Foundations of Indefinite Constraint Databases

Manolis Koubarakis*

IC-Parc
Imperial College
London SW7 2AZ
United Kingdom
msk@doc.ic.ac.uk

Abstract. We lay the foundations of a theory of constraint databases with indefinite information based on the relational model. We develop the scheme of indefinite $\mathcal{L}$-constraint databases where $\mathcal{L}$, the parameter, is a first-order constraint language. This scheme extends the proposal of Kanellakis, Kuper and Revesz to include indefinite information in the style of Imielinski and Lipski. We propose declarative and procedural query languages for the new scheme and study the semantics of query evaluation.

1 Introduction

In this paper we lay the foundations of a theory of *indefinite constraint databases* based on the relational model [Mai83]. As a starting point of our investigation, we take the model of constraint databases proposed in [KKR90]. This model is useful for the representation of *unrestricted* (i.e., finite or infinite) *definite* information. However, indefinite information is also important in many applications e.g., planning and scheduling, medical expert systems, geographical information systems and natural language processing systems. Motivated by these practical considerations, we develop the model of *indefinite constraint databases* which allows the representation of *definite, indefinite, finite* and *infinite* information in a single unifying framework.

Our contributions to the theory of constraint databases can be summarized as follows:

– We develop the scheme of *indefinite $\mathcal{L}$-constraint databases* where $\mathcal{L}$, the parameter, is a first-order *constraint language*. This parameterized model extends the scheme of [KKR90] to include indefinite information in the style of [IL84, Gra89] (section 3).
– We propose *modal relational calculus* with $\mathcal{L}$-constraints as a declarative query languages for indefinite $\mathcal{L}$-constraint databases (section 4). We also

* Most of this research was carried out while the author was with the Computer Science Division, Dept. of Electrical and Computer Engineering, National Technical University of Athens, Greece.

propose a procedural query language: the *modal $\mathcal{L}$-constraint algebra* (section 5). The introduction of modal operators gives us the ability to ask "possibility" and "certainty" queries - a very useful feature in any indefinite information setting.

- We show that expressions of modal relational calculus with $\mathcal{L}$-constraints can be evaluated *bottom-up in closed form* on indefinite $\mathcal{L}$-constraint databases. This is a direct consequence of the fact that every expression of modal relational calculus with $\mathcal{L}$-constraints has an equivalent expression in modal $\mathcal{L}$-constraint algebra (section 7). This result could be the first step in developing optimization techniques for $\mathcal{L}$-constraint databases and indefinite $\mathcal{L}$-constraint databases. Our analysis is carried out in an abstract setting and therefore subsumes previous work on specific classes of constraints [KKR90, KSW90, Kou93, Kou94c, KG94, PVdBVG94].

This paper is organized as follows. The next section presents some examples of constraint languages and defines the relevant abstract concepts. In section 3 we present the scheme of indefinite $\mathcal{L}$-constraint databases. In sections 4 and 5 we discuss the modal relational calculus with $\mathcal{L}$-constraints and the modal $\mathcal{L}$-constraint algebra. In section 6 we present several results concerning algebraic query evaluation in $\mathcal{L}$-constraint databases and indefinite $\mathcal{L}$-constraint databases. In section 7 we discuss the translation of expressions of modal relational calculus with $\mathcal{L}$-constraints into expressions of modal $\mathcal{L}$-constraint algebra. Finally, section 8 presents related work and future research. All proofs are omitted. The interested reader can consult [Kou94a].

2 Constraint Languages

In this paper we consider many-sorted languages, structures and theories [End72]. Every language $\mathcal{L}$ will be interpreted over a *fixed* structure, called the *intended structure*, which will usually be denoted by $\mathbf{M}_{\mathcal{L}}$. If $\mathbf{M}$ is a structure then $Th(\mathbf{M})$ will denote the theory of $\mathbf{M}$ i.e., the set of sentences which are true in $\mathbf{M}$. For every language $\mathcal{L}$, we will distinguish a class of quantifier free formulas called *$\mathcal{L}$-constraints*. The atomic formulas of $\mathcal{L}$ will be included in the class of $\mathcal{L}$-constraints. There will also be two distinguished $\mathcal{L}$-constraints *true* and *false* with obvious semantics. Similar assumptions have been made in [Mah93] in the contex of the CLP scheme. A set of $\mathcal{L}$-constraints will be the algebraic counterpart of the logical conjunction of its members. Thus we will freely mix the terms "set of $\mathcal{L}$-constraints" and "conjunction of $\mathcal{L}$-constraints". We will assume that the reader is familiar with the notions of *solution, consistency* and *equivalence* of sets of constraints [Mah93].

Let us now give some examples of constraint languages.

Example 1. The language *ECL* (*E*quality *C*onstraint *L*anguage) with predicate symbols $=$, $\neq$ and an infinite number of constants has been defined in [KKR90]. The intended structure for this language interprets $=$ as equality, $\neq$ as non-equality and constants as "themselves". An *ECL-constraint* is an ECL formula of the form $x_1 = x_2$ or $x_1 \neq x_2$ where x_1, x_2 are variables or constants. ECL has been used by [KKR90] for the development of an extended relational model based on ECL-constraints.

We now present a language for expressing *temporal constraints*.

Example 2. The language *dePCL* (*d*ense *P*oint *C*onstraint *L*anguage) allows us to make stamements about points in dense time. dePCL is a first-order language with equality and the following set of non-logical symbols: the set of rational numerals, function symbol $-$ of arity 2 and predicate symbol $<$ of arity 2. The *terms* and *atomic formulas* of dePCL are defined as follows. Constants and variables are terms. If t_1 and t_2 are variables or constants then $t_1 - t_2$ is a term. An *atomic formula* of dePCL is a formula of the form $t \sim c$ or $c \sim t$ where $\sim$ is $<$ or $=$ and t is a term.

The intended structure for dePCL is $\mathbf{Q}$. $\mathbf{Q}$ interprets each rational numeral by its corresponding rational number, function symbol $-$ by the subtraction operation over the rationals and $<$ by the relation "less than". The theory $Th(\mathbf{Q})$ is a subtheory of real addition with order [Rab77].

A *dePCL-constraint* is a dePCL formula of the form $t \sim c$ where t is a term, c is a constant and $\sim$ is $=$, $<$, $>$, $\leq$ or $\geq$. For example, the formulas $p_1 < p_2$, $p_3 - p_4 \geq 15$, $p_3 = 5/4$ are dePCL-constraints.

Example 3. Let us also consider the many-sorted language ECL+dePCL which is the union of ECL and dePCL. The sorts of ECL+dePCL are $\mathcal{D}$ (for the infinite set of constants of ECL) and $\mathcal{Q}$ (for the rational numerals of dePCL). The symbols of ECL+dePCL are interpreted by the many-sorted structure which is the union of the intended structures for ECL and dePCL.

Let us now define the concept of *variable elimination*.[2]

Definition 1. Let $\mathcal{L}$ be a many-sorted first-order language. The class of $\mathcal{L}$-constraints *admits variable elimination* iff for every boolean combination ϕ of $\mathcal{L}$-constraints in variables $\overline{x}$, and every vector of variables $\overline{z} \subseteq \overline{x}$, there exists a disjunction ϕ' of conjunctions of $\mathcal{L}$-constraints in variables $\overline{x} \setminus \overline{z}$ such that

1. If $\overline{x}^0$ is a solution of ϕ then $\overline{x}^0 \setminus \overline{z}^0$ is a solution of ϕ'.
2. If $\overline{x}^0 \setminus \overline{z}^0$ is a solution of ϕ' then this solution can be extended to a solution $\overline{x}^0$ of ϕ.

[2] Notation: The vector of symbols $(o_1, \ldots, o_n)$ will be denoted by $\overline{o}$. The natural number n will be called the *size* of $\overline{o}$ and will be denoted by $|\overline{o}|$. This notation will be used for vectors of variables but also for vectors of domain elements. Variables will be denoted by x, y, z, t etc. and vectors of variables by $\overline{x}, \overline{y}, \overline{z}, \overline{t}$ etc. If $\overline{x}$ and $\overline{y}$ are vectors of variables then $\overline{x} \setminus \overline{y}$ will denote the vector obtained from $\overline{x}$ by deleting the variables in $\overline{y}$. If $\overline{x}$ is a vector of variables then $\overline{x}^0$ will be a vector of constants of the same size.

Some people might find the above definition overly strong. But requiring ϕ' to be just a boolean combination of $\mathcal{L}$-constraints would turn out to be unsatisfactory for the database models discussed in section 3. The reason is very simple: when we eliminate variables, we would have to deal with negations of $\mathcal{L}$-constraints. Similar arguments and definitions appear in [Stu91].

The following definition will be useful in the forthcoming sections.

Definition 2. Let $\mathcal{L}$ be a many-sorted first-order language. The class of $\mathcal{L}$-constraints is *weakly closed under negation* if the negation of every $\mathcal{L}$-constraint is equivalent to a disjunction of $\mathcal{L}$-constraints.

In the rest of this paper we will only be interested in constraints which admit variable elimination and are weakly closed under negation. Many interesting classes of constraints fall under this category. The following proposition shows that this is also the case for the constraint classes defined in this section.

Proposition 3. *The classes of ECL-constraints, dePCL-constraints and ECL+dePCL-constraints admit variable elimination and are weakly closed under negation.*

3 Indefinite Constraint Databases

We will now extend the $\mathcal{L}$-constraint database model of [KKR90] to account for indefinite information in the style of [IL84, Gra89]. For the rest of this section, let $\mathcal{L}$ be a many-sorted language and $\mathbf{M}_{\mathcal{L}}$ be the *intended $\mathcal{L}$-structure*. Let us also assume that the class of $\mathcal{L}$-constraints admits variable elimination and is weakly closed under negation.

For each sort $s \in sorts(\mathcal{L})$, let U_s be a countably infinite set of *attributes* of sort s. The set of all attributes, denoted by $\mathcal{U}$, is $\bigcup_{s \in sorts(\mathcal{L})} U_s$. The sort of attribute A will be denoted by $sort(A)$. With each $A \in \mathcal{U}$ we associate a set of values $dom(A) = dom(s, \mathbf{M}_{\mathcal{L}})$ called the *domain* of A.[3] A *relation scheme* R is a finite subset of $\mathcal{U}$.

We will first define $\mathbf{M}_{\mathcal{L}}$-relations which are unrestricted (i.e., finite or infinite) standard relations. $\mathbf{M}_{\mathcal{L}}$-relations are a theoretical device for giving semantics to indefinite $\mathcal{L}$-constraint relations.

Definition 4. Let R be a relation scheme. An $\mathbf{M}_{\mathcal{L}}$*-relational tuple* t over scheme R is a mapping from R to $\bigcup_{s \in sorts(\mathcal{L})} dom(s, \mathbf{M}_{\mathcal{L}})$ such that $t(A) \in dom(sort(A), \mathbf{M}_{\mathcal{L}})$. An $\mathbf{M}_{\mathcal{L}}$*-relation* r over scheme R is an *unrestricted* set of $\mathbf{M}_{\mathcal{L}}$-relational tuples over R.

For every $s \in sorts(\mathcal{L})$, we now assume the existence of two disjoint countably infinite sets of *variables*: the set of *u-variables* $UVAR^s_{\mathcal{L}}$ and the set of *e-variables* $EVAR^s_{\mathcal{L}}$. Let $UVAR_{\mathcal{L}}$ and $EVAR_{\mathcal{L}}$ denote $\bigcup_{s \in sorts(\mathcal{L})} UVAR^s_{\mathcal{L}}$ and $\bigcup_{s \in sorts(\mathcal{L})} EVAR^s_{\mathcal{L}}$ respectively. The intersection of the sets $UVAR_{\mathcal{L}}$ and $EVAR_{\mathcal{L}}$ with the domains of attributes is empty.

[3] If s is a sort and $\mathbf{M}$ is a structure then $dom(s, \mathbf{M})$ denotes the domain of s in structure $\mathbf{M}$.

Notation 3.1 U-variables will be denoted by letters of the English alphabet, usually x, y, z, t, possibly subscripted. E-variables will be denoted by letters of the Greek alphabet, usually $\omega, \lambda, \zeta, \nu$, possibly subscripted.

Definition 5. Let R be a relation scheme. An *indefinite $\mathcal{L}$-constraint tuple t* over scheme R is a mapping from $R \cup \{CON\}$ to $UVAR_{\mathcal{L}} \cup WFF(\mathcal{L})$ such that (i) $t(A) \in UVAR_{\mathcal{L}}^{sort(A)}$ for each $A \in R$, (ii) $t(A_i)$ is different than $t(A_j)$ for all distinct $A_i,\ A_j \in R$, (iii) $t(CON)$ is a conjunction of $\mathcal{L}$-constraints and (iv) the free variables of $t(CON)$ are included in $\{t(A) :\ A \in R\} \cup EVAR_{\mathcal{L}}$. $t(CON)$ is called the *local condition* of the tuple t while $t(R)$ is called the *proper part* of t.

Definition 6. Let R be a relation scheme. An *indefinite $\mathcal{L}$-constraint relation* over scheme R is a *finite* set of indefinite $\mathcal{L}$-constraint tuples over R. Each indefinite $\mathcal{L}$-constraint relation r is associated with a boolean combination of $\mathcal{L}$-constraints $G(r)$, called the *global condition* of r.

Similarly we can define database schemes, $\mathbf{M}_{\mathcal{L}}$-relational databases and indefinite $\mathcal{L}$-constraint databases [Kou94a]. Database schemes and databases will usually be denoted by $\widetilde{R}$ and $\widetilde{r}$ respectively.

The above definitions extend the model of [KKR90] by introducing *e-variables* which have the semantics of marked nulls of [IL84]. As in [Gra89], the possible values of the e-variables can be constrained by a *global condition*.

Example 4. BOOKED is an indefinite ECL+dePCL-constraint relation giving the times that rooms are booked. The first tuple says that room WP212 is booked from 1:00 to 7:00. For room WP219 the information is indefinite: it is booked from 1:00 until some time between 5:00 and 8:00. This indefinite information is captured by the *e-variable* ω and its global condition $5 \leq \omega \leq 8$. *E-variables* can be understood as being existentially quantified and their scope is the entire database. They represent values that exist but are not known precisely [IL84, Gra89]. All we know about these values is captured by the global condition. *U-variables* (e.g., x_1, x_2, t_1, t_2) can be understood as being universally quantified and their scope is the tuple in which they appear [KKR90].

BOOKED

Room	Time	CON
x_1	t_1	$x_1 = WP212,\ 1 \leq t_1 < 7$
x_2	t_2	$x_2 = WP219,\ 1 \leq t_2 < \omega$

$$G(BOOKED) :\ 5 \leq \omega \leq 8$$

3.1 Semantics

Let us first define two special kinds of valuations. An *e-valuation* in $\mathbf{M}_{\mathcal{L}}$ is a valuation whose domain is restricted to the set $EVAR_{\mathcal{L}}$. Similarly, a *u-valuation* in $\mathbf{M}_{\mathcal{L}}$ is a valuation whose domain is restricted to the set $UVAR_{\mathcal{L}}$. The symbols

$Val^e_{\mathbf{M}_\mathcal{L}}$ and $Val^u_{\mathbf{M}_\mathcal{L}}$ will denote the set of e-valuations and u-valuations in $\mathbf{M}_\mathcal{L}$ respectively. The result of applying an e-valuation v to an indefinite $\mathcal{L}$-constraint relation r over R will be denoted by $v(r)$. $v(r)$ is an $\mathcal{L}$-constraint relation over R obtained from r by substituting each e-variable ω of r by the constant symbol whose denotation in structure $\mathbf{M}_\mathcal{L}$ is $v(\omega)$. The result of applying a u-valuation of $\mathbf{M}_\mathcal{L}$ to the proper part of a tuple can be defined as follows. If t is an $\mathcal{L}$-constraint tuple on scheme R and u is a u-valuation in $\mathbf{M}_\mathcal{L}$ then $u(t)$ is an $\mathbf{M}_\mathcal{L}$-tuple over R such that for each $A \in R$, $u(t)(A) = u(t(A))$.

The semantics of an $\mathcal{L}$-constraint relation is given by the function *points* [KKR90]. *points* takes as argument an $\mathcal{L}$-constraint relation r over R and returns the $\mathbf{M}_\mathcal{L}$-relation over R which is finitely represented by r:

$$points(r) = \{u(t) : \ t \in r, \ u \in Val^u_{\mathbf{M}_\mathcal{L}} \text{ and } \mathbf{M}_\mathcal{L} \models t(CON)[u]\}.$$

The *semantics* of an indefinite $\mathcal{L}$-constraint relation r over scheme R is defined to be the following set of $\mathbf{M}_\mathcal{L}$-relations:

$$sem(r) = \{points(v(r)) : \text{ there exists } v \in Val^e_{\mathbf{M}_\mathcal{L}} \text{ s.t. } \mathbf{M}_\mathcal{L} \models G(r)[v]\}.$$

The function *rep* will also be useful in the rest of this paper. If r is an indefinite $\mathcal{L}$-constraint relation over scheme R then *rep* gives the set of $\mathcal{L}$-constraint relations represented by r:

$$rep(r) = \{v(r) : \text{ there exists } v \in Val^e_{\mathbf{M}_\mathcal{L}} \text{ s.t. } \mathbf{M}_\mathcal{L} \models G(r)[v]\}$$

The functions *points*, *sem* and *rep* can be extended to databases in the obvious way.[4]

4 Declarative Query Languages

[KKR90] proposed *relational calculus with $\mathcal{L}$-constraints* as a declarative query language for $\mathcal{L}$-constraint databases. In this section we propose *modal relational calculus with $\mathcal{L}$-constraints* as a declarative query language for indefinite $\mathcal{L}$-constraint databases. Similar modal query languages have been investigated in [Lip79, Lev84, Rei88].

Definition 7. Let $\widetilde{R}$ be a database scheme and $R(C_1, \ldots, C_m)$ be a relation scheme. An expression over $\widetilde{R}$ in *modal relational calculus with $\mathcal{L}$-constraints* is $\{R(C_1, \ldots, C_m), x_1/s_1, \ldots, x_m/s_m \ : \ OP \ \phi(x_1, \ldots, x_m)\}$ where $s_i \in sorts(\mathcal{L})$ is the sort of C_i, OP is an *optional* modal operator $\Diamond$ or $\Box$, ϕ is a well-formed formula of relational calculus with $\mathcal{L}$-constraints and $x_1, \ldots, x_m$ are the only free variables of ϕ. If an expression does not contain a modal operator then it will be called *pure*, otherwise it will be called *modal*.

[4] The above definitions imply that indefinite $\mathcal{L}$-constraint relations are interpreted in a *closed-world* fashion. They are assumed to represent all facts relevant to an application domain. However the exact value of any attribute of these facts may not be known precisely.

We will now define the *value* of expressions in modal relational calculus.

Definition 8. Let f be the pure expression $\{R(C_1, \ldots, C_m), x_1/s_1, \ldots, x_m/s_m : \phi(x_1, \ldots, x_m)\}$ over $\widetilde{R}$ in modal relational calculus with $\mathcal{L}$-constraints. If $\widetilde{r}$ is an indefinite $\mathcal{L}$-constraint database over $\widetilde{R}$ then the *value* of f on the set of $\mathbf{M}_{\mathcal{L}}$-relational databases $sem(\widetilde{r})$, whose finite representation is $\widetilde{r}$, is the following set of $\mathbf{M}_{\mathcal{L}}$-relations:

$$f(sem(\widetilde{r})) = \{ \ \{(a_1, \ldots, a_m) \in dom(s_1) \times \cdots \times dom(s_m) : (\mathbf{M}_{\mathcal{L}}, Dom, \widetilde{r}') \models \phi(a_1, \ldots, a_m)\} \ : \widetilde{r}' \in sem(\widetilde{r})\}$$

The question left open by the above definition is whether we can guarantee *closure* as required by the constraint query language principles laid out in [KKR90]. In other words, given a pure expression f of modal relational calculus with $\mathcal{L}$-constraints, and an indefinite $\mathcal{L}$-constraint database $\widetilde{r}$, is it possible to find an indefinite $\mathcal{L}$-constraint relation which finitely represents $f(sem(\widetilde{r}))$? In section 7, we show that the answer to this question is in the affirmative.

Example 5. The query "Find all rooms that are booked at 6:00" over the database of example 4 can be expressed as $\{BOOKED\_AT\_6(Room), x/\mathcal{D} : BOOKED(x, 6)\}$. The answer to this query is given by the following relation:

BOOKED_AT_6

Room	CON
x_1	$x_1 = WP212$
x_2	$x_2 = WP219, \ \omega > 6$

This answer is *conditional*. Room WP212 is booked on time 6. However, room WP219 is booked on time 6 *only under the condition* that ω is greater than 6. In section 7 we will show how to evaluate calculus queries and compute a finite representation of the answer.

Definition 9. Let f be the modal expression $\{R(C_1, \ldots, C_m), x_1/s_1, \ldots, x_m/s_m : \Box \ \phi(x_1, \ldots, x_m)\}$ over $\widetilde{R}$ in modal relational calculus with $\mathcal{L}$-constraints. If $\widetilde{r}$ is an indefinite $\mathcal{L}$-constraint database over $\widetilde{R}$ then the *value* of f on the set of $\mathbf{M}_{\mathcal{L}}$-relational databases $sem(\widetilde{r})$, whose finite representation is $\widetilde{r}$, is the following *singleton* set of $\mathbf{M}_{\mathcal{L}}$-relations:

$$f(sem(\widetilde{r})) = \{ \ \{(a_1, \ldots, a_m) \in dom(s_1) \times \cdots \times dom(s_m) : \text{for every } \mathbf{M}_{\mathcal{L}}\text{-relational database } \widetilde{r}' \in sem(\widetilde{r}) \ (\mathbf{M}_{\mathcal{L}}, Dom, \widetilde{r}') \models \phi(a_1, \ldots, a_m)\} \ \}$$

The value of a $\Diamond$-expression is defined in the same way but now the quantification over $\mathbf{M}_{\mathcal{L}}$-relational databases in $sem(\widetilde{r})$ is existential. Section 7 demonstrates that expressions of modal relational calculus with $\mathcal{L}$-constraints can also be evaluated in closed form. In summary, for every expression f (pure or modal) in modal relational calculus with $\mathcal{L}$-constraints and indefinite $\mathcal{L}$-constraint database $\widetilde{r}$, it is possible to find an indefinite $\mathcal{L}$-constraint relation which finitely represents $f(\widetilde{r})$.

Example 6. The query "Find all rooms that are possibly booked at 6:00" over the database of example 4 can be expressed as $\{POSS\_BOOKED\_AT\_6(Room), x/\mathcal{D} : \Diamond BOOKED(x, 6)\}$. If this query is evaluated using the method of section 7, the answer will be the following relation:

POSS_BOOKED_AT_6

Room	CON
x_1	$x_1 = WP212$
x_2	$x_2 = WP219$

The above answer is *unconditional*. It is possible that both rooms WP212 and WP219 are booked on time 6.

The next lemma demonstrates an intuitive property of modal relational calculus with $\mathcal{L}$-constraints. If $\mathcal{S}$ is a set of sets then $\bigcap \mathcal{S}$ (resp. $\bigcup \mathcal{S}$) denotes the set $\{\cap_{s \in \mathcal{S}} s\}$ (resp. $\{\cup_{s \in \mathcal{S}} s\}$).

Lemma 10. *Let f be a $\Box$-expression (resp. $\Diamond$-expression) over $\widetilde{R}$ in modal relational calculus with $\mathcal{L}$-constraints. Let f' be the pure expression which corresponds to f. Then for all indefinite $\mathcal{L}$-constraint databases $\widetilde{r}$ over $\widetilde{R}$, $f(sem(\widetilde{r})) = \bigcap f'(sem(\widetilde{r}))$ (resp. $f(sem(\widetilde{r})) = \bigcup f'(sem(\widetilde{r}))$).*

5 Procedural Query Languages

In this section, we briefly sketch three procedural query languages, one for each of the models discussed in section 3: the $\mathbf{M}_{\mathcal{L}}$*-relational algebra*, the $\mathcal{L}$*-constraint algebra* and the *modal* $\mathcal{L}$*-constraint algebra*. The $\mathbf{M}_{\mathcal{L}}$-relational algebra is a procedural query language for $\mathbf{M}_{\mathcal{L}}$-relational databases. It is interesting only from a theoretical point of view because $\mathbf{M}_{\mathcal{L}}$-relations are unrestricted. The operations of $\mathbf{M}_{\mathcal{L}}$-relational algebra can be defined verbatim as in the case of finite relations [Kan90].

The operations of the $\mathcal{L}$-constraint algebra are extensions of similar operations of standard relational algebra [Kan90]. The $\mathcal{L}$-constraint algebra has not been presented in [KKR90] where the model of $\mathcal{L}$-constraint databases was originally defined. However its definition is straightforward and can be found in [Kou94a] and in the full version of this paper. The algebras of [KSW90, Kou93, Kou94c, KG94, PVdBVG94] are essentially instances of the $\mathcal{L}$-constraint algebra.[5]

The operations of the *modal* $\mathcal{L}$*-constraint algebra* take as input one (or two) indefinite $\mathcal{L}$-constraint relations associated with a common global condition and return an indefinite $\mathcal{L}$-constraint relation associated with the *same* global

[5] The algebra of [KG94] also explores *canonical forms* for databases with dense order constraints. This issue is not discussed in this paper because our model is abstract. One might also add that canonical forms are important *only* in the context of a specific implementation.

condition. The modal $\mathcal{L}$-constraint algebra contains an operation for every $\mathcal{L}$-constraint algebra operation. The definitions of these operations were originally given in [Kou93] for the special case of indefinite dePCL-constraint relations.[6] These operations treat e-variables as uninterpreted parameters thus they are defined exactly as the $\mathcal{L}$-constraint algebra operations. Similar operations were defined in [IL84, Gra89] for the special case of conditional tables.

The modal algebra also includes two additional operations $POSS$ and $CERT$, which take a more active stand towards e-variables. Given an indefinite $\mathcal{L}$-constraint relation r, the expression $POSS(r)$ evaluates to an $\mathcal{L}$-constraint relation which finitely represents the set of all tuples contained in *any* relation of $sem(r)$. The expression $CERT(r)$ evaluates to an $\mathcal{L}$-constraint relation which finitely represents the set of all tuples contained in *every* relation of $sem(r)$.

Possibility. Let r be an indefinite $\mathcal{L}$-constraint relation on scheme R. Then $POSS(r)$ is an *$\mathcal{L}$-constraint relation* defined as follows:

1. $sch(POSS(r)) = sch(r)$
2. $POSS(r) = \{poss(t) : t \in r\}$.

For each tuple t on scheme R, $poss(t)$ is a tuple on scheme R such that $poss(t)(R) = t(R)$ and $poss(t)(CON) = \psi$ where ψ is obtained by eliminating all e-variables from the boolean combination of $\mathcal{L}$-constraints $G(r) \wedge t(CON)$. The expression $poss(t)(CON)$ is well-defined since the class of $\mathcal{L}$-constraints admits variable elimination.

Certainty. Let r be an indefinite $\mathcal{L}$-constraint relation on scheme R. Then $CERT(r)$ is an $\mathcal{L}$-constraint relation defined as follows:

1. $sch(CERT(r)) = sch(r)$
2. $CERT(r) = \{cert(t) : t \in r^{\downarrow}\}^{\uparrow}$.

For each tuple t on scheme R, $cert(t)$ is a tuple on scheme R such that $cert(t)(R) = t(R)$ and $cert(t)(CON) = \neg\psi$ where ψ is obtained by eliminating all e-variables from the boolean combination of $\mathcal{L}$-constraints $G(r) \wedge \neg t(CON)$. The expression $cert(t)(CON)$ is well-defined since the class of $\mathcal{L}$-constraints admits variable elimination.

The operation $r^{\downarrow}$ has the effect of *denormalizing* $\mathcal{L}$-constraint relation r. This is achieved by collecting all tuples $\{t_1, \ldots, t_{|r|}\}$ of r into a single tuple t' on scheme R such that $t'(R) = (x_1, \ldots, x_{|R|})$ and $t'(CON) = t'_1(CON) \vee \cdots \vee t'_{|r|}(CON)$. In the new tuple t' u-variables have been standardized apart: $x_1, \ldots, x_{|R|}$ are brand new u-variables, and for $1 \leq i \leq |r|$, $t'_i(CON)$ is the same as $t_i(CON)$ except that $t(X)$ has been substituted by $t'(X)$ for each $X \in R$.

The operation $r^{\uparrow}$ has the effect of *normalizing* the local conditions of a relation r in order to obtain a true $\mathcal{L}$-constraint relation. This is done by the following three steps:

- Application of De Morgan's laws to transform the negated parts of each local condition of r into a disjunction whose disjuncts are $\mathcal{L}$-constraints.

[6] [Kou93] uses the term *temporal tables* for indefinite dePCL-constraint relations.

This operation is well-defined since the class of $\mathcal{L}$-constraints is weakly closed under negation.
- Application of the law of associativity of conjunction with respect to disjunction to transform each local condition of r into a disjunction of conjunctions of $\mathcal{L}$-constraints.
- Splitting of disjuncts into different tuples.

Let us now define modal $\mathcal{L}$-constraint algebra expressions.

Definition 11. A *pure expression* over scheme $\widetilde{R}$ in modal $\mathcal{L}$-constraint algebra is any well-formed expression built from constant $\mathcal{L}$-constraint relations, relation schemes from $\widetilde{R}$ and the above operators excluding $POSS$ and $CERT$. A *modal $\mathcal{L}$-constraint algebra expression* is a pure expression, or an expression of the form $CERT(g)$ or $POSS(g)$ where g is a pure expression. Expressions of the form $CERT(g)$ or $POSS(g)$ are called *$CERT$-expressions* or *$POSS$-expressions* respectively.

Modal $\mathcal{L}$-constraint algebra expressions define functions from indefinite $\mathcal{L}$-constraint databases to indefinite $\mathcal{L}$-constraint relations. The result of applying an expression e to an indefinite $\mathcal{L}$-constraint database $\widetilde{r}$ is defined as for the $\mathcal{L}$-constraint algebra. Let us simply stress that $G(e(\widetilde{r})) = G(\widetilde{r})$ for all indefinite $\mathcal{L}$-constraint databases $\widetilde{r}$ and expressions e over $\widetilde{R}$.

The following lemma gives an intuitive property of $POSS$ and $CERT$.

Lemma 12. *Let e be a pure expression over scheme $\widetilde{R}$ in modal $\mathcal{L}$-constraint algebra. Then for all indefinite $\mathcal{L}$-constraint databases $\widetilde{r}$ over $\widetilde{R}$*

$$sem(CERT(e(\widetilde{r}))) = \bigcap sem(e(\widetilde{r})) \text{ and } sem(POSS(e(\widetilde{r}))) = \bigcup sem(e(\widetilde{r})).$$

6 On the Semantics of Algebraic Query Evaluation

Let $\widetilde{r}$ be an $\mathcal{L}$-constraint database, e an $\mathcal{L}$-constraint algebra expression and $e1$ its corresponding $\mathbf{M}_{\mathcal{L}}$-relational algebra expression. Recall that an $\mathcal{L}$-constraint relation $\widetilde{r}$ is a finite representation of the unrestricted set of tuples $points(\widetilde{r})$. The following theorem shows that the operations of $\mathcal{L}$-constraint algebra "behave" according to our intuitions: when we evaluate e on $\widetilde{r}$, we essentially evaluate $e1$ on the unrestricted relation $points(\widetilde{r})$. This theorem generalizes the results of [KG94, Kou94c, PVdBVG94] who consider *specific* constraint languages.

Theorem 13. *Let e be an $\mathcal{L}$-constraint algebra expression over $\widetilde{R}$ and $e1$ be its corresponding $\mathbf{M}_{\mathcal{L}}$-relational algebra expression. If $\widetilde{r}$ is an $\mathcal{L}$-constraint database over scheme $\widetilde{R}$, then $points(e(\widetilde{r})) = e1(points(\widetilde{r}))$.*

Let us now assume that $\widetilde{r}$ is an indefinite $\mathcal{L}$-constraint database, e is a pure expression of modal $\mathcal{L}$-constraint algebra and $e1$ is its corresponding expression in $\mathcal{L}$-constraint algebra. Recall that the semantic function $sem(\widetilde{r})$ returns all the "possible worlds" represented by $\widetilde{r}$. When we evaluate e on indefinite

$$
\begin{tikzcd}
\mathcal{L}\text{-}ICDB \arrow[r, "\mathbf{e}"] \arrow[d, "\mathbf{rep}"'] \arrow[dd, bend right=60, "\mathbf{sem}"'] & \mathcal{L}\text{-}ICDB \arrow[d, "\mathbf{rep}"] \arrow[dd, bend left=60, "\mathbf{sem}"] \\
\{\mathcal{L}\text{-}CDB\} \arrow[r, "\mathbf{e1}"] \arrow[d, "\mathbf{points}"'] & \{\mathcal{L}\text{-}CDB\} \arrow[d, "\mathbf{points}"] \\
\{\mathbf{M}_{\mathcal{L}}\text{-}RDB\} \arrow[r, "\mathbf{e2}"] & \{\mathbf{M}_{\mathcal{L}}\text{-}RDB\}
\end{tikzcd}
$$

Fig. 1. Relating the three algebras

$\mathcal{L}$-constraint database $\widetilde{r}$ using the operations defined above, we essentially evaluate the corresponding $\mathbf{M}_{\mathcal{L}}$-relational algebra expression on each possible world in $sem(\widetilde{r})$. As discussed in [Imi89], an extension of an $\mathbf{M}_{\mathcal{L}}$-relational algebra expression $e1$ to an expression e for an $\mathcal{L}$-constraint representation of indefinite information can claim to be "faithful to the underlying semantics" if and only if for every indefinite $\mathcal{L}$-constraint database $\widetilde{r}$,

$$sem(e(\widetilde{r})) = e1(sem(\widetilde{r})) = \{e1(r_1) : \ r_1 \in sem(\widetilde{r})\}.$$

Equivalently, one would like to guarantee that there is always an indefinite $\mathcal{L}$-constraint database $\widetilde{r}'$ such that $sem(\widetilde{r}') = e1(sem(\widetilde{r}))$. The following theorem demonstrates that $\mathcal{L}$-constraint databases satisfy this form of algebraic closure. This theorem generalizes a result of [Kou94c] who considers databases with dePCL-constraints.

Theorem 14. *Let e be a pure expression of modal $\mathcal{L}$-constraint algebra over $\widetilde{R}$, $e1$ its corresponding $\mathcal{L}$-constraint algebra expression and $e2$ its corresponding $\mathbf{M}_{\mathcal{L}}$-relational algebra expression. If $\widetilde{r}$ is an indefinite $\mathcal{L}$-constraint database over scheme $\widetilde{R}$ then $rep(e(\widetilde{r}))$ is equivalent to $e1(rep(\widetilde{r}))$ and $sem(e(\widetilde{r})) = e2(sem(\widetilde{r}))$.*

The above theorems are summarized graphically in the commutative diagram of figure 1 where $\mathbf{M}_{\mathcal{L}}$-RDB denotes the set of all $\mathbf{M}_{\mathcal{L}}$-relational databases, $\mathcal{L}$-CDB denotes the set of all $\mathcal{L}$-constraint databases and $\mathcal{L}$-$ICDB$ denotes the set of all indefinite $\mathcal{L}$-constraint databases. Since the above results have been proved in our general framework, specific cases of constraint databases [KKR90, KSW90, Kou93, Kou94c, KG94, PVdBVG94] can simply refer to these theorems to demonstrate the "correctness" of the operations of their algebraic query languages.

7 Translating Calculus Expressions into Algebraic Expressions

In this section we show that expressions of modal relational calculus with $\mathcal{L}$-constraints have equivalent expressions in modal $\mathcal{L}$-constraint algebra. Thus we can evaluate a calculus expression by evaluating an equivalent algebraic expression. As we have seen in section 5, algebraic query evaluation can be done bottom-up and the answer is obtained in closed form. Therefore calculus expressions can also be evaluated bottom-up in closed form on indefinite $\mathcal{L}$-constraint databases. [Kou94a] gives an alternative proof of this result by employing quantifier elimination techniques as suggested in [KKR90].

We start by considering the simpler case of $\mathcal{L}$-constraint databases. The following theorem generalizes the analogous results of [KKR90, Kou94c, PVdBVG94] who consider *specific* constraint languages.

Theorem 15. *For every expression f over $\widetilde{R}$ in relational calculus with $\mathcal{L}$-constraints there exists an $\mathcal{L}$-constraint algebra expression e over $\widetilde{R}$ such that the following property holds. If $\widetilde{r}$ is an $\mathcal{L}$-constraint database over $\widetilde{R}$ then $f(points(\widetilde{r})) = points(e(\widetilde{r}))$.*

Let us note here that the analogous proofs of [KKR90] rely on quantifier elimination methods which achieve good data complexity lower bounds but do not seem to have practical implementations. In contrast, the above theorem (as well as the analogous theorems of [KG94, Kou94c, PVdBVG94]) provides a translation of calculus expressions into algebraic expressions. This translation can be the first step in optimizing the evaluation of expressions in relational calculus with $\mathcal{L}$-constraints.

Let us now turn to modal relational calculus with $\mathcal{L}$-constraints and modal $\mathcal{L}$-constraint algebra.

Lemma 16. *Let e be a pure expression over $\widetilde{R}$ in modal $\mathcal{L}$-constraint algebra and e' be its corresponding $\mathcal{L}$-constraint algebra expression. Then $sem(e(\widetilde{r})) = \{points(e'(\widetilde{r}')) : \widetilde{r}' \in rep(\widetilde{r})\}$ for all indefinite $\mathcal{L}$-constraint databases $\widetilde{r}$ over $\widetilde{R}$.*

The following theorem demonstrates that pure expressions of modal relational calculus with $\mathcal{L}$-constraints over indefinite $\mathcal{L}$-constraint databases can also be evaluated bottom-up in closed form.

Theorem 17. *For every pure expression f over $\widetilde{R}$ in relational calculus with $\mathcal{L}$-constraints there exists a pure expression e over $\widetilde{R}$ in modal $\mathcal{L}$-constraint algebra such that the following property holds. If $\widetilde{r}$ is an indefinite $\mathcal{L}$-constraint database over $\widetilde{R}$ then $f(sem(\widetilde{r})) = sem(e(\widetilde{r}))$.*

Example 7. The algebraic expression equivalent to the calculus expression of example 5 is

$$\pi_{Room}(\sigma_{Time=6}(BOOKED)).$$

Finally we turn to modal expressions.

Theorem 18. *Let f be a $\Box$-expression (resp. $\Diamond$-expression) over $\widetilde{R}$ in modal relational calculus with $\mathcal{L}$-constraints. Then there exists a $CERT$-expression (resp. $POSS$-expression) e over $\widetilde{R}$ in modal $\mathcal{L}$-constraint algebra such that the following property holds. If $\widetilde{r}$ is an indefinite $\mathcal{L}$-constraint database over $\widetilde{R}$ then $f(sem(\widetilde{r})) = sem(e(\widetilde{r}))$.*

Example 8. The algebraic expression equivalent to the calculus expression of example 6 is

$$POSS(\pi_{Room}(\sigma_{Time=6}(BOOKED))).$$

8 Related Work and Future Research

The results of this study are extended in [Kou94b, Kou94a] where we concentrate on *temporal constraint databases* (with or without indefinite information). In particular, we study the complexity of query evaluation in $\mathcal{L}$-constraint databases and indefinite $\mathcal{L}$-constraint databases where $\mathcal{L}$ ranges over several temporal constraint languages (including dePCL). Our analysis shows that the worst-case data/combined complexity of query evaluation *does not change* when we move from queries in relational calculus over relational databases, to queries in relational calculus with temporal constraints over temporal constraint databases. This fact remains true even if we consider indefinite relational databases vs. indefinite temporal constraint databases. Unfortunately, the presence of indefinite information makes query evaluation intractable in many cases. Our analysis complements the results of [Rev90, CM93] and extends the results of [KKR90, vdM92].

The research reported in [SRR] is closely related to our own. This paper investigates the use of constraints for the representation of infinite and indefinite information in a simple object-oriented model. The authors do not consider the properties of an abstract constraint-object model; instead, they concentrate on a special case which uses order constraints and simple set constraints (e.g., constraints using $\in$, $\subseteq$ but not $\cap$ or $\cup$). The important contribution of this paper is the study of quantifier-elimination for this simple class of set constraints and the development of a Datalog-like query language which incorporates them. This query language is intended for the expression of "certainty" queries only; "possibility" queries are not considered.

In other related work [ACGK, GS94, PVdBVG94] study the expressive power of query languages with linear and polynomial constraints. [GS94] in particular demonstrate that the theory of finitely representable models differs substantially from classical model theory and finite model theory. This suggests a need for developing new tools for considering definability of queries in the constraint database context.

We are currently considering query languages based on first-order constraint languages more expressive than diPCL and dePCL. The interesting question is whether the results of [Kou94a] carry over to the new classes. Another interesting question is the study of tractable cases of indefinite diPCL/dePCL-constraint databases. The work of [vdM92] can serve as the basis for such an effort.

Acknowledgements

I would like to thank Timos Sellis and Barry Richards for their support and encouragement. I would also like to thank Peter Revesz and Paris Kanellakis for promptly answering my questions concerning their work on constraint databases.

References

[ACGK] F. Afrati, S. Cosmadakis, S. Grumbach, and G. Kuper. Linear vs. Polynomial Constraints in Database Query Languages. In this volume.

[CM93] J. Cox and K. McAloon. Decision Procedures for Constraint Based Extensions of Datalog. In F. Benhamou and A. Colmerauer, editors, *Constraint Logic Programming: Selected Research.* MIT Press, 1993. Originally appeared as Technical Report No. 90-09, Dept. of Computer and Information Sciences, Brooklyn College of C.U.N.Y.

[End72] H.B. Enderton. *A Mathematical Introduction to Logic.* Academic Press, 1972.

[Gra89] Gosta Grahne. The Problem of Incomplete Information in Relational Databases. Technical Report Report A-1989-1, Department of Computer Science, University of Helsinki, Finland, 1989. Also published as Lecture Notes in Computer Science 554, Springer Verlag, 1991.

[GS94] S. Grumbach and J. Su. Finitely representable databases. In *Proceedings of the 13th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 289–300, 1994.

[IL84] T. Imielinski and W. Lipski. Incomplete Information in Relational Databases. *Journal of ACM*, 31(4):761–791, 1984.

[Imi89] T. Imielinski. Incomplete Information in Logical Databases. *Data Engineering*, 12(2):29–39, 1989.

[Kan90] Paris Kanellakis. Elements of Relational Database Theory. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 17. North-Holland, 1990.

[KG94] P.C. Kanellakis and D. Goldin. Constraint Programming and Database Query Languages. In *Proceedings of Theoretical Aspects of Computer Software (TACS)*, volume 789 of *Lecture Notes in Computer Science*, pages 96–120. Springer-Verlag, April 1994.

[KKR90] Paris C. Kanellakis, Gabriel M. Kuper, and Peter Z. Revesz. Constraint Query Languages. In *Proceedings of the 9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 299–313, 1990. Long version to appear in Journal of Computer and System Sciences.

[Kou93] Manolis Koubarakis. Representation and Querying in Temporal Databases: the Power of Temporal Constraints. In *Proceedings of the 9th International Conference on Data Engineering*, pages 327–334, April 1993.

[Kou94a] M. Koubarakis. *Foundations of Temporal Constraint Databases.* PhD thesis, Computer Science Division, Dept. of Electrical and Computer Engineering, National Technical University of Athens, February 1994.

[Kou94b] Manolis Koubarakis. Complexity Results for First-Order Theories of Temporal Constraints. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR'94)*, pages 379–390. Morgan Kaufmann, San Francisco, CA, May 1994.

[Kou94c] Manolis Koubarakis. Database Models for Infinite and Indefinite Temporal Information. *Information Systems*, 19(2):141–173, March 1994.

[KSW90] F. Kabanza, J.-M. Stevenne, and P. Wolper. Handling Infinite Temporal Data. In *Proceedings of ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 392–403, 1990.

[Lev84] H.J. Levesque. Foundations of a Functional Approach to Knowledge Representation. *Artificial Intelligence*, 23:155–212, 1984.

[Lip79] Witold Jr. Lipski. On Semantic Issues Connected with Incomplete Information Databases. *ACM Transcactions on Database Systems*, 4(3):262–296, September 1979.

[Mah93] M. Maher. A Logic Programming View of CLP. In *Proceedings of the 10th International Conference on Logic Programming*, pages 737–753, 1993.

[Mai83] David Maier. *The theory of relational databases.* Computer Science Press, 1983.

[PVdBVG94] J. Paredaens, J. Van den Bussche, and D. Van Gucht. Towards a theory of spatial database queries. In *Proceedings of the 13th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 279–288, 1994.

[Rab77] M.O. Rabin. Decidable theories. In *Handbook of Mathematical Logic*, volume 90 of *Studies in Logic and the Foundations of Mathematics*, pages 595–629. North-Holland, 1977.

[Rei88] Ray Reiter. On Integrity Constraints. In *Proceedings of the 2nd Conference on Theoretical Aspects of Reasoning About Knowledge*, pages 97–111, Asilomar, CA, 1988.

[Rev90] Peter Z. Revesz. A Closed Form for Datalog Queries with Integer Order. In *Proceedings of the 3rd International Conference on Database Theory*, pages 187–201, 1990. Long version to appear in Theoretical Computer Science.

[SRR] D. Srivastava, R. Ramakrishnan, and P. Revesz. Constraint Objects. In this volume.

[Stu91] P.J. Stuckey. Constructive Negation for Constraint Logic Programming. In *Proceedings of Symposium on Logic in Computer Science*, pages 328–339, 1991.

[vdM92] Ron van der Meyden. The Complexity of Querying Indefinite Data About Linearly Ordered Domains (Preliminary Version). In *Proceedings of the 11th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 331–345, 1992.