Domain Abstraction and Limited Reasoning

Tomasz Imielinski

Department of Computer Science
Rutgers University and
Polish Academy of Science

I Abstract

We are investigating the possibility of constructing meaningful and computationally efficient approximate reasoning methods for the first order logic. In particular, we study a situation when only certain aspects of the domain are of interest to the user. This is reflected by an equivalence relation defined on the domain of the knowledge base. The whole mechanism is called domain abstraction and is demonstrated to lead to significant computational advantages. The domain abstraction discussed in the paper is only a very special case of the more general notion of abstraction which is discussed shortly here and is a subject of the currently ongoing research.

II INTRODUCTION

In this paper we are interested in providing a basis for meaningful reasoning with low complexity. Such reasoning will be called limited to indicate that it is weaker than the general first order logic proof methods. Recently many authors have tried to develop different logic systems, which would have better complexity properties than the classical propositional or predicate logic [Patel-Schneider 85], [Lakemeyer 86], [Konolidge 85] and in a way [Fagin 85]. Here, we take a different approach by developing approximate methods of reasoning *within* the same logic.

Approximate methods are widely accepted in numerical computations. Unfortunately automated reasoning, which is computationally at least as expensive as numerical computing does not have a proper notion of approximation and has still to follow the ambitious "All or nothing" approach. The main problem is lack of the proper notion of error, without which it is difficult to provide any meaning to an approximation.

In this paper we demonstrate a notion of error which is sufficiently general to cover both automated reasoning and numerical computations. The following example illustrates the point:

Example 1

Suppose we want to compute a volume of a certain cube A. Assume that we round up the measurements of A, say to the closest integer in meters. If we calculate now, say, the volume of A our result will be biased with error, say 1 m^3.

Let us now take a look at this simple example from the more general point of view. Let the measurements of A form a *knowledge base* and let Volume(A,x), where A is our cube and x stands for volume be a *query.* (We could also have other queries asking for the diameter of A, total area of faces, etc.) The process of rounding up the measurements of A can be now viewed as replacement of the original knowledge base by a new one, less precise but presumably easier to deal with. The price of this simplification is paid in the loss of precision - we will not compute a "true" volume of the cube anymore but some other value. This new, approximate value will *share* certain properties with the real answer. Indeed, let the answer to our query be represented in the form of atomic formula Volume(A, 124 m^3). Although this formula may not necessarily be true, the formula $\phi = \exists_x (123 \leq x \leq 125) \wedge$ Volume(A,x) will be true (Since our error is equal to $1\ m^3$). In other words <*t*> form a property of the real answer, which is *preserved* by the approximate answer. In fact all formulas of the form $\phi' = \exists_x (a \leq x \leq b) \wedge$Volume(A,x), where $a \leq 123$ and $b \geq 125$ will be preserved. On the other hand the preservation is not guranteed if $123 \leq a$ or $b \leq 125$. For instance the formula $\phi'' = \exists_x (123.9 \leq x \leq 124.1)) \wedge$ Volume(A,x) will not necessarly be preserved (i.e it is true for our approximate answer, but could be false for the real answer). ■

Our notion of error is motivated by this example - it will be the set of all formulas (notice that error is a set) which are *not* preserved by the approximate answer. In our case ϕ'' belongs to the error, while ϕ does not.

This notion of an error will be called *local error* since it is related to a particular query. We define also a *global* error resulting from the replacement of our knowledge base by the "rounded up" one. The global error will be simply a set of all queries, which are not quranteed to be answered correctly by the "rounded up" knowledge base. In the Example 1 we could imagine the whole variety of queries, asking for a diameter, total area of faces, color of a cube etc (assuming the proper data is in the knowledge base) . Some of these queries will be answered correctly, because the "round up" does not affect them. Therefore, while the global error will divide queries

Research support*) by NSF grant DCR 85-04140

into two categories (correctly answered, incorrectly answered), the local error will indicate "how far" each individual answer (for each individual query) is from the real answer. Needless to say the local error will be empty for queries, which are answered correctly, that is which are not in the global error set.

This paper is intended to serve as a "case study" for the notions introduced above, for a very special type of approximate reasoning method, called domain abstraction. Under domain abstraction only certain aspects of the domain will be of interest to the user. Instead of domain constants, he will deal with equivalence classes of them. In consequence he will deal with "rounded up" knowledge base, similar to the numerical one just described.

Important observations about the fundamental role of abstraction in approximate reasoning were made by Hobbs in [Hobbs 85]. The notion of abstraction is also studied in [Imielinski 85]. Here we concentrate on the domain abstraction (to be defined in the next section) by discussing it's computational benefits and (global) errors.

The paper is divided into two parts. In the first part of the paper, in section three, four and five we define formally the notion of abstraction and discuss the issues of query processing and the error of domain abstraction. Finally, we briefly discuss applications of domain abstraction in the limited reasoning.

III BASIC NOTIONS

By the *Knowledge base,* denoted by KB (or DB) we understand any finite collections of formulas of some first order language L. We also use the term *database* specially when the KB is a collection of atomic formulas corresponding to the relational database.

By the query we mean any open formula of L. and by the answer to a query the set of all substitutions of domain constants for variables such that the resulting closed formula is a logical consequence of KB.

By the *domain* of the knowledge base we mean the set of all objects occurring in the KB.

By the *equivalence relation* on the set D we mean a binary relation on D which is reflexive, symmetric and transitive. Equivalence relation is *selective* on the subset D_0 of D iff for any element $a \in D_0$ the equivalence class of a: $[a] = \{a\}$.

IV DOMAIN ABSTRACTION

Let D be the domain of our Knowledge base and let R be an equivalence relation on D. Let L be the first order language of the knowledge base. We can extend R in the natural way to models of L. Two models, m and m' will be R-equivalent iff for any atomic formula $Ra_r..a_B$ which is true in m (m^1) there is an atomic formula $Rb_r..b_n$ which is true in m' (m), such that ajRb, for $i=1,...n$.

The equivalence relation R can be given one of the following interpretations:

1. It may correspond to the relevant features of the external world which are of interest to the user
2. It may be used by the system to hide certain features of the external world from the user
3. It may correspond to the error with which data is entered into the database. As a consequence we do not entirely believe what is stated in the database, but take it with "the grain of salt", which is reflected by equivalence relation R.

All these interpretations have similar consequences: i.e the user's view of external world is even more incomplete that the view of the knowledge base. The "noise" is introduced on the interface between the user and knowledge base. However, there is an important difference between these two approaches: In the first approach the choice *of* abstracted interpretation is made by user, while in the second interpretation the choice is made by administrator of a system. This distinction will have further consequences later in the paper.

Let KB be a knowledge base and let R be the equivalence relation on the set of models of KB. The equivalence relation R determines a new, weaker, semantic consequence relation N_R on L.

$KB \models_R \sigma$ iff for any model $m \in M(KB)$ σ is true not only in m but also in all models which are R-equivalent to R.

This definition corresponds to the truth definition for the necessity operator in the Kripke model with R as its access relations. This is studied in more detail in the paper [Imielinski 87]. Intuitively, the external user whose information is filtered out by the relation R cannot distinguish between two models which are equivalent, therefore all models which are equivalent to models of KB are, for this user "as good" as models of KB. Obviously some of the formulas of KB will be lost if they are not "filtered out" through R.

The above definition could be interpreted also in the different way, as abstraction of KB. Indeed, our user no longer sees the knowledge base KB but rather some logically weaker set of formulas corresponding to his, less precise now, set of possible worlds namely:

$M(KB,R) = \{m$: There is m' such that m'Rm and $m' \in M(KB)\}$

Clearly any formula o is a semantic consequence of KB in the new sense iff it is true in each model from M(KB,R).

The equivalence relation R which is defined on the domain of knowledge base induces equivalence relation on the name constants of the knowledge base language L. The equivalence classes of this relation will be denoted by [a] where "a" is a name constant.

The key question, which we are going to investigate, is whether $\models_R$ is computationally more attractive than standard $\models$? We are also interested in estimating the "error" if the reasoning is performed in the abstracted

knowledge base instead of the original one.

Example 2

Let our knowledge base have a form of a very long disjunction, say of the form:

$$Pab \vee Pa_1b_1 \vee \ldots Pa_nb_n$$

Let us assume that the user's equivalence relation R is defined in such a way on the domain of the knowledge base that all elements a,...a_n belong to the same equivalence class, say [a] and all elements $b_1..b_n$ belong to the other equivalence class, say [b]. In such case the resulting knowledge base can be viewed simply as the atomic formula: $P[a][b]$. One can visualize very long disjunctions reducing its size to very short, if not atomic formulas, after applying this kind of abstraction. There are therefore obvious computational benefits of this technique here. ■

We use here two different languages: the abstracted language L_{ft} and the basic language L. The abstracted language is the first order language with name constants $\{[a]:a\in D\}$. The formulas of L_a are interpreted either as second order formulas (if we allow quantification on equivalence classes) or as first order formulas of some extension of L.

Second Order Interpretation of L.

The second order interpretation of the formula o of L_a will be denoted by a_u (where "u" stands for "unaware", which will be clear later on). The satisfiability relation for this interpretation will be defined on the basis of the *quotient* models which is equal to the set of all equivalence classes (with respect to R) of models of KB. Notice that quotient models are simply built as relations defined on equivalence classes [a] of domain constants aeD. We will say the $<T_U$ is true in KB iff it is true in each quotient model of KB.

This interpretation corresponds to the situation when the external user is *unaware* of the fact that he is using any abstracted language. In fact he treats or equivalence constants as if they were names of singular objects. He is simply unaware of the fact that they are really unary predicates.

First order Interpretation of L_A

In this interpretation denoted by 0a the user is aware of "cheating" and he treats all the constants of the language as unary predicates. Formally let L be the language L extended by unary predicates $u_{iv}..,u_n$ corresponding to the equivalence classes of R on the name constants of L. Any formula of L_t can be translated into the formula of L by replacing all constants [a] by existential quantifiers range restricted to the unary predicate $[a](x)$ corresponding to [a]. All quantifiers in L_a are treated as ranging over the objects of the domain D.

Example 3

The atomic formula $P([a],[b])$ will be translated to the formula:

$\exists_{x,y}\ [a](x)\wedge[b](y)\wedge P(x,y)$, where $[a](x)$ and $[b](x)$ are unary predicates corresponding to equivalence classes [a] and [b] (i.e. $[a](x)$ is true iff x is in the equivalence class [a]).

Example 4

Let KB be a knowledge base storing all direct flight connections in the United States. One natural abstraction which can be considered is provided by the equivalence relation putting all cities which are in the same state into the same equivalence class. The abstracted language L_a is going to use names of the states instead of using the names of the cities. In the second order interpretation the user will not be "aware" that states are really predicates. Let us now consider the query *a:* Give me all direct or indirect connections with one stop over from New York to Seattle. According to the second order interpretation o_u the user will get some erroneous connections. For example, if there is a flight from New York to LA and the flight from San Francisco to Seattle then this will be printed as connection because LA and San Francisco are in the same state ! Therefore, instead of the real answer to the query we will get some approximation of it which is complete but not sound (i.e we get more tuples than necessary). In the same time this approximation will give us a correct information about which connections are *not* possible. On the other hand take now the first order interpretation *oa.* This will be a very conservative interpretation, which in fact will lead to a subset of the real answer, i.e to the approximation which is *sound* but not complete. Indeed, even if there is a flight from NY to Seattle with one stopover in San Francisco it will not be printed out, since there are other models in M(KB,R) in which the place of arrival of the flight from NY is different from the place of departure of the flight to Seattle. In other words, even if the knowledge base contain formulas Connected(NewYork State,California) and Connected(California, Washington) no matching between two occurrences of California will occur (since the user is aware that there may be many cities there. These two approximations differ in their treatment of equivalence classes, in the former approach equivalence classes are always unifiable with themselves, in the latter one they are never unifiable with themselves.

I

Let us now investigate the basic questions related to the very notion of abstraction: How good this approximation is? Does it lead to computational benefits ? .

We will approach these questions by investigating first the following problems:

Actually it it a possibility operator in the Kripke model with access relation R defined as domain equivalence relation defined on models

1. Given a theory KB how to construct theory $[KB]_R$, called first order R-reconstruction in L* such that $M([KB]_R) = M(KB,R)$?
2. The same question but with regards to a theory $[KB]_R'$ in L_A, called second order R-reconstruction on L_A such that or that the set of models of $[KB]_R'$ is equal to the set of quotient models of KB.
3. What is the error between KB and $[KB]_R$, i.e. for which formulas KB and KB_R give different answers?

The first two questions will be investigated in the next two subsections. The last question is a subject of section 5.

IV.I. First order R-reconstructions of the knowledge base KB

We will assume here that our knowledge base is a set of positive (no negation) formulas. Later we discuss deductive (with implication) databases.

Here we are interested in representing the set of all formulas of the language L* which are true in all models of M(KB,R). For a formula ^ we would like to establish a new formula $[\phi]_R$, or shortly $[\phi]$ if R is clear from the context, such that:

$$M([\phi]) = M(\phi,R).$$

It easy to check that [ ] behaves like a possibility operator'*. Therefore:

$$[A \vee B] = [A] \vee [B]$$

$$[\exists_x A] = \exists_x [A]$$

but

$$[A \wedge B] \neq [A] \wedge [B]$$

Therefore we cannot generate $[T]_R$ on the formula by formula basis. Instead we first transform T to the disjunctive normal form and treat each disjunct separately. For each individual disjunct U we form a set of formulas $[U]_R$ by the following procedure:

1. First rename all variables in U in such a way that no variable occur twice in U. We add proper equality conditions to reflect the fact that several different occurrences of the same variable were renamed differently. As the result we get the set of literals and the set of equalities.
2. Replace each individual constant "a" occurring in any of the literals by the variable with range restricted existential quantification over unary predicate [a](x), corresponding to the equivalence class generated by a. $[U]_R$ will be formed by ignoring all equality conditions and making a conjunction of all literals possibly with such a range restricted existential quantification.

Therefore, in general only single literal formulas of D will be preserved by abstraction.

Example 5

Let our knowledge base have the following form:

$$KB = \{\exists_x P(a,x) \wedge P(b,x) \wedge P(c,x)\}$$

It is clear that, if we know nothing about abstraction relation R, the first order reconstruction of this knowledge base will have a form:

$$[KB] = \{\exists_x \exists_y [a](y) \wedge P(y,x),\ \exists_{u,v} [b](u) \wedge P(u,z),\ \exists_{v,w} [c](v) \wedge P(v,w)\}$$

Indeed, after renaming all occurrences of the variable x in the original formula KB by three different variables: x, u and w and adding proper equality conditions (x=u=w) we get three different literals which after replacement by unary predicates corresponding to equivalence classes will lead to [KB] displayed above.

However, in case we know that R is selective on the second attribute of the relation we will get

$$[KB] = \exists_{x,v,y,z} [a](v) \wedge [b](y) \wedge [c](z) \wedge P(v,x) \wedge P(y,x) \wedge P(z,x)$$

i.e. the original formula in the knowledge base will be preserved modulo substitution of name constants by unary predicates.

If we know more about the equivalence relation R then we loose less information in the reconstruction of the knowledge base KB and more formulas will be preserved. In particular if we distinguish subsets of the domain D as *types* or domains of attributes then we can make some separate assumptions about the behavior of equivalence relation R for individual types. The important special case occurs when R is selective on domains of certain attributes. In such a case we cannot ignore equality conditions between variables ranging on this attribute (called *selective* variables) and the construction of $[U]_R$ is more complex. We should now, in the final step of construction of $[U]_R$, take a conjunction of all literals with all these equality conditions which hold between selective variables. More knowledge about the equivalence relation R may however lead to more complex reasoning: Let our knowledge base KB = { Pe_1b_1, Pe_2b_2, Pe_3b_2} and let the equivalence classes of R have a form $\{b_1,b_2\}$, $\{e_1\},\{e_2\},\{e_3\}$.

Let query Q have a form:

$$Q = \exists_{x,y,z} (x \neq y) \wedge P(x,z) \wedge P(y,z)$$

If we reconstruct our knowledge base according to the above domain abstraction we will get the following set of formulas:

$P(e_1,B);P(e_2,B);P(e_3,B)$, where "B" is the equivalence class containing b_1 and b_2.

Without knowing the equivalence relation R the answer to the above query would be "false" since no two different occurrences of B could be equal for sure. Suppose, however, that we know cardinalities of equivalence classes of R. In this case the answer to our query will still be true. Indeed, since B has

exactly two elements ,and there are three occurrences of B in the database, therefore at least two of these occurrences have to be equal.

In general these kind of "combinatorial" inferences about equality are much harder to perform that even the standard first order derivations (i.e derivations when we assume that the equivalence relation is selective). We can conclude the above considerations in the following:

"Ignorance" Principle

Abstraction is computationally beneficial if we either don't know anything about the cardinality of equivalence classes or we know that there are singular (i.e contain one element)

IV.2. Second Order R-reconstruction

The second order reconstruction is much simpler than the one described above. We simply take each formula of the knowledge base KB and map all constants into their equivalence classes. Some of the terms will collapse into one (for example in the case of disjunctions) and in the result we will always obtain a simpler theory. These is in fact a pure *language abstraction.*

Example 6

If we take the formula from the above example we will get the following second order reconstruction

$$[KB]_R' = \exists_x P([a],x) \wedge P([b],x) \wedge P([c],x)$$

where $[a]$, $[b]$, $[c]$ are new name constants for equivalence classes

I

It is easy to see that both transformations lead to the proper R-reconstructions of the knowledge base both in the first order and in the second order case.

V GLOBAL ERROR OF THE APPROXIMATION

Here we will evaluate the error which arises from assuming the R-reconstruction of the KB instead of KB itself. We want to characterize the set of formulas of L* which are preserved under domain abstraction. Let D_L be the set of constants of the language L* and let U_L be the set of unary predicates in L' corresponding to equivalence classes. Without loss of generality we assume that all our predicates are untyped, i.e., all database constants can occur on all positions of the predicates. We will construct a sublanguage L_0 as follows:

1. For any database predicate p and unary predicates u,,...u where each u is either a predicate corresponding to equivalence class or a universal domain predicate $D(x)$[1] ,the formulas of the form:

$\exists_{x_1} \ldots \exists_{x_n} u_1(x_1) \wedge \ldots u_n(x_n) p(x_1, \ldots x_n)$ are in L_0^*

2. If A and B are in L_0^* then so are $A \wedge B$ $A \vee B$ and $\neg A$
3. Nothing else is in L_0^*

Given database KB and equivalence relation R by ext*ended* database we will mean the database KB extended by the unary predicates corresponding to the equivalence classes.

Lemma 4

The set of all formulas preserved by domain abstraction R is equal to the set of all formulas of L_0^* which are consequences of the extended database. In other words the difference between the KB and its R-reconstruction can only be detected by the formulas out of L_0^*

I

Notice that the language L_0^* is in principle a propositional language generated by quantificational atomic formulas. Disallowing arbitrary existential quantification is the consequence of the fact that there are no free variables in the formulas of L_0^*.

In a similar way we can obtain the estimation of error in terms of the abstracted language L_e.

V.I. Domain Abstraction and Approximations

The above considerations provided an error for the semantic consequence relation $\models_d$. For the queries belonging to the "error set" domain abstraction is not expected to give correct values: What will be the relation of the answers to queries computed according to the first order and second order interpretation? Let σ be a query, R be an equivalence relation and let σ_{KB} denote an answer to this query in the knowledge base KB, while σ_a denote the R-answer returned to σ treated as first order interpretation and σ_u the R-answer returned due to the second order interpretation. We have the following lemma:

Lemma 5

For any knowledge base KB, any abstraction R and any query σ

$$\sigma_a \subseteq \sigma_{KB} \subseteq \sigma_u$$

That is σ_a is a sound but not always complete approximation, while σ_u (The second order interpretation) is complete but not always sound approximation, i.e. in general it will return more tuples then necessary. If, however σ belongs to L * then both answers will coincide with σ_{KB}.

I

The analysis of local error would tell us "how close" are these approximations to the real a n s σ_{KB}.r W e will do it in the full version of the paper.

In the next section we will discuss the influence of domain abstraction on deduction process.

[1] D(x) it true iff x belongs to the domain D of the knowldgebasr

VI R-resolution

Here we would like to consider simple *deductive* database built from atomic facts and universally quantified Horn formulas without function symbols. Let DB denote the set of atomic facts of this database and let Σ denote the set of Horn rules. In database theory, given a query Q we can either look at the database as a single theory T or modify the query Q to the form Q^{Σ} in such a way that Q^{Σ} can be directly evaluated on DB forgetting about the formulas from Σ. These two ways are equivalent, although the first one treats rules as a part of the database while the second one in treats them as a part of query. In abstracted database these two ways are no longer equivalent. The incorporation of rules into the query leads to a much less costly evaluation which we will call *R-rtBolution.* In R-resolution (R, from the equivalence relation R) all formulas before being transformed to clausal forms have all variables renamed, so no variable occurs twice, and proper equality conditions are added. Then all formulas and negation of the query are transformed to the clausal form and standard resolution is performed with one slight modification of great computational consequences: all equality conditions imposed between two variables fail. This condition can be weaken again, if some additional conditions are imposed on equivalence relation R (selectivness of R on some subdomains, etc). In general, in this approach, the query Q in the presence of deductive rules Σ is treated as Q^{Σ}. By the *R-an$wer* to Q we will understand the answer to Q^{Σ} computed from the R-reconstruction of the DB (the set of atomic facts).

Example 7

Let Σ be the following set of rules:

$$Pxy \wedge Qyz \rightarrow Wxyz$$

$W_Q xyz$—Wxyz, where P, Q and W_0 are predicates whose extensions are stored in the database and W is a derived predicate. Let our query have a form {xyzrWxyz}. In order to answer this query we can either treat the whole database (both facts and the above rules) as one theory, or modify a query Q to the form $Q^{\Sigma} = W_0xyz \vee (Pxy \wedge Qyz)$. These two ways are equivalent, unless domain abstraction is applied. With domain abstraction and no knowledge about selectivness of R, the second disjunct of Q^ will fail (unification will fail). If we treat the database as a single theory, including rules, then the R-reconstruction of it will lead to the different result and in fact will not be computationally attractive (effectively, it will require computing a fixpoint of the database before the R-reconstruction, which could be very expensive) I

The general failure of unification, in case we know nothing about R makes R-resolution very easy. Even, if we have some partial knowledge about selectivity of R, R-resolution will still be easier than the standard resolution. A short analysis is provided in the next section:

VI.1. Complexity of Abstracted Reasoning

There are two major computational advantages of abstraction: The R-reconstruction of the theory T is simpler than T itself and the reasoning is much less demanding because of the limited unification. In this way we obtain substantial computational improvements, which will be illustrated for two types of theories: Horn theories without functions and Horn Theories with function symbols.

Horn Theories without function symbols

For a Knowledge base intensions in the form of sets of Horn Clauses without function symbols any query could be processed in time which is a polynomial function of the size of the database [Vardi 82]. We will demonstrate here that under domain abstraction a very large class of queries can be processed in time which is a *linear* function of the database size. Let us first define the notions of a *proper* query and of a *proper* rule.

Definition

By the *proper* query Q we mean a query such that in any conjunct of Q with more then one database literal involved (i.e when the predicate is the database predicate) there is an equality literal $(x=y)$, where x occurs in l_1 and y occurs in l_2.s

By a *proper* rule we mean a universally quantified Horn rule, which either has a single literal in it's body or, if it has more than one literal, for any literal I in the body there is some other literal *Y* such that 1 and i" share the same variable.

This definitions eliminates expensive and rather rare "cartesian product" like conjunctions of the *form* $R(x,y) \wedge S(u,w)$. It still contains a very large family of practical queries and rules.. We have now the following easy fact:

Fact 1

Let Σ be a knowledge base intension built from proper Horn Clauses without function symbols and let Q be a proper query then the R-answer to a query Q can be generated in time which is *linear* with respect to the database size.

Proof

As we have pointed out before "pure" equality conditions of the form $(x=y)$ are always evaluated to *False" under abstraction. Therefore, if such conditions occur in conjuncts the whole such conjuncts will be evaluated to false. Hence, the only parts of the query which will "get through" will be single literals and disjunctions or existential quantifications of them. Such queries however can be clearly processed in the linear time with respect to the size of the database. ■

More interesting situation occurs, when R is partially selective. We will discuss it in more detail in the full version of the paper.

We will now discuss a situation when the function symbols are present:

Function Symbols and Abstraction

It is well known that the problem of query processing is undecidable in the general case when function symbols are. present in the database. Can domain abstraction help to make this problem more tractable?

In the same way as before we will distinguish two different interpretations, first and second order. Under the first order interpretation, when nothing is known about the equivalence relation, unification will still fail uniformly, as in the function-free case. In the same trivial way as before, we will always be able to compute an answer to a query (in most cases in time which is a linear function of the database size). A more interesting case occurs when the equivalence relation is partially selective. For instance, in case of natural numbers, the equivalence relation can be selective for all natural numbers up to n_0 and map all larger numbers into one equivalence class called "LARGE". Such an abstraction will preserve all formulas which were proved "using small numbers" and will distort all formulas which "depend" on large numbers. This could be viewed as another interpretation of some "intuitionistic" principle that proofs using large nonconstructive numbers are invalid.

Situation is different when one considers a second order interpretation. Here, in order to have a well defined Herbrand Universe, we want R to be a *congruence* relation with respect to all functions, namely we want the following formula to be true:

$$(*)\ [a] = [b] \rightarrow f([a]) = f([b])\ ?$$

Indeed, otherwise f([a]) will even not be defined. If (*) is the case and R has a finite number of equivalence classes, then we can effectively compute answers to queries under second order interpretation (i.e. treating equivalence classes as constants). This is the case because a new Herbrand universe (built from equivalence classes) is finite.

VII LIMITED INFERENCE

Given a database KB in a predicate language L we can treat L as abstraction of some unknown "prelanguage" L_0. This will result in a very cautious treatment of object identity in the database, namely by treating all constants as equivalence classes of some unknown equivalence relation R. In more intuitive terms this treatment corresponds to somehow impersonal interpretation of names of objects: Instead of saying "Smith" we will say "Some Smith" and instead of saying "Smith brings an apple" we will say "Some Smith brings Some apple". The resulting reasoning will have low complexity characteristic to the, described before, abstracted reasoning. We could gradually introduce more information about R (selectivness etc) and get better and better approximations of the "real" answers to queries σ, both in terms of already discussed upper - σ^* and lower - σ_* approximations

VIII OTHER TYPES OF ABSTRACTION

In [Imielinski 87] the different types of abstractions were introduced. If we define abstraction equivalence relation by fixing some formula $\mathbf{Q(x)}$ and making two models: m_1 and m_2 equivalent iff the diagram of $\mathbf{Q(x)}$ (i.e. roughly speaking the set of all constants c such that Q(c) is true in the model) is identical in m_1 and m2 This formula $\mathbf{Q(x)}$ is called a *view.* It turns out that the R-reconstruction of any theory KB is equal to the set of all consequences of KB which can be constructed using formula Q(x) as the basic predicate and all logical connectives and quantifiers. This is discussed in detail in [Imielinski 87].

DC CONCLUSIONS

We introduced here new notions of errors in logic and studied one particular approximation, called domain abstraction, as an illustration of these formal notions. We argued that the notion of abstraction provides meaningful approximate method, which is computationally attractive with a clear notion of error. It can be used for a preliminary computation of the answer to a query, which can then be followed, if necessary, by some precise procedure. The notion of abstraction is much more general than the specific domain abstraction introduced in this paper. We indicated this in the last section of the paper. Investigating different types of abstractions AS approximation mechanisms in logical reasoning is a promising area for further research.

References

(Fapin 851 Fagin,R and Hal pern, J.
Belief, Awarness and Limited Reasoning.
In *Proceedings of IJCAJ 85.* 1985.

|Hob; 85 Jerry R.Hobbs.
Granularity.
In *Proceedings of IJCAJ 85.* 1985.

[Imielinski 85| Imielinski,T.
Abstraction in Query Processing.
December. 1985.

imiei nski 87 ImielinskLT.
Relative Knowledge in the Distributed Database.
In *Proceedings of the ACM Symposium on Princn'ce of Database Systems.* 1987.

[Konolidge 85] Kurt Konoliige.
A Computational Theory of Belief Introspection
In *Proceedings of IJCAJ 85.* 1985.

Lakemeyer 86] Lakemeyer,G.
Steps Forwards a First Order Logic of Explicit and Implicit Belief.
In *Proceedings of the Conference on Theoretical Aspects of Reasoning about Knowledge.* 1986.

[Patel-Schneider 85j
Peter F. Patel-Schneider.
A Decidable First Order Logic for Knowledge Representation.
In *Proceedings of IJCAJ 1985.* 1985.