# Hybrid Knowledge Bases*


**James J. Lu**
*Bucknell University*
*Lewisburg, PA 17837*
*lu@sol.cs.bucknell.edu*

**Anil Nerode**
*Cornell University*
*Ithaca, NY 14853*
*anil@math.cornell.edu*

**V. S. Subrahmanian**
*University of Maryland*
*College Park, MD 20742*
*vs@cs.umd.edu*



### Abstract

Deductive databases that interact with, and are accessed by, reasoning agents in the real world (such as logic controllers in automated manufacturing, weapons guidance systems, aircraft landing systems, land-vehicle maneuvering systems, and air-traffic control systems) must have the ability to deal with multiple modes of reasoning. Specifically, the types of reasoning we are concerned with include, amongst others, reasoning about time, reasoning about quantitative relationships that may be expressed in the form of differential equations or optimization problems, and reasoning about *numeric* modes of uncertainty about the domain which the database seeks to describe. Such databases may need to handle diverse forms of data structures, and frequently they may require use of the assumption-based non-monotonic representation of knowledge.

A hybrid knowledge base is a theoretical framework capturing all the above modes of reasoning. The theory tightly unifies the Constraint Logic Programming Scheme of Jaffar and Lassez [11], the Generalized Annotated Logic Programming Theory of Kifer and Subrahmanian [16], and the Stable Model semantics of Gelfond and Lifschitz [6]. New techniques are introduced which extend both the work on Annotated Logic Programming and the Stable Model semantics.



**Keywords:** annotated logic, constraint logic programming, heterogeneous system, mediator, stable model



*This work was supported by the Army Research Office under grant numbers DAAG-29-85-C-0018 and DAAL-03-92-G-0225, by the Air Force Office of Scientific Research under grant number F49620-93-1-0065, by ARPA/Rome Labs contract Nr. F30602-93-C-0241 (Order Nr. A716), and by the National Science Foundation under grant numbers IRI9109755, IRI9357756 and CCR9225037.

Proofs are omitted for conciseness. Details may be found in [23]

# 1 Introduction

Wiederhold and his colleagues [35, 36, 37, 38] have proposed the important concept of a *mediator* between heterogeneous databases. Intuitively, a mediator is a program that accesses different data and knowledge representation paradigms and works with *existing* implementations of those paradigms. Mediators typically express methods to resolve conflicts, unify mismatches in measurement units (e.g. data expressed in centimeters and inches), and generate sophisticated conclusions based on information contained in a variety of data structures and data repositories. Till recently, mediators have primarily been developed on an ad-hoc basis, with large amounts of C-like code being written to implement them. Maintaining such software is almost impossible because reading any large body of C code is a formidable task. In this paper, we propose a formalism called *hybrid knowledge bases* (**HKB**s, for short) that has the capability of expressing and reasoning with a variety of information represented in various ways. The language of **HKB**s is completely declarative, and hence, maintaining and understanding the rules in an **HKB** is relatively easy. **HKB**s form a simple theoretical framework within which mediators can be expressed.

The theory that we develop is based on the recent works in Constraint Logic Programming (CLP) by Jaffar and Lassez [11], in Non-monotonic Logic Programming (NMLP), and in Generalized Annotated Logic Programming (GAP) by Kifer and Subrahmanian [16]. Each of these developments allows us to represent and manipulate certain forms of automated reasoning that are critical in hybrid reasoning.

Our work shows that the three aforementioned extensions of classical logic programming complement one another naturally, and that such an integration provides a powerful tool for sophisticated knowledge base development. Moreover, our study extends previous approaches in the following ways. First of all, the annotated logic programming component of our framework is different from the development of [16] in that annotations are not elements taken directly from the underlying lattice $\Delta$ of truth values. Rather, a different set of syntactic objects are introduced as annotations that map onto $\Delta$. Doing so allows for increased flexibility in the manipulation of the annotations at the expense of a slight loss of expressive power. More importantly, the new annotation system greatly simplifies the query processing procedure developed in [16] for GAPs. Secondly, we show that the problem of computing the Gelfond-Lifschitz transform that defines the semantics of nonmonotonic logic programs may be viewed as a constraint solving problem over the domain of substitutions. This revelation gives a tight coupling of the three theories via constraint solving since it has already been shown [16, 2, 20, 21] that query answering in GAPs may be accomplished through constraint solving. In other words, the theory that we develop is not a simple combination of several disjoint theories. Rather, our results show that constraint solving, over various domains, provides a thread through which the three parts may be unified.

We demonstrate through examples how the system may be applied to information mediation. In particular, we show how important and diverse classes of data structures may be viewed as constraint domains. The framework proposed in this paper has already led to one concrete implementation for the US Army Corps of Engineers [3] and another application in mobile robotics for the National Institute of Standards and Technology[10].

The organization of this paper is as follows. In Section 2, we introduce the basic syntax of hybrid knowledge bases. It's two main components are constraints (Section 2.1) and annotations (Section 2.2). Section 3 develops the semantics of positive hybrid knowledge bases. A sound and complete proof procedure is given. Based on the semantics of positive hybrid knowledge bases, Section 4 develops the semantics of general hybrid knowledge bases that may contain non-monotonic negation. Section 5 presents an extended example illustrating the application of hybrid knowledge bases to problems involving hybrid reasoning.

# 2 Hybrid Knowledge Bases: Syntax

We assume that we have an underlying datalog language $L$, i.e. $L$ is a first order language generated by a finite set of constant symbols, a finite set of predicate symbols, and an infinite set of variable symbols, called *object variables.* $L$ contains the usual connectives: $\neg$ (classical negation), & (conjunction), $\leftarrow$ (implication), $\vee$ (disjunction), $\exists$ (existential quantification) and $\forall$ (universal quantification). In this section, we will define two other types of syntactic objects, each reflecting the domains we wish to reason about (constraints, on the one hand, and time and uncertainty on the other).

## 2.1 Constraints

The introduction of *explicit* constraints into logic programming was due to Jaffar and Lassez [11] in their Constraint Logic Programming framework (CLP). The motivation behind CLP is to augment the expressive power of conventional logic programming with constraint solving in specific domains.

We begin with a domain $D$ of objects, called the domain of discourse. The function space generated by $D$ consists of the set of all functions over $D$, as well as the set of all functionals over functions of $D$. A *constraint domain* defined by $D$, denoted $\Sigma_D$ is a pair $(F, R)$ where $F$ is a subset of the function space generated by $D$, and $R$ is a set of relations over $F$.

Most software packages, databases, and knowledge representation schemes consist of a set of "data objects" together with a set of operations to manipulate those data objects. Intuitively, the set $D$ may be thought of as the set of "data objects" that we wish to reason about and manipulate.

A constraint domain $\Sigma_D$ specifies a set $F$ of functions and functionals that are used to manipulate these data objects. The set $R$ represents the relations over $F$ that we are interested in. Thus, constraint domains may be thought of as an abstraction of existing pieces of software including special-purpose databases and knowledge representation systems.

For example, consider a numerical computation package manipulating the real numbers. In this case, $\mathbf{R}$ (the set of reals) is our domain of discourse; functions implemented in such a system may include the definite integral $DEFINT$ which may be viewed as a higher order function that takes as inputs, a real-valued function $f$ and two real numbers $a$ and $b$, and returns a real number. e.g.

$$DEFINT(2x, 0, 1) = x^2(1) - x^2(0).$$

Likewise, we may have a relation $r \in R$ that says if $f, g : \mathbf{R} \to \mathbf{R}$, then

$$f\, r\, g \leftrightarrow (\forall x \in \mathbf{R}) f(x) \leq g(x).$$

In this case, for the pair $(f, g)$ to be in the relation $r$, both $f$ and $g$ must be unary functions on $\mathbf{R}$ and they must satisfy the above condition of "belowness." Note that an element $e$ of $D$ may be viewed as a 0-ary function that always returns $e$ as its output. Thus $D$ is a subset of the function space generated by $D$.

Given a constraint domain $\Sigma_D = (F, R)$, we assume the existence of a first order language whose function and predicate symbols are interpreted over $\Sigma_D$. A *constraint term* is built in the usual way ¿from the non-logical symbols of the language. If $p$ is an n-ary predicate symbol and $\alpha_1, \ldots, \alpha_n$ are constraint terms, then $p(\alpha_1, \ldots, \alpha_n)$ is an *atomic* constraint. More generally, a *constraint* is either an atomic constraint, or is the expression $\neg C$ where $C$ is a constraint, or is one of the expressions $(C_1 \,\&\, C_2)$, $(C_1 \vee C_2)$ where $C_1$ and $C_2$ are constraints. To ease the presentation, whenever possible, we do not distinguish between objects in $\Sigma_D$ and expressions of the first order language whose denotations are in $\Sigma_D$. Moreover, we often leave out the subscript $D$ in $\Sigma_D$ as that will be clear from the context.

Given a constraint domain $\Sigma_D = (F, R)$ and a constraint $C$, $\Sigma_D$ is said to satisfy $C$, denoted $\Sigma_D \rhd C$, if:

- $C$ is the atomic constraint $p(\alpha_1, \ldots, \alpha_n)$ then $(\alpha^1, \ldots, \alpha^n) \in p$ where $p \in R$.
- $C$ is the constraint $\neg D$, then $\Sigma_D \not\rhd D$.
- $C$ is the constraint $(D \,\&\, E)$, then $\Sigma_D \rhd D$ and $\Sigma_D \rhd E$.
- $C$ is the constraint $(D \vee E)$, then $\Sigma_D \rhd D$ or $\Sigma_D \rhd E$.

Thus, for every constraint $C$, either $\Sigma_D \rhd C$ or $\Sigma_D \rhd \neg C$.

## 2.2 Annotation Theory

Kifer and Subrahmanian have shown in [16] that the *Generalized Annotated Logic Programs* (GAP's) framework naturally captures certain temporal reasoning applications using multiple valued logic. In the current paper, we develop an amalgamation of time and uncertainty using GAP's. We assume that the model of time being used is linear, metric time, and the model of uncertainty being used is fuzzy logic.

Underlying any GAP is a complete lattice $\Delta$. As discussed in [16], elements in $\Delta$ may be thought of as confidence factors, as degrees of belief, or as truth values. The distinguishing feature of GAP's is the incorporation of names for $\Delta$ directly into the language of the logics. Such signing of formulas has subsequently been considered elsewhere including by [26] for analyzing multiple-valued logics, by [8, 9] using analytic tableaux, and by [34] for fuzzy reasoning. For the current work, the lattice $\Delta$ of interest consists of the set of all functions from $\mathbf{R}^+$, the set of non-negative reals, to $[0,1]$. Elements in $\Delta$ are ordered by the relation $\preceq$, defined by: $f_1 \preceq f_2$ iff for all $r \in \mathbf{R}^+$, $f_1(r) \leq f_2(r)$ where $\leq$ is the usual less-than-or-equals ordering on the reals.

Using the symbol $\sqcup$ to denote the least upper bound operator, it is straightforward to verify that given a subset $X$ of $\Delta$, $\sqcup X$ is the function such that

$$(\sqcup X)(r) = \max\{f(r) \mid f \in X\}.$$

Conversely, the greatest lower bound of $X$, denoted $\sqcap X$, is the function

$$(\sqcap X)(r) = \min\{f(r) \mid f \in X\}.$$

The top element of the lattice $\Delta$ is the function that assigns 1 to each element of $\mathbf{R}^+$, while the bottom element of the lattice assigns 0 to each element of $\mathbf{R}^+$. We denote the top and the bottom elements by $f_\top$, $f_\perp$, respectively.

Each element of $\Delta$ is a mapping from $\mathbf{R}^+$ to $[0,1]$. Intuitively, the domain $\mathbf{R}^+$ of each mapping represents time points, and the range $[0,1]$ represents certainty. Thus if a proposition $A$ is associated with a function $f$ in $\Delta$ that maps 1 and 2 to 0.5, 3 to 0.7, and all other elements in $\mathbf{R}^+$ to 0, then informally this may be interpreted as saying that the proposition $A$ has a certainty value of at least 0.5 at time points 1 and 2, a certainty value of at least 0.7 at point 3, and has a certainty value of 0 at all other time points.

Suppose $f$ is a function such that for all inputs $S \subseteq \mathbf{R}^+$ to which $f$ assigns a non-zero output, $f$ gives the same output, say $\alpha$, to each $x \in S$. Then we will often denote $f$ by the pair $(\alpha, S)$. Conversely, a pair $(\alpha, S)$ where $\alpha \in [0,1]$ and $S \subseteq \mathbf{R}^+$ uniquely determines a function $f$ in $\Delta$ such that $f(x) = \alpha$ for each $x \in S$, and $f(x) = 0$ otherwise. A function $f$ is said to be *$\ell$-representable* if there is a $\alpha \in [0,1]$ and $S \subseteq \mathbf{R}^+$ such that $f = (\alpha, S)$. The pair $(\alpha, S)$ is called the *$\ell$-representation* of $f$. Not every function in $\Delta$ is $\ell$-representable; any function whose set of outputs contains more than two elements cannot be so represented. This fact will play an important role later when we introduce the semantics of hybrid knowledge bases. Clearly, both $f_\top$ and $f_\perp$ are $\ell$-representable. The unique $\ell$-representation of $f_\top$ is $(1, \mathbf{R}^+)$. On the other hand, any one of $(0, S)$, where $S \subseteq \mathbf{R}^+$ is an $\ell$-representation of $f_\perp$.

We are interested in expressing statements of the form: "Proposition $A$ has certainty at least $\alpha$ at all time points in $S$." If $f$ is a function that assigns $\alpha$ to all elements in $S$ and 0 everywhere else, then the above statement may be expressed as the proposition $A$ annotated with the function name $f$, i.e. $A : f$. Indeed, this is the syntax used by Annotated Logics [13, 16]. However, in this paper, we use the $\ell$-representation of $f$ as the annotation and write $A : (\alpha, S)$ instead. The reason for this change will become clear.

Thus unlike ordinary GAPs, there is a subtle distinction between the elements in the set $\Delta$, and the objects used for annotating atoms of $L$ within our hybrid knowledge bases. We further enhance the expressiveness of our annotation system as follows. For each integer $i \geq 1$, $\mathcal{U}_i$ and $\mathcal{T}_i$ are two families of total continuous (hence monotonic) functions of the types $[0,1]^i \to [0,1]$ and $(2^{\mathbf{R}^+})^i \to (2^{\mathbf{R}^+})$, respectively. Each function in $\mathcal{U}_i$ and $\mathcal{T}_i$ is assumed to be computable. The set of annotation functions include, for each $j \geq 0$, the $j$-ary function $\sqcup_j$ such that, given inputs $\mu_1, \ldots, \mu_j$, it returns the least upper bound of the set $\{\mu_1, \ldots, \mu_j\}$. $AV_U$ and $AV_T$ are two sets of *annotation variables*. Each variable in $AV_U$ (resp. $AV_T$) ranges over the elements in $[0,1]$ (resp. $(2^{\mathbf{R}^+})$).

A *$U$-annotation* is either an element of $[0,1]$, an element of $AV_U$ or a *complex $U$-annotation term* constructed from annotation functions in $\mathcal{U}_i$, annotation variables in $AV_U$, and elements in $[0,1]$ in the usual ways. A *$T$-annotation* is defined similarly over the sets $(2^{\mathbf{R}^+})$, $AV_T$, and $\mathcal{T}_i$. Finally, an *annotation* is a pair $(u,t)$ where $u$ is a $U$-annotation and $t$ is a $T$-annotation. We remind the readers that an annotation $(u,t)$ corresponds to, given a particular choice for the variables in $u$ and $t$, an $\ell$-representation of a function in $\Delta$. Following the terminology of [16], an annotation $(u,t)$ is a *c-annotation* if $u \in [0,1]$ and $t \in 2^{\mathbf{R}^+}$. When both components of an annotation are variables, it is called a *v-annotation*. If either of the components is a complex annotation, then it is called a *t-annotation*. An *annotation substitution* $\eta$, sometimes referred to as an a-substitution, is a map from annotation variables (i.e. $AV_U \cup AV_T$) to the set $[0,1] \cup 2^{\mathbf{R}^+}$ such that for all $x \in AV_U$, $\eta(x) \in [0,1]$ and for all $y \in AV_T$, $\eta(y) \in 2^{\mathbf{R}^+}$. Implied by the definition is that an annotation substitution is necessarily ground, in the sense that each annotation variable, whether in $AV_U$ or $AV_T$, is mapped to an object in the corresponding set, i.e. either $[0,1]$ or $2^{\mathbf{R}^+}$.

Using this annotation system, we may construct independent expressions over the domain and the range of an $\ell$-representable function. For example, $(0.5 + x, s_1 \cup s_2)$ denotes an $\ell$-representable function schema where, depending on the values given for $x, s_1$ and $s_2$, denotes a function that maps each element in both $s_1$ and $s_2$ to the value $0.5 + x$. Hence although we have eliminated certain expressiveness in the annotation system by considering only $\ell$-representable functions, we nevertheless have a very rich, flexible language whereby a large collection of $\ell$-representable functions may be succinctly expressed. The above example would have been somewhat more difficult to represent had we simply used the names of functions directly from $\Delta$ as our annotations.

## 2.3 Hybrid Knowledge Bases

Recall that $L$ is an underlying Datalog language. Suppose $A$ is an atom constructed from constants, variables, and predicates in $L$ in the usual way. Moreover, suppose $(u, t)$ is an annotation. Then the expression $A : (u, t)$ is an *annotated atom.* In the case that an annotated atom is free of any object variables, it is said to be *ground.* An annotated atom whose annotation is a c-annotation is said to be a *c-annotated atom.* Similarly, an annotated atom may be v-annotated or t-annotated. A substitution for object variables is defined in the same way as for classical logic programming [24]. To distinguish such a substitution from an annotated substitution, we refer to it as an o-substitution to indicate that it is a substitution for object variables. We write $(A : (u, t))\theta$ to denote the result of applying a substitution $\theta$ to the annotated atom $A : (u, t)$. If $\theta$ is an annotation substitution, then $(A : (u, t))\theta$ is the same expression as $A : ((u, t)\theta)$. Otherwise $\theta$ is an o-substitution and $(A : (u, t))\theta$ is just $A\theta : (u, t)$.

Suppose each of $A : (u_0, t_0)$, $B_1 : (u_1, t_1), \ldots, B_k : (u_k, t_k), C_1 : (v_1, s_1), ..., C_m : (v_m, s_m)$ is an annotated atom. Then the formula

$$A : (u_0, t_0) \leftarrow B_1 : (u_1, t_1) \,\&\, \ldots \,\&\, B_k : (u_k, t_k) \&\mathbf{not}\,(C_1 : (v_1, s_1))\&...\&\mathbf{not}\,(C_m : (v_m, s_m))$$

is called an *annotated clause.* The annotated atom $A : (u_0, t_0)$ is called the *head* of this clause, while the expression $B_1 : (u_1, t_1) \,\&\, \ldots \,\&\, B_k : (u_k, t_k) \&\mathbf{not}\,(C_1 : (v_1, s_1))\&...\&\mathbf{not}\,(C_m : (v_m, s_m))$ is called the *body* of the clause. All variables (object and annotation) are implicitly universally quantified. A set of annotated clauses is called a *Generalized Annotated Program*, abbreviated GAP.

A *c-annotated instance* of an annotated clause $C$ is an annotated clause obtained from $C$ by applying an annotation substitution to each annotated atom occurring in $C$. Note that different occurrences of the same annotation variable within $C$ will be replaced by the same element. Suppose $C$ is an annotated clause. A *strictly ground instance* of $C$ is a ground c-annotated instance of $C$. As each annotation function is computable, any variable-free annotation term $\mu$ may be replaced by the element, in either $[0, 1]$ or in $2^{\mathbf{R}^+}$, that is the result of carrying out the computation of all functions in $\mu$ on their arguments. Throughout this paper, we make the assumption that such a replacement is always made whenever possible.

A *constrained clause* is an expression of the form

$$A \leftarrow \Xi \parallel B_1 \& \ldots \& B_n \& \mathbf{not}\,(C_1) \& ... \& \mathbf{not}\,(C_m)$$

where $A, B_1, \ldots, B_n, C_1, ..., C_m$ are annotated atoms, and $\Xi$ is a constraint. $\Xi$ is the *constraint part* of the constrained clause, and $A \leftarrow B_1 \& \ldots \& B_n$ is the *annotated clause part* of the constrained clause. A *hybrid knowledge base* (**HKB**) consists of a set of constrained clauses. A *strictly ground instance* of a constrained clause is obtained by replacing the annotated clause part with a strictly ground instance. If a **HKB** is absent of any occurrences of **not** , then it is called a *positive* hybrid knowledge base. The symbol **not** is used to represent a form of negation as failure, similar to those considered frequently in logic programming. To define the meaning of hybrid knowledge bases, we need to first define the semantics of positive hybrid knowledge bases.

# 3 Positive Hybrid Knowledge Bases

The semantics of positive hybrid knowledge bases can be derived naturally by combining the formalisms developed for GAPs [16] and for CLPs [11]. Throughout this section, we consider only positive **HKB**. We assume a fixed constraint domain $\Sigma_D$.

## 3.1 $\Sigma$-Model and Fixpoint Theory

A $\Sigma$-*interpretation* $I$ is a function from the Herbrand Base of our underlying first order language to $\Delta$. The ordering $\preceq$ on elements of $\Delta$ may be naturally extended to $\Sigma$-interpretations in the following ways: Given two interpretations $I_1$ and $I_2$, $I_1 \preceq I_2$ iff for each ground atom $A$, $I_1(A) \preceq I_2(A)$.

**Definition.** A $\Sigma_D$-interpretation $I$ *satisfies* a formula $F$ iff:

1. if $F$ is a c-annotated ground atom $A : (u, t)$, then $(u, t) \preceq I(A)$.
2. if $F = (G \,\&\, H)$, then $I \models G$ and $I \models H$
3. if $F = (G \vee H)$, then $I \models G$ or $I \models H$
4. if $F = (G \leftarrow H)$, then $I \models G$ or $I \not\models H$
5. if $F = (\forall x)G$ then $I \models G'$ for each instance $G'$ of $G$ obtained by replacing each occurrence of $x$ in $G$ by a constant of $L$ (if $x$ is an ordinary variable) or a c-annotation if $x$ (if $x$ is an annotation variable)

6. if $F = (\exists x)G$ then $I \models G'$ for some instance $G'$ of $G$ obtained by replacing each occurrence of $x$ in $G$ by a constant of $L$ (if $x$ is an ordinary variable) or a c-annotation if $x$ (if $x$ is an annotation variable).

7. if $F$ is the constrainted-clause $A : (u_0, t_0) \leftarrow \Xi \parallel B_1 : (u_1, t_1) \,\&\ldots\&\, B_n : (u_n, t_n)$, then either $\Sigma_D \rhd \neg\Xi$, or $I \models A : (u_0, t_0) \leftarrow B_1 : (u_1, t_1) \,\&\ldots\&\, B_n : (u_n, t_n)$.

When $I$ satisfies a formula $F$, $I$ is said to be a $\Sigma$-*model* of $F$; this relationship is denoted $I \models F$. When the constraint domain $\Sigma$ is clear from context, we usually refer to $I$ simply as a model of $F$. A **HKB** $\Sigma$-entails $F$, denoted **HKB** $\models F$, iff whenever $I$ is a model of all the constraint clauses in **HKB**, it is also the case that $I$ is a model of $F$.

The above semantics of **HKB** appears to be a fairly straightforward adaptation of the semantics of GAP's where the additional conditions required for the constraints occurring in each clause is dependent upon the specific constraint domain. However, as noted earlier in the paper, the use of $\ell$-representations of functions in $\Delta$ as annotations, rather than the functions themselves, presents subtleties in the semantics to which we now turn our attention.

A strictly ground fact $A : (u, t) \leftarrow$ in an **HKB** intuitively states that $A$ has certainty at least $u$ at all time points in $t$. However, this fact may only partially determine the certainty of $A$ since other parts of the knowledge base may also contain information that contributes to the truth value of $A$. Suppose we have another strictly ground fact $A : (u', t') \leftarrow$. Then the certainty of $A$ becomes more defined. Specifically, if $u > u'$, the certainty of $A$ at all time points in $t$ is at least $u$, while the certainty of $A$ at all time points $t' - t$ is at least $u'$. In other words, intuitively, the "truth value" of $A$ implied by the above facts is the function $f$ where

$$\begin{aligned} f(x) \quad & = u \text{ if } x \in t \\ & = u' \text{ if } x \in t' - t \\ & = 0 \text{ otherwise.} \end{aligned}$$

The symbol $-$ denotes set subtraction. Note that $f = \sqcup\{(u, t), (u', t')\}$. By earlier remarks, the function $f$ is not $\ell$-representable. Therefore, there does not exist an annotation $(v, s)$ that is the $\ell$-representation of $f$ such that **HKB** $\models A : (v, s)$. This is the one aspect of the semantics of positive **HKB**s that makes it different from GAPs. Note that in choosing our annotations in this way, we lose some expressiveness in the language. As the last example illustrates, it is not possible to directly represent certain entailed information. To state that $A$ has the truth value $f$, we must distribute that information over two annotated atoms, viz. $A : (u, t) \& A : (u', t' - t)$. However, we gain several advantages by our choice of annotation. First, $\ell$-representations make explicit the elements in the domain and the range of a function, which enables the incorporation of $U$-annotation functions and variables, as well as $T$-annotation functions and variables, easily. This reiterates the point made in the end of Section 2.2. Moreover, as we will see in the next section, $\ell$-representations allow for a simplification of the resolution proof procedure developed for GAPs. It is not necessary to use the expensive *reduction* inference rule. Additionally, the proofs thus produced are also more readable.

As in the case of classical logic programming, we may associate an operator, $\mathbf{T_{HKB}}$, with any positive **HKB**. The function $\mathbf{T_{HKB}}$ maps $\Sigma$-interpretations to $\Sigma$-interpretations, and is defined as follows:

> $\mathbf{T_{HKB}}(I)(A) = \sqcup\{(u,t) \mid A : (u,t) \leftarrow \Xi \parallel B_1 : (u_1, t_1) \,\&\ldots\&\, B_n : (u_n, t_n)$ is a strictly ground instance of a clause in **HKB** such that $\Sigma \rhd \Xi$ and $I \models B_1 : (u_1, t_1) \,\&\ldots\&\, B_n : (u_n, t_n)\}$.

The upward iteration of the $\mathbf{T_{HKB}}$ operator can be defined in the standard way as follows:

$\mathbf{T_{HKB}} \uparrow 0(A)(t) = 0$

$\mathbf{T_{HKB}} \uparrow \alpha(A)(t) = \mathbf{T_{HKB}}((\mathbf{T_{HKB}} \uparrow \beta))(A)(t)$ where $\alpha = \beta + 1$

$\mathbf{T_{HKB}} \uparrow \gamma(A)(t) = \sqcup_{\alpha<\gamma}\mathbf{T_{HKB}} \uparrow \alpha(A)(t)$ for limit ordinals $\gamma$.

**Theorem 1** Suppose $DB$ is any positive hybrid knowledge base. Then:

1. $\mathbf{T_{HKB}}$ is monotone.
2. $I$ is a $\Sigma$-model of $DB$ iff $\mathbf{T_{HKB}}(I) \preceq I$.
3. $DB \models A : (u,t)$ iff $(u,t) \preceq lfp(\mathbf{T_{HKB}})$.
4. $lfp(\mathbf{T_{HKB}}) = \mathbf{T_{HKB}} \uparrow \eta$ for some ordinal $\eta$. ∎

Similar to GAPs, the property that the fixpoint operator associated with a GAP may not always achieve a fixpoint after $\omega$ iterations holds for **HKB**'s as well. Consider the positive **HKB** containing the following three clauses:

$$\begin{aligned} p : (0, \{0\}) &\leftarrow \\ p : (\frac{V+1}{2}, \{0\}) &\leftarrow p : (V, \{0\}) \\ q : (1, \{0\}) &\leftarrow p : (1, \{0\}) \end{aligned}$$

It is easy to see that $\mathbf{T_{HKB}} \uparrow \omega \models p : (1, \{0\})$ even though, $\mathbf{T_{HKB}} \uparrow k \not\models p : (1, \{0\})$ for all $k < \omega$. Hence, $\mathbf{T_{HKB}} \uparrow \omega \not\models q : (1, \{0\})$. However, $\mathbf{T_{HKB}} \uparrow (\omega + 1) \models q : (1, \{0\})$. In fact, $\mathbf{T_{HKB}} \uparrow (\omega + 1)$ is the least fixpoint of $\mathbf{T_{HKB}}$.

We are interested in **HKB**s having a reachability property, which says that if $lfp(\mathbf{T_{HKB}})(A) = f$, then at all time points $t \in \mathbf{R}^+$, $t$ is assigned the value $f(t)$ after a finite iteration of the $\mathbf{T_{HKB}}$

operator. More formally, an **HKB** is said to have the *fixpoint reachability property* iff for all ground atoms $A$, $lfp(\mathbf{T_{HKB}})(A) = f$ implies that there exists an integer $k$ such that $(\mathbf{T_{HKB}} \uparrow k(A)) = f$. It is easy to see that any positive **HKB** that has the fixpoint reachability property is guaranteed to obtain a fixpoint at $\omega$.

**Multiple Constraint Domains:** Note that the constraint domain $\Sigma$ may be comprised of several disjoint constraint domains $\Sigma_1, ..., \Sigma_n$. In general, when we attempt to mediate between multiple data structures, each of these data structures (together with the corpus of implemented algorithms manipulating those data structures) may be viewed as a constraint domain. In such a case, we assume that the constraint satisfaction relation $\triangleright$ respects the individual domains. Thus, for instance, if $(\Xi_1 \,\&\, \Xi_2)$ is a constraint where $\Xi_1$ is over $\Sigma_1$ and $\Xi_2$ is over $\Sigma_2$, then this conjunctive constraint is satisfied iff each of its two conjuncts is satisfied over its specified domain. The introduction of multiple constraint domains does not raise any additional theoretical issues, and the correspondence between the model theory and the fixpoint theory of a **HKB**, as given in Theorem 1, still holds. Indeed, we will see in Section 4 that adding non-monotonicity to **HKB**'s can be viewed as adding a constraint domain over substitutions.

## 3.2 Query Processing in Positive Hybrid Databases

A *query* $Q$ is a formula of the form:

$$\leftarrow \Xi' \| A_1 : (v_1, s_1), \ldots, A_m : (v_m, s_m).$$

$\Xi'$ is called the *constraint* part of $Q$, while $A_1 : (u_1, t_1) \,\&\, \ldots \,\&\, A_m : (u_m, t_m)$ is called the *atomic* part of $Q$.

Since, for all $u \in [0,1]$, $(u, \emptyset) = f_\perp$ is the function that maps all time points in $\mathbf{R}^+$ to 0, it is less than any function in $\Delta$ under the ordering $\preceq$. It follows that, according to the definition of satisfaction, an annotated atom of the form $A : (u, \emptyset)$ is satisfied by every interpretation. We will therefore assume, throughout the rest of this paper, that such annotated atoms do not occur in any query.

**Definition.** (Resolution) Suppose $C =$

$$A : (u, t) \leftarrow \Xi \,\|\, B_1 : (u_1, t_1) \,\&\, \ldots \,\&\, B_n : (u_n, t_n)$$

is a constrained clause and $Q = \leftarrow \Xi' \,\|\, A_1 : (v_1, s_1), \ldots, A_m : (v_m, s_m)$ is a query with no variables in common with $C$. If $A_i$, for some $1 \leq i \leq m$, is unifiable with $A$ via mgu $\theta$, then the resolvent of $Q$ and $C$ with respect to $A_i$ is the new query

$$(\leftarrow \Xi \,\&\, \Xi' \,\&\, v_i \leq u \| A_1 : (v_1, s_1) \,\&\, \ldots \,\&\, A_{i-1} : (v_{i-1}, s_{i-1}) \,\&$$

$$A_i : (v_i, (s_i - t)) \,\&\, B_1 : (u_1, t_1), \ldots, \&\, B_n : (u_n, t_n)$$

$$A_{i+1} : (v_{i+1}, s_{i+1}) \& \ldots \& A_m : (v_m, s_m))\theta.$$

In case $s_i - t$ is a ground expression and evaluates to $\emptyset$, then by the earlier remark, we may simplify the resulting query by removing the atom $A_i : (v_i, (s_i - t))$.

The intuition behind the above notion of "resolvent" is the following. In order to establish $A_i : (v_i, s_i)$, we need to show that $A_i$'s truth value is at least $v_i$ at all time points in $s_i$. If $v_i \leq u$, then by establishing $\Xi$ and $B_1 : (u_1, t_1) \,\&\ldots\&\, B_n : (u_n, t_n)$, we would have proved that $A_i$ has the desired uncertainty value, $v_i$, at all time points in $t$. Hence, all we would need to show is that $A_i$ has truth value $v_i$ at all remaining time points, which is $(s_i - t)$.

A *deduction* from a query $Q_0$ and an ALP $P$ is a sequence

$$\langle Q_0, C_0, \theta_0\rangle, \ldots, \langle Q_n, C_n, \theta_n\rangle, \ldots$$

such that $Q_{i+1}$ is a resolvent of $Q_i$ and $C_i$ via mgu $\theta_i$ and $C_i$ is a clause of $P$. The deduction is called a refutation if it is finite and the last query, say $Q_{n+1}$ contains a solvable constraint part, and for some solution $\sigma$ of the constraint, $\sigma$ applied to the atomic part of $Q_{n+1}$ yields an empty atomic part. Note if the atomic part of $Q_{n+1}$ is $A_1 : (u_1, t_1) \& \ldots \& A_k : (u_k, t_k)$, then the last statement of implies that $t_i\sigma = \emptyset$, for each $1 \leq i \leq k$.

Consider the positive hybrid knowledge base $P$ given below

$$p : (1, \{2, 3\}) \quad \leftarrow$$

$$p : (1, \{4, 5\}) \quad \leftarrow$$

It is easy to see $P \models p : (1, \{2, 3, 4, 5\})$. A refutation of the query $\leftarrow p : (1, \{2, 3, 4, 5\})$ is obtained by resolution with the first clause, obtaining as the resolvent the new query $\leftarrow p : (1, \{4, 5\})$. This resolves with the second clause to yield the query $\leftarrow p : (1, \emptyset)$. This atom can be deleted (as it is a tautology), leading to the empty clause.

As we have pointed out, our resolution procedure does not contain the reduction rule of inference found in GAPs [16]. In the simplest case, the reduction rule allows one to infer, from two ground unit annotated clauses $p : \mu_1 \leftarrow$ and $p : \mu_2 \leftarrow$, the new unit annotated clause $p : \sqcup\{\mu_1, \mu_2\} \leftarrow$. In the case of **HKB**s, such an inference is generally not possible. Consider the two unit clauses $p : (0.5, \{2, 3\}) \leftarrow$ and $p : (0.6, \{3, 4\}) \leftarrow$. The two clauses entails that $p$ has the truth value denoted by the function

$$\begin{aligned} f(x) \quad &= 0.6 \text{ if } x = 3, 4 \\ &= 0.5 \text{ if } x = 2 \\ &= 0 \text{ otherwise.} \end{aligned}$$

In other words, the certainty of $p$ is at least 0.6 at time points 3 and 4, and is at least 0.5 at time point 2. The function $f$ corresponds to the least upper bound of $(0.5, \{2,3\})$ and $(0.6, \{3,4\})$. However, $f$ is not $\ell$-representable. Hence it is not possible to infer a constrained clause $p : (u,t) \leftarrow$ where $f = (u,t)$. This relates to the earlier point that certain true information may not be directly expressible.

Yet the fact that $f$ represents the largest possible truth value that one may infer about $p$ does not change. Therefore one should answer affirmatively to any query about $p$ whose annotation is an $\ell$-representation of a function smaller than $f$, under the ordering $\preceq$. In particular, each of the following queries possesses a refutation:

$\leftarrow p : (0.6, \{3,4\})$;

$\leftarrow p : (0.5, \{2,3,4\})$;

$\leftarrow p : (0.5, \{2\}) \& p : (0.6, \{3,4\})$.

In the first query $(0.6, \{3,4\}) \preceq f$ since $(0.6, \{3,4\})(2) = 0 \leq f(2) = 0.5$. It follows that $p : (0.6, \{3,4\})$ is entailed by the **HKB**. It is straightforward to verify that each of the above indeed may be proved using our resolution procedure. Observe that the last expression queries the "maximum truth value" of $p$ implied by the **HKB**.

The above example demonstrates that due to our restriction that only a subset of the set of truth values have corresponding annotations (viz. those $\ell$-representable functions), it may be necessary at times to pose a query using several annotated atoms to express the desired truth value. However, this is a relatively minor inconvenience when compared to the computational advantage gained by the elimination of the reduction rule. The following two results show that resolution forms a sound and complete proof system for positive hybrid databases.

**Theorem 2 (Soundness of Resolution)** Suppose $P$ is a positive hybrid knowledge base and $Q$ is a query

$$\leftarrow \Xi' || A_1 : (v_1, s_1), \ldots, A_m : (v_m, s_m)$$

such that there exists a refutation of $Q$ from $P$. Then $P \models (\exists)(\Xi' || A_1 : (v_1, s_1), \ldots, A_m : (v_m, s_m))$. ∎

**Theorem 3 (Completeness of Resolution)** Suppose

$$P \models (\exists)(\Xi' || A_1 : (v_1, s_1), \ldots, A_m : (v_m, s_m))$$

where $P$ is a positive hybrid knowledge base that possesses the fixpoint reachability property. Then there is a refutation of $\leftarrow \Xi' || A_1 : (v_1, s_1), \ldots, A_m : (v_m, s_m)$ from $P$. ∎

The proof of the completeness result uses the mgu and the lifting lemmas in the standard way [24]. In [20, 21], a query processing procedure for GAPs that interleaves an SLD-like resolution procedure, and a constraint solving procedure for acyclic constraint graphs is introduced. As the theory of GAPs figures prominently in the development of hybrid knowledge bases, the relationship between constraint solving and query processing in hybrid knowledge bases also exists. Indeed, the solvability of a query

$$\leftarrow \Xi || Q$$

where $Q$ consists of a conjunction of annotated atoms depends first of all on the solvability of $\Xi$, and secondly on finding a substitution for the annotation variables so that each annotation reduces to an $\ell$-representation of $f_\perp$. Similar to the method described in [20, 21], the latter problem may often be posed as a constraint satisfaction problem involving acyclic constraint graphs whose solutions are obtainable by local propagation. We do not expand on the details here.

The next section demonstrates that the semantics of non-monotonic hybrid knowledge bases may be given in terms of constraints, thus showing that constraint solving over multiple domains is the important thread through which the various aspects of hybrid knowledge bases are linked.

# 4 Non-monotonic Hybrid Knowledge Bases

Section 3.1 introduced the semantics of positive **HKB**s. We are interested in hybrid knowledge bases that are capable of non-monotonic reasoning. In this section, we extend the semantics so far defined to cover hybrid knowledge bases in its full generality, where the negation operator **not** may occur in the body of constrained clauses.

The intuitive reading of a constrained clause

$$A \leftarrow \Xi \mid\mid B_1 \& \ldots \& B_n \& \mathbf{not}\,(C_1) \& ... \& \mathbf{not}\,(C_m)$$

is: "If the constraint $\Xi$ is satisfied by $\Sigma$, i.e. $\Sigma \rhd \Xi$, and each of the annotated atoms $B_i$ is provable, and none of the annotated atoms $C_j$ are provable, then $A$ is provable." The difficulty with this definition, which is a reincarnation of a similar phenomenon in non-monotonic deductive databases, is that the provability of an annotated atom is defined in terms of the unprovability of other annotated atoms. In the case that such a dependency structure is acyclic, then an intuitively obvious semantics can be given by the *stratified semantics* of Apt, Blair, and Walker [1]. However, should it turn out that the dependency structure of some annotated atom is defined in terms of itself i.e. it is cyclic, then the semantics of such a knowledge base becomes difficult to pinpoint.

Two major semantics currently exist for non-monotonic negation in logic programming: the stable model semantics as defined by Gelfond and Lifschitz [6], and the well-founded semantics of van Gelder, Ross and Schlipf [32, 33]. We examine the stable model semantics in the context of hybrid knowledge bases. The stable model semantics of logic programming given in [6] is defined based on ground programs (i.e. variable-free). Hence, to give meaning to programs that contain variables, one first considers the program's ground equivalent where variables are replaced by all possible ground terms within the Herbrand Base. Existing works on deductive databases that attempt to implement the stable model semantics have used the definition of the semantics directly by grounding out all clauses [4]. Such a process is computationally expensive. We introduce a novel technique through which the stable model semantics of hybrid knowledge base may be captured via a transformation. The resulting product of the transformation is a positive hybrid knowledge base in which the non-monotonic aspect is expressed as a new constraint over the domain of substitutions.

## 4.1 Blocking Sets

In defining the transformation of a hybrid knowledge base to a positive hybrid knowledge base, certain constraints are added in place of those annotated atoms preceded with the negation operator **not**. Solutions to these constraints are formed from the domain of substitutions. We call such substitutions *blocking substitutions* and define them shortly following a brief discussion of their intuition.

Informally, each expression of the form **not** $(A : \mu)$ in a constrained clause stipulates that, in order to infer the head of the clause in question, $A : \mu$ must be unprovable from the knowledge base. In other words, for each $\Sigma$-interpretation $I$ that satisfies the knowledge base, it must be the case that $I$ does not make $A : \mu$ true. To satisfy $A : \mu$, for each substitution $\rho$ that grounds out $A : \mu$, $I(A\rho^o)$ must denote a function such that $\mu\rho^a \preceq I(A\rho^o)$. Blocking substitutions are exactly those substitutions that can, with possible further instantiations, lead to these ground substitutions. Then the stipulation that $I$ does not make $A : \mu$ true can be regarded as specifying a constraint that says $A : \mu$ is applied with a substitution that is not equal to nor less general than any of the most general blocking substitutions. This is the intuitive basis behind the transformation that we will develop.

Suppose $I$ is a $\Sigma$-interpretation, and $A : (u, t)$ is a c-annotated atom. A *blocking substitution* of $A : (u, t)$ with respect to $I$ is an o-substitution $\theta$ such that $I$ satisfies $(\forall)(A : (u, t))\theta$. As usual, we may speak of one o-substitution being more general than another. Hence, a blocking substitution $\theta$ of $A : (u, t)$ is most general if there does not exist a block substitution $\lambda$ of $A : (u, t)$ such that $\lambda$ is more general than $\theta$. The *blocking set* of a c-annotated atom $A : (u, t)$ is the set of most general blocking substitutions of $A : (u, t)$.

The advantage in considering most general blocking substitutions is that the cardinality of the domain over which the aforementioned constraints are solved is considerably reduced. A simple example provides an illustration. Assume the language $L$ contains constant symbols $a, b_1, \ldots, b_n$ and at least one binary predicate symbol $p$. Let $g$ be the function that assigns 0.5 to all time points $t \in \mathbf{R}^+$. Suppose $I$ is an interpretation that assigns $f_\top$ to all ground instances of $p(a, X)$ and $g$ to all atoms whose predicate symbol is $p$, but where the first argument is a symbol other than $a$. Then consider the annotated atom $A = p(X, Y) : (0.7, \mathbf{R}^+)$. Each of the substitutions $\theta_1, ..., \theta_n$ such that $\theta_i$ maps $X$ to $a$, and $Y$ to $b_i$, is a blocking substitution of $A$ with respect to $I$. Observe that any blocking substitution of $A$ must assign to $X$ the constant $a$ since otherwise $I$ assigns the truth value $g$ to such an instance of $A$, which is less than $(0.7, \mathbf{R}^+)$ in the ordering $\preceq$. There exist numerous other blocking substitutions of $A$, including many that are non-ground. However, there exists a single most general blocking substitution, namely the substitution that maps $X$ to $a$, but it makes no assignment to $Y$. With respect to the intuition developed in the second paragraph of this section concerning the role of blocking substitutions, the most general blocking substitutions are those blocking substitutions that make the "minimum" substitutions necessary to ensure the satisfiability of $A : \mu$ within $I$.

The next lemma relates the ordering $\preceq$ on interpretations to blocking substitutions.

**Lemma 1** Suppose $I_1 \preceq I_2$ and $A$ is a c-annotated atom. If $\rho$ is a blocking substitution of $A$ with respect to $I_1$, then $\rho$ is a blocking substitution of $A$ with respect to $I_2$. ∎

Hence, the set of blocking substitutions with respect to a given interpretation is a subset of the set of blocking substitutions with respect to a larger interpretation.

Generalizing blocking substitutions to arbitrary annotated atoms is straightforward. Each substitution $\rho$ consists of two parts, one that corresponds to an o-substitution, denoted $\rho^o$, and the other corresponds to an a-substitution, denoted $\rho^a$. On the other hand, the notion of most general blocking substitution for such a substitution does not translate directly from o-substitutions. It depends on the meaning of generality in the context of a-substitutions, which is given by the ordering $\preceq$ on $\Delta$.

Suppose $\alpha_1$ and $\alpha_2$ are two a-substitutions. $\alpha_1$ is said to be more general than $\alpha_2$ if for each annotation $(u, t)$, $(u, t)\alpha_1 \preceq (u, t)\alpha 2$. Recall that any a-substitution grounds out all annotation variables. So both $(u, t)\alpha_1$ and $(u, t)\alpha_2$ denote $\ell$-representable functions within $\Delta$, and the comparison $\preceq$ is well-defined. Finally, if $\rho_1$ and $\rho_2$ are two substitutions that contain an a-substitution and an o-substitution component, then $\rho_2$ is more general than $\rho_1$ if $\rho_2^o$ and $\rho_2^a$ are more general than $\rho_1^o$ and $\rho_1^a$, respectively. Although the notion of generality is used with respect to different domains (e.g. o-substitutions and a-substitutions), the specific ordering to which it refers should be clear from context. We let $\mathbf{mgb}(A : \mu, I)$ represent the set of most general blocking substitutions for the annotated atom $A : \mu$ with respect to the $\Sigma$-interpretation $I$.

Given an annotated atom $A$ and two interpretation $I_1$ and $I_2$ where $I_1 \preceq I_2$, each blocking substitution for $A$ with respect to $I_1$ is also a blocking substitution for $A$ with respect to $I_2$. As this holds for arbitrary annotated atoms, it generalizes Lemma 1.

**Lemma 2** Suppose $I_1 \preceq I_2$ and $A$ is an annotated atom. Then each blocking substitution of $A$ with respect to $I_1$ is a blocking substitution of $A$ with respect to $I_2$. ∎

## 4.2 Transforming Hybrid Knowledge Bases

Given an o-substitution

$$\theta = \{X_1 = a_1, ..., X_n = a_n\},$$

we use $\neg\theta$ to denote the disjunction of inequalities $X_1 \neq a_1 \vee ... \vee X_n \neq a_n$. Such a disjunction expresses a constraint over the set of constants in the language $L$ the solution of which is, for some $1 \leq i \leq n$, $X_i = b_i$, where $b_i$ and $a_i$ are syntactically distinct constants. On the other hand, given an annotation $(u,t)$ and a function $f \in \Delta$, the expression $(u,t) \not\preceq f$ is a constraint over a-substitutions. A solution $\beta$ is a function, denoted $(u,t)\beta$, that satisfies $(u,t)\beta \not\preceq f$.

Suppose **HKB** is a hybrid knowledge base and $I$ is a $\Sigma$-interpretation. The *transformation* of **HKB** with respect to $I$, denoted $\mathbf{tr}(\mathbf{HKB}, I)$, is the positive hybrid knowledge base obtained by the following.

1. Suppose

$$\begin{array}{rcl} A : (u_0, t_0) & \leftarrow & \Xi \,||\, B_1 : (u_1, t_1) \,\&\, \ldots \,\&\, B_n : (u_n, t_n) \,\& \\ & & \mathbf{not}\,(D_1 : (v_1, s_1)) \,\&\, \ldots \,\&\, \mathbf{not}\,(D_m : (v_m, s_m)) \end{array}$$

   is a constrained clause $C$ in **HKB**.

   For each $1 \leq j \leq m$, let $\Xi'_j$ be the conjunction of constraints

$$(\neg\rho^o \vee ((v_j, s_j) \not\preceq (v_j, s_j)\rho^a)$$

   where $\rho$ is a substitution in $\mathbf{mgb}(D_j : (v_j, s_j), I)$. Then the constrained-clause

$$A \leftarrow \Xi \,\&\, \Xi'_1 \& ... \& \Xi'_m \,||\, B_1 : (u_1, t_1) \,\&\, \ldots \,\&\, B_n : (u_n, t_n),$$

   denoted $\mathbf{tr}(C, I)$, is in $\mathbf{tr}(\mathbf{HKB}, I)$.

2. Nothing else is in $\mathbf{tr}(\mathbf{HKB}, I)$.

Let's examine several examples.

**Example.** Suppose we have the c-annotated constrained clause $C$:

$$\begin{aligned} q(X,Y):(1,\mathbf{R}^+) \quad \leftarrow \quad & \Xi \,\| \\ & s(X,Y):(0.7,\mathbf{R}^+)\,\& \\ & \mathbf{not}\, p(X,Y):(0.7,\mathbf{R}^+)\,\& \\ & \mathbf{not}\, r(X):(0.4,\mathbf{R}^+). \end{aligned}$$

Let $I$ be an interpretation that assigns the function $(1,\mathbf{R}^+) = f_\top$ to any ground atom whose predicate symbol is $p$, and whose first argument is the constant $a$. Let $I$ also assign the function $(0.5,\mathbf{R}^+)$ to every other ground atom whose predicate symbol is $p$. Moreover, suppose $I$ assigns the function $(0.1,\mathbf{R}^+)$ to each of $r(b_2),\ldots,r(b_n)$, and it assigns $(0.6,\mathbf{R}^+)$ to $r(a)$ and $r(b_1)$.

There exist two most general blocking substitutions for $r(X):(0.4,\mathbf{R}^+)$ with respect to $I$, namely $\theta_1 = \{X = a\}$ and $\theta_2 = \{X = b_1\}$. $\theta_1$ is a blocking substitution since $r(X)\theta_1 : (0.4,\mathbf{R}^+)$ yields the ground annotated atom $r(a):(0.4,\mathbf{R}^+)$, which is satisfied by $I$. To obtain $\mathbf{tr}(C,I)$, the constraint $\Xi_1$ obtained from $\mathbf{not}\,(r(X):(0.4,\mathbf{R}^+))$ is

$$(X \neq a)\,\&\,(X \neq b_1).$$

For the other negative atom $\mathbf{not}\, p(X,Y):(0.7,\mathbf{R}^+)$, observe that any substitution that replaces $X$ by a constant other than $a$ makes the p-atom false in $I$. This is because $I$ assigns $(0.5,\mathbf{R}^+)$ to such an atom, which is not larger than the function $(0.7,\mathbf{R}^+)$ denoted by the annotation. Any other substitution that replaces $X$ by $a$ however, makes the p-atom satisfied in $I$. As the determination of whether an instance of $p(X,Y):(0.7,\mathbf{R}^+)$ depends solely on the substitution made on the first argument $X$, it follows that the most general blocking substitution in this case is $\{X = a\}$. The constraint $\Xi_2$ derived from $\mathbf{not}\, p(X,Y):(0.7,\mathbf{R}^+)$ is hence $X \neq a$.

The conjunction $\Xi_1 \& \Xi_2$ reduces to $\Xi_1$ since $\Xi_2$ is subsumed by $\Xi_1$. Therefore, the transformation $\mathbf{tr}(C,I)$ is the $\mathbf{not}$-free constrained clause

$$q(X,Y):(1,\mathbf{R}^+) \quad \leftarrow \quad \Xi\,\&\,\Xi_1 \,\|\, s(X,Y):(0.7,\mathbf{R}^+).$$

∎

**Example.** Suppose we have the constrained clause

$$\begin{aligned} q(X,Y):(\frac{v_1+v_2}{2},\mathbf{R}^+) \quad \leftarrow \quad & \Xi \,\| \\ & \mathbf{not}\, s(X,Y):(v_1,\mathbf{R}^+)\,\& \\ & \mathbf{not}\, r(X,Y):(v_2,\mathbf{R}^+). \end{aligned}$$

Let our underlying datalog language contain constants $a, b, c_1, \ldots, c_n$, and let $I$ be the interpretation that assigns $(0.7,\mathbf{R}^+)$ to each ground atom of the form $s(a,-)$ (i.e. the second argument of $s$ may be any constant), and $(0, \mathbf{R}^+)$ to all other ground atoms of the form $s(-,-)$. In addition, $I$ assigns $(0.4, \mathbf{R}^+)$ to all ground atoms of the form $r(b,-)$, and $(0, \mathbf{R}^+)$ to all other ground atoms of the form $r(-,-)$. Finally, let $I$ assign $(0, \mathbf{R}^+)$ to all ground atoms of the form $q(-,-)$.

The blocking set $\mathbf{mgb}(s(X,Y) : (v_1, \mathbf{R}^+))$ with respect to $I$ contains $n+2$ substitutions $\rho, \rho_0, \rho_1, ..., \rho_n$, which satisfy:

$$
\begin{array}{ll}
\rho^o = \{X = a\} & \rho^a = \{v_1 = 0.7\} \\
\rho_0^o = \{X = b\} & \rho_0^a = \{v_1 = 0\} \\
\rho_1^o = \{X = c_1\} & \rho_1^a = \{v_1 = 0\} \\
 & \ldots \\
\rho_n^o = \{X = c_n\} & \rho_n^a = \{v_1 = 0\}
\end{array}
$$

The blocking set $\mathbf{mgexb}(r(X,Y) : (v_2, \mathbf{R}^+))$ with respect to $I$ also contains $n+2$ substitutions, $\gamma, \gamma_0, \gamma_1, ..., \gamma_n$, that satisfy:

$$
\begin{array}{ll}
\gamma^o = \{X = b\} & \gamma^a = \{v_2 = 0.4\} \\
\gamma_0^o = \{X = a\} & \gamma_0^a = \{v_2 = 0\} \\
\gamma_1^o = \{X = c_1\} & \gamma_1^a = \{v_2 = 0\} \\
 & \ldots \\
\gamma_n^o = \{X = c_n\} & \gamma_n^a = \{v_2 = 0\}
\end{array}
$$

Let $\Xi_1'$ be the conjunction of the constraints

$$
\begin{array}{l}
(X \neq a) \vee (v_1, \mathbf{R}^+) \not\preceq (0.7, \mathbf{R}^+) \\
(X \neq b) \vee (v_1, \mathbf{R}^+) \not\preceq (0, \mathbf{R}^+) \\
(X \neq c_1) \vee (v_1, \mathbf{R}^+) \not\preceq (0, \mathbf{R}^+) \\
\qquad \ldots \\
(X \neq c_n) \vee (v_1, \mathbf{R}^+) \not\preceq (0, \mathbf{R}^+),
\end{array}
$$

and let $\Xi_2'$ be the conjunction of the constraints

$$
\begin{array}{l}
(X \neq b) \vee (v_2, \mathbf{R}^+) \not\preceq (0.4, \mathbf{R}^+) \\
(X \neq a) \vee (v_2, \mathbf{R}^+) \not\preceq (0, \mathbf{R}^+) \\
(X \neq c_1) \vee (v_2, \mathbf{R}^+) \not\preceq (0, \mathbf{R}^+) \\
\qquad \ldots \\
(X \neq c_n) \vee (v_2, \mathbf{R}^+) \not\preceq (0, \mathbf{R}^+).
\end{array}
$$

The constraint $\Xi_1'$ is obtained according to the definition of transformation, using each substitution $\rho_i \in \mathbf{mgb}(s(X,Y) : (v_1, \mathbf{R}^+))$. Similarly, the constraint $\Xi_2'$ is obtained using each substitution

$\gamma_j \in \mathbf{mgb}(r(X,Y) : (v_2, \mathbf{R}^+))$. Then the transformation of the initial constrained clause with respect to interpretation $I$ leads to the **not**-free constrained clause

$$q(X,Y) : (\frac{v_1 + v_2}{2}, \mathbf{R}^+) \quad \leftarrow \quad \Xi \,\&\, \Xi_1' \& \Xi_2'$$

∎

Suppose $I$ is a $\Sigma$-interpretation, and **HKB** is a hybrid knowledge base. We associate with **HKB**, an operator, denoted $\mathbf{F}_{\mathbf{HKB}}$, that maps $\Sigma$-interpretations to $\Sigma$-interpretations as follows:

$$\mathbf{F}_{\mathbf{HKB}}(I) \quad = \quad lfp(\mathbf{T}_{\mathbf{tr}(\mathbf{HKB},I)}).$$

Recall here that $\mathbf{tr}(\mathbf{HKB}, I)$ is the transformation of $HKB$ with respect to the interpretation $I$.

**Lemma 3** Suppose $I_1 \preceq I_2$ and $C$ is the constrained clause

$$\begin{aligned} A : (u_0, t_0) \quad \leftarrow \quad & \Xi \,||\, B_1 : (u_1, t_1) \,\&\, \ldots \,\&\, B_n : (u_n, t_n) \,\&\, \\ & \mathbf{not}\,(D_1 : (v_1, s_1)) \,\&\, \ldots \,\&\, \mathbf{not}\,(D_m : (v_m, s_m)). \end{aligned}$$

Then $\mathbf{tr}(C, I_2) \models \mathbf{tr}(C, I_1)$. ∎

Suppose $I$ is a $\Sigma$-interpretation, and **HKB** is a hybrid knowledge base. $I$ is said to be a *hybrid stable set* with respect to **HKB** iff $I = \mathbf{F}_{\mathbf{HKB}}(I)$.

**Theorem 4** If $I$ is a hybrid stable set with respect to **HKB**, then $I$ is a minimal model, with respect to $\preceq$, of **HKB**. ∎

It is important to observe that the converse of the statements in the above theorem are not true, i.e. there are minimal models of **HKB** that are not hybrid stable sets and there are minimal fixpoints of $T_{\mathbf{HKB}}$ that are not hybrid stable sets.

### 4.3 Well-Founded Semantics for Hybrid Knowledge Bases

Hybrid stable sets do not always exist. There are hybrid knowledge bases that are "meaningless" due to the absence of any hybrid stable sets. A simple example is the hybrid knowledge base containing the clauses:

$$\begin{aligned} p : (1, \{1\}) \quad &\leftarrow \quad \mathbf{not}\,(p : (1, \{1\})). \\ q : (1, \{1\}) \quad &\leftarrow \end{aligned}$$

It is easy to verify that this hybrid knowledge base has no hybrid stable set. However, this non-existence of stable models is due to the first clause.

The well-founded semantics for logic programs [32] was designed to address the above problem where a single clause may corrupt the meaning of other, unrelated clauses. For completeness, we briefly consider in this section how the well-founded semantics may be applied to hybrid knowledge bases.

**Theorem 5** If $I_1 \preceq I_2$ are $\Sigma$-interpretations, then $\mathbf{F_{HKB}}(I_2) \preceq \mathbf{F_{HKB}}(I_1)$, i.e. $\mathbf{F_{HKB}}$ is a anti-monotone map. ∎

**Corollary.** $\mathbf{F^2_{HKB}}$, the operator that applies $\mathbf{F_{HKB}}$ twice is monotone. ∎

The *hybrid well-founded semantics* is given by the following.

1. $\mathbf{HKB} \models_{\mathbf{wfs}} A : \mu$, where $\mu$ is a c-annotation, iff $lfp(F_P^2) \models A : \mu$.
2. $\mathbf{HKB} \models_{\mathbf{wfs}} \mathbf{not}\,(A : \mu)$, where $\mu$ is a c-annotation, iff $gfp(F_P^2) \not\models A : \mu$.

In the above example, it is straightforward to verify that $\mathbf{HKB} \models_{\mathbf{wfs}} q : (1, \{1\})$, $\mathbf{HKB} \not\models_{\mathbf{wfs}} p : (1, \{1\})$, and $\mathbf{HKB} \not\models_{\mathbf{wfs}} \mathbf{not}\, p : (1, \{1\})$.

**Theorem 6** The hybrid well-founded semantics is a sound approximation of the hybrid stable semantics in the sense that:

1. If $\mathbf{HKB} \models_{\mathbf{wfs}} A : \mu$, then every hybrid stable model $M$ satisfies $A : \mu$.
2. If $\mathbf{HKB} \models_{\mathbf{wfs}} \mathbf{not}\,(A : \mu)$, then every hybrid stable model $M$ satisfies $\neg(A : \mu)$. ∎

# 5 An Example: Mobile Robots

In this section, we describe an example of how the hybrid knowledge based framework may be used for intelligent reasoning by robots. Though the example is small, it is illustrative of the flexibility that hybrid knowledge bases presents. In particular, if we view data structures that are used for storing various information as constraint domains, then hybrid knowledge bases enables an integration of distributed databases, where information may be stored in diverse forms. The important underlying assumption that enables all of the various forms to be integrated via hybrid knowledge bases is that the information stored provides a set of predicates and functions, describing the desired behaviors and relations of interest. In this regard, the specific form that the information take, whether it be quad-trees, or some other class of data structure, is not important.

For this toy example, the robot's workspace is a $(4 \times 4)$ grid. We assume that the robot is mobile from one grid coordinate to another. Figure 1 shows the workspace of the robot – there are seven objects $A, B, C, D, E, F$ and $G$ present in workspace. Their locations are shown in the figure. This

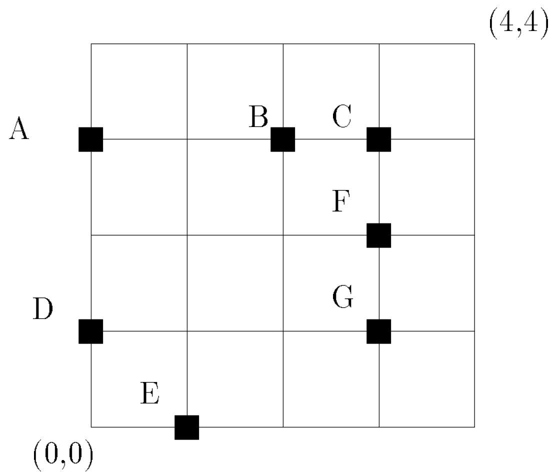


Figure 1: Two-Dimensional Robot Workspace

spatial information may be represented as a spatial database. The typical data structures that one might use for representing such spatial information includes point quadtrees, MX-quadtrees, PM-quadtrees, and R-trees [28]. Figure 2 shows a point quadtree for storing the above seven objects, assuming the insertion order $A, B, C, D, E, F, G$.

We assume a second database (which we will refer to simply as the relational database) that contains information about objects including those in the workspace. In particular, the database contains the relations:

COLOR having the form

$$\langle \mathrm{obj\_name}, \mathrm{color} \rangle,$$

SIZE having the form

$$\langle \mathrm{obj\_name}, \mathrm{height(cm.)}, \mathrm{length(cm.)}, \mathrm{width(cm.)}, \mathrm{weight(kg.)} \rangle,$$

TEMP having the form

$$\langle \mathrm{obj\_name}, \mathrm{temperature} \rangle.$$

It is important to note that the spatial database and the relations in the relational database represent *a priori* information. It therefore is possible that one or both may contain errors, and may be incomplete.

At any given point in time, the robot's sensors generate information on the above relations. The hybrid knowledge base expresses what to do given the fact that:

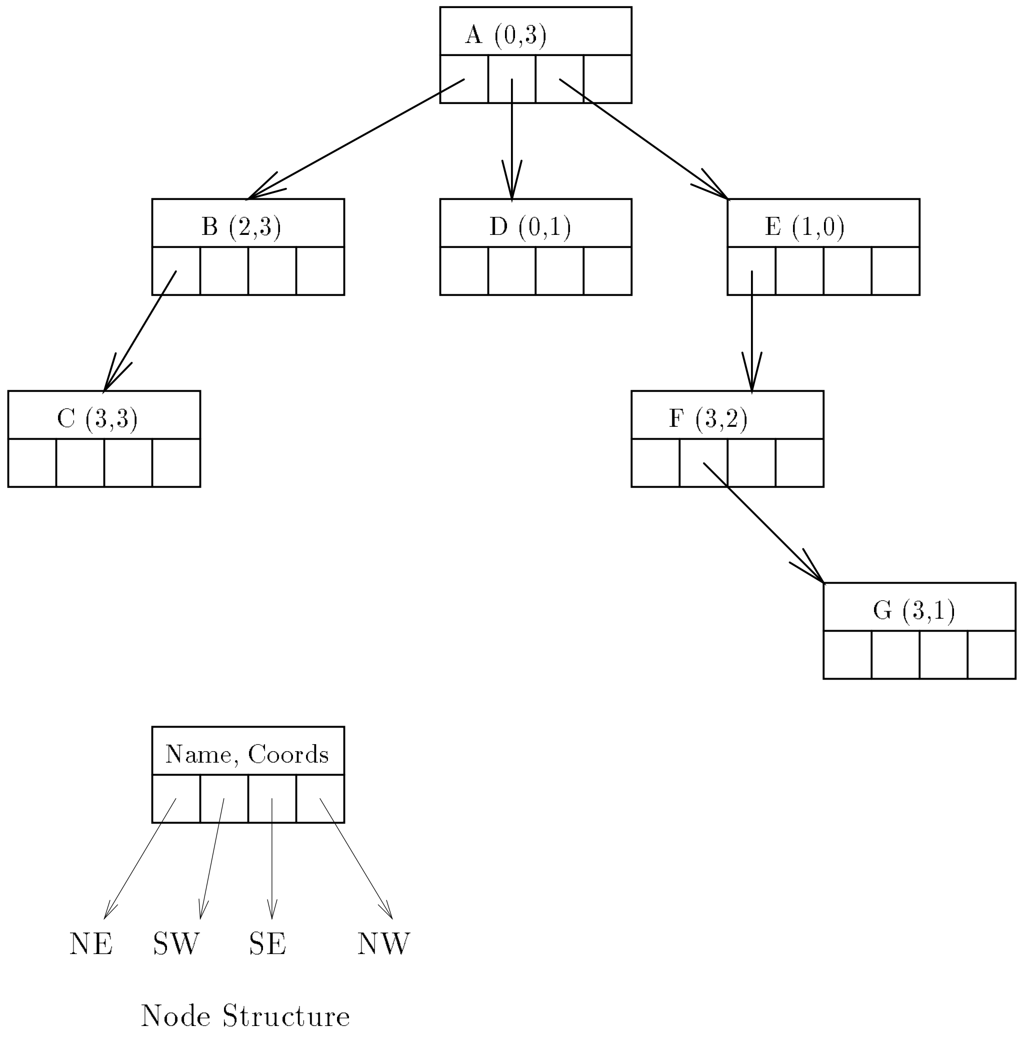


Figure 2: Quadtree for Robot Example

- (Uncertainty) sensor information may be unreliable, and
- (Conflict Resolution and Updating) the sensor information may contradict information in the *a priori* databases,
- (Temporal Prediction) the history of changes in the environment may enable us to make predictions about future states of the workspace and base our actions on them (e.g. if object E is at location $(1,0)$ at time 0, at $(1,1)$ at time 1, and at $(1,2)$ at time 2, we may predict that it will be at location $(1,3)$ at time 3),
- (Nonmonotonicity) it may turn out that predictions such as the one above allow a conclusion to be drawn that must later be retracted (e.g. if the robot moved to location $(2,2)$ at time 3).

The robot may have the following rules that specify what kind of action to take.

1. "If the robot is at location $(X,Y)$ (and the positional uncertainty that the robot is at this location is at least 50%) and there is an object $O$ within a distance of 2 units from the robot that has a temperature over 100 degrees, then aim a cooling spray at object $O$." This can be represented as:

$$\begin{aligned} cool\_spray(O,X,Y):(1,T) \quad \leftarrow \quad & (X_1,Y_1)\ \underline{\mathsf{in}}\ \mathtt{RANGE}((X,Y),2)\ \& \\ & \underline{\mathsf{at}}(O,X_1,Y_1)\ \& \\ & temp(O,Temp)\ \&\ Temp \geq 100 \ \| \\ & robot\_at(X,Y):(0.5,T). \end{aligned}$$

   This rule encompasses the aforementioned ideas. It contains:

   (a) **use of uncertainty:** this is the positional uncertainty of the robot (is the robot where it thinks it is ?),

   (b) **use of time:** the robot's location changes with time,

   (c) **simultaneous use of three constraint domains:** spatial databases are accessed when evaluating the `RANGE` subquery and the <u>in</u> relation, the relational database is used to evaluate the "temp" relation, and a real number constraint domain is used to evaluate the "$Temp \geq 100$" constraint.

2. Temporal prediction can also be expressed. The statement: "If the robot is at location $(X,Y)$ with certainty $V$ at time $t$ and at location $(X,Y+1)$ with certainty $V$ at time $(t+1)$, then

the robot will be at location $(X, Y+2)$ with certainty $V$ at time $(t+2)$" may be expressed in our framework as follows:

$$\begin{aligned} robot\_at(X,Y+2):(V,\{T+2\}) \quad \leftarrow \quad & robot\_at(X,Y):(V,\{T\})\,\& \\ & robot\_at(X,Y+1):(V,\{T+1\}). \end{aligned}$$

3. To suppress using the above rule if the robot behaves in a manner different from the prediction, we may modify it as:

$$\begin{aligned} robot\_at(X,Y+2):(V,\{T+2\}) \quad \leftarrow \quad & (X',Y') \neq (X,Y+2)\,\| \\ & robot\_at(X,Y):(V,\{T\})\,\& \\ & robot\_at(X,Y+1):(V,\{T+1\})\& \\ & \mathbf{not}\; robot\_at(X',Y'):(1,\{T+2\}). \end{aligned}$$

   The rule says that if we have *explicit* information saying that the robot is at a location other than $(X, Y+2)$ at time $T+2$, then the (non-monotonically derived) conclusion $robot\_at(X, Y+2):(V,\{T+2\})$ is suppressed.

4. Finally, we may wish to specify rules for conflict resolution between sensed and a priori information. For example, suppose the robot is functioning on a production shop floor. It is common for objects to be fixed to the ground (i.e. they are immobile). If $O$ is one such an immobile object, then a sensor reading placing object $O$ at a location other than that specified by the spatial database can be suppressed. Such a rule can be expressed as the constrained rule:

$$\begin{aligned} loc(O,X,Y):(1,\{T\}) \quad \leftarrow \quad & \underline{\mathsf{at}}(O,X,Y)\,\| \\ & sensor\_says(O,X',Y'):(V,\{T\})\,\& \\ & immobile(O):(1,\mathbf{R}^{+}). \end{aligned}$$

   A different rule for movable objects may give priority to the sensor reading rather than to the location specified by the *a priori* spatial database.

To reiterate, functions such as **RANGE**, and relations like in are provided over the constraint domain of spatial databases. Having such a set of functions and relations is a key to being able to integrate multiple databases using hybrid knowledge bases. Certainly it is not unreasonable to assume the existence of such functions and relations. Such a set of relations can be characterized as an abstract data type that satisfies a particular set of axioms.

# 6 Related Work

The work described in this paper may be viewed as an integration of three major extensions for logic programming. These are:

- Constraint Logic Programming (CLP), due to Jaffar and Lassez [11, 12]
- Non-Monotone Logic Programming (NMLP), due to Gelfond and Lifschitz [6], and
- Generalized Annotated Programs (GAP), due to Kifer and Subrahmanian [15, 16].

The resulting framework, called hybrid knowledge bases, provides a powerful tool for building sophisticated deductive knowledge bases. A hybrid knowledge base is capable of accessing a wide variety of heterogeneous databases and data sources. In many applications domains, these heterogeneous pieces of data include diverse data structures stored in different ways depending on the intended mode of access. The hybrid knowledge base handles incomplete and uncertain information, along with time dependent information. Figure rel-fig shows how hybrid knowledge bases extend the CLP, NMLP and GAP frameworks. Constraint logic programming provides a mechanism for reasoning about different domains within logic programming. Hybrid knowledge bases extend CLPs to include uncertainty, time, and non-monotonicity. With respect to the non-monotonic semantics given by stable models, hybrid knowledge bases show that the semantics may be viewed equivalently as constraint solving over the domain of substitutions. Lastly, in addition to adding constraints and non-monotonicity, hybrid knowledge bases modifies the approach of generalized annotated programs whereby the annotations are not taken directly from the lattice of truth values. This allows for flexibility in working with the annotations, as discussed in the end of Section 2.2, as well as simplifies the proof procedure required for positive knowledge bases, demonstrated in Section 3.2. A very important observation to point out is that the integration of the three formalisms is not a loose combination of three disjoint theories. There is a tight coupling of these formalisms where the consistent thread through which all three are linked is constraint solving.

Stickel [30] has introduced *theory resolution* where resolution may be integrated with a number of different theories. The work is set in the context of general theorem proving and it does not consider non-monotonic or temporal reasoning. Lehmann and his colleagues [23, 18] have studied methods for integrating time and probabilities. Though this work is related to our work, we have not included probabilities in our treatment of knowledge. In [27], a probabilistic theory of annotated logic programming is introduced. Extending hybrid knowledge bases with such a capability is a topic for future work. Frisch [5] provides a framework for hybrid reasoning, but only in the monotonic case. Moreover, it is not clear how Frisch's system can handle time and uncertainty.

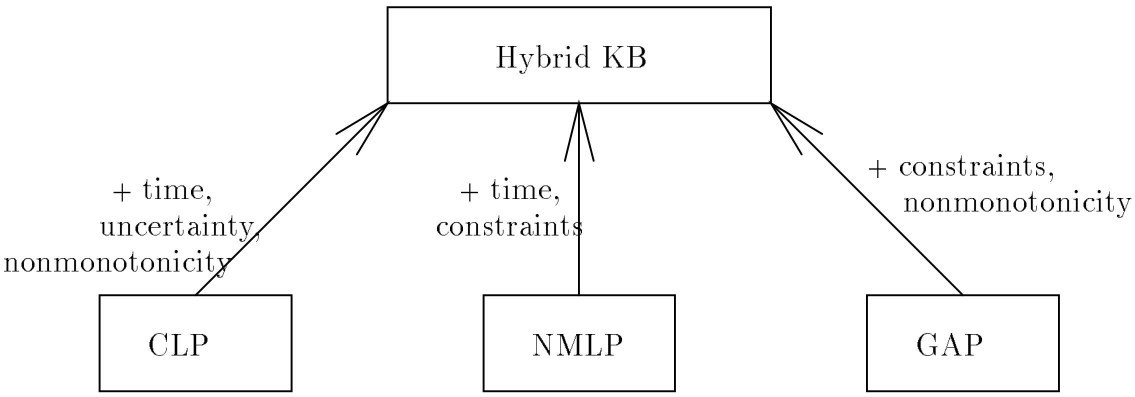


Figure 3: The Hybrid Knowledge Bases Framework

A great deal of work has been done in *multidatabase* systems and *interoperable* database systems [37, 38, 19, 17, 29, 22]. Most of this work combines standard relational databases (no deductive capabilities). The development of a semantical foundation for multidatabases has been studied by Grant et. al. [7] who introduced a calculus and an algebra for integrating information from multiple databases. This calculus extends the standard relational calculus. Our research addresses a different topic – that of integrating multiple databases containing (possibly) different schematic representations (in terms of diverse data structures), as well as inconsistencies, uncertainty, and possibly even temporal information. Krishnamurthy et. al. [19] argue that a higher-order form of Horn clause logic is better than relational languages for schema integration. A language, called IDL, has been developed to express various queries spanning multiple relational databases. In contrast, our language will allow accessing multiple databases (deductive and otherwise) including those containing uncertainties. Lefebvre et. al. [22] have a similar aim – they base their language on F-logic due to Kifer et. al. [14].

A large-scale development effort for controlling a real mobile robot using hybrid knowledge bases is currently under way as a joint effort with the National Institute of Standards and Technology in Gaithersburg, Maryland. Preliminary details are contained in [10].

**Acknowledgement.** We are thankful for the useful comments provided by two anonymous referees. We have also benefited from comments by Jeff Remmel.

## References


[1] K. Apt, H. A. Blair, A. Walker, Towards a Theory of Declarative Knowledge, *Foundations of Deductive Databases and Logic Programming* (J. Minker ed.), Morgan-Kaufmann, 89 – 148, 1988.

[2] S. Adalı, V.S. Subrahmanian, Amalgamating knowledge bases III: Algorithms, data structures and query processing, Technical Report CS-TR-3124, Computer Science Department, University of Maryland, 1993.

[3] J. Benton, V.S. Subrahmanian, Hybrid Knowledge Bases for Missile Siting Problems, *Proceedings of the IEEE Conference on Applications of Artificial Intelligence*, 1994.

[4] C. Bell, A. Nerode, R. Ng, V.S. Subrahmanian, Implementing Deductive Databases by Linear Programming, *Proceedings of the ACM SIGACT/SIGART/SIGMOD Symposium on Principles of Database Systems*, 283–292, 1992.

[5] A. Frisch, The Substitutional Framework for Sorted Deduction, *Artificial Intelligence*, 49, 161–198, 1991.

[6] M. Gelfond, V. Lifschitz, The Stable Model Semantics for Logic Programming, *Proceedings of the 5th International Conference and Symposium on Logic Programming* (R. A. Kowalski and K. A. Bowen eds.), MIT Press, 1070–1080, 1988.

[7] J. Grant, W. Litwin, N. Roussopoulos, T. Sellis, An Algebra and Calculus for Relational Multidatabase Systems, *Proceedings of the First International Workshop on Interoperability in Multidatabase Systems*, IEEE Computer Society Press, 118–124, 1991.

[8] R. Hähnle, Towards an Efficient Tableau Rules for Multiple-valued Logics, *Proceedings of the Workshop on Computer Science Logic*, Springer, 248–260, 1990.

[9] R. Hähnle, Uniform Notation of Tableau Rules for Multiple-valued Logics, *Proceedings of the International Symposium on Multiple-Valued Logic*, 26–29, 1991.

[10] J. Horst, E. W. Kent, H. Rifky, V.S. Subrahmanian, Intelligent, Real-Time Robotic Reasoning with Hybrid Knowledge Bases, *Proceedings of the 4th International Workshop on Pattern Recognition in Practice*, North-Holland Elsevier, 1994.

[11] J. Jaffar, J.L. Lassez, Constraint Logic Programming, *Proceedings of the ACM Principles of Programming Languages*, 111–119, 1987.

[12] J. Jaffar, S. Michaylov, P. Stuckey, R. Yap, The CLP($\Re$) Language and System, *ACM Transactions on Programming Languages and Systems*, 339–395, 1992.

[13] M. Kifer, E. Lozinskii, *A Logic for Reasoning with Inconsistency*, Journal of Automated Reasoning, 9, 179–215, 1992.

[14] M. Kifer, G. Lausen, J. Wu, Logical Foundations of Object-Oriented and Frame-Based Languages, Technical Report 90/14, SUNY at Stonybrook, 1990.

[15] M. Kifer, V.S. Subrahmanian, On the Expressive Power of Annotated Logics, *Proceedings of the North American Conference on Logic Programming*, MIT Press, 1069–1089, 1989.

[16] M. Kifer, V. S. Subrahmanian, Theory of Generalized Annotated Logic Programming and its Applications, *Journal of Logic Programming*, 12, 4, 335–368, 1992.

[17] W. Kim, J. Seo, Classifying Schematic and Data Heterogeneity in Multidatabase Systems, *IEEE Computer*, December 1991.

[18] S. Kraus, D. Lehmann, Decision procedures for time and chance, *Proceedings of the IEEE Symposium on the Foundation of Computer Science*, 202–209, 1983.

[19] R. Krishnamurthy, W. Litwin, W. Kent, Language Features for Interoperability of Databases with Schematic Discrepancies, *Proceedings of the ACM SIGMOD Conference on Management of Data*, 1991.

[20] S. Leach, J. Lu, Computing Annotated Logic Programs, *Proceedings of the International Conference on Logic Programming*, MIT Press, 257–271, 1994.

[21] S. Leach, J. Lu, Query Processing in Annotated Logic Programming: Theory and Implementation, *Journal of Intelligent Information Systems*, accepted to appear.

[22] A. Lefebvre, P. Bernus, R. Topor, Querying Heterogeneous Databases: A Case Study, draft manuscript, 1992.

[23] D. Lehmann, S. Shelah, Reasoning with Time and Chance, *Information and Control*, 53, 3, 165–198, 1982.

[24] J. W. Lloyd, *Foundations of Logic Programming*, 2nd edition, Springer, 1987.

[25] J. Lu, A. Nerode, J. Remmel, V.S. Subrahmanian Towards a Theory of Hybrid Knowledge Bases, TR93-14, Mathematical Sciences Institute, Cornell University, 1993.

[26] N.V. Murray, E. Rosenthal, Signed Formulas: A Liftable Meta-Logic for Multiple-Valued Logics, *Proceedings of International Symposium on Methodologies for Intelligent Systems*, 275–284, 1993.

[27] R. Ng, V.S. Subrahmanian, Probabilistic Logic Programming, *Information and Computation*, 101, 2, 150–201, 1992.

[28] H. Samet, *The Design and Analysis of Spatial Data Structures*, Addison-Wesley, 1989.

[29] A. Sheth and J. Larson, Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases, *ACM Computing Surveys*, 22, 3, 183–236, 1990.

[30] M. Stickel, Automated Deduction by Theory Resolution, *Journal of Automated Reasoning*, 1, 333–355, 1985.

[31] V.S. Subrahmanian, Amalgamating Knowledge Bases, *ACM Transactions on Database Systems*, 19, 2, 291–331, 1994.

[32] A. van Gelder, K. Ross, J. Schlipf, Unfounded Sets and Well-founded Semantics for General Logic Programs, *Proceedings of the Symposium on Principles of Database Systems*, 221–230, 1988.

[33] A. van Gelder, The Alternating Fixpoint of Logic Programs with Negation, *Proceedings of the ACM Symposium on Principles of Database Systems*, 1 – 10, 1989.

[34] T.J. Weigert, J-P. Tsai, X. Liu, Fuzzy Operator Logic and Fuzzy Resolution, *Journal of Automated Reasoning*, 10, 59–78, 1993.

[35] G. Wiederhold, Mediators in the Architecture of Future Information Systems, *IEEE Computer*, 38–49, March 1992.

[36] G. Wiederhold, Intelligent Integration of Information, *Proceedings of the ACM SIGMOD Conference on Management of Data*, 434–437, 1993.

[37] G. Wiederhold, S. Jajodia, W. Litwin, Dealing with granularity of time in temporal databases, *Proceedings of the Nordic Conference on Advanced Information Systems Engineering* (R. Anderson et al. eds.), Springer, 124–140, 1991,

[38] G Wiederhold, S. Jajodia, and W. Litwin, Integrating temporal data in a heterogeneous environment, *Temporal Databases*, Benjamin Cummings, 1993.