observations made by the sensor about a specific object may be viewed as facts.

Let us suppose that we have three sensors, leading to databases DB_1 (containing radar information), DB_2 containing gun characteristics, and DB_3 containing information on the speed of the hostile object. DB_1, DB_2, DB_3 are given below, together with the observations made by them:

$\boxed{DB_1}$ Radar reading r_1 indicates that the object is a T-72 tank.
Radar reading r_2 indicates that the object is a T-60 tank.
Radar reading r_3 indicates that the object cannot be a T-72.
Radar reading r_4 indicates that the object is a T-80 tank.
(Observation) The object has radar reading r_3.

$\boxed{DB_2}$ Gun characteristic c_1 indicates that the object is a T-60 tank.
Gun characteristic c_2 indicates that the object cannot be a T-80.
Gun characteristic c_3 indicates that the object is a T-72 tank.
(Observation) The object has gun characteristic c_3.

$\boxed{DB_3}$ High speed indicates that the object cannot be a T-72.
High speed indicates that the object cannot be a T-80.
Medium speed indicates that the object cannot be a T-80.

(Observation) The object has high speed.

Additionally, we know that the object is either a T-72, T-60 or a T-80 tank. Note that by using DB_1 alone, we would conclude that the object is either a T-60 or a T-80. Using DB_2 alone, we would conclude that the object is a T-72 (thus contradicting the conclusion yielded by DB_1). Finally, DB_3 by itself, indicates that the object is neither a T-72 nor a T-80 (and hence presumably a T-60). This contradicts DB_2.

Notwithstanding the contradictions present, in the absence of any other data, the autonomous vehicle must draw some conclusion about what the sensed object is. In order to do this, some mechanism for resolving conflicts (fusing sensor information) is necessary. For instance, it may be known that that the gun characteristic sensor is less reliable than the other two. Let us say that this is the only conflict resolution information that can be reliably used.

Conflict Resolution

- if DB_2 says "the object is X" has truth value v and DB_1 says it has the truth value $\neg v$, then the conclusion drawn should be that of DB_1.
- if DB_2 says "the object is X" has truth value v and DB_3 says it has the truth value $\neg v$, then the conclusion drawn should be that of DB_3.

Given this conflict resolution information, we may conclude that the sensed object is a T-60. In future sections, we will formalize this knowledge, and show to conclude that the object is a T-60) tank.

2.2 Fault Identification

In the Target Recognition example, uncertainty was not specified using real numbers. Here, we will present an example where numeric modes of uncertainty are present. Let us suppose that we are dealing with a system where we have three components c_1, c_2, c_3 and at least one of them is broken. We have two sensors, thus leading to two knowledge bases, DB_1 and DB_2. They contain the following information:

DB_1

- If sensor 1 is in state s_1, then there is a 75% chance that component 1 is broken, and a 25% chance that component 2 is broken.
- If sensor 1 is not in state s_1, then there is an 80% chance that component 3 is broken, and a 20% chance that component 2 is broken.
- (Observation) Sensor 1 is in state s_1.

DB_2

- If sensor 2 is in state w_1, then there is a 60% chance that component 2 is broken, a 30% chance that component 3 is broken, and a 10% chance that component 1 is broken.
- If sensor 2 is not in state w_1, then there is a 90% chance that component 3 is broken, a 5% chance that component 1 is broken, and a 5% chance that component 2 is broken.

- If sensor 2 is in state w_1, then if there is an over 30% chance that a component is broken, then check it.
- (Observation) Sensor 2 is in state w_1.

Here, we are faced with the following options. According to sensor 1, it is most likely that component 1 is broken, while according to sensor 2, it is most likely that component 2 is broken. However, sensor 1 assigns a higher certainty that component 2 is broken (25%) than sensor 2 assigns to component 1 being broken (10%). Integrating these pieces of information in order to determine what action to take involves resolving this conflict. Such a resolution may well take into account, specific knowledge about the characteristics of the components, of the sensors' reliability, and the impact of these probabilities. The conflict resolution scheme may contain the following rules:

Conflict Resolution

- Check component 2 if both sensors 1 and 2 say it has over 50% chance of being broken.
- Check component 1 if both sensors 1 and 2 say it has over 75% chance of being broken.
- Check component 2 if sensor 2 says it has over 55% chance of being broken and sensor 1 says that it has over 20% chance of being broken.

Given this conflict resolution information, we may conclude (intuitively) that component 2 should be checked. This example contains decisions being made with uncertain information.

2.3 Employee Example: Time-Stamped Information

Information in knowledge base(s) may come armed with a "time-stamp" recording when the piece of information was inserted into the knowledge base. Inconsistencies may arise because old information (in one or more of the multiple databases being considered) has not been appropriately modified/deleted.

For example, in a large corporation, different sites may maintain different knowledge bases about personnel employed at that site, with all these "local" knowledge bases being accessible to a central knowledge base at the corporation's head office. It may turn out that some individual, John Doe, was transferred from Boston to Seattle. When reaching Seattle, John Doe's records (reflecting his new position) were entered into Seattle's local database, but (perhaps mistakenly) his records were not deleted from the knowledge base in Boston. In this very simple example, let us assume we have the following data:

Boston DB

- John Doe is a Manager (entered on June 1, 1990).
- John Doe makes 42K (entered on June 1, 1991).

Seattle DB

- John Doe is a Vice-President (entered on June 1, 1992).

- John Doe makes 50K (entered on June 1, 1992).

In addition, we assume that there are background axioms specifying that nobody has two positions, nobody can make two different salaries, etc. In this case, conflict resolution says that we accept the most recent data.

Conflict Resolution

- If $p(X)$ held at time t and and $p(Y)$ holds at time $t' > t$ and $p(Y) \rightarrow \neg p(X)$ follows from the knowledge base, then conclude $p(Y)$.

In this situation, if somebody from the corporate head office wanted to know John Doe's current salary, then they could conclude (correctly) that it is 50K.

3 Amalgamation Logic: Syntax

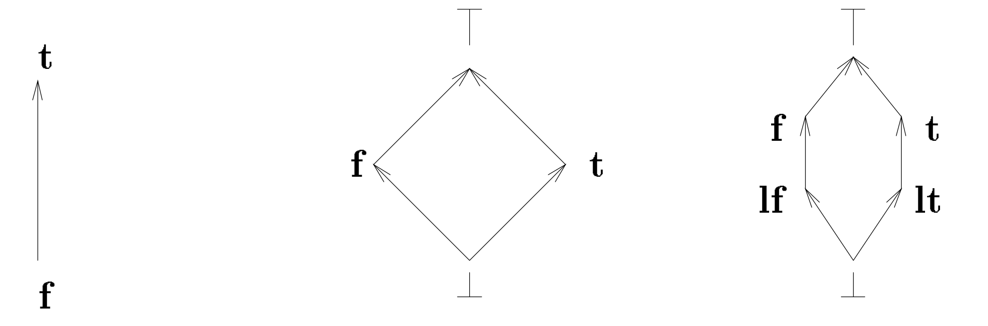
In this section, we describe the basic syntax of the language in which databases are expressed.

3.1 The Underlying Languages

The base language we will use is that of annotated logic developed by Subrahmanian [41] and studied further in [6, 8, 26, 27, 28, 29, 31, 39, 43].

Our framework will be based on a slight extension of annotated logic that takes the cross product of M (where $\mathsf{M} \geq 1$) complete lattices $\mathcal{T}_1, \ldots, \mathcal{T}_{\mathsf{M}}$ – in the case when $\mathsf{M} = 1$ and negations are absent, we will obtain the framework of Kifer and Subrahmanian[30] as a special case. We use the notation, $\mathcal{T}^{\mathrm{loc}} = \mathcal{T}_1 \times \cdots \times \mathcal{T}_{\mathsf{M}}$. Some examples of individual lattices that could be involved in the cross product are (however, users are free to choose other lattices to reflect their individual preferences):

1. TWO: the set $\{\mathbf{t}, \mathbf{f}\}$ of "classical" truth values ordered as shown in Figure 2(a).
2. FOUR: the set $\{\mathbf{t}, \mathbf{f}, \bot, \top\}$ ordered as shown in Figure 2(b). This set of truth values was initially studied by Belnap and used in logic programming by Blair and Subrahmanian [6] and Fitting [15, 16]. FOUR is extremely useful in reasoning about inconsistency: intuitively, $\top$ stands for the truth value "inconsistent in classical logic," while $\bot$ stands for "unknown."
3. SIX: This lattice was introduced in [2] and consists of the set $\{\mathbf{t}, \mathbf{f}, \mathbf{lt}, \mathbf{lf}, \bot, \top\}$ ordered as shown in Figure 2(c). Intuitively, **lt** and **lf** stand, respectively, for "likely to be true" and "likely to be false." This lattice contains the lattice FOUR embedded in it.
4. DEF: Ginsberg [20] has proposed a lattice of truth values – the elements in this lattice are $\{\mathbf{t}, \mathbf{f}, \top, \bot, \mathbf{d}\top, \mathbf{dt}, \mathbf{df}\}$. Intuitively, **dt** stands for "true by default."
5. $\Re[0, 1]$: the set of real numbers between (and including) 0 and 1. This is a complete lattice under the usual $\leq$ ordering. The real numbers in the $[0, 1]$ interval may be viewed as "fuzzy" or, in other cases, as "possibilistic" truth values[9].
6. TIME_1: the power-set of non-negative integers is a complete lattice when ordered under the $\subseteq$ ordering.



(a) TWO **(b) FOUR** **(c) SIX**

7. TIME_2: the set of all closed intervals of non-negative real numbers is a complete lattice when ordered under the $\subseteq$ ordering.

Base Language: We assume that we have an underlying logical language L generated by a finite set of constant symbols, function symbols, and predicate symbols. In addition, L contains an infinite set, OV, of variable symbols (called *ordinary variables*). The concepts of terms, atoms, and literals are defined in the usual way (cf. Lloyd [32]). L is the language in which all the local knowledge bases are expressed.

$\mathcal{T}_j$ Languages: Each member of $\mathcal{T}_j$, $1 \leq j \leq \mathsf{M}$, is a constant symbol in the $\mathcal{T}_j$-language. In addition, for each $1 \leq j \leq \mathsf{M}$, and each $i \geq 1$, we postulate that there is a family $\mathcal{F}_j^i$ of total continuous (hence monotonic, [32]) functions, each of type $(\mathcal{T}_j)^i \rightarrow \mathcal{T}_j$, called *annotation functions over lattice* $\mathcal{T}_j$. We assume that each $(\mathcal{F}_j^i)$ contains an i-ary function, $\sqcup_i$, derived from the lattice LUB operator $\sqcup$ for lattice $\mathcal{T}_j$, which, given inputs $\mu_1, \ldots, \mu_i$, returns the least upper bound, in lattice $\mathcal{T}_j$, of $\{\mu_1, \ldots, \mu_i\}$. Slightly abusing notation, we will often write $\sqcup$ instead of $\sqcup_i$.

Without loss of generality, we will assume that the base language and the $\mathcal{T}_j$ languages possess no symbols in common. Terms in the base language (resp. $\mathcal{T}_j$-languages) are constructed out of the constant symbols, function symbols, and variable symbols of that language. Ground terms in these languages are terms that contain no variables from that language. The concept of an atom is defined (in the usual way) only for the base language (as the other two languages contain no predicate symbols).

Definition 3.1 (Annotated Atom) If ψ_j, $1 \leq j \leq \mathsf{M}$, is a $\mathcal{T}_j$-term, and A is an atom in the base language, then $A : [\psi_1, \ldots, \psi_{\mathsf{M}}]$ is an *annotated atom.* ∎

When $\mathsf{M} = 1$, we will often abuse notation and write $A : \psi_1$ instead of $A : [\psi_1]$. In the sequel, we will use ψ to denote a metavariable ranging over objects in the $\mathcal{T}_j$ language, while μ will stand for an M-tuple of the form $[\psi_1, \ldots, \psi_{\mathsf{M}}]$.

Definition 3.2 (Annotated Clause) If $A_0 : \mu_0, \ldots, A_n : \mu_n, A_{n+1} : \mu_{n+1}, \ldots, A_{n+m} : \mu_{n+m}$ are annotated atoms, then

$$\begin{array}{rcl} A_0 : \mu_0 & \leftarrow & A_1 : \mu_1 \,\&\, \ldots \,\&\, A_n : \mu_n \,\& \\ & & \mathbf{not}(A_{n+1} : \mu_{n+1}) \,\&\, \ldots \,\&\, \mathbf{not}(A_{n+m} : \mu_{n+m}). \end{array}$$

is an *annotated clause*. All variables occurring in an annotated clause are assumed to be implicitly universally quantified at the front of the clause. The above annotated clause is said to be *negation-free* just in case $m = 0$. ∎

Definition 3.3 (Generalized Annotated Program) A *generalized annotated program* (GAP, for short) is a finite set of annotated clauses. A GAP is said to be *negation-free* just in case it contains only negation-free annotated clauses. ∎

In [30], Kifer and Subrahmanian studied the semantics of negation-free GAPs. In Sections 5.1 and 6, we will characterize the semantics of GAPs containing negation.

3.2 Amalgamated Knowledge Bases

Using the notation just set out, we are now in a position to define amalgamated logic programs. Suppose $DB_1, \ldots, DB_{\mathsf{N}}$ are "local databases" expressed in annotated logic using the product lattice $\mathcal{T}^{\mathrm{loc}} = \mathcal{T}_1 \times \cdots \times \mathcal{T}_{\mathsf{M}}$.

<u>DNAME *Language:*</u> In addition to the base language and the $\mathcal{T}_j$-languages ($1 \leq j \leq \mathsf{M}$), we assume that we have a language to name sets of databases. For simplicity, we will assume that the databases are represented by the integers $1, \ldots, \mathsf{N}$ and s is the name of the supervisor. These are called DNAME constants. In addition, we have an infinite set, DV, of variables, called DNAME variables which range over the DNAME constants $1, \ldots, \mathsf{N}, \mathsf{s}$. A DNAME term is either a *set* of DNAME constants, or a DNAME variable.

If $A = p(\vec{t})$ is an atom of our base language, and D is a DNAME term, then $p_D(\vec{t})$ is called a DNAME atom and is denoted A_D. For example, if we consider the target recognition example, then $obj_{\{1,3\}}(t72)$ is a DNAME atom.

Definition 3.4 If ψ_j, $1 \leq j \leq \mathsf{M}$, is a $\mathcal{T}_j$-term, and $p_D(\vec{t})$ is a DNAME atom, then $p_D(\vec{t}) : [\psi_1, \ldots, \psi_{\mathsf{M}}]$ is an *amalgamated atom.*

Note that amalgamated atoms are just annotated atoms – the only difference is that the "base language" over which they are expressed is constructed from the base language of the local knowledge bases, as well as the names of the local databases. In other words, if L was the base language over which the local databases are constructed, then $\mathsf{amal}(L)$, the base language over which the amalgamation is constructed, is identical to L except in one respect – the set of predicate symbols of $\mathsf{amal}(L)$ is defined as: if p is an n-ary predicate symbol in L, and if $D \subseteq \{1, \ldots, \mathsf{N}, \mathsf{s}\}$, then p_D is an n-ary predicate symbol of $\mathsf{amal}(L)$.

Thus, given a collection of local databases $DB_1, \ldots, DB_{\mathsf{N}}$, amalgamated knowledge bases are just GAPs whose atoms are of the form $A_D : [\psi_1, \ldots, \psi_{\mathsf{M}}]$, where $A : [\psi_1, \ldots, \psi_{\mathsf{M}}]$ is an annotated atom. Hence, the entire theory of GAPs carries over to the case of amalgamated KBs.

In this paper, we will abuse notation in one way – instead of writing $A_D : [\psi_1, \ldots, \psi_{\mathsf{M}}]$, we will write $A : [D, \psi_1, \ldots, \psi_{\mathsf{M}}]$ – this prevents potentially long subscripts. The next example shows how the conflict resolution rules for the Target Recognition example may be represented using amalgamated KBs.

Example 3.1 (Representing Conflict Resolution Axioms for the Target Recognition Example) The axioms may be represented as follows:

$$\begin{array}{rcl} obj(X):[\{\mathbf{s}\},U] & \leftarrow & obj(X):[\{1\},U]\,\&\,obj(X):[\{2\},neg(U)]. \\ obj(X):[\{\mathbf{s}\},V] & \leftarrow & obj(X):[\{3\},V]\,\&\,obj(X):[\{2\},neg(V)]. \\ obj(X):[\{\mathbf{s}\},V] & \leftarrow & obj(X):[\{1,3\},V]. \end{array}$$

Here, X is an ordinary variable, while U,V are $\mathcal{T}^{\mathrm{loc}}$-variables. neg is a function over FOUR that maps $\mathbf{t}$ to $\mathbf{f}$, $\mathbf{f}$ to $\mathbf{t}$, and $\perp$ and $\top$ to themselves. The first rule says that if DB_1 and DB_2 are in conflict about the truth value of $obj(X)$, then the truth value assigned to $obj(X)$ is the truth value assigned by DB_1. The other rules are similar. ∎

Example 3.2 (Fault Identification Example) The amalgamated clause

$$check(c_2):[\{\mathbf{s}\},1] \quad \leftarrow \quad broken(c_2):[\{1\},0.5]\,\&\,broken(c_2):[\{2\},0.5].$$

represents one of the conflict resolution statements in the fault identification example. It says that "According to the supervisor, component c_2 must be checked with certainty 1 if both DB_1 and DB_2 say that component c_2 is broken with certainty at least 0.5. ∎

3.3 Model Theory for GAPs

In this section, we recapitulate the model theory of GAPs developed in [30]. The reader familiar with [30] may skip ahead to section 3.4.

Definition 3.5 (r-Interpretation) [30] A *restricted* Herbrand interpretation (*r-interpretation*, for short) of language L, is any map from the Herbrand Base, B_L, to $\mathcal{T}^{\mathrm{loc}}$. ∎

In particular, observe that when local GAPs are being considered, L is the *base language.*

Throughout this paper, we will use I^{loc} to denote "local" interpretations – interpretations associated with local GAPs that assign values in $\mathcal{T}^{\mathrm{loc}}$ to atoms in L.

Before explaining the notion of satisfaction, we observe that when $\mathcal{T}$ is a complete lattice (under ordering $\leq$), the ordering $\leq$ can be extended pointwise to make the set of r-interpretations of the Base Language L a complete lattice as follows: $I_1 \leq I_2$ iff for all $A \in B_L$, $I_1(A) \leq I_2(A)$. Here, I_1 and I_2 are r-interpretations.

Definition 3.6 (r-Satisfaction) Suppose I is an r-interpretation and $\mu \in \mathcal{T}$. Then:

1. If A is a ground atom, $I \models^r A:\mu$ if and only if $I(A) \geq \mu$.
2. $I \models^r \mathbf{not}(A:\mu)$ iff I is not an r-model of $A:\mu$.
3. $I \models^r F_1 \,\&\, F_2$ if and only if $I \models^r F_1$ and $I \models^r F_2$.
4. $I \models^r F_1 \vee F_2$ if and only if $I \models^r F_1$ or $I \models^r F_2$.
5. $I \models^r F_1 \leftarrow F_2$ if and only if $I \models^r F_1$ or $I \not\models^r F_2$.
6. $I \models^r F_1 \leftrightarrow F_2$ if and only if $I \models^r (F_1 \leftarrow F_2)$ and $I \models^r (F_2 \leftarrow F_1)$.
7. $I \models^r (\forall x)F$ if and only if $I \models^r F(x/t)$ for all ground terms t. Here x is an object variable or an annotation variable and t must be of the same sort as x (i.e., either a usual ground first-order term, or an element of $\mathcal{T}$). $F(x/t)$ denotes the replacement of all free occurrences of x in F by t.

8. $I \models^r (\exists x)F$ if and only if $I \models^r F(x/t)$ for some ground term t, where x is an object or annotation variable.[2]
9. If F is not a closed formula, then $I \models^r F$ if and only if $I \models^r (\forall)F$, where $(\forall)F$ denotes the universal closure of F. ∎

I is an r-model of a GAP DB iff $I \models^r C$ for all annotated clauses in DB.

Definition 3.7 (r-Logical Consequence) A formula F is said to be an *r-logical-consequence* of a negation-free GAP, DB, (denoted $DB \models^r F$) iff for every r-model I of DB, it is also the case that $I \models^r F$. ∎

3.4 Model Theory for Amalgamated KBs

As we have observed earlier, given a collection of local databases $DB_1, \ldots, DB_{\mathsf{N}}$ over lattice $\mathcal{T}^{\mathrm{loc}}$, an amalgamated KB is just a GAP over the same lattice – the only difference is that the base language, L, is replaced by a new language, $\mathsf{amal}(L)$, that subscripts predicate symbols in L with subsets of $\{1, \ldots, \mathsf{N}, \mathsf{s}\}$.

As amalgamated KBs are GAPs, it follows that the entire model theory of GAPs applies to amalgamated KBs as well. We use I^{aml} to denote amalgamated interpretations (**A**-interpretations, for short) – interpretations that are maps from $B_{\mathrm{amal(L)}}$ to $\mathcal{T}^{\mathrm{loc}}$. We will often abuse notation by considering **A**-interpretations to be maps from $B_{\mathrm{amal(L)}}$ to $(\{1, \ldots, \mathsf{N}, \mathsf{s}\} \longrightarrow \mathcal{T}^{\mathrm{loc}})$, i.e. $I^{\mathrm{aml}}(A)$ is considered to be a map from $\{1, \ldots, \mathsf{N}, \mathsf{s}\}$ to $\mathcal{T}^{\mathrm{loc}}$. There is no loss of generality in this when we observe that $I^{\mathrm{aml}}(A)(D) = \mu$ can be viewed as syntactic sugar for $I^{\mathrm{aml}}(A_D) = \mu$. When $D = \{i\}$ is a singleton, we will write $I^{\mathrm{aml}}(A)(i)$ instead of $I^{\mathrm{aml}}(A)(\{i\})$.

Question 1 At this point, the reader may find himself/herself asking the question: if I^{aml} is an **A**-interpretation, and $D_1 \subseteq D_2$, then surely there should be some connection between $I^{\mathrm{aml}}(A)(D_1)$ and $I^{\mathrm{aml}}(A)(D_2)$?

The answer is "yes." We will address this point later on in this section via the notion of a "combination axiom scheme" which is used to define the notion of the *amalgam* of local knowledge bases $DB_1, \ldots, DB_{\mathsf{N}}$ with a supervisory knowledge base S.

What does it mean for an **A**-interpretation I^{aml} to satisfy $A : [D, \mu]$? According to the definition of r-satisfaction, I^{aml} satisfies $A : [D, \mu]$ iff $\mu \leq I(A_D)$ iff $\mu \leq I(A)(D)$. Later on in this section, we will introduce a "combination axiom scheme" which will cause $I(A)(D) = \sqcup_{D' \subset D} I(A)(D')$. In other words, if $D = \{d_1, d_2\}$, then the *truth value that databases d_1 and d_2 jointly assign* to A is the least upper bound of the truth values assigned to A by d_1 and d_2 individually. However, if the supervisor contains information that allows it to prefer one over the other, then the supervisor can make a different choice.

We use the $\models^{\mathbf{A}}$ to distinguish between satisfaction by local interpretations and **A**-interpretations. The following example illustrates the notion of **A**-satisfaction.

[2] If t is a ground $\mathcal{T}$-term, it can be identified with a constant in $\mathcal{T}$, since all annotation functions are evaluable.

Example 3.3 (Target Recognition Example) Consider the amalgamated clause

$$obj(t72) : [\{\mathbf{s}\}, \mathbf{t}] \;\leftarrow\; obj(t72) : [\{1,3\}, \mathbf{t}].$$

This clause is a ground instance of the last clause of Example 3.1. Consider the $\mathbf{A}-interpretation$ I_1^{aml} where $I_1^{\text{aml}}(obj(t72))$ is the function:

$I_1^{\text{aml}}(obj(t72))(1) = \mathbf{t}$ $\quad$ $I_1^{\text{aml}}(obj(t72))(2) = \mathbf{f}$
$I_1^{\text{aml}}(obj(t72))(3) = \mathbf{t}$ $\quad$ $I_1^{\text{aml}}(obj(t72))(\mathbf{s}) = \mathbf{t}$

When D is not a singleton set, let $I_1^{\text{aml}}(A)(D) = \sqcup_{d\in D} I_1^{\text{aml}}(A)(d)$. I_1^{aml} is an $\mathbf{A}$-model of the above clause. $I_1^{\text{aml}} \models^{\mathbf{A}} obj(t72) : [\{1,3\}, \mathbf{t}]$ as $I_1^{\text{aml}}(obj(t72))\ (\{1,3\}) = \sqcup\{I_1^{\text{aml}}(obj(t72))(1), I_1^{\text{aml}}(obj(t72))(3)\} = \mathbf{t}$. As I_1^{aml} $\mathbf{A}$-satisfies the body of the above clause, for it to be an $\mathbf{A}$-model, it must satisfy the head as well. This occurs because $I_1^{\text{aml}}(obj(t72))(s) = \mathbf{t}$. ∎

We now prove that GAPs may be successfully embedded into amalgamated knowledge bases. To prove this embedding, we need a simple definition.

Definition 3.8 Suppose DB_i is a GAP and $C =$

$$A_0 : \mu_0 \;\leftarrow\; A_1 : \mu_1 \,\&\, \ldots \,\&\, A_n : \mu_n \,\&\, \mathbf{not}(A_{n+1} : \mu_{n+1}) \,\&\, \ldots \,\&\, \mathbf{not}(A_{n+m} : \mu_{n+m})$$

is an annotated clause in DB_i. Then the *amalgamation transform* of C, denoted $AT(C)$, is the amalgamated clause

$$\begin{array}{rcl} A_0 : [\{i\}, \mu_0] & \leftarrow & A_1 : [\{i\}, \mu_1] \,\&\, \ldots \,\&\, A_n : [\{i\}, \mu_n] \,\&\, \\ & & \mathbf{not}(A_{n+1} : [\{i\}, \mu_{n+1}]) \,\&\, \ldots \,\&\, \mathbf{not}(A_{n+m} : [\{i\}, \mu_{n+m}]). \end{array}$$

The *amalgamation transform*, $AT(DB_i)$, of DB_i, is the set $\{AT(C) \mid C \in DB_i\}$. ∎

It turns out (as theorem 3.1 below shows), that the negation-free GAP, DB_i, and the amalgamated knowledge base, $AT(DB_i)$, are equivalent in a very precise sense. Before formalizing this equivalence, we need to establish a link between r-interpretations and $\mathbf{A}-interpretations$.

Definition 3.9 (Locale) Suppose I^{loc} is a local interpretation. The *locale* of I^{loc} w.r.t. a GAP DB_i, is the set $\{I^{\text{aml}} \mid I^{\text{aml}}$ is an $\mathbf{A}$-interpretation and for all ground atoms A, $I^{\text{aml}}(A)(i) = I^{\text{loc}}(A)\}$. ∎

Intuitively, the locale of a local interpretation associated with DB_i is the set of all $\mathbf{A}$-interpretations that agree with that local interpretation as far as database DB_i is concerned.

Theorem 3.1 (Amalgamation Theorem) Suppose DB_i is a GAP. Then: $I^{\text{loc}} \models^r DB_i$ implies that for each and every I^{aml} in the locale of I^{loc} w.r.t. DB_i, it is the case that $I^{\text{aml}} \models^{\mathbf{A}} AT(DB_i)$. ∎

Theorem 3.2 (Second Amalgamation Theorem) Suppose DB_i is a GAP and I^{loc} is a local interpretation such that there exists an I^{aml} in the locale of I^{loc} (w.r.t. DB_i) which is an $\mathbf{A}$-model of $AT(DB_i)$. Then I^{loc} is an r-model of DB_i. ∎

Corollary 3.1 Suppose DB_i is a GAP, and $A : \mu$ is a ground annotated atom. Then: $DB_i \models^r A : \mu$ iff $AT(DB_i) \models^{\mathbf{A}} A : [\{i\}, \mu]$. ∎

The above results say that as far as GAPs are concerned, using the amalgamation transformation converts them into "equivalent" amalgamated knowledge bases. The essence of our technique for amalgamating knowledge bases may now be summed up as follows:

> Given GAPs $DB_1, \ldots, DB_n$, and an amalgamated knowledge base S (the supervisory knowledge base), the semantics of the combination of $DB_1, \ldots, DB_n$ with S is the same as the semantics of the amalgamated knowledge base $S \cup \bigcup_{i=1}^{n} AT(DB_i) \cup$ special axioms.

What are these "special axioms" referred to above ? They are related to Question 1. Intuitively, if database 1 says that A's truth value is at least μ_1 and database 2 says that A's truth value is at least μ_2, we say that the *databases* $\{1, 2\}$ *jointly say* that A's truth value is at least $\mu_1 \sqcup \mu_2$. However, in the definition of **A**-interpretations, there is no rule that enforces this, i.e. it is possible for an **A**-interpretation, I^{aml}, over lattice FOUR, to assign, say, $\mathbf{t}$ to $A_{\{1\}}$, $\mathbf{f}$ to $A_{\{2\}}$, and $\perp$ to $A_{\{1,2\}}$, which is not what we want. To prevent such interpretations from occurring, we add the following axiom scheme:

$$(\mathbf{CombinationAxiomScheme}): \qquad \mathbf{B} : [D, \bigsqcup_{\emptyset \subset D' \subset D} V_{D'}] \leftarrow \bigwedge_{\emptyset \subset D' \subset D} \mathbf{B} : [D', V_{D'}]$$

where $D \subseteq \{1, \ldots, \mathsf{N}\}$. Note that this is an axiom scheme – it says that the truth value of $\mathbf{B}$ according to the databases in D is at least the lub of the truth value of $\mathbf{B}$ according to the non-empty subsets of D. This axiom scheme can be converted to a set of implications by instantiating D and D' to subsets of $\{1, \ldots, \mathsf{N}, \mathbf{s}\}$. This causes an extra $(2^{\mathsf{N}+1} - 1) \times 2 = 2^{\mathsf{N}+2} - 2$ amalgamated clauses to be added to the amalgamated KB. However, there is no need to explicitly do this in practice, though in the development of the declarative semantics, we will assume that this has been done. In the rest of the paper, we will study the semantics of amalgamated knowledge bases in the presence of this combination axiom scheme (unless stated otherwise, all amalgamated KBs will be assumed to contain the Combination Axioms).

In particular, note that when $D = \{i\}$ is a singleton, then $\mathbf{B} : [\{i\}, \perp]$ is the instance of the combination axiom scheme (the lub of the empty set is $\perp$) obtained by instantiation D to i.

Definition 3.10 Suppose $DB_1, \ldots, DB_n$ are GAPs. A *supervisory database* S is a set of clauses such that every ground instance of a clause in S is of the form:

$$\begin{aligned} A_0 : [\{\mathbf{s}\}, \mu] \quad \leftarrow \quad & A_1 : [D_1, \mu_1] \,\&\, \ldots \,\&\, A_n : [D_n, \mu_n] \,\&\, \\ & \mathbf{not}(A_{n+1} : [D_{n+1}, \mu_{n+1}]) \,\&\, \ldots \,\&\, \mathbf{not}(A_{n+m} : [D_{n+m}, \mu_{n+m}]) \end{aligned}$$

where, for all $1 \leq i \leq (n + m)$, $D_i \subseteq \{1, \ldots, n, \mathbf{s}\}$.

The above definition of a supervisory database S allows us great flexibility in specifying it – we may place whatever clauses we feel appropriate inside S. The reason for placing (at this time) no restrictions on what is in S is that this allows us the greatest possible generality when studying the semantics of amalgamated databases (described below). In practice, the supervisory database may often contain clauses such as:

$$p(\vec{X}) : [\{\mathbf{s}\}, V] \quad \leftarrow \quad p(\vec{X}) : [\{1\}, V] \,\&\, \ldots \,\&\, p(\vec{X}) : [\{n\}, V].$$

The above clause says that if all local databases agree on the truth value of an atom $A = p(\ldots)$, then the supervisor will assign at least that consensus value to atom A.

Definition 3.11 Suppose $DB_1, \ldots, DB_n$ are GAPs and S is an amalgamated knowledge base (the supervisory DB). The *amalgam* of $(DB_1, \ldots, DB_n, S)$ is the amalgamated knowledge base $(S \cup \bigcup_{i=1}^{n} AT(DB_i) \cup$ Combination Axioms). ∎

It is very important to note that the amalgam $(DB_1, \ldots, DB_n, S)$ of the local databases $DB_1, \ldots, DB_n$ with the supervisor S is a GAP over an expanded set of predicate symbols (i.e. where predicate symbols may be subscripted with sets D of database names). In particular, each instance of the combination axiom scheme is a negation-free annotated clause over this expanded alphabet.

4 Fixpoint Theory

Given any GAP P (possibly containing negation), we can associate an operator, denoted $\mathbf{R}_P^e$, that maps interpretations to interpretations.

Definition 4.1 Suppose I is an r-interpretation and $A \in B_{\mathcal{L}}$. Then: $\mathbf{R}_P^e(I)(A) = \sqcup\{f(\mu_1, \ldots, \mu_n) \,|\, A : f(\mu_1, \ldots, \mu_n) \leftarrow B_1 : \mu_1 \,\&\, \ldots \,\&\, B_n : \mu_n \,\&\, \mathbf{not}(B_{n+1} : \mu_{n+1}) \,\&\, \ldots \,\&\, \mathbf{not}(B_{n+m} : \mu_{n+m})$ is a ground instance of a clause in P such that $I \models^r (B_1 : \mu_1 \& \ldots \& B_n : \mu_n)$ and for all $1 \leq j \leq m$ $I \not\models^r B_{n+j} : \mu_{n+j}\}$. ∎

Kifer and Subrahmanian [30] have proved that when P contains no negation, $\mathbf{R}_P^e$ is monotone (but not continuous), that the pre-fixpoints of $\mathbf{R}_P^e$ coincide with the r-models of P, and that $\mathbf{R}_P^e$ has a unique "least" r-model. In addition:

Theorem 4.1 Suppose P is any GAP (possibly containing negation). Then:

1. I is an r-model of P iff $\mathbf{R}_P^e(I) \leq I$.
2. If I is a $\leq$-minimal model of P, then $I = \mathbf{R}_P^e(I)$. ∎

4.1 Fixpoint Characterization of Amalgamated KBs

As amalgamated KBs are just GAPs over an expanded base language, Definition 4.1 tells us how to associate, with any amalgamated KB, Q, a fixpoint operator $\mathbf{R}_Q^e$. As Q contains the combination axioms, $\mathbf{R}_Q^e(I)(A_D) = \{\mu \mid A : [D', \mu] \leftarrow B_1 : [D_1, \mu_1] \,\&\, \ldots \,\&\, B_n : [D_n, \mu_n] \,\&\, \mathbf{not}(B_{n+1} : [D_{n+1}, \mu_{n+1}]) \,\&\, \ldots \,\&\, \mathbf{not}(B_{n+m} : [D_{n+m}, \mu_{n+m}])$ is a ground instance of a clause in Q and for all $1 \leq i \leq n$, $\mu_i \leq I(B_i)(D_i)$ and for all $(n+1) \leq j \leq (n+m)$, $\mu_j \not\leq I(B_j)(D_j)$ and $D' \subseteq D\}$. Instead of writing $\mathbf{R}_Q^e(I)(A_D)$, we will write $\mathbf{A}_Q(I)(A)(D)$ to emphasize the fact that we are considering the fixpoint operator associated with an amalgamated KB.

Example 4.1 (Fault Identification Example Revisited) Let Q denote the amalgam of the local databases DB_1 and DB_2 with the supervisory DB shown in Appendix C. Consider any $\mathbf{A}$-interpretation, I^{aml} such that:

$I^{\text{aml}}(sensor1(s_1))(1) = 1.$	$I^{\text{aml}}(sensor2(s_1))(2) = 0.$
$I^{\text{aml}}(sensor1(w_1))(1) = 0.$	$I^{\text{aml}}(sensor2(w_1))(2) = 1.$
$I^{\text{aml}}(broken(c_1))(1) = 0.9$	$I^{\text{aml}}(broken(c_2))(1) = 0.5$
$I^{\text{aml}}(broken(c_1))(2) = 0.9$	$I^{\text{aml}}(broken(c_2))(2) = 0.5.$

Then, $\mathbf{A}_Q(I^{\text{aml}})$ makes the following assignments:

$\mathbf{A}_Q(I^{\text{aml}})(sensor1(s_1))(1) = 1.$	$\mathbf{A}_Q(I^{\text{aml}})(sensor2(s_1))(2) = 0.$
$\mathbf{A}_Q(I^{\text{aml}})(sensor1(w_1))(1) = 0.$	$\mathbf{A}_Q(I^{\text{aml}})(sensor2(w_1))(2) = 1.$
$\mathbf{A}_Q(I^{\text{aml}})(broken(c_1))(1) = 0.75$	$\mathbf{A}_Q(I^{\text{aml}})(broken(c_2))(1) = 0$
$\mathbf{A}_Q(I^{\text{aml}})(broken(c_3))(1) = 0$	$\mathbf{A}_Q(I^{\text{aml}})(broken(c_1))(2) = 0.1$
$\mathbf{A}_Q(I^{\text{aml}})(broken(c_2))(2) = 0.6$	$\mathbf{A}_Q(I^{\text{aml}})(broken(c_3))(2) = 0.3$
$\mathbf{A}_Q(I^{\text{aml}})(check(c_1))(\{\mathsf{s}\}) = 0$	$\mathbf{A}_Q(I^{\text{aml}})(check(c_2))(\{\mathsf{s}\}) = 1.$

Before proceeding to study the relationship between the semantics of local KBs and the semantics of the amalgam of the local KBs with a supervisor, we define the concept of *projection* of an $\mathbf{A}$-interpretation. Projections of $\mathbf{A}$-interpretations are analogous to locales (Definition 3.9); given an r-interpretation, a "locale" is the set of $\mathbf{A}$-interpretations that "agree" with that r-interpretation as far as a particular local database is concerned. Conversely, given an $\mathbf{A}$-interpretation, the projection of that $\mathbf{A}$-interpretation on a given local database is the (unique) r-interpretation obtained by restricting the $\mathbf{A}$-interpretation to a specific database.

Definition 4.2 (Projection) Suppose Q is the amalgam of $(DB_1, \ldots, DB_{\mathsf{N}}, S)$ and I^{aml} is an $\mathbf{A}$-interpretation. The *projection of I^{aml} on DB_i for* $1 \leq i \leq \mathsf{N}$ is the r-interpretation I^{loc} defined as follows: $I^{\text{loc}}(A) = I^{\text{aml}}(A)(i)$. ∎

Example 4.2 Suppose there are two local databases, DB_1 and DB_2 and only two propositions, p and q, in the base language. Suppose the lattice $\mathcal{T}$ being considered is FOUR. Let I^{aml} be the $\mathbf{A}$-interpretation defined below:

$I^{\text{aml}}(p)(1) = \mathbf{t}.$	$I^{\text{aml}}(p)(2) = \mathbf{f}.$
$I^{\text{aml}}(q)(1) = \top.$	$I^{\text{aml}}(q)(2) = \mathbf{t}.$

The *projection* of I^{aml} on DB_1 is the r-interpretation, I^{loc} such that $I^{\text{loc}}(p) = \mathbf{t}$ and $I^{\text{loc}}(q) = \top$. ∎

In the next section, we will use projections of an $\mathbf{A}$-interpretation to relate the semantics of an amalgamated knowledge base to the semantics of the local knowledge bases.

4.2 Relating Local Knowledge Bases and Amalgams: The Case of Negation-Free GAPs

When $DB_1, \ldots, DB_{\mathsf{N}}$ are negation-free local knowledge bases, and S is a supervisory knowledge base, then the amalgam of $(DB_1, \ldots, DB_{\mathsf{N}}, S)$ is closely related to the individual local knowledge bases.

Theorem 4.2 (Third Amalgamation Theorem) Suppose $DB_1, \ldots, DB_{\mathsf{N}}$ are negation-free GAPs and S is any supervisory knowledge base. Let Q be the amalgam of databases $(DB_1, \ldots, DB_{\mathsf{N}}, S)$. Suppose I^{aml} is a fixpoint of $\mathbf{A}_Q$. For $1 \leq j \leq \mathsf{N}$, let I_j^{loc} be the projection of I^{aml} on DB_j. Then: I_j^{loc} is a fixpoint of $\mathbf{R}^e_{DB_j}$. ∎

Intuitively, I_j^{loc} is the r-interpretation obtained by extracting the information supplied about the local knowledge base DB_j by the $\mathbf{A}$-interpretation I^{aml}. The first part of the theorem shows that every fixpoint of the amalgam of $(DB_1, \ldots, DB_{\mathsf{N}}, S)$ is an expansion of a corresponding fixpoint of the local knowledge base DB_j. It is important to note that the theorem does *not* say that every fixpoint of the operator associated with a local knowledge base can be expanded to a corresponding fixpoint of the amalgam of $(DB_1, \ldots, DB_{\mathsf{N}}, S)$. The following example shows that this may not be true in general:

Example 4.3 Suppose we have only two knowledge bases, DB_1 and DB_2. Suppose the lattice $\mathcal{T}^{\mathrm{loc}}$ being considered is FOUR. DB_1 contains only one fact: $p : \mathbf{t}$. Similarly, DB_2 contains only one fact, $p : \mathbf{f}$. The supervisor, S, contains the rule[3]:

$$p : [\{\mathbf{s}\}, \top] \;\leftarrow\; \mathbf{not}(p : [\{\mathbf{s}\}, \top]).$$

The operator, $\mathbf{R}^e_{DB_1}$, associated with DB_1 has a unique fixpoint, viz. the r-interpretation I_1^{loc} which assigns $\mathbf{t}$ to p. Likewise, the operator, $\mathbf{R}^e_{DB_2}$, associated with DB_2 has a unique fixpoint, viz. the r-interpretation I_2^{loc} which assigns $\mathbf{f}$ to p. The amalgam of (DB_1, DB_2, S) contains the clause in S above, as well as the two facts:

$$p : [\{1\}, \mathbf{t}] \leftarrow; \qquad\qquad p : [\{2\}, \mathbf{f}] \leftarrow$$

together with the combination axiom scheme. If we use Q to denote the amalgam, it is easy to verify that $\mathbf{A}_Q$ has no fixpoint. Hence, there is no way the fixpoints of $\mathbf{R}_{DB_1}$ (resp. $\mathbf{R}_{DB_2}$) may be expanded to a fixpoint of $\mathbf{A}_Q$. Theorem 4.2 says that the converse is true, i.e. every "global" fixpoint (i.e. fixpoint of the amalgam) is an expansion of local fixpoints. ∎

What this suggests is that supervisory knowledge bases should not be allowed to be arbitrary amalgamated KBs. Below, we formally define a strong supervisory database.

Definition 4.3 Suppose $DB_1, \ldots, DB_{\mathsf{N}}$ are GAPs. A *strong supervisory* knowledge base is a finite set of amalgamated clauses such that each ground instance of a clause in the strong supervisory KB is of the form:

$$\begin{aligned} A_0 : [\{\mathbf{s}\}, \mu] \;\leftarrow\; & A_1 : [D_1, \mu_1] \,\&\, \ldots \,\&\, A_n : [D_n, \mu_n] \,\&\, \\ & \mathbf{not}(A_{n+1} : [D_{n+1}, \mu_{n+1}]) \,\&\, \ldots \,\&\, \mathbf{not}(A_{n+m} : [D_{n+m}, \mu_{n+m}]) \end{aligned}$$

where each D_j, $1 \leq j \leq (n+m)$, is a subset of $\{1, \ldots, \mathsf{N}\}$. In particular, the D_j's are not allowed to evaluate to a set with $\mathbf{s}$ in it. ∎

[3]This is an incarnation of the familiar program $p \leftarrow \mathbf{not}\, p$ in logic programming that is well known not to possess a stable model[19].

Thus, strong supervisory databases differ from arbitrary amalgamated knowledge bases in the sense that ground instances of clauses can:

- only have heads of the form $A_0 : [\{\mathbf{s}\}, \mu]$, i.e. the first component of the annotation is fixed to $\{\mathbf{s}\}$, and
- the first component of all annotations in the clause body must be a subset of $\{1, \ldots, \mathsf{N}\}$.

The amalgams of the local databases and the supervisory databases pertaining to the Target Recognition Example, the Employee Example, and the Fault Identification Example are all strong supervisory knowledge bases.

The following result shows that when S is a strong supervisory database, then every fixpoint of a local knowledge base can be expanded to a fixpoint of the amalgam of the local knowledge bases with a strong supervisory knowledge base.

Theorem 4.3 (Fourth Amalgamation Theorem) Suppose $DB_1, \ldots, DB_{\mathsf{N}}$ are negation-free GAPs and S is any *strong* supervisory knowledge base. Let Q be the amalgam of $(DB_1, \ldots, DB_{\mathsf{N}}, S)$. Let I_j^{loc} be an r-interpretation. Then: If I_j^{loc} is a fixpoint of $\mathbf{R}_{DB_j}$ then there is an $\mathbf{A}$-interpretation I^{aml} in the locale of I_j^{loc} such that I^{aml} is a fixpoint of $\mathbf{A}_Q$ (i.e. I_j^{loc} can be "expanded" to a fixpoint, I^{aml}, of $\mathbf{A}_Q$). ∎

The theorem has the following corollary:

Corollary 4.1 Suppose $DB_1, \ldots, DB_{\mathsf{N}}$ are negation-free GAPs and S is any *strong* supervisory knowledge base. Let Q be the amalgam of $(DB_1, \ldots, DB_{\mathsf{N}}, S)$, A be any ground atom, $D \subseteq \{1, \ldots, \mathsf{N}\}$, and $\mu \in \mathcal{T}^{\mathrm{loc}}$. Then: If $Q \models^{\mathbf{A}} A : [D, \mu]$, then for all $i \in D$, there exists a $\mu_i \in \mathcal{T}^{\mathrm{loc}}$ such that

1. $DB_i \models A : [\{i\}, \mu_i]$ and
2. $\mu \leq \sqcup\{\mu_1, \ldots, \mu_{\mathsf{N}}\}$. ∎

It can also be easily seen from the proof of Theorem 4.3 that Theorem 4.3 holds when S is negation-free (even if it is not strong).

5 Scenarios and Stable Models

In the preceding section, we studied the semantics of amalgamated knowledge bases, and amalgams of $(DB_1, \ldots, DB_{\mathsf{N}}, S)$ when $DB_1, \ldots, DB_{\mathsf{N}}$ were negation-free. In this section, we study the situation that arises when the DB_i's, $1 \leq i \leq n$, are allowed to contain the non-monotonic negation symbol **not**. Under these circumstances, multiple plausible scenarios may arise when the supervisory database is under-specified, i.e. S does not contain enough information to definitively resolve conflicts between two or more knowledge bases.

5.1 r-Stable Models of GAPs

Suppose P is a GAP (possibly containing negation). Given an r-interpretation I, we may transform P into a negation-free program, $\mathbf{G}(P, I)$ as follows:

1. If $A_0 : \mu_0 \leftarrow A_1 : \mu_1 \,\&\, \ldots \,\&\, A_n : \mu_n \,\&\, \mathbf{not}(A_{n+1} : \mu_{n+1}) \,\&\, \ldots \,\&\, \mathbf{not}(A_{n+m} : \mu_{n+m})$ is a ground instance of a clause in P, and for all $1 \leq j \leq m$, $\mu_{n+j} \not\leq I(A_{n+j})$, then $A_0 : \mu_0 \leftarrow A_1 : \mu_1 \,\&\, \ldots \,\&\, A_n : \mu_n$ is in $\mathbf{G}(P, I)$.
2. Nothing else is in $\mathbf{G}(P, I)$.

As $\mathbf{G}(P, I)$ is a negation-free program, the operator $\mathbf{R}_{\mathbf{G}(P,I)}$ has a least fixpoint, denoted $\mathsf{lfp}(\mathbf{R}_{\mathbf{G}(P,I)})$. We use these observations to associate with a GAP, P, an operator, denoted $\mathcal{G}_P$ defined as follows: $\mathcal{G}_P(I) = \mathsf{lfp}(\mathbf{R}_{\mathbf{G}(P,I)})$.

Definition 5.1 I is said to be an *r-stable model* of P iff $I = \mathcal{G}_P(I)$. ∎

Example 5.1 (Fault Identification Example Revisited) Consider the local database DB_1 of the Fault Identification example (Appendix C). This database has exactly one r-stable model, viz. the r-interpretation I^{loc} that assigns 0.75 to $broken(c_1)$, 0.25 to $broken(c_2)$ and 0 to all other atoms. ∎

Example 5.2 Consider the following GAP over the lattice $\mathcal{T}$ = FOUR.

$$\begin{aligned} p : \mathbf{t} &\leftarrow \mathbf{not}(q : \mathbf{t}). \\ q : \mathbf{t} &\leftarrow \mathbf{not}(p : \mathbf{t}). \end{aligned}$$

This GAP has two r-stable models, I_1^{loc} and I_2^{loc} such that: $I_1^{\mathrm{loc}}(p) = \mathbf{t}$, $I_1^{\mathrm{loc}}(q) = \bot$, $I_2^{\mathrm{loc}}(p) = \bot$, $I_2^{\mathrm{loc}}(q) = \mathbf{t}$. ∎

Theorem 5.1 Suppose P is any GAP and I_1 and I_2 are r-interpretations. Then:

1. $I_1 \leq I_2$ implies that $\mathbf{G}(P, I_2) \subseteq \mathbf{G}(P, I_1)$.
2. $I_1 \leq I_2$ implies that $\mathcal{G}_P(I_2) \leq \mathcal{G}_P(I_1)$. ∎

Theorem 5.2 Suppose P is any GAP and I is an r-stable model of P. Then I is a $\leq$-minimal model of P. ∎

As amalgamated KBs are just GAPs over a different alphabet (by expanding the set of predicate symbols and adding instances of the combination axiom scheme), the notion of a stable model for amalgamated KBs follows immediately from the definitions given above. We give an example below to illustrate the concept of a stable model for amalgamated KBs.

Example 5.3 (Fault Identification Example Revisited) Let us return to the fault identification example (Appendix C), and consider the $\mathbf{A}$-stable models of the amalgam (DB_1, DB_2, S), which we will denote by Q. Q has exactly one stable model, I^{aml}, which is defined as follows:

$I^{\text{aml}}(sensor1(s_1))(1) = 1.$ $\quad$ $I^{\text{aml}}(sensor2(w_1))(2) = 1.$
$I^{\text{aml}}(sensor1(s_1))(\{\mathrm{s}\}) = 1.$ $\quad$ $I^{\text{aml}}(sensor2(w_1))(\{\mathrm{s}\}) = 1.$
$I^{\text{aml}}(broken(c_1))(1) = 0.75.$ $\quad$ $I^{\text{aml}}(broken(c_1))(2) = 0.10.$
$I^{\text{aml}}(broken(c_1))(\{\mathrm{s}\}) = 0.$ $\quad$ $I^{\text{aml}}(broken(c_2))(1) = 0.25.$
$I^{\text{aml}}(broken(c_2))(2) = 0.6.$ $\quad$ $I^{\text{aml}}(broken(c_2))(\{\mathrm{s}\}) = 0.$
$I^{\text{aml}}(check(c_1))(1) = 0.$ $\quad$ $I^{\text{aml}}(check(c_1))(2) = 0.$
$I^{\text{aml}}(check(c_1))(\{\mathrm{s}\}) = 0.$ $\quad$ $I^{\text{aml}}(check(c_2))(1) = 0.$
$I^{\text{aml}}(check(c_2))(2) = 1.$ $\quad$ $I^{\text{aml}}(check(c_2))(\{\mathrm{s}\}) = 1.$
$I^{\text{aml}}(check(c_3))(1) = 0.$ $\quad$ $I^{\text{aml}}(check(c_1))(2) = 1.$
$I^{\text{aml}}(check(c_3))(\{\mathrm{s}\}) = 0.$

For all other ground atoms A, $I^{\text{aml}}(A)(\{\mathrm{s}\}) = 0.$ ∎

The following theorem is of fundamental importance: it says that whenever each local database has an r-stable model, if the supervisory database is "strong," then the amalgam of the local databases with the supervisor has a stable model.

Theorem 5.3 (Fifth Amalgamation Theorem) Suppose Q is the amalgam of databases $(DB_1, \ldots, DB_{\mathsf{N}}, S)$ where S is a strong supervisory knowledge base. If each local database DB_i, $1 \le i \le \mathsf{N}$, has an r-stable model, then Q has a stable model. ∎

5.2 Relating Stable Models of Local KBs and Amalgams

In this section, we show that whenever Q is the amalgam of $(DB_1, \ldots, DB_{\mathsf{N}}, S)$ where S is a strong supervisory knowledge base, there is a strong relationship between the stable models of Q and those of the local databases $DB_1, \ldots, DB_{\mathsf{N}}$.

The following theorem shows that if I^{loc} is an r-stable model of a local database, then I^{loc} "generates" a stable model of the amalgam (assuming all local databases have r-stable models).

Theorem 5.4 (Sixth Amalgamation Theorem) Suppose Q is the amalgam of databases $(DB_1, \ldots, DB_{\mathsf{N}}, S)$ where S is a strong supervisory knowledge base and where each local database DB_j, $1 \le j \le \mathsf{N}$, has an r-stable model. Suppose I_j^{loc} is an r-stable model of the GAP DB_j. Then there exists a stable model in the locale of I_j^{loc}. ∎

The converse is true as well.

Theorem 5.5 (Seventh Amalgamation Theorem) Suppose Q is the amalgam of $(DB_1, \ldots, DB_{\mathsf{N}}, S)$ where S is a strong supervisory knowledge base and where each local database DB_i, $1 \le i \le \mathsf{N}$, has an r-stable model. Furthermore, suppose I^{aml} is a stable model of Q. Then the *projection*, I_j^{loc}, of I^{aml} on DB_j, $1 \le j \le \mathsf{N}$, is an r-stable model of DB_j. ∎

Theorem 5.5 says that every stable model of the amalgam generates an r-stable model of each local database.

6 Well-Founded Models

In this section, we will first define the well-founded semantics of GAPs, and then study the relationship between the semantics of the local knowledge bases and the amalgam of the local knowledge base with the supervisory knowledge base.

Theorem 5.1 shows us that the operator $\mathcal{G}_P$ is anti-monotonic – hence, the function $\mathcal{G}_P^2$, which applies $\mathcal{G}_P$ twice is monotonic and hence, is always guaranteed to have a fixpoint.

Proposition 6.1 Let P be any GAP. Then $\mathcal{G}_P^2$ is monotonic, and is guaranteed to possess a unique $\leq$-minimal fixpoint, denoted $\mathsf{lfp}(\mathcal{G}_P^2)$ and a unique, $\leq$-maximal fixpoint, denoted $\mathsf{gfp}(\mathcal{G}_P^2)$. ∎

Note that the fact that $\mathcal{G}_P^2$ has a fixpoint does not necessarily mean that $\mathcal{G}_P$ always has a fixpoint. To see this, consider the following example:

Example 6.1 Let P be the single clause program: $p : \mathbf{t} \leftarrow \mathbf{not}(p : \mathbf{t})$, where the lattice $\mathcal{T}$ being considered in FOUR. $\mathcal{G}_P$ has no fixpoint (and hence P has no r-stable model). However, the interpretation I_1 which assigns $\mathbf{t}$ to p is a fixpoint of $\mathcal{G}_P^2$ because $\mathcal{G}_P(I)(p) = \perp$, and $\mathcal{G}_P^2(I)(p) = \mathbf{t}$. In fact, the interpretation I_2 which assigns $\perp$ to p is also a fixpoint of $\mathcal{G}_P^2$. ∎

Definition 6.1 The *well-founded semantics* of a GAP, P, is defined by the following entailment relation:

- $P \overset{\text{wfs}}{\models_r} A : \mu$ iff $\mu \leq \mathsf{lfp}(\mathcal{G}_P^2)(A)$.
- $P \overset{\text{wfs}}{\models_r} \mathbf{not}(A : \mu)$ iff $\mu \not\leq \mathsf{gfp}(\mathcal{G}_P^2)(A)$. ∎

The following theorem shows that the well-founded semantics approximates the r-stable semantics in the sense that whenever P WFS-entails a ground atom $A : \mu$, then every r-stable model also satisfies $A : \mu$ (and similarly for negative information).

Theorem 6.1 (GAPs: Well-Founded Semantics Approximates Stable Semantics) Suppose P is any GAP, and suppose I is any r-stable model of P. Then:

1. If $P \overset{\text{wfs}}{\models_r} A : \mu$, then $\mu \leq I(A)$, i.e. I satisfies $A : \mu$.
2. If $P \overset{\text{wfs}}{\models_r} \mathbf{not}(A : \mu)$, then $\mu \not\leq I(A)$, i.e. $I(A)$ does not satisfy $A : \mu$. ∎

Finally, we observe that as amalgamated KBs are just GAPs of a different sort, the notion of well-founded semantics applies equally well to amalgamated KBs and is captured by the above definition. As in the case of stable semantics, there is a close correspondence between the well-founded semantics of local knowledge bases, and the well-founded semantics of the amalgam resulting from the combination of local knowledge bases with a supervisory knowledge base.

Theorem 6.2 (Eighth Amalgamation Theorem) Suppose Q is the amalgam of databases $(DB_1, \ldots, DB_{\mathsf{N}}, S)$, $A : \mu$ is an annotated atom, and $1 \leq i \leq \mathsf{N}$ is an integer. Then:

1. $DB_i \overset{\text{wfs}}{\models}_r A : \mu$ iff $Q \overset{\text{wfs}}{\models}_{\mathbf{A}} A : [\{i\}, \mu]$.
2. $DB_i \overset{\text{wfs}}{\models}_r \mathbf{not}(A : \mu)$ iff $Q \overset{\text{wfs}}{\models}_{\mathbf{A}} \mathbf{not}(A : [\{i\}, \mu])$.

Here, $\overset{\text{wfs}}{\models}_{\mathbf{A}}$ denotes the well-founded entailment relationship for amalgamated KBs. ∎

The theorem says that if DB_i says (using WFS) that $A : \mu$ is true, then the amalgam of $(DB_1, \ldots, DB_{\mathsf{N}}, S)$ will say (also by using WFS) that "DB_i says that $A : \mu$." Note that this is quite different from the supervisor saying that $A : \mu$ is true.

7 Multiple Truth Value Domains

Thus far, we have (implicitly) assumed that all local databases $DB_1, \ldots, DB_{\mathsf{N}}$ are over the same complete lattice $\mathcal{T}^{\text{loc}}$. In other words, we have not specified how to amalgamate databases $DB_1, \ldots, DB_{\mathsf{N}}$ where DB_i is a GAP over lattice $\mathcal{W}_i$.

This problem is quite straightforward. Suppose $\mathcal{W}_i$ is a complete lattice under ordering $\preceq_i$. Then the set $\mathcal{W} = \mathcal{W}_1 \times \cdots \times \mathcal{W}_{\mathsf{N}}$ is a complete lattice under the pointwise ordering $\preceq$ defined as:

$$(x_1, \ldots, x_{\mathsf{N}}) \preceq (y_1, \ldots, y_{\mathsf{N}}) \text{ iff } x_1 \preceq_1 y_1 \,\&\, \cdots \,\&\, x_{\mathsf{N}} \preceq_n y_{\mathsf{N}}.$$

It is straightforward to verify that the $\preceq$ ordering makes $\mathcal{W}$ a complete lattice. Given a specific local database DB_i, $1 \leq i \leq \mathsf{N}$, we can now replace the clause (in DB_i) of the form:

$$A_0 : \mu_0 \quad \leftarrow \quad A_1 : \mu_1 \,\&\, \ldots \,\&\, A_k : \mu_k \,\&\, \mathbf{not}(B_1 : \rho_1) \,\&\, \ldots \,\&\, \mathbf{not}(B_m : \rho_m)$$

by the following clause over the lattice $\mathcal{W}$:

$$\begin{aligned} A_0 : [\perp_1, \ldots, \perp_{i-1}, \mu_0, \perp_{i+1}, \ldots, \perp_{\mathsf{N}}] \quad \leftarrow \quad & A_1 : [\perp_1, \ldots, \perp_{i-1}, \mu_1, \perp_{i+1}, \ldots, \perp_{\mathsf{N}}] \,\&\, \ldots \,\&\, \\ & A_k : [\perp_1, \ldots, \perp_{i-1}, \mu_k, \perp_{i+1}, \ldots, \perp_{\mathsf{N}}] \,\&\, \\ & \mathbf{not}(B_1 : [\perp_1, \ldots, \perp_{i-1}, \rho_1, \perp_{i+1}, \ldots, \perp_{\mathsf{N}}] \,\&\, \ldots \,\&\, \\ & \mathbf{not}(B_m : [\perp_1, \ldots, \perp_{i-1}, \rho_m, \perp_{i+1}, \ldots, \perp_{\mathsf{N}}]. \end{aligned}$$

where $\perp_i$ is the bottom element of lattice $\mathcal{W}_i$. This converts each DB_i to a set of clauses over the complete lattice $\mathcal{W}$. When this transformation is applied to all local databases, the local databases that were initially over different truth value domains are now over the common product lattice $\mathcal{W}$.

To illustrate this concept further, we present an example below.

Example 7.1 Let us consider an example where we have two databases DB_1 and DB_2 over lattices FOUR and SIX, respectively (Figure 2 shows these two lattices). One can see that FOUR is embedded within SIX and hence (in an informal sense), SIX provides finer granularity than FOUR. Suppose DB_1 and DB_2 are as given below:

DB_1:	DB_2:
$p : \mathbf{t} \leftarrow$	$p : \mathbf{lt} \leftarrow$
$q : \mathbf{t} \leftarrow$	$q : \mathbf{lf} \leftarrow$

We can consider $\mathcal{T}$ to be the lattice FOUR $\times$ SIX. The supervisor, S, may be the following amalgamated KB:

$$\begin{array}{rcl} p : [\{\mathbf{s}\}, (\perp, V)] & \leftarrow & p : [\{1\}, V]. \\ q : [\{\mathbf{s}\}, (\perp, V_1 \sqcap_{\text{SIX}} V_2]] & \leftarrow & q : [\{1\}, V_1] \ , \& \, q : [\{2\}, V_2] \end{array}$$

Intuitively, the first clause in S says that the supervisor always prefers DB_1's truth value (over that assigned by DB_2) as far as proposition p is concerned, while the second clause says that the supervisor will only conclude the joint consensus opinion of DB_1 and DB_2 as far as q is concerned. Note that $\sqcap_{\text{SIX}}$ denotes greatest lower bounds in the lattice SIX ($\sqcap_{\text{FOUR}}$ likewise denotes greatest lower bounds in FOUR). It is easy to see that for all $x, y \in$ FOUR, $x \sqcap_{\text{FOUR}} y = x \sqcap_{\text{SIX}} y$.

7.1 Related Work

In the logic programming and deductive database arena, cooperative problem solving by manipulating multiple knowledge bases has been studied in three contexts: metalogic programming, database updates, and multidatabases.

Metalogic programming was introduced by Bowen and Kowalski [7] who showed how to amalgamate object language and metalanguage via the device of Godel numbering. They showed that this allowed a database to refer to components of the database. Subrahmanian [42, pps. 88-89] showed how metalogic programming may be used to address cooperative reasoning problems, as well as the database update problem [42, pps. 91-94]. However, the development of a formal semantics for (even monotonic) metalogic programming has remained a rather difficult and contentious issue.

Database updates are also closely related to combining multiple knowledge bases. Given a database DB and a new fact F, updating DB by asserting F is tantamount to combining the database DB with a database containing the single fact F with priority being given to the update, F. This has been studied by Fagin et. al. [13, 14].

A great deal of work has been done in *multidatabase* systems and *interoperable* database systems[21, 46, 47]. However, most of this work combines standard relational databases (no deductive capabilities). Not much has been done on the development of a semantic foundation for such databases. The work of Grant et. al. [21] is an exception: the authors develop a calculus and an algebra for integrating information from multiple databases. This calculus extends the standard relational calculus. Further work specialized to handle inter-operability of multidatabases is critically needed. However, our paper addresses a different topic – that of integrating multiple deductive databases containing (possibly) inconsistencies, uncertainty, non-monotonic negation, and possibly even temporal information. Zicari et. al [47] describe how interoperability may be achieved between a rule-based system (deductive DB) and an object-oriented database using special *import/export* primitives. No formal theory is developed in [47]. Perhaps closer to our goal is that of Whang et. al. [46] who argue that Prolog is a suitable framework for schema integration. In fact, the approach of Whang et. al. is in the same spirit as that of metalogic programming discussed earlier. Whang et. al. do not give a formal semantics for multi-databases containing inconsistency and/or uncertainty and/or non-monotonicity and/or temporal information.

Baral et. al. [3, 4] have developed algorithms for combining different logic databases which generalizes the update strategy by giving priorities to some updates (when appropriate) and as well as not giving priorities to updates (which corresponds to combining two theories without any preferences). Combining two theories corresponds, roughly, to finding maximally consistent subsets (also called flocks by Fagin et. al. [13, 14]). As we shall argue in Section 7.2, our framework can express maximal consistency as well. [3, 4] do not develop a formal model-theoretic treatment of combining multiple knowledge bases, whereas our method does provide such a model theory. [3, 4] are unable to handle non-monotonicity (in terms of stable/well-founded semantics), nor uncertainty, nor time-stamped information – our framework is able to do so.

Dubois, Lang and Prade [11], also suggest that formulas in knowledge bases can be annotated with, for each source, a lower bound of a degree of certainty associated with that source. The spirit behind their approach is similar to ours, though interest is restricted to the $[0,1]$ lattice, the stable and well-founded semantics are not addressed, and amalgamation theorems are not studied. However, for the $[0,1]$ case, their framework is a bit richer than ours when nonmonotonic negations are absent. The authors have extended their work to accommodate time in [10].

In [17], Fitting generalizes results in [5, 45], to obtain a well-founded semantics for bilattice-based logic programs.[4] Unlike annotated logics, bilattice-valued logic programs are just (unannotated) sets of clauses $C \equiv$

$$A \;\leftarrow\; B_1 \,\&\ldots\&\, B_n \,\&\, \neg D_1 \,\&\ldots\&\, \neg D_m.$$

An interpretation, ν, assigns truth values (contained in an algebraic structure called a bilattice) to atoms. Given a bilattice-valued logic program $\mathcal{Z}$, it is possible to associate a function $\Psi_{\mathcal{Z}}$, which takes two valuations ν_1, ν_2, and returns as output, the valuation $\Psi_{\mathcal{Z}}(\nu_1,\nu_2)(A) = \sqcup_t\{\sqcap_t\{\nu_1(B_1),\ldots,\nu_1(B_n),\neg\nu_2(D_1),\ldots,\neg\nu_2(D_m)\} \,|\, A \leftarrow B_1 \,\&\ldots\&\, B_n \,\&\, \neg D_1 \,\&\ldots\&\, \neg D_m$ is a ground instance of a clause in $\mathcal{Z}\}$. Here, $\sqcup_t$ (resp. $\sqcap_t$) denotes the least upper bound (resp. greatest lower bound) w.r.t. the truth-ordering, $\leq_t$. Given clause C as shown above, let C_{ann} be the annotated clause:

$$\begin{aligned} A : \sqcap_t(V_1,\ldots,V_n,\neg(W_1),\ldots,\neg(W_m)) \;\leftarrow\;& B_1 : V_1 \& \ldots \& B_n : V_n \& \\ & D'_1 : W_1 \& \ldots \& D'_m : W_m. \end{aligned}$$

Here, the $\neg$ occurring in the annotation in the head of C_{ann} is a "negation-function" that maps $\mathcal{B}$ to $\mathcal{B}$. The D'_i's are new atoms, obtained by introducing new "primed" predicates. Given an (unannotated) logic program $\mathcal{Z}$, let $\mathcal{Z}_{\mathsf{ann}}$ be the annotated program $\{C_{\mathsf{ann}} \mid C \in \mathcal{Z}\}$.[5] Then:

Theorem 7.1 Let $(\mathcal{B},\leq_t,\leq_k)$ be a bilattice, ν_1,ν_2 be $\mathcal{B}$-valued interpretations, and $\mathcal{Z}$ be a bilattice-valued logic program. Consider $\mathcal{Z}_{\mathsf{ann}}$ to be an annotated program over the lattice $(\mathcal{B},\leq_t)$. Then, for all (unprimed) ground atoms A in the language of $\mathcal{Z}$, $\Psi_{\mathcal{Z}}(\nu_1,\nu_2)(A) = \mathbf{R}_{\mathcal{Z}_{\mathsf{ann}}}(I)$ where $I(A) = \nu_1(A)$ for all un-primed atoms, and $I(A) = \nu_2(A)$ for all primed atoms. ∎

The theorem shows that Fitting's $\Psi_{\mathcal{Z}}$ can be accurately captured by the $\mathbf{R}$ operator associated with the annotated logic program $\mathcal{Z}_{\mathsf{ann}}$.

[4] A bilattice, due to M. Ginsberg [20] is a set that is a complete lattice under two orderings, a "truth" ordering, $\leq_t$, and a "knowledge" ordering, $\leq_k$. In addition, LUBs and GLBs according to each of these orderings have to satisfy certain additional conditions – namely, that they be monotone w.r.t. the other ordering and that various distributivity properties hold.

[5] An important point to note is that in Fitting's semantics, $\neg$ is antimonotonic w.r.t. the $\leq_t$ operator, and hence, this clause transforms non-monotonicity from the body of C to a non-monotonic annotation function in the head of C.

Fitting [16] defines both the well-founded and stable semantics of bilattice-valued logic programs in terms of an operator, $\Psi'_{\mathcal{Z}}$ where $\Psi'_{\mathcal{Z}}(\nu)$ is the least fixpoint, according to the $\leq_t$ ordering, of the operator $\lambda x.\Psi_{\mathcal{Z}}(x,\nu)$. This corresponds to keeping the second argument of $\Psi_{\mathcal{Z}}$ fixed, and varying the first argument of Ψ_P. As both well-founded and stable semantics of bilattice-valued logic programs can be captured in terms of the fixpoints $\Psi'_{\mathcal{Z}}$, which is defined in terms of the $\Psi_{\mathcal{Z}}$ operator, it follows immediately that the annotated logic program, $\mathcal{Z}_{\mathsf{ann}}$ can capture the semantics of the bilattice valued logic program $\mathcal{Z}$. In particular, the "consensus" operator, $\otimes$, defined by Fitting [17], which takes the glb (in the $\leq_k$-ordering) of the least and greatest fixpoints of $\Psi'_{\mathcal{Z}}$ can be described in terms of annotated logic.

From results of Kifer and Subrahmanian [30], it is already known (even in the negation-free case), that bilattices cannot subsume GAPs. There are a number of reasons for this: (1) the truth value associated with the head of a bilattice logic program can never be less than the value associated with its body according to the definition of Ψ_P, but this is permitted according to the annotated logic semantics. (2) A totally ordered set S consisting of more than two elements is not a bilattice (though it is possible to make $S \times S$ into a bilattice – but $S \times S$ is not the same as S !).

7.2 Merits of our Framework

For complex reasoning tasks requiring cooperation amongst multiple reasoning agents (and multiple knowledge bases), we need an integrated system such as the one proposed here. Our system has a number of advantages:

$\boxed{\text{Local vs. Global Queries.}}$ Let Q denote the amalgam of $(DB_1,\ldots,DB_{\mathsf{N}},S)$. A user who is interacting with Q should not have to worry about the "internals" of Q, i.e. (s)he should not have to know which specific local database (or set of local databases) a particular query is being processed by.

Our framework accomplishes this neatly. If the user wishes to ask about an atom A, the query is simply treated (perhaps by a user interface) as the amalgamated query

$$?-\ A : [\{\mathbf{s}\}, V],$$

i.e. "find the truth value V assigned to A by the supervisor."

On the other hand, a *knowledge-able* user who wishes to delve into the "internals" of the database should have the flexibility to ask queries of specific local knowledge bases. For instance, a user may have reason to believe (based on knowledge not represented in the conflict resolution axioms) that DB_2 is most knowledge-able about an atom A. In that case, the user is free to ask the amalgamated knowledge base Q the query:

$$?-\ A : [\{2\}, V],$$

i.e. "find the truth value V assigned to A by DB_2."

$\boxed{\text{Representing Prioritized Knowledge About Predicates.}}$ Given a particular predicate p, our framework is rich enough to represent the fact that database DB_i, for some $1 \leq i \leq n$, is most authoritative on predicate p (in comparison with the other local databases). For instance, when representing information about physical disorders in a medical knowledge base, information about the predicate *heart_attack* that is contained in a knowledge base designed by consulting a cardiologist may be more authoritative than other local knowledge bases (constructed by consulting, say, a gynecologist, an oncologist or a general practitioner). Preference to the views of a single knowledge base may be expressed using a clause such as:

$$p(\vec{X}):[\{\mathbf{s}\},V] \quad \leftarrow \quad p(\vec{X}):[\{i\},V].$$

$\boxed{\text{Representing Prioritized Knowledge About Objects.}}$ Conversely, it may turn out that a specific local database DB_i is better informed about the properties of a particular object (constant) than the other local databases. For instance, continuing the example of the previous paragraph, it may turn out that the views of the general practitioner on the case of a *specific* patient, John, may be more authoritative than that of the other specialists, because the GP in question has treated John for years and is fully acquainted with his case history. In general, to represent the fact that the local database DB_i is most authoritative about a specific constant, c, we add axioms of the form:

$$p_j(c):[\{\mathbf{s}\},V] \quad \leftarrow \quad p_j(c):[\{i\},V]$$

for each predicate p_j. A word of caution is in order here: some care is needed in specifying the interaction between rules specifying prioritized knowledge about predicates and prioritized knowledge about constants.

$\boxed{\text{Achieving Consensus.}}$ In our framework, it is very easy to express information such as: "The supervisor accepts only information that all the local databases agree on." This may be expressed as the clause:

$$A:[\{\mathbf{s}\},\sqcap\{V_1,\ldots,V_n\}] \quad \leftarrow \quad A:[\{1\},V_1]\,\&\ldots\&\,A:[\{n\},V_n].$$

Note that this clause requires annotation variables (the V_i's) as well as annotation functions (the $\sqcap$ function in the annotation in the head of this clause is an example of this).

$\boxed{\text{Representing Maximal Consistent Subsets.}}$ It is well known [12] that maximal consistent subsets of a theory may be viewed as the extensions of a corresponding normal default theory. In other words, given theory T, we can associate a default logic theory $\Delta(T)=(W,D)$ where $W=\emptyset$ and $D=\{\frac{:F}{F}\mid F\in T\}$. The maximal consistent subsets of T correspond precisely to the extensions of $\Delta(T)$ in the sense that T' is a maximal consistent subset of T iff $Cn(T')$, the set of first order consequences of T', is an extension of $\Delta(T)$. Intuitively, the default

$$\frac{a_1,\ldots,a_n,\neg a_{n+1},\ldots,\neg a_{n+m} \;:\; b_1,\ldots,b_s,\neg b_{s+1},\ldots,\neg b_{s+t}}{c}$$

where the a's, b's and c's are atoms can be read as "If $a_1,\ldots,a_n,\neg a_{n+1},\ldots,\neg a_{n+m}$ are true, and it is consistent to assume that $b_1,\ldots,b_s,\neg b_{s+1},\ldots,\neg b_{s+t}$ are true, then c is true." Using FOUR-valued annotated logic, this may be expressed intuitively as:

$$\begin{aligned} c:\mathbf{t} \quad \leftarrow \quad & a_1:\mathbf{t}\,\&\ldots\&\,a_n:\mathbf{t}\,\&\,a_{n+1}:\mathbf{f}\,\&\ldots\&\,a_{n+m}:\mathbf{f}\,\& \\ & \mathbf{not}(b_1:\mathbf{f})\,\&\ldots\&\,\mathbf{not}(b_s:\mathbf{f})\,\&\,\mathbf{not}(b_{s+1}:\mathbf{t})\,\&\ldots\&\,\mathbf{not}(b_{s+t}:\mathbf{t}). \end{aligned}$$

Intuitively, this statement[6] has the same reading as the default rule shown above. For example, if we take $T=\{a,b,\neg b\}$, T has two maximally consistent subsets, $\{\{a,b\},\{a,\neg b\}$. The default theory $\Delta(T)$ associated with T is: $\{\frac{:a}{a},\frac{:b}{b},\frac{:\neg b}{\neg b}\}$. The annotated clauses associated with $\Delta(T)$ are:

$$a:\mathbf{t}\leftarrow\mathbf{not}(a:\mathbf{f}); \qquad b:\mathbf{t}\leftarrow\mathbf{not}(b:\mathbf{f}); \qquad b:\mathbf{f}\leftarrow\mathbf{not}(b:\mathbf{t}).$$

[6]Note that if c is the literal $\neg d$, then we can replace $c:\mathbf{t}$ in the head of the above rule by $d:\mathbf{f}$.

This has two stable models: the first assigns $\mathbf{t}$ to a and $\mathbf{t}$ to b; the second assigns $\mathbf{t}$ to a and $\mathbf{f}$ to b. These correspond precisely to the (consequences of) the maximal consistent subsets of T. As defaults can be expressed in this way, and as maximal consistent subsets can be represented using a class of defaults called normal defaults, it seems clear that our framework can represent maximal consistent subsets as well.

$\boxed{\text{Complexity Issues}}$ As an amalgamated knowledge base is just an annotated logic program over an expanded set of atoms, procedures for processing queries to annotated logics (e.g. [23]) can be used to process queries to amalgamated logic programs. Given local knowledge bases $DB_1, \ldots, DB_{\mathsf{N}}$ and a supervisor S, the number of clauses (excluding the combination axiom scheme) in the amalgam is identical to the number of clauses in $(DB_1 \cup \ldots \cup DB_{\mathsf{N}} \cup S)$; as discussed earlier, it is possible that $O(2^{\mathsf{N}})$ new ground clauses will result from explicitly adding the instantiated combination axioms. There are two mitigating factors here – first, N is the *number of local databases being integrated* which is a relatively small number in practice. There are not too many applications that require more than 4 or 5 databases to be integrated. Second, in [1], we will show how to implement a sound and complete OLDT-resolution based query processing scheme that does not require an explicit instantiation of the combination axiom scheme. The basic idea behind [1] is the following: during query processing, at any given point in time, it is only necessary to store (in an OLDT-table), atoms of the form $A : [\{i\}, \mu]$ where $\{i\}$ is a *singleton* set. The only place where annotated atoms of the form $B : [D, \rho]$ (for D not a singleton) can occur is in the bodies of clauses. To evaluate these, all one needs to do is to compute $\sqcup_{d \in D} \rho_d$ where $A : [\{d\}, \rho_d]$ is stored in the OLDT-table. [1] gives algorithms and data structures for doing this.

7.3 Flaws in our Framework: Directions for Future Work

No system is without flaws – our framework is certainly no exception. In this section, we briefly overview some of the immediate flaws, and suggest some open problems.

$\boxed{\text{Indecisive Supervisors}}$ There is nothing in our current framework which assures us that supervisors will decisively assign a specific truth value to a given atom. The reason for this is that we wanted to develop a very general framework which would allow the designers of the supervisory database to incorporate into it, a broad range of supervisory information. For example, the user may easily specify various priorities and conflict resolution strategies (e.g. achieving consensus, maximal consistent sets, representing prioritized knowledge about objects, representing prioritized knowledge about predicates, etc.) as described in the previous section. In technical terms:

Definition 7.1 The *plausible answer set* of a ground atom A w.r.t. an amalgamated knowledge base Q is the set: $\{I(A)(\{\mathbf{s}\}) \mid I \text{ is an } \mathbf{A}\text{-stable model of } Q\}$. There is said to be a *unique* answer for A iff the plausible answer set of A is a singleton set. ∎

Given a ground atom A, it may turn out that the plausible answer set of A contains two or more elements. In such a case, what conclusion (if any) can we draw about A's truth value ? One conclusion, clearly, is that A's truth value w.r.t. the supervisor is at least $\sqcap_{j=1,\ldots,n} I_j(A)(\{\mathbf{s}\})$. Requiring that supervisory databases be strong is inadequate in ensuring unique answers because we may still end up with different truth values for A in different $\mathbf{A}$-stable models. There is one thing we can say, though:

Theorem 7.2 Suppose Q is the amalgam of $(DB_1, \ldots, DB_{\mathsf{N}}, S)$ where S is a strong supervisory database. If each DB_i, $1 \leq i \leq \mathsf{N}$, has a unique r-stable model, then Q has a unique $\mathbf{A}$-stable model. ∎

Open Problem 1 Further research is needed to find syntactic restrictions on local knowledge bases so that for all ground atoms A, there is a unique answer for A. ∎

$\boxed{\text{No Plausible Answers}}$ It is possible that the plausible answer set of a ground atom A w.r.t. Q, the amalgam of $(DB_1, \ldots, DB_{\mathsf{N}}, S)$, is empty. There is hope, however: we know, by Theorem 5.3, that if the plausible answer set of a ground atom A w.r.t. Q is empty and if S is strong, then one of the local databases must have no stable model. Hence, one way of ensuring the existence of plausible answers is to ensure that each local knowledge base have stable models (not so easy !), and to ensure that the supervisory database is strong (a relatively easy syntactic check).

Open Problem 2 How much can we weaken the restriction that S be strong, while still guaranteeing that the amalgam of $(DB_1, \ldots, DB_{\mathsf{N}}, S)$ has an **A**-stable model given that each local database has an r-stable model.

$\boxed{\text{Specifying Truth in the Supervisor.}}$ Note that in the form stated, if we choose to place no clauses at all in the supervisory database, then for all queries of the form

$$? - \ A : [\{\mathbf{s}\}, V],$$

the answer returned will be $\perp$ (where $\perp$ is the bottom element of the $\mathcal{T}$ lattice). This will be true even if all local databases agree on the truth value of A. Hence, if we wish to make a statement of the form: "If each local database assigns V to A, then so does the supervisor," we need to explicitly code this and place it in the supervisory database. One way of coding this is:

$$A : [\{\mathbf{s}\}, V] \leftarrow A : [\{1\}, V] \,\&\ldots\, \& A : [\{n\}, V].$$

Explicitly stating such rules may often be cumbersome. In an implementation, it is conceivable that such rules are implicitly added by the system.

8 Conclusions

In the context of integrating multiple databases, Silberschatz, Stonebraker and Ullman [40, p.119] have remarked that: "As the problems of fusion and semantic inconsistency are so severe, there is need for a class of information sources that stand between the user and the heterogeneous databases." They refer to this class of information sources as *mediators* because they mediate between multiple knowledge bases, and coordinate, collate, and render coherent (in terms of consistency), the data provided by these different databases. (In our paper, supervisory databases play the role of the mediators).

From a theoretical point of view, this raises the following questions:

1. In what language (L_M) should the mediator be described ?
2. What should be the (syntactic) relationship between the language in which the mediator is described, and the language in which the local knowledge bases are described ?
3. What are the semantic relationships between the local knowledge bases and the mediator ?

The above questions do not have a unique answer, but rather, admit a multiplicity of answers. In this paper we have given one set of answers to these questions. We have developed a formal declarative foundation (model theoretic and fixpoint) for amalgamated knowledge bases. This foundation has the following features:

- It allows a user to interact directly with the amalgam by asking questions of the supervisor. Thus, if a user asks the supervisor the atomic query A, this may be viewed as the annotated atom $A : [\{\mathbf{s}\}, V]$ where V is a $\mathcal{T}$-variable.
- On the other hand, if the user wishes to know the answer of a specific database, then (s)he can directly ask a query of that database (the query can be $A : [\{i\}, V]$, for instance).
- Our language enables us to study the relationship between the semantics of the local knowledge bases (in terms of stable models, and/or well-founded models) in a uniform way.
- Furthermore, the language used for expressing the local KBs and the supervisory database is an extension of the language of generalized annotated programs [30] (GAPs). GAPs are based on annotated logics which have been fairly well-studied from a foundational point of view, as well as from the point of view of reasoning about temporal data, object-oriented databases [31], and inheritance nets [26].

Acknowledgements. I have benefited from discussions with Sibel Adalı, Bobbie Cochrane, Sarit Kraus, Jim Lu and Raymond Ng. I am particularly grateful to Sibel Adalı, John Grant, Ashish Gupta, Sarit Kraus, Jim Lu and the anonymous referees for carefully reading through the manuscript and pointing out errors, typos, corrections in proofs and avenues for improvement.

References

[1] S. Adalı and V.S. Subrahmanian. (1993) *Integrating Multiple Knowledge Bases*, submitted for conference publication. Journal version in preparation.

[2] R. Anand and V.S. Subrahmanian. (1987) *FLOG: A Logic Programming System Based on a Six-Valued Logic*, AAAI/Xerox Second Intl. Symp. on Knowledge Engg., Madrid, Spain.

[3] C. Baral, S. Kraus and J. Minker. (1991) *Combining Multiple Knowledge Bases*, IEEE Trans. on Knowledge and Data Engineering, 3, 2, pps 200-220.

[4] C. Baral, S. Kraus, J. Minker and V.S. Subrahmanian. (1992) *Combining Knowledge Bases Consisting of First Order Theories*, Computational Intelligence, 8, 1, pps 45–71.

[5] C. Baral and V.S. Subrahmanian. (1991) *Dualities between Alternative Semantics for Logic Programming and Non-Monotonic Reasoning*, to appear in: Journal of Automated Reasoning. Preliminary version in: Proc. 1991 Intl. Workshop on Logic Programming and Non-Monotonic Reasoning (eds. A. Nerode, W. Marek and V.S. Subrahmanian), MIT Press.

[6] H. A. Blair and V.S. Subrahmanian. (1987) *Paraconsistent Logic Programming*, Theoretical Computer Science, 68, pp 35-54. Preliminary version in: LNCS 287, Dec. 1987, Springer.

[7] K. Bowen and R. Kowalski. (1982) *Amalgamating Language and Metalanguage in Logic Programming*, in "Logic Programming" (eds. K. L. Clark and S.-A. Tarnlund), pps 153–172, Academic Press.

[8] N.C.A. da Costa, V.S. Subrahmanian and C. Vago. (1991) *The Paraconsistent Logics* $\mathcal{P}\mathbf{T}$, Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik, Vol. 37, pps 139–148.

[9] D. Dubois, J. Lang and H. Prade. (1991) *Towards Possibilistic Logic Programming*, Proc. 1991 Intl. Conf. on Logic Programming, ed. K. Furukawa, pps 581–595, MIT Press.

[10] D. Dubois, J. Lang and H. Prade. (1991) *Timed Possibilistic Logic*, Fundamenta Informaticae, Vol. XV, No. 3-4, pps 211-234.

[11] D. Dubois, J. Lang and H. Prade. (1992) *Dealing with Multi-Source Information in Possibilistic Logic*, Proc. 10th European Conf. on Artificial Intelligence, Wiley.

[12] D. Etherington. (1988) *Reasoning with Incomplete Information*, Pitman Publishers.

[13] R. Fagin, J.D. Ullman, and M.Y. Vardi. (1983) *On the Semantics of Updates in Databases*, Proc. ACM SIGACT/SIGMOD Symposium on Principles of Database Systems, pps 352–365.

[14] R. Fagin, G. Kuper, J. Ullman, and M. Vardi. (1986) *Updating Logical Databases*, In *Advances in Computing Research*, volume 3, pages 1–18, 1986.

[15] M. C. Fitting. (1989) *Negation as Refutation*, Proc. 4th Symp. on Logic in Computer Science, pps 63–70.

[16] M. C. Fitting. (1991) *Bilattices and the Semantics of Logic Programming*, Journal of Logic Programming, 11, pps 91–116.

[17] M. C. Fitting. (1991) *Well-Founded Semantics, Generalized*, Proc. 1991 Intl. Logic Programming Symposium, pps 71–83, MIT Press.

[18] M. C. Fitting. (1992) *The Family of Stable Models*, to appear in: Journal of Logic Programming.

[19] M. Gelfond and V. Lifschitz. (1988) *The Stable Model Semantics for Logic Programming*, in: Proc. 5th International Conference and Symposium on Logic Programming, ed R. A. Kowalski and K. A. Bowen, pp 1070–1080.

[20] M. L. Ginsberg. (1988) *Multivalued Logics: A Uniform Approach to Reasoning in Artificial Intelligence*, Computational Intelligence, 4, pps 265–316.

[21] J. Grant, W. Litwin, N. Roussopoulos and T. Sellis. (1991) *An Algebra and Calculus for Relational Multidatabase Systems*, Proc. First International Workshop on Interoperability in Multidatabase Systems, IEEE Computer Society Press (1991) 118-124.

[22] L. J. Henschen and J.J. Lu. (1991) *The Paraconsistent Closed World Assumption*, to appear in: Theoretical Computer Science.

[23] L. J. Henschen and J.J. Lu. (1992) *The Completeness of Gp-resolution for Annotated Logics*, Information Processing Letters, 44, pps 135–140.

[24] Y. Ioannidis and T. Sellis. (1989) *Conflict Resolution of Rules Assigning Values to Virtual Attributes*, Proc. ACM SIGMOD Symp. on Management of Data.

[25] M. Kifer and T. Krishnaprasad. (1989) *An Evidence Based Framework for a Theory of Inheritance*, Proc. 1989 International Joint Conference on Artificial Intelligence, pps 1093–1098, Morgan-Kauffman. Full version to appear in: *Artificial Intelligence.*

[26] M. Kifer, T. Krishnaprasad and D.S. Warren. (1989) *On the Declarative Semantics of Inheritance Networks*, Proc. 1989 International Joint Conference on Artificial Intelligence, pps 1099-1103, Morgan-Kauffman.

[27] M. Kifer and A. Li. (1988) *On the Semantics of Rule-Based Expert Systems with Uncertainty*, 2-nd Intl. Conf. on Database Theory, Springer Verlag LNCS 326, (eds. M. Gyssens, J. Paredaens, D. Van Gucht), Bruges, Belgium, pp. 102–117.

[28] M. Kifer and E. Lozinskii. (1989) *RI: A Logic for Reasoning with Inconsistency*, 4-th Symposium on Logic in Computer Science, Asilomar, CA, pp. 253-262.

[29] M. Kifer and E. Lozinskii. (1992) *A Logic for Reasoning with Inconsistency*, Journal of Automated Reasoning, 9, 2, pps 179–215.

[30] M. Kifer and V.S. Subrahmanian. (1989) *Theory of Generalized Annotated Logic Programming and its Applications*, Journal of Logic Programming, 12, 4, pps 335–368, 1992. Preliminary version in: Proc. 1989 North American Conf. on Logic Programming, MIT Press.

[31] M. Kifer and J. Wu. (1989) *A Logic for Object Oriented Logic Programming*, Proc. 8th ACM SIGACT/SIGMOD/SIGART Symp. on Principles of Database Systems, Philadelphia, pps 379–393.

[32] J. W. Lloyd. (1987) *Foundations of Logic Programming*, Springer.

[33] J. J. Lu. (1992) *Automated Deduction in Paraconsistent Logics*, Ph.D. thesis, Northwestern University, May 1992.

[34] W. Marek, A. Nerode and J. Remmel. (1990) *A Theory of Non-Monotonic Rule Systems, I*, Annals of Mathematics and Artificial Intelligence, 1991. Preliminary version in: Proc. 1990 Conf. on Logic in Computer Science.

[35] W. Marek, A. Nerode and J. Remmel. (1992) *The Stable Models of a Predicate Logic Program* , in: Proc. 1992 Intl. Conf. on Logic Programming (ed. K.R. Apt), MIT Press.

[36] N.V. Murray and E. Rosenthal. (1991) *Signed Formulas: A Classical Approach to Multiple-Valued Logics*, TR-91-12, State University of New York at Albany.

[37] R. T. Ng and V.S. Subrahmanian. (1990) *Probabilistic Logic Programming*, to appear in Information and Computation. Preliminary version in: Proc. 1990 Intl. Symp. on Methodologies for Intelligent Systems, North Holland.

[38] R. T. Ng and V.S. Subrahmanian. (1991) *A Semantical Framework for Supporting Subjective and Conditional Probabilities in Deductive Databases,*, Journal of Automated Reasoning, 10, 2, pps 191–235, 1993. A preliminary version of this paper appeared in the Proc. of the 1991 Intl. Conf. on Logic Programming, MIT Press.

[39] C. Sakama. (1992) *Extended Well-Founded Semantics for Paraconsistent Logic Programs*, Proc. 1992 Intl. Conf. on Fifth Generation Computer Systems, Tokyo.

[40] A. Silberschatz, M. Stonebraker and J. D. Ullman. (1991) *Database Systems: Achievements and Opportunities*, Comm. of the ACM, 34, 10, pps 110–120.

[41] V.S. Subrahmanian. (1987) *On the Semantics of Quantitative Logic Programs*, Proc. 1987 IEEE Symp. on Logic Programming, pps 173–182.

[42] V.S. Subrahmanian. (1989) *A Simple Formulation of the Theory of Metalogic Programming*, in "Meta-Programming in Logic Programming" (eds. H. Abramson and M. Rogers), pps 65–101, MIT Press.

[43] V.S. Subrahmanian. (1992) *Paraconsistent Disjunctive Deductive Databases*, Theoretical Computer Science, 93, pps 115–141.

[44] A. van Gelder, K. Ross and J. Schlipf. (1988) *Unfounded Sets and Well-founded Semantics for General Logic Programs*, in Proc. 7th Symposium on Principles of Database Systems, pp 221-230.

[45] A. van Gelder. (1989) *The Alternating Fixpoint of Logic Programs with Negation*, Proc. 8th ACM Symp. on Principles of Database Systems, pps 1 – 10.

[46] W.K. Whang, S. B. Navathe and S. Chakravarthy. (1991) *Logic-Based Approach for Realizing a Federated Information System*, Proc. First International Workshop on Interoperability in Multidatabase Systems, IEEE Computer Society Press (1991) 92–100.

[47] R. Zicari, S. Ceri, and L. Tanca. (1991) *Interoperability between a Rule-Based Database Language and an Object-Oriented Language*, Proc. First International Workshop on Interoperability in Multidatabase Systems, IEEE Computer Society Press (1991) 125-135.

Appendix A: Proofs of Theorems and Lemmas

Proof of Theorem 3.1. Suppose $I^{\mathrm{loc}} \models^r DB_i$ and I^{aml} is in the locale of I^{loc} w.r.t. DB_i. We need to show that $I^{\mathrm{aml}} \models^{\mathbf{A}} AT(DB_i)$. Each clause in $AT(grd(DB_i)$ is of the form $AT(C)$:

$$\begin{aligned} A_0 : [\{i\}, \mu_0] \quad \leftarrow \quad & A_1 : [\{i\}, \mu_1] \,\&\, \ldots \,\&\, A_n : [\{i\}, \mu_n] \,\&\, \\ & \mathbf{not}(A_{n+1} : [\{i\}, \mu_{n+1}]) \,\&\, \ldots \,\&\, \mathbf{not}(A_{n+m} : [\{i\}, \mu_{n+m}]). \end{aligned}$$

and is obtained from a clause $C \in grd(P)$ of the form

$$\begin{aligned} A_0 : \mu_0 \quad \leftarrow \quad & A_1 : \mu_1 \,\&\, \ldots \,\&\, A_n : \mu_n \,\&\, \\ & \mathbf{not}(A_{n+1} : \mu_{n+1}) \,\&\, \ldots \,\&\, \mathbf{not}(A_{n+m} : \mu_{n+m}). \end{aligned}$$

Suppose $I^{\mathrm{aml}} \models^{\mathbf{A}} A_1 : [\{i\}, \mu_1] \,\&\, \ldots \,\&\, A_n : [\{i\}, \mu_n] \,\&\mathbf{not}(A_{n+1} : [\{i\}, \mu_{n+1}]) \,\&\, \ldots \,\&\, \mathbf{not}$ $(A_{n+m} : [\{i\}, \mu_{n+m}])$. In other words, for all $1 \le j \le n$, $\mu_i \le I^{\mathrm{aml}}(A_j)(i)$ and for all $1 \le k \le m$, $\mu_{n+k} \not\le I^{\mathrm{aml}}(A_{n+k})(i)$. As I^{aml} is in the locale of I^{loc} w.r.t. DB_i, for all ground atoms A, $I^{\mathrm{loc}}(A) = I^{\mathrm{aml}}(A)(i)$ and hence, it follows that for all $1 \le j \le n$, $\mu_i \le I^{\mathrm{loc}}(A_j)$ and for all $1 \le k \le m$, $\mu_{n+k} \not\le I^{\mathrm{loc}}(A_{n+k})$. In other words, $I^{\mathrm{loc}} \models^r A_1 : [\{i\}, \mu_1] \,\&\, \ldots \,\&\, A_n : [\{i\}, \mu_n] \,\&\, \mathbf{not}(A_{n+1} : [\{i\}, \mu_{n+1}]) \,\&\, \ldots \,\&\, \mathbf{not}(A_{n+m} : [\{i\}, \mu_{n+m}])$. As I^{loc} is a model of DB_i (and hence of $grd(C)$), $\mu_0 \le I^{\mathrm{loc}}(A_0)$. As I^{aml} is in the locale of I^{loc} w.r.t. DB_i, $I^{\mathrm{aml}}(A_0)(i) = I^{\mathrm{loc}}(A_0)$ and hence, $\mu_0 \le I^{\mathrm{aml}}(A_0)(i)$, i.e.

$I^{\mathrm{aml}} \models^{\mathbf{A}} AT(C)$. As $AT(C)$ is a ground instance of an arbitrary clause of $AT(DB_i)$, it follows that $I^{\mathrm{aml}} \models^{\mathbf{A}} AT(DB_i)$. ∎

Proof of Theorem 3.2. The proof is similar to the proof of Theorem 3.1. ∎

Proof of Corollary 3.1.

$\boxed{\text{Part 1 } (\rightarrow)}$ Suppose $DB_i \models^r A : \mu$ and suppose $AT(DB_i) \not\models^{\mathbf{A}} A : [\{i\}, \mu]$. Then there exists an $\mathbf{A}$-interpretation, I^{aml}, such that $I^{\mathrm{aml}} \models^{\mathbf{A}} AT(DB_i)$, but $I^{\mathrm{aml}} \not\models^{\mathbf{A}} A : [\{i\}, \mu]$. Consider the r-interpretation I^{loc} defined as follows: $I^{\mathrm{loc}}(A) = I^{\mathrm{aml}}(A)(i)$. Then:

1. I^{aml} is in the locale of I^{loc} and
2. every $\mathbf{A}$-interpretation I_1^{aml} in the locale of I^{loc} is such that $I_1^{\mathrm{aml}} \not\models^{\mathbf{A}} A : [\{i\}, \mu]$ because for all such I_1^{aml}, $I_1^{\mathrm{aml}}(A)(i) = I^{\mathrm{loc}}(A) = I^{\mathrm{aml}}(A)(i)$.

Consequently, $I^{\mathrm{loc}}(A) \not\models^r A : \mu$. But, by Theorem 3.2, as I^{aml} is an $\mathbf{A}$-model of $AT(DB_i)$ in the locale of I^{loc}, I^{loc} is an r-model of DB_i. This contradicts the assumption that $DB_i \models^r A : \mu$.

$\boxed{\text{Part 2 } (\leftarrow)}$ Similar to the proof of Part I. ∎

Proof of Theorem 4.1. (1) $\boxed{\text{Part 1 } (\rightarrow)}$ Suppose I is an r-model of P, and A is any ground atom and that $\mathbf{R}_P^e(I)(A) = \mu$. If $\mu = \perp$ (where $\perp$ is the bottom element of the $\mathcal{T}$-lattice), then $\mu \leq I(A)$ holds automatically.

Otherwise, let $CL(A) = \{C \mid C$ is a ground instance of a clause in P of the form $A : f(\mu_1, \ldots, \mu_n) \leftarrow B_1 : \mu_1 \,\&\ldots\&\, B_n : \mu_n \,\&\, \mathbf{not}(B_{n+1} : \mu_{n+1}) \,\&\ldots\&\, \mathbf{not}(B_{n+m} : \mu_{n+m})$ such that $I \models^r (B_1 : \mu_1 \,\&\ldots\&\, B_n : \mu_n)$ and for all $1 \leq j \leq m$, $I \not\models^r B_{n+j} : \mu_{n+j}\}$. Let $ANN(A) = \{f(\mu_1, \ldots, \mu_n) \mid A : f(\mu_1, \ldots, \mu_n)$ is the head of a clause in $CL(A)\}$. Clearly, $\mathbf{R}_P^e(I)(A) = \sqcup ANN(A)$. Furthermore, as I is an r-model of P, I must be an r-model of each clause in $CL(A)$. Hence, $f(\mu_1, \ldots, \mu_n) \leq I(A)$ for all clauses C in $CL(P)$ having $A : f(\mu_1, \ldots, \mu_n)$ as the head. Hence, $I(A)$ is an upper bound for $ANN(A)$, i.e. $\sqcup ANN(A) \leq I(A)$. But $\sqcup ANN(A) = \mathbf{R}_P^e(I)(A)$ and hence, $\mathbf{R}_P^e(I)(A) \leq I(A)$. As A is an arbitrary ground atom, it follows that $\mathbf{R}_P^e(I) \leq I$.

$\boxed{\text{Part 2 } (\leftarrow)}$ Conversely, suppose $\mathbf{R}_P^e(I) \leq I$. We need to show that $I \models^r P$. Let C be any ground instance of a clause in P of the form:

$$\begin{aligned} A_0 : \mu_0 \quad \leftarrow \quad & A_1 : \mu_1 \,\&\ldots\&\, A_n : \mu_n \,\&\\ & \mathbf{not}(A_{n+1} : \mu_{n+1}) \,\&\ldots\&\, \mathbf{not}(A_{n+m} : \mu_{n+m}). \end{aligned}$$

and such that $I \models^r A_1 : \mu_1 \,\&\ldots\&\, A_n : \mu_n \,\&\mathbf{not}(A_{n+1} : \mu_{n+1}) \,\&\ldots\&\, \mathbf{not}(A_{n+m} : \mu_{n+m})$. Then, $\mu_0 \leq \mathbf{R}_P^e(I)(A_0)$. As $\mathbf{R}_P^e(I) \leq I$, it follows that $\mu_0 \leq \mathbf{R}_P^e(I)(A_0) \leq I(A_0)$, i.e. I is an r-model of C. The result follows.

(2) Suppose I is a $\leq$-minimal model of P. As I is an r-model of P, by part (1) above, it follows that $\mathbf{R}_P^e(I) \leq I$. We need to show that $I \leq \mathbf{R}_P^e(I)$.

We claim that $\mathbf{R}_P^e(I)$ is a model of P. To see this, suppose C is a ground instance of a clause in P of the form shown in part (2) above such that $\mathbf{R}_P^e(I) \models^r A_1 : \mu_1 \,\&\ldots\&\, A_n : \mu_n \,\&\mathbf{not}(A_{n+1} : \mu_{n+1}) \,\&\ldots\&\, \mathbf{not}(A_{n+m} : \mu_{n+m})$. As $\mathbf{R}_P^e(I) \leq I$, it follows that $I \models^r A_1 : \mu_1 \,\&\ldots\&\, A_n : \mu_n \,\&\mathbf{not}(A_{n+1} : \mu_{n+1}) \,\&\ldots\&\, \mathbf{not}(A_{n+m} : \mu_{n+m})$, and hence, $\mu_0 \leq \mathbf{R}_P^e(I)(A_0)$. Therefore, $\mathbf{R}_P^e(I)$ is a model of C. As C was an arbitrary ground instance of an arbitrary clause in P, it follows that $\mathbf{R}^e_P(I)$ is a model of P. But $\mathbf{R}^e_P(I) \le I$ and we are supposing that I is a $\le$-minimal model of P. This is possible only if $\mathbf{R}^e_P(I) = I$. ∎

Proof of Theorem 4.2. Suppose I^{aml} is a fixpoint of $\mathbf{A}_Q$ and A is any ground atom. Let $I^{\text{loc}}_j(A) = \mu$. We need to show that $\mu \le \mathbf{R}_{DB_j}(I^{\text{loc}}_j)(A)$. There are two cases depending upon whether $\mu = \perp$.

$\boxed{\text{Case 1 } (\mu \ne \perp)}$ As I^{loc}_j is a projection of I^{aml}, it follows that $I^{\text{aml}}(A) = \mu = \mathbf{A}_Q(I^{\text{aml}})(A)(j)$. (The last equality follows as I^{aml} is a fixpoint of $\mathbf{A}_Q$. Furthermore, any clause in the amalgam of $(DB_1, \ldots, DB_n, S)$ that has a head of the form $A : [D, \rho]$ with $D \subseteq \{j\}$ must have $D = \{j\}$, and all such clauses must have been obtained either from database DB_j or by instantiating the combination axiom scheme. Hence, by this observation and by using the definition of $\mathbf{A}_Q$, we obtain:

μ

$= \sqcup\{\rho | A : [D, \rho] \leftarrow B_1 : [D_1, \mu_1] \,\&\ldots\&\, B_n : [D_n, \mu_n] \,\&\, \mathbf{not}(B_{n+1} : [D_{n+1}, \mu_{n+1}]) \,\&\ldots\&\, \mathbf{not}(B_{n+m} : [D_{n+m}, \mu_{n+m}])$ is a ground instance of a clause in Q and $D' = \{j\}$ and for all $1 \le i \le n$, $\mu_i \le I^{\text{aml}}(B_i)(D_i)$ and for all $(n+1) \le s \le (n+m)$, $\mu_s \not\le I^{\text{aml}}(B_s)(D_s)\}$

$= \sqcup(X_1 \cup X_2)$ where

$X_1 = \sqcup\{\rho | A : \rho \leftarrow B_1 : \mu_1 \,\&\ldots\&\, B_n : \mu_n \,\&\, \mathbf{not}(B_{n+1} : \mu_{n+1}) \,\&\ldots\&\, \mathbf{not}(B_{n+m} : \mu_{n+m}])$ is a ground instance of a clause in DB_j and for all $1 \le i \le n$ $\mu_i \le I^{\text{loc}}_j(B_i)$ and for all $(n+1) \le s \le (n+m)$, $\mu_s \not\le I^{\text{loc}}_j(B_s)\}$
The fact that I^{loc}_j is a projection of I^{aml} is used in the last equality above.
and
$X_2 = \{\rho \mid A : [\{j\}, \rho] \leftarrow$ is an instance of the combination axiom scheme $\}$

$= \mathbf{R}^e_{DB_j}(I^{\text{loc}}_j)(A)$.

Note that $X_2 = \{\perp\}$ and that $\mathbf{R}^e_{DB_j}(I^{\text{loc}}_j)(A) = \sqcup X_1$. In particular, note that the body of the axiom scheme in X_2 must contain no atoms because $\{j\}$ has no non-empty, strict subsets.

$\boxed{\text{Case 2 } (\mu = \perp)}$ In this case, by a similar argument as in Case 1, there is no clause C in $grd(Q)$ with head $A : [\{j\}, \rho]$ for $\rho \ne \perp$ such that the body of C is satisfied by I^{aml}. It follows, by an argument analogous to Case 1 that there is no clause C_0 in $grd(DB_j)$ such that the body of C_0 is satisfied by I^{loc}_j. Hence, $\mathbf{R}^e_{DB_j}(I^{\text{loc}}_j)(A) = \perp$. ∎

Proof of Theorem 4.3. Suppose I^{loc}_j is a fixpoint of $\mathbf{R}_{DB_j}$. We can construct an $\mathbf{A}$-interpretation I^{aml} as follows:

$$I^{\text{aml}}(A)(i) = \begin{cases} I^{\text{loc}}_j(A) & \text{if } i = j \\ \mathsf{gfp}(\mathbf{R}^e_{DB_i})(A) & \text{if } i \ne j \text{ and } i \ne \mathbf{s} \\ ? & \text{otherwise} \end{cases}$$

The only thing that still needs to be specified to complete the construction of I^{aml} is the value of $I^{\text{aml}}(A)(\mathbf{s})$. It is clear that I^{aml} is in the locale of I^{loc}_j.

Proceed as follows: For every clause in $grd(S)$ of the form:

$$A_0 : [\{\mathbf{s}\}, \mu] \;\leftarrow\; A_1 : [D_1, \mu_1] \,\&\, \ldots \,\&\, A_n : [D_n, \mu_n] \,\&\, \mathbf{not}(A_{n+1} : [D_{n+1}, \mu_{n+1}]) \,\&\, \ldots \,\&\, \mathbf{not}(A_{n+m} : [D_{n+m}, \mu_{n+m}])$$

check the following: if there is either an $(n+1) \leq r \leq (n+m)$ such that $\mu_r \leq \sqcup_{d \in D_r} I^{\mathrm{aml}}(A_r)(d)$, or if there is an $1 \leq s \leq n$ such that $\mu_s \not\leq \sqcup_{d \in D_s} I^{\mathrm{aml}}(A_s)(d)$, then delete the clause from $grd(S)$; otherwise, delete the entire body of the clause. As S is a strong supervisory knowledge base, the D_r's and D_s's are all subsets of $\{1, \ldots, n\}$, and hence, for all clauses in $grd(S)$, either the clause gets deleted by this process, or its body gets deleted. This leaves an amalgamated knowledge base of facts in $grd(S)$, i.e. of clauses of the form, $A_0 : [\{\mathbf{s}\}, \mu] \leftarrow$. For each ground atom A_0, set $I^{\mathrm{aml}}(A_0)(\mathbf{s}) = \sqcup\{\mu \mid A_0 : [\{\mathbf{s}\}, \mu] \leftarrow$ is a clause in $grd(S)$ after the above deletions have been made $\}$.

This completes the construction of $I^{\mathrm{aml}}(A_0)$. Using the fact that the annotation $\mathbf{s}$ occurs nowhere in the bodies of ground instances of clauses in S, it is now straightforward to establish that I^{aml} is a fixpoint of $\mathbf{A}_Q$. ∎

Proof of Corollary 4.1. Immediate. ∎

Proof of Theorem 5.1.

(1) Suppose $I_1 \leq I_2$ and $A_0 : \mu_0 \leftarrow A_1 : \mu_1 \,\&\, \ldots \,\&\, A_n : \mu_n$ is in $\mathbf{G}(P, I_2)$. Then there was a clause in P having a ground instance of the form

$$A_0 : \mu_0 \;\leftarrow\; A_1 : \mu_1 \,\&\, \ldots \,\&\, A_n : \mu_n \,\&\, \mathbf{not}(A_{n+1} : \mu_{n+1}) \,\&\, \ldots \,\&\, \mathbf{not}(A_{n+m} : \mu_{n+m}).$$

such that for all $1 \leq j \leq m$, $\mu_{n+j} \not\leq I_2(A)$. As $I_1 \leq I_2$, it follows that for all $1 \leq j \leq m$, $\mu_{n+j} \not\leq I_1(A)$. Hence, by definition of $\mathbf{G}$, the clause $A_0 : \mu_0 \leftarrow A_1 : \mu_1 \,\&\, \ldots \,\&\, A_n : \mu_n$ is in $\mathbf{G}(P, I_1)$. This completes the proof of (1).

(2) As $\mathbf{G}(P, I_2) \subseteq \mathbf{G}(P, I_1)$, and as both $\mathbf{G}(P, I_2)$ and $\mathbf{G}(P, I_1)$ are negation-free, it follows immediately that for all ground atoms A, $\mathcal{G}_P(I_2)(A) \leq \mathcal{G}_P(I_1)(A)$, i.e. $\mathcal{G}_P(I_2) \leq \mathcal{G}_P(I_1)$. ∎

Proof of Theorem 5.2. We need to show two things: first, that I is an r-model of P, and second, that it is a $\leq$-minimal model.

$\boxed{I \text{ is an r-model}}$ Suppose $A_0 : \mu_0 \leftarrow A_1 : \mu_1 \,\&\, \ldots \,\&\, A_n : \mu_n \,\&\mathbf{not}(A_{n+1} : \mu_{n+1}) \,\&\, \ldots \,\&\, \mathbf{not}(A_{n+m} : \mu_{n+m})$. is a ground instance of a clause in P such that for all $1 \leq i \leq n$, $\mu_i \leq I(A_i)$ and for all $1 \leq j \leq m$, $\mu_{n+j} \not\leq I_2(A)$. Then $A_0 : \mu_0 \leftarrow A_1 : \mu_1 \,\&\, \ldots \,\&\, A_n : \mu_n$ is in $\mathbf{G}(P, I)$. But $I = \mathsf{lfp}(\mathbf{R}_{\mathbf{G}(P,I)})$. As $\mathbf{G}(P, I)$ is negation-free, it follows from results of Kifer and Subrahmanian [30]) (pre-fixpoints of $\mathbf{R}_P$ are models) that I is an r-model of P.

$\boxed{\text{r-Minimality of } I}$ Suppose there is an r-model J of P such that $J \leq I$. Then we show that $I \leq J$. As I is an r-stable model of P, there is an ordinal ζ such that $I = \mathbf{R}_{\mathbf{G}(P,I)} \uparrow \zeta$. We first show, by transfinite induction on ordinals γ, that if $J(A) \leq \mathbf{R}_{\mathbf{G}(P,I)} \uparrow \gamma(A)$, then $\mathbf{R}_{\mathbf{G}(P,I)} \uparrow \gamma(A) \leq J(A)$ for all ground atoms A.

$\boxed{\text{Base Case } (\gamma = 1)}$ In this case, there are (either finitely many or infinitely many) clauses $C_1, \ldots, C_k, \ldots$, $k \geq 1$, in $\mathbf{G}(P, I)$ with each C_i having the form[7]: $A : \mu_i \leftarrow$

[7]Strictly speaking, the body could contain literals of the form $B : \perp$, but as these are tautologies that are true in all r-interpretations, we may ignore them.

such that $J(A) \leq \sqcup\{\mu_1, \ldots, \mu_k, \ldots\} \leq I(A)$. As $J \leq I$, by Theorem 5.1, we know that $\mathbf{G}(P, I) \subseteq \mathbf{G}(P, J)$, i.e. the clauses $C_1, \ldots, C_k, \ldots$ are all in $\mathbf{G}(P, J)$. For each $r \geq 1$, C_r is obtained from a clause C_r' in $grd(P)$ of the form $A : \mu_r \leftarrow \mathbf{not}(B_1 : \rho_1) \,\&\ldots\&\, \mathbf{not}(B_{w_r} : \rho_{w_r})$ such that for all $1 \leq s \leq w_r$, $\rho_{w_s} \not\leq J(B_s)$. In other words, J satisfies the body of C_r'. As C_r' is in $grd(P)$, and as J is a model of P, $\mu_r \leq J(A)$. Therefore, $J(A)$ is an upper bound for $\{\mu_1, \ldots, \mu_k, \ldots\}$. Thus, $\mathbf{R}_{\mathbf{G}(P,I)} \uparrow 1(A) = \sqcup\{\mu_1, \ldots, \mu_k, \ldots\} \leq J(A)$.

$\boxed{\text{Inductive Case } (\gamma > 1)}$ In this case, there are are two subcases to consider.

Subcase 1 ($\gamma = (\lambda + 1)$ is a Successor Ordinal) In this case, there are (either finitely many or infinitely many) clauses $C_1, \ldots, C_k, \ldots$, $k \geq 1$, in $\mathbf{G}(P, I)$ with each C_r having the form: $A : \mu_r \leftarrow D_1 : \delta_1 \,\&\ldots\&\, D_{v_r} : \delta_{v_r}$ such that $\delta_i \leq \mathbf{R}_{\mathbf{G}(P,I)} \uparrow \lambda(D_i)$ for all $1 \leq i \leq v_r$. By the induction hypothesis, $\delta_i \leq J(D_i)$ for all $1 \leq i \leq v_r$. As $J \leq I$, by Theorem 5.1, we know that $\mathbf{G}(P, I) \subseteq \mathbf{G}(P, J)$, i.e. the clauses $C_1, \ldots, C_k, \ldots$ are all in $\mathbf{G}(P, J)$. For each $r \geq 1$, C_r is obtained from a clause C_r' in $grd(P)$ of the form

$$\begin{aligned} A : \mu_r \quad \leftarrow \quad & D_1 : \delta_1 \,\&\ldots\&\, D_{v_r} : \delta_{v_r} \,\& \\ & \mathbf{not}(B_1 : \rho_1) \,\&\ldots\&\, \mathbf{not}(B_{w_r} : \rho_{w_r}). \end{aligned}$$

such that for all $1 \leq s \leq w_r$, $\rho_{w_s} \not\leq J(B_s)$. Furthermore, recall that $\delta_i \leq J(D_i)$ for all $1 \leq i \leq v_r$. In other words, J satisfies the body of C_r'. As C_r' is in $grd(P)$, and as J is a model of P, $\mu_r \leq J(A)$. Therefore, $J(A)$ is an upper bound for $\{\mu_1, \ldots, \mu_k, \ldots\}$. Thus, $\mathbf{R}_{\mathbf{G}(P,I)} \uparrow \gamma = \sqcup\{\mu_1, \ldots, \mu_k, \ldots\} \leq J(A)$.

Subcase 2: γ is a Limit Ordinal In this case, by the induction hypothesis, $\mathbf{R}_{\mathbf{G}(P,I)} \uparrow \beta(A) \leq J(A)$ for all $\beta < \lambda$. Hence, $\mathbf{R}_{\mathbf{G}(P,I)} \uparrow \lambda(A) = \sqcup_{\beta<\lambda} \mathbf{R}_{\mathbf{G}(P,I)} \uparrow \beta(A) \leq J(A)$. This completes the proof that for all ordinals γ, and all ground atoms A, if $J(A) \leq \mathbf{R}_{\mathbf{G}(P,I)} \uparrow \gamma(A)$, then $\mathbf{R}_{\mathbf{G}(P,I)} \uparrow \gamma(A) \leq J(A)$. The result follows by setting $\gamma = \zeta$. ∎

Proof of Theorem 5.3. Suppose $I_1, \ldots, I_n$ are r-stable models of $DB_1, \ldots, DB_n$, respectively. We can construct an $\mathbf{A}$-stable model, I, of Q as follows:

$$I(A)(j) = \begin{cases} I_j(A) & \text{if } j \in \{1, \ldots, n\} \\ ? & \text{if } j = \mathbf{s} \end{cases}$$

To determine $I(A)(\mathbf{s})$, we proceed as follows: let $(C_i)_{i \in \mathcal{A}}$ be all the ground instances of clauses in Q of the form:

$$\begin{aligned} A : [\{\mathbf{s}\}, \mu_i] \quad \leftarrow \quad & B_1 : [D_1 : \rho_1] \,\&\ldots\&\, B_n : [D_n, \rho_n] \,\& \\ & \mathbf{not}(B_{n+1} : [D_{n+1}, \rho_{n+1}]) \,\&\ldots\&\, \mathbf{not}(B_{n+m} : [D_{n+m}, \rho_{n+m}]) \end{aligned}$$

such that for all $1 \leq j \leq n$, $\rho_j \leq I(B_j)(D_j) = \sqcup_{d \in D_j} I(B_j)(d)$ and for all $(n+1) \leq k \leq (n+m)$, $\rho_k \not\leq I(B_k)(D_k)$. Even though I is not yet fully defined, both the above conditions are fully evaluable w.r.t. the partially specified I because S is a strong supervisory database (which implies that there are no occurrences of $\mathbf{s}$ in any of the D_r's, for $1 \leq r \leq (n+m)$). Set $I(A)(\{\mathbf{s}\}) = \sqcup\{\mu_i \mid i \in \mathcal{A}\}$. It is easy to verify that I is a stable model of Q. ∎

Proof of Theorem 5.4. Follows imediately from the construction in the proof of Theorem 5.3. It is easy to verify that I is in the locale of I_j^{loc} for all $1 \leq j \leq n$. ∎

Proof of Theorem 5.5. To show that I_j^{loc} is an r-stable model of DB_j, we need to show that $I_j^{\text{loc}} = \mathsf{lfp}(\mathbf{R}_{\mathbf{G}(DB_j, I_j^{\text{loc}})})$.

$\boxed{I_j^{\text{loc}} \leq \mathsf{lfp}(\mathbf{R}_{\mathbf{G}(DB_j, I_j^{\text{loc}})})}$ Suppose $I_j^{\text{loc}}(A) = \mu$. If $\mu = \bot$, then $I_j^{\text{loc}}(A) \leq \mathsf{lfp}(\mathbf{R}_{\mathbf{G}(DB_j, I_j^{\text{loc}})})(A)$ holds immediately.

If $\mu \neq \bot$, then, as I^{aml} is an $\mathbf{A}$-stable model of Q, it follows that $I^{\text{aml}}(A)(j) = \mu = \mathcal{G}_Q(I^{\text{aml}})(A)(j)$, i.e. $\mathcal{G}_Q(I^{\text{aml}}) \models A : [\{j\}, \mu]$. Hence, $\mathsf{lfp}(\mathbf{A}_{\mathbf{G}(Q, I^{\text{aml}})}) \models A : [\{j\}, \mu]$. We proceed to show, by induction on γ, that if $\mathbf{A}_{\mathbf{G}(Q, I^{\text{aml}})} \uparrow \gamma \models A : [\{j\}, \mu]$. then $\mathsf{lfp}(\mathbf{A}_{\mathbf{G}(Q, I^{\text{aml}})}) \models A : [\{j\}, \mu]$.

<u>Base Case ($\gamma = 1$)</u> $\mathbf{A}_{\mathbf{G}(Q, I^{\text{aml}})} \uparrow 1 \models A : [\{j\}, \mu]$ means that there are clauses $(A : [\{j\}, \mu_r] \leftarrow)_{r \in \mathcal{A}}$ in $\mathbf{G}(Q, I^{\text{aml}})$ such that $\mu \leq \sqcup_{r \in \mathcal{A}} \mu_r$. But then, by the construction of Q, there are clauses $A : [\{j\}, \mu_r] \leftarrow \mathbf{not}(B_1^r : [\{j\}, \rho_1^r]) \ldots \& \mathbf{not}(B_{v_r}^r : [\{j\}, \rho_{v_r}^r])$ in Q such that for all $1 \leq i \leq v_r$, $\rho_i \not\leq I^{\text{aml}}(B_i)(\{j\})$. But then, by construction of Q, $A : \mu_r \leftarrow \mathbf{not}(B_1^r : \rho_1^r) \ldots \& \mathbf{not}(B_{v_r}^r : \rho_{v_r}^r)]$ is a clause in DB_j, and furthermore, as I_j^{loc} is the projection of I^{aml} on DB_j, $I_j^{\text{loc}}(B_i) = I^{\text{aml}}(B_i)(j)$ for all $1 \leq i \leq v_r$. Hence, $A : \mu_r \leftarrow$ is in $\mathbf{G}(P, I_j^{\text{loc}})$ for all $r \in \mathcal{A}$. Therefore, $\mathbf{R}_{\mathbf{G}(DB_j, I_j^{\text{loc}})} \uparrow 1 \models A : \mu_r$ for all $r \in \mathcal{A}$, and hence, $\mathbf{R}_{\mathbf{G}(DB_j, I_j^{\text{loc}})} \uparrow 1 \models A : \mu$ as $\mu = \sqcup_{r \in \mathcal{A}} \mu_r$.

$\boxed{\text{Inductive Case } (\gamma > 1)}$ There are two possibilities to consider.

<u>Subcase 1</u> (γ is a Successor Ordinal, $\gamma = \delta + 1$) In this case, $\gamma = \sqcup_{r \in \mathcal{A}} \mu_r$. Then, for all $r \in \mathcal{A}$, $A : [\{j\}, \mu_r] \leftarrow C_1 : [\{j\}, \beta_1] \& \ldots \& C_m : [\{j\}, \beta_m]$ is in $\mathbf{G}(Q, I^{\text{aml}})$ and $\mathbf{A}_{\mathbf{G}(Q, I^{\text{aml}})} \uparrow \delta \models C_i : [\{j\}, \mu_i]$ for all $1 \leq i \leq m$. By the induction hypothesis, $\mathsf{lfp}(\mathbf{R}_{\mathbf{G}(DB_j, I_j^{\text{loc}})}) \models C_i : [\{j\}, \mu_i]$ for all $1 \leq i \leq m$. Furthermore, there is a clause in $\mathbf{G}(Q, I^{\text{aml}})$ of the form:

$$\begin{array}{rcl} A : [\{j\}, \mu_r] & \leftarrow & C_1 : [\{j\}, \beta_1] \& \ldots \& C_m : [\{j\}, \beta_m] \\ & & \mathbf{not}(B_1 : [\{j\}, \rho_1]) \& \ldots \& \mathbf{not}(B_s 1 : [\{j\}, \rho_s]) \end{array}$$

Using a similar argument as in the base case, $A : \mu_r \leftarrow C_1 : \beta_1 \& \ldots \& C_m : \beta_m$ is a clause in $\mathbf{G}(DB_j, I_j^{\text{loc}})$. As $\mathsf{lfp}(\mathbf{R}_{\mathbf{G}(DB_j, I_j^{\text{loc}})})$ satisfies the body of this clause, it follows that $\mu_r \leq \mathsf{lfp}(\mathbf{R}_{\mathbf{G}(DB_j, I_j^{\text{loc}})})(A)$. As this holds for all $r \in \mathcal{A}$, it follows that $\mu = \sqcup_{r \in \mathcal{A}} \mu_r \leq \mathsf{lfp}(\mathbf{R}_{\mathbf{G}(DB_j, I_j^{\text{loc}})})(A)$.

<u>Subcase 2 (γ a Limit Ordinal)</u> Then $\mu = \sqcup_{\beta < \gamma} \mathbf{A}_{\mathbf{G}(Q, I^{\text{aml}})} \uparrow \beta(A)$. By the induction hypothesis, for all $\beta < \gamma$, $\mathbf{A}_{\mathbf{G}(Q, I^{\text{aml}})} \uparrow \beta(A) \leq \mathsf{lfp}(\mathbf{R}_{\mathbf{G}(DB_j, I_j^{\text{loc}})})(A)$. Therefore,

$$\mu = \sqcup_{\beta < \gamma} \mathbf{A}_{\mathbf{G}(Q, I^{\text{aml}})} \uparrow \beta(A) \leq \mathsf{lfp}(\mathbf{R}_{\mathbf{G}(DB_j, I_j^{\text{loc}})})(A).$$

This completes the proof that $I_j^{\text{loc}} \leq \mathsf{lfp}(\mathbf{R}_{\mathbf{G}(DB_j, I_j^{\text{loc}})})$.

$\boxed{\mathsf{lfp}(\mathbf{R}_{\mathbf{G}(DB_j, I_j^{\text{loc}})}) \leq I_j^{\text{loc}}}$ $\mathsf{lfp}(\mathbf{R}_{\mathbf{G}(DB_j, I_j^{\text{loc}})})(A) = \mathbf{R}_{\mathbf{G}(DB_j, I_j^{\text{loc}})} \uparrow \alpha(A)$ for some ordinal α. The proof is by induction on α and mirrors the above proof. ∎

Proof of Proposition 6.1. Immediate. ∎

Proof of Theorem 6.1. (1) $P \overset{\text{wfs}}{\models}_r A : \mu$
$\Longrightarrow$ $\mu \leq \mathsf{lfp}(\mathcal{G}_P^2)(A)$
$\Longrightarrow$ $\mu \leq I_1(A)$ for all fixpoints I_1 of $\mathcal{G}_P^2$
$\Longrightarrow$ $\mu \leq I(A)$ as any r-stable model is a fixpoint of $\mathcal{G}_P$, and hence of $\mathcal{G}_P^2$.

(2) $P \overset{\text{wfs}}{\models_r} \mathbf{not}(A:\mu)$
$\Longrightarrow \mu \not\leq \mathsf{gfp}(\mathcal{G}_P^2)(A)$
$\Longrightarrow \mu \not\leq I_2(A)$ for all fixpoints I_2 of $\mathcal{G}_P^2$
$\Longrightarrow \mu \not\leq I(A)$ as any r-stable model is a fixpoint of $\mathcal{G}_P$, and hence of $\mathcal{G}_P^2$. ∎

Proof of Theorem 6.2. We show that $DB_i \overset{\text{wfs}}{\models_r} A:\mu$ implies that $Q \overset{\text{wfs}}{\models_{\mathbf{A}}} A:[\{i\},\mu]$ and $DB_i \overset{\text{wfs}}{\models_r} \mathbf{not}(A:\mu)$ implies that $Q \overset{\text{wfs}}{\models_{\mathbf{A}}} \mathbf{not}(A:[\{i\},\mu])$. The reverse direction follows along analogous lines.

Suppose $DB_i \overset{\text{wfs}}{\models_r} A:\mu$. Then there is an ordinal α such that $\mu \leq \mathcal{G}_{DB_i}^2 \uparrow \alpha(A)$. Likewise, $DB_i \overset{\text{wfs}}{\models_r} \mathbf{not}(B:\delta)$ means that there is an ordinal β such that $\delta \not\leq \mathcal{G}_{DB_i}(\mathcal{G}_{DB_i}^2 \uparrow \beta)(B)$. We show, by induction on ordinals, γ, that the conjunction of (1) and (2) below holds:

1. $\mu \leq \mathcal{G}_{DB_i}^2 \uparrow \gamma(A)$ implies that $Q \overset{\text{wfs}}{\models_{\mathbf{A}}} A:[\{i\},\mu]$, and
2. $\delta \not\leq \mathcal{G}_{DB_i}(\mathcal{G}_{DB_i}^2 \uparrow \gamma)(B)$ implies that $Q \overset{\text{wfs}}{\models_{\mathbf{A}}} \mathbf{not}(B:[\{i\},\delta]$.

$\boxed{\text{Base Case } (\gamma = 0)}$ **(1)** In this case, $\mu = \perp$. Hence, $Q \overset{\text{wfs}}{\models_{\mathbf{A}}} A:[\{i\},\perp]$ is immediately true as all atoms of the form $A:[\{i\},\perp]$ are true in all $\mathbf{A}$-models.

(2) In this case, $\delta \not\leq (\mathcal{G}_{DB_i}(\mathcal{G}_{DB_i}^2 \uparrow 0)(B)$, i.e. $\mathbf{G}(DB_i, \mathcal{G}_{DB_i}^2 \uparrow 0) \not\models B:\delta$. Now $\mathcal{G}_{DB_i}^2 \uparrow 0$ assigns $\perp$ to all ground atoms in the language of DB_i. Hence, $\mathbf{G}(DB_i, \mathcal{G}_{DB_i}^2 \uparrow 0)$ is the set of all clauses in DB_i with negative literals in them (if any) deleted, i.e. $\mathbf{G}(DB_i, \mathcal{G}_{DB_i}^2 \uparrow 0)$ $= \{A_0:\mu_0 \leftarrow A_1:\mu_1 \,\&\ldots\&\, A_n:\mu_n \mid (\exists m \geq 0)$ such that $A_0:\mu_0 \leftarrow A_1:\mu_1 \,\&\ldots\&\, A_n:\mu_n \,\&\, \mathbf{not}(A_{n+1}:\mu_{n+1}) \,\&\ldots\&\, \mathbf{not}(A_{n+m}:\mu_{n+m})$ is a clause in $grd(DB_i)\}$.

Clearly, $X = \{A_0:[\{i\},\mu_0] \leftarrow A_1::[\{i\},\mu_1], \&\ldots\&\, A_n:[\{i\},\mu_n] \mid A_0:\mu_0 \leftarrow A_1:\mu_1 \,\&\ldots\&\, A_n:\mu_n$ is in $\mathbf{G}(DB_i, \mathcal{G}_{DB_i}^2 \uparrow 0)\}$ is a subset of $\mathbf{G}(Q, I^{\text{aml}})$ where I^{aml} is the $\mathbf{A}$-interpretation that assigns $\perp$ to all $A \in B_L$ and all $1 \leq j \leq n$, i.e. $I^{\text{aml}}(A)(j) = \perp$. Furthermore, the only clauses in Q with heads of the form $A:[\{i\},--]$ are from the set X. Consequently, as $\mathbf{G}(DB_i, \mathcal{G}_{DB_i}^2 \uparrow 0) \not\models B:\delta$, it follows that $\mathbf{G}(Q, I^{\text{aml}}) \not\models B:\delta$ where I^{aml} is as defined earlier. Hence, $Q \overset{\text{wfs}}{\models_{\mathbf{A}}} \mathbf{not}(B:\delta)$. This completes the base case.

$\boxed{\text{Inductive Case}}$ There are two cases, depending upon whether γ is a successor ordinal or a limit ordinal.

$\underline{\gamma \text{ a Successor Ordinal}}$ In this case, $\gamma = (\kappa + 1)$ for some ordinal κ.

(1) Suppose $\mu \leq \mathcal{G}_{DB_i}^2 \uparrow \gamma(A)$. Then $\mathbf{G}(DB_i, \mathcal{G}_{DB_i}(\mathcal{G}_{DB_i}^2 \uparrow \kappa)) \models_r A:\mu$. Hence, there are clauses in $\mathbf{G}(DB_i, \mathcal{G}_{DB_i}(\mathcal{G}_{DB_i}^2 \uparrow \kappa))$ having ground instances, $(C_i)_{i\in\mathcal{A}}$ of the form: $A:\mu_i \leftarrow B_1^i:\rho_1^i \,\&\ldots\&\, B_{k_i}^i:\rho_{k_i}^i$ such that: (1) $\mu \leq \sqcup_{i\in\mathcal{A}}\mu_i$ and (2) for all $i \in \mathcal{A}$ and all $1 \leq j \leq k_i$, $\rho_j^i \leq \mathcal{G}_{DB_i}^2 \uparrow \kappa(B_j^i)$. By the induction hypothesis, $Q \overset{\text{wfs}}{\models_{\mathbf{A}}} B_j^i:\rho_j^i$ for all $i \in \mathcal{A}$ and all $1 \leq j \leq k_i$. Furthermore, corresponding to the C_i's above, there is a clause in Q having a ground instance, C_i' of the form: $A:[\{i\},\mu_i] \leftarrow B_1^i:[\{i\},\rho_1^i] \,\&\ldots\&\, B_{k_i}^i:[\{i\},\rho_{k_i}^i]$. As $Q \overset{\text{wfs}}{\models_{\mathbf{A}}} (B_1^i:[\{i\},\rho_1^i] \,\&\ldots\&\, B_{k_i}^i:[\{i\},\rho_{k_i}^i])$ for all $i \in \mathcal{A}$, it follows that

$Q \overset{\text{wfs}}{\models}_{\mathbf{A}} A : [\{i\}, \mu]$.

(2) Proceeds along analogous lines.

γ a Limit Ordinal Immediate. ∎

Proof of Theorem 7.1. Follows immediately from the specification of I; I behaves exactly like ν_1 on positive (i.e. unprimed) atoms in $\mathcal{Z}_{\mathsf{ann}}$ and exactly like ν_2 on negative (i.e. primed) atoms in $\mathcal{Z}_{\mathsf{ann}}$. The result follows immediately from this observation, the construction of $\mathcal{Z}_{\mathsf{ann}}$, and the definition of the operators $\mathbf{R}_{\mathcal{Z}}(I)$ and $\Psi_{\mathcal{Z}}(\nu_1, \nu_2)$. ∎

Proof of Theorem 7.2. Immediate consequence of Theorems 5.4 and 5.5. ∎

Appendix B: Target Recognition Example

$\boxed{DB_1}$

$$
\begin{array}{rcl}
obj(t72) : \mathbf{t} & \leftarrow & radar(r_1) : \mathbf{t}. \\
obj(t60) : \mathbf{t} & \leftarrow & radar(r_2) : \mathbf{t}. \\
obj(t72) : \mathbf{f} & \leftarrow & radar(r_3) : \mathbf{t}. \\
obj(t80) : \mathbf{t} & \leftarrow & radar(r_4) : \mathbf{t}. \\
radar(r_3) : \mathbf{t} & \leftarrow &
\end{array}
$$

$\boxed{DB_2}$

$$
\begin{array}{rcl}
obj(t60) : \mathbf{t} & \leftarrow & gun(c_1) : \mathbf{t}. \\
obj(t80) : \mathbf{f} & \leftarrow & gun(c_2) : \mathbf{t}. \\
obj(t72) : \mathbf{t} & \leftarrow & gun(c_3) : \mathbf{t}. \\
gun(c_3) : \mathbf{t} & \leftarrow &
\end{array}
$$

$\boxed{DB_3}$

$$
\begin{array}{rcl}
obj(t72) : \mathbf{f} & \leftarrow & speed(high) : \mathbf{t}. \\
obj(t80) : \mathbf{f} & \leftarrow & speed(high) : \mathbf{t}. \\
obj(t80) : \mathbf{f} & \leftarrow & speed(medium) : \mathbf{t}. \\
speed(high) : \mathbf{t} & \leftarrow &
\end{array}
$$

$\boxed{\text{Supervisory DB}}$

$$
\begin{array}{rcl}
obj(X) : [\{\mathsf{s}\}, U] & \leftarrow & obj(X) : [\{1\}, U] \,\&\, obj(X) : [\{2\}, neg(U)]. \\
obj(X) : [\{\mathsf{s}\}, V] & \leftarrow & obj(X) : [\{3\}, V] \,\&\, obj(X) : [\{2\}, neg(U)]. \\
obj(X) : [\{\mathsf{s}\}, V] & \leftarrow & obj(X) : [\{1, 3\}, V].
\end{array}
$$

Here, X is an ordinary variable, while U, V are $\mathcal{T}$-variables. neg is a $\mathcal{T}$-function that maps $\mathbf{t}$ to $\mathbf{f}$, $\mathbf{f}$ to $\mathbf{t}$, and $\perp$ and $\top$ to themselves.

Appendix C: Fault Identification Example

$\boxed{DB_1}$

$$\begin{aligned}
broken(c_1) : 0.75 &\leftarrow sensor1(s_1) : 1. \\
broken(c_2) : 0.25 &\leftarrow sensor1(s_1) : 1. \\
broken(c_3) : 0.80 &\leftarrow \mathbf{not}(sensor1(s_1) : 1). \\
broken(c_2) : 0.20 &\leftarrow \mathbf{not}(sensor1(s_1) : 1). \\
sensor1(s_1) : 1 &\leftarrow
\end{aligned}$$

$\boxed{DB_2}$

$$\begin{aligned}
broken(c_1) : 0.10 &\leftarrow sensor2(w_1) : 1. \\
broken(c_2) : 0.60 &\leftarrow sensor2(w_1) : 1. \\
broken(c_3) : 0.30 &\leftarrow sensor2(w_1) : 1. \\
broken(c_1) : 0.05 &\leftarrow \mathbf{not}(sensor2(w_1) : 1). \\
broken(c_2) : 0.05 &\leftarrow \mathbf{not}(sensor2(w_1) : 1). \\
broken(c_3) : 0.90 &\leftarrow \mathbf{not}(sensor2(w_1) : 1). \\
check(C) : 1 &\leftarrow sensor2(w_1) : 1 \,\&\, broken(C) : 0.3. \\
sensor2(w_1) : 1 &\leftarrow
\end{aligned}$$

$\boxed{\text{Supervisory DB}}$

$$\begin{aligned}
check(c_2) : [\{\mathbf{s}\}, 1] &\leftarrow broken(c_2) : [\{1\}, 0.5] \,\&\, broken(c_2) : [\{2\}, 0.5]. \\
check(c_1) : [\{\mathbf{s}\}, 1] &\leftarrow broken(c_1) : [\{1\}, 0.75] \,\&\, broken(c_1) : [\{2\}, 0.75]. \\
check(c_2) : [\{\mathbf{s}\}, 1] &\leftarrow broken(c_2) : [\{2\}, 0.55] \,\&\, broken(c_2) : [\{1\}, 0.2].
\end{aligned}$$

Appendix D: Employee Example

$\boxed{\text{Boston DB}}$

$$\begin{aligned}
position(john_doe, manager) : 1990 &\leftarrow \\
salary(john_doe, 42) : 1991 &\leftarrow
\end{aligned}$$

$\boxed{\text{Seattle DB}}$

$$\begin{aligned}
position(john_doe, vice_pres) : 1992 &\leftarrow \\
salary(john_doe, 50) : 1992 &\leftarrow
\end{aligned}$$

$\boxed{\text{Supervisory DB}}$

$$\begin{aligned}
position(X, Z) : [\{\mathbf{s}\}, T] &\leftarrow position(X, Z) : [D, T] \,\& \\
&\quad\ \mathbf{not}(position(X, Z) : [\{1, \ldots n\} - D, T']) \,\& \\
&\quad\ T \geq T'. \\
salary(X, S) : [\{\mathbf{s}\}, T] &\leftarrow salary(X, S) : [D, T] \,\& \\
&\quad\ \mathbf{not}(salary(X, S) : [\{1, \ldots n\} - D, T]) \\
&\quad\ T \geq T'.
\end{aligned}$$