ISSN:0975-9646

Tractable Form of Non-Monotonic Reasoning to Generate Belief Set

Dipanjan kumar Dey

Department of Computer Science & Engineering,
Assistant professor of Prajnanananda Institute of technology and management (PITM)
Dist-Kolkata, West Bengal (India)

1. ABSTRACT

Nonmonotonic reasoning is intended to apply specifically in situation where the initial information is incomplete. The most important property of traditional system is monotonicity, i.e. addition of new facts to the database or to the theory does not result in any previous fact being retracted. There does not exist any inconsistency between the old statements and newly added statements and is assume that situation do not change. But in real world problem situation changes and so many new assumptions are generated. A monotonic reasoning system cannot work efficiently in real life environments because information available is always incomplete. These problem can be solved using nonmonotonic reasoning.

Nonmonotonic reasoning tend to be introduced proof theoretically and little attention is paid to their semantic characteristics or their computational tractability.

I present a different approach for construction of consistent belief set using least fix point semantics, declarative semantics and procedural semantics.

Keywords: *Nonmonotonic Reasoning, Fixpoint, Disposition, Extended Herbrand Base, Resolution,*least fix point semantics, declarative semantics and procedural semantics.

2. INTRODUCTION

Development of tractable form of nonmonotonic reasoning to generate belief set from a database which is a combination of disposition and proposition.

Classical logic was developed for mathematical formalisation of human reasoning. One of the major aspects of classical logic is its monotonicity property, which tells, if a formula P is derivable from a set of premises Q, then P is also derivable from each superset of Q. However, human commonsense is frequently nonmonotonic, i.e. in many cases conclusion, drawn on the basis of present knowledge, is given up in the light of further information. For instance, we know that birds can fly. Given the information that Tweety is a bird, we conclude that or common people will understand that Tweety flies. Now if we get further information about Tweety, it is not necessary that Tweety should fly because or a variety of reasons that Tweety is a penguin,Tweety'swings are broken,Tweety is too weak to fly,Tweetyis in caged, then we have to withdraw our previous conclusion and revise it by saying that Tweety doesn't fly. Most importantly, this belief revision is done without invalidating any of our premises. This form of logic, that allows us to invalidate our old conclusions are called 'Nonmonotonic' logic and it's more suitable in the field of commonsense reasoning than its monotonic counterpart.

Non-monotonic reasoning systems are more complex than monotonic reasoning systems. Monotonic reasoning systems generally do not provide facilities for altering facts, deleting rules because it will have an adverse effect on the reasoning process.

3. MCDERMOTT AND DOYLE APPROACH

McDermott and Doyle [12, 13] proposed that, nonmonotonic logic (NML) must consider not only the classical derivability of formulas but also the consistency of formulas for drawing inferences. The concept can be explained with a simple example [5]:

German typically drinks bear

which can be syntactically represented as follows:

∀. German(x) Λ MDrink_Beer (x) ⊃Drinks_Beer(x).

i.e. if x is a German and it is consistent to assume that x drinks beer then x drinks beer. The modal operator 'M' is used to represent consistency. For any predicate p, Mp stands for 'it is consistent to assume that p' and it is defined as

If p is not derivable from given premises then infer Mp.

This 'M' operator bears the essence of nonmonotonicity.

Based on this concept McDermott and Doyle illustrated the proof-theoretic procedure for constructing fixed point or belief set from a given set of premises.

In Autoepistemic Logic (AEL) Moore [14], instead of considering the consistency of formulas, tried to formalise the reasoning of an ideal agent having both positive and negative introspective capabilities. This means that the agent knows that he knows p, whenever he knows p and he knows that he doesn't know p, whenever he doesn't know p.

In AEL the statement 'German typically drinks beer' is represented as,

∀. German(x) Λ L Drinks_Beer(x) ⊃Drinks_Beer(x).

Where, 'L' is read as "it's believed that". In other words M in NML is replaced by L in AEL. Based on this concept Moore also calculated the proof-theoretic fixpoint.

3.1 *Limitations of McDermott and Doyle approach*

McDermott and Doyle's Nonmonotonic Logic does not fully capture the notion of consistency even though it is based on the modal operator M which denotes "is consistent". Consider the theory T4 has only one fixed point containing ¬p. But {¬Mp} is inconsistent as it does not have any fixed point. Adding Mp to any set of formulae renders the set inconsistent, so a set containing Mp cannot be a fixed point, but ¬p is not derivable with any other assumptions besides Mp.For example, If the following assertions are introduced: the following fixed – point can be proved: If the following axiom is addedthen cannot be proven. It is seen that is consistent with a theory and it is false, that is, is not inconsistent with . So, Nonmonotonic Logic is inconsistent.

4.DISPOSITION

Definition 4.1: A disposition is a proposition that is preponderantly but not necessarily always true.

The disposition are usually interpreted as 'if p, then q' for example
the disposition 'Birds fly'.
It is interpreted as
if x is a bird then x flies
Formally this can be written as:
( ) ∀x (usually) Fly(x) ←Bird(x).

We characterize dispositions by means of two sets: (i). the usual set and (ii). the exception set. The usual set corresponding to a disposition consists of the elements that satisfy the isposition, i.e elements for which the disposition manifests itself. The exception set contains the elements for which the disposition doesn't hold. These elements are the cause of nonmonotonicity. The rest of the literature for simplicity. This clause is to be called dispositional clause.

4.1 *Inference Rule for Dispositions:*

The modus ponens rule that is used in first order logic is also valid for dispositions. This inference rule is called dispositional modus ponens.
Given the clauses
1. (usually) Fly(x) ←Bird(x)
and 2. Bird (Tweety)
we get 3. (usually) Fly (Tweety).

In the above example the clause 1 is a disposition. Next Bird (Tweety) (in 2.) is a ground atomic formula. Using dispositional modus ponens on these two formulas we get another disposition (usually) Fly (Tweety), which can be represented as Tweety Flies(d). This disposition doesn't contain any free variable. Thus we refer to this kind of dispositions as dispositional ground atoms. Thus it's evident that by dispositional modus ponens we get another disposition or more precisely a dispositional ground atom. This dispositional ground atom can be considered as dispositional fact, in analogy to the facts in predicate logic.

5. PROPOSITION

Propositional logic is the simplest logic_illustrates basic ideas using propositions
P1 , Birds fly
P2 , Today it is raining
P3 , This automated reasoning course is boring Pi is an atom or atomic formula
Each Pi can be either true or false but never both.
The values true or false assigned to each proposition is called truth value of the proposition

6. LIMITATION OF MONOTONIC SYSTEM

Logic base system are monotonic in nature i.e., if a proposition is made which is true, it remains true under all circumstances. All theorems are proved by this methodology only but in real life world problem situations changes and new assumptions are generated. That all statements made do not necessary mean that they are correct under all circumstances.

Whenever we make a statement, we do not make it in a ad hoc fashion. The statement is made by manipulating a set of beliefs. Experts predict, diagnose and perform majority of their mental activity by relying on their beliefs. It is possible that during the course of action, events may take place which can either enhance the beliefs or reduce the dependency on the beliefs already existing.

This problem can be solved using non-monotonic reasoning. A monotonic reasoning system can not work effectively in real life environment because

- Information available always incomplete
- As process goes by, situations change and so are the solutions.
- Default assumptions are made in order to reduce the search time and for quick arrival of solutions.

7.BASIC CONCEPTS OF NON-MONOTONIC REASONING SYSTEMS

To understand this let us take an example , if we say that Rohini is a bird, the conclusion that is arrived at (default) is that Rohini can fly. But on the other hand, it is not necessary that Rohini should fly becauseor a variety of reasons similar to those given below :

- Rohini could be an Ostrich.
- Rohini's wings are broken.
- Rohini is too weak to fly.
- Rohini could be caged.
- Rohini could be a dead bird etc.

As one makes a statement like Rohini is a bird, people assume that it can fly. If another statement like Rohini in an ostrich , people retract the assumptions that were made.

Lot of day to day activities involve such instances wherein assumptions that have been made are forced to be withdrawn by the occurrence of an event or by getting a new piece of information.

Why do humans make such assumptions as done in Rohini example. The reason could be that they identify such statements with the most likely characteristics of the object under consideration. Based on the most likely characteristics one can make some statements like the following:

- Indian Railway maintain punctuality.
- Indian Airlines regularly operate their flights.
- Letters are delivered in time.
- Telephones are working properly and no cross-talks.

8.PREDICATE

A relation that binds two atoms together for example Ram likes aeroplanes. Here like is predicate and two atoms Ram and aeroplanes. Symbolically predicate-like (Ram, aeroplane)

9. FACTS

A facts must start with a predicate (which is an atom and ends with a fullstop.)

10.RULES

A rules is a predicate expression that uses Logical implication ( ) to describe a Relationship among facts.
For example

A= {Fly(x) ← Bird(x),
Bird (Tweety) ←
Bird(x) ← Penguin(x),
Give _egg(x)← Bird(x)
Penguin (Fred) ←

Fly(x) ←Penguin(x), }

Here basically two facts
1 .Bird (Tweety)
2. Penguin (Fred)

And four types of rules:--
1. Bird(x) ←Penguin(x)
2. Gives_egg(x)← Bird(x)
3. Fly(x) ← Penguin(x)
4. Fly(x) ← Bird(x)

11.BELIEF SET :

This is usually a set of beliefs which justify their action.
Belief set = Proposition + Disposition
Belief set is a subset of ground attom.

12. HERBRAND'S APPROACH:-

A very important approach to mechanical theorem proving was given by Herbrand in 1930.By definition; a valid formula is that is true under all interpretations. Herbrand developed an algorithm to find interpretations that can falsify a given formula, that is, instead of proving a formula is valid, they prove that the negation of the formula is inconsistent.

13. RESOLUTION

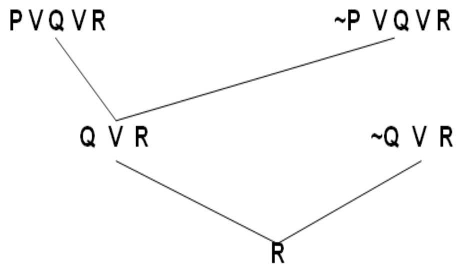
The resolution Principle is "Given any two clauses A and B, if there is a literal P1 in A which has a complementary literal P2 in B, delete P1 and P2 from A and B and construct a disjunction of the remaining clauses. The clause so constructed is called the resolvent of A and B."

For example, consider the following clauses
A : P V Q V R
B : ~P V Q V R
C : ~Q V R

Clauses A has the literal P which is complementary to ~P in B. Hence both of them are deleted and a resolvent(disjunction of A and B after the complementary clauses are removed) is generated. That resolvent has again a literal Q whose negation is available in C. Hence resolving those two, one has the final resolvent.

A : P V Q V R	(given in the problem)
B : ~P V Q V R	(given in the problem)
D : Q V R	(resolvent of A and B)
C : ~Q V R	(given in the problem)
E : R	(resolvent of C and D)

It is possible to picturise the path of the problem using a deduction tree.



14. BASIC CONCEPT OF DEDUCTIVE DATABASES

A deductive database may be defined as a triple DB = <C, P, I>. C is a finite set of nonlogical symbols (constant and predicate) that define a specific first-order language. It is assumed that C has at least one constant symbol and at least one predicate symbol. P consists of a finite set of axioms in the language and may contain metarules (that is, rules not expressible in the language) as well. I is afinite set of sentences in the **language,** the integrity constraints, that must be satisfied by the database. Updates to the database typically involve P only; the updated database must still satisfy I.

15. RELATIONAL DATABASES

Relational databases are a special case of deductive databases that do not allow deductive rules. In this case, P consists of ground atomic formulas only. These formulas represent database facts: tuples in relations (rows in database tables). Practical examples of relational databases are numerous and books on databases contain many examples. Some typical relations include Supplier, Part, Employee, and Department, with appropriate tuples, such as the 4-tuple <Jones H., 31, secretary, 19000> in the Employee relation, indicating an employee's name, age,title, and salary. No specific example will be given here. The following abstract example is used for illustration.

Example 1 A relational database
C contains the constants a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, the unary predicate symbol *R1,* the binary predicate symbol *R2,* and the ternary predicate symbol *R3.*
P contains
$R1(a_1) \longleftarrow$
$R1(a_2) \longleftarrow$
$R2(a_1, a_3) \longleftarrow$
$R_2(a_2, a_4) \longleftarrow$
$R3(a_1, a_5, a_7,) \longleftarrow$
$R3(a_2, a_3, a_4) \longleftarrow$
$R3(a_8, a_1, a_7) \longleftarrow$
The tables in the corresponding relational database are R1 with the elements a_l and a_2, *R2* with the pairs $<a_1, a_3>$and $<a_2, a_4)'$, and *R3* with the triples $<a_1, a_s, a_7>$, $<a_2, a_3, a_4)'$, and $<a_8, a_1, a_7)'$.
Consider now the following queries:
1.1 $\longleftarrow R_1(x)$ (Find the x's for which the relation $R_1(x)$ is true.)

There are two answers: a_l and a_2, since $R_1(a,)$ and $R_1(a_2)$ are facts.

1.2 $\longleftarrow R_2(x_1, X_2), R_3(X_1, X_3, X_2)$

There is one answer: $<a_2, a_4, a_3>$, since $R_2(a_2, a_4)$ and $R_3(a_2, a_4, a_3)$ are facts.

1.3 $\longleftarrow R_1(a_3)$

The answer is No, since $R_1(a_3)$ is not a fact.

Relational databases are the simplest in the classification. Such theories are very important because of the prominence of commercially available relational database systems, and the development of many sophisticated concepts, both theoretical and applied, for relational databases(Ullman, 1988). While only facts can be represented directly in relational databases, Horn databasesallow for the representation of both facts and rules. This yields a significant advantage in expressivepower, as rules provide a concise way of expressing knowledge and are particularly useful in know-ledge based systems.

16. HORN DATABASES

It is the combination of facts and rules.

For Example, as in above example of facts & rules A is an Horn Database.

In a Horn database, P consists of formulas of the form B A_l, ...,A. where B, A_l, ..., A,,, are atoms (n may be 0). The non-atomic Horn formulas represent database rules. These rules constitute the deductive part of the database. Horn databases in the sense of this paper, that is, function-free, are also referred to as DATALOG programs (Ullman, 1988).

An example of a Horn database is the example involving family relationships. The predicate parent isgiven in the form of ground atoms,such as parent(mary,jim)•-. Horn definitions can then be given to define concepts such as grandparent and ancestor.

grandparent(x,y) parent(x,z),parent(z,y)

ancestor(x,y) F-- parent(x,y)

ancestor(x,y) parent(x,z),ancestor(z,y).

In words, a grandparent is a parent of a parent; an ancestor is either a parent or an ancestor of a parent. A definition such as the one for ancestor is called recursive because it appears both on the left and right hand side, that is, ancestor is defined partially in terms of itself.

The following abstract example is an illustration of Horn databases.

Example 2 A Horn database

C contains the constants $a_l, a_2, a_3, a_4, a_5, a_6, a_7, a_8$, the binary predicate symbols R_1, R_2, and the ternary predicate symbol R_3

P contains

$R_1(a_l,a_3) \leftarrow$

$R_1(a_3,a_4) \leftarrow$

$R_l(a_4,a_6) \leftarrow$

$R_2(x,y) \leftarrow R_1(x, y)$

$R_2(x, y) \leftarrow R_1(x, z), R_2(z, y)$

$R_3(a_l, a_2, a_3) \leftarrow$

$R_3(x, y, z) \leftarrow R_1(x, y), R_2(y, z)$.

Note that R_2 has a recursive definition and that the contents of R_3 are described in terms of a fact and rule. To make the example more meaningful, one may think of R_l as the parent predicate,R_2 as the ancestor predicate, and R_3 as a ternary predicate that is a special combination of parent and ancestor.

Consider now the following queries.

2.1 $\leftarrow R_2(x, y)$

There are six answers: $<a_1, a_3>$, $<a_3, a_4>$, $<a_1, a_4>$, $<a_4, a_6>$, $<a_3, a_6>$, $<a_1, a_6>$ 2.2 f-

2.2 $\leftarrow R_3(a_1, x, y)$

There are three answers: $<a_2, a_3>$, $<a_3, a_4>$, $<a_3, a_6>$

2.3 $\leftarrow R_2(a_2, x), R_1(x, y)$

There is no answer.

17. DECLARATIVE, FIXPOINT, AND PROCEDURAL SEMANTICS

Semantics deals with meaning, a relational database and a Horn database. For each database, several questions, along with answers, were given. The answers appear to be correct intuitively. In this section a more formal approach is taken to determine the meaning of a deductive database and the correctness of an answer to a query. Three standard semantics are discussed: declarative, fixpoint, and procedural. The main result of this section is that these differently defined semantics yield identical results for Horn databases. The theorems of this section come from van Emden and Kowalski (1976).

17.1 *Declarative semantics for Horn databases*

Declarative semantics is based on interpretations, as discussed in section 1. Within the framework of the basic axioms given in the previous section, the domain of every model must contain a distinct element for each constant symbol. But in logic programming in general, or if some of the basic axioms are not included, there may be many different domains for a theory. There is one domain thatisparticularly useful. The *Herbrand universe* for a set of formulas (axioms) is the set of all symbolsbuilt up using the constant symbols (and function symbols, if any). An interpretation whose domainis the Herbrand universe is called*a Herbrand interpretation. A* Herbrandinterpretationfor a set of sentences which is also a model (that is, in which the sentences are all true) is called a *Herbrandmodel.* In logic programming the Herbrand universe may be infinite because of a function symbol or infinitely many constant symbols. Using the definition of a deductive database from section 2 including the basic axioms, all models are Herbrand models modulo renaming the elements of the domain.

Another useful concept is the notion of a *Herbrandbase, HBc:* the set of all ground atomic formulas that can be formed using C. *HBc*may also be thought of as the set of all possible facts about the database. For a deductive database, with a finite set of constants and predicates, the Herbrand base is finite. The Herbrand base is always a Herbrand model for a Horn database, because it satisfies the axioms of P including the fundamental axioms. However, the Herbrand base is usually not the intended model: it is too big. In general, we do not intend all possible facts to be true. The idea is to look for small subsets of the Herbrand base that are Herbrand models, in order to make the least number of assumptions concerning what is true in the database. Theorem 2 will justify this restriction.

For Horn databases there is a unique smallest Herbrand model, the minimal model, by the following theorem.

Thorem 1 (van Emden and Kowalski, 1976). The intersection of every (non-empty) set of Herbrand models for a Horn database is a Herbrand model.

Hence it suffices to take the intersection of all Herbrand models,M_P, to obtain the intended meaning of the deductive database. The following theorem provides an important property *of* M_p

Theorem 2 (van Emden and Kowalski, 1976). For a Horn database, $M_p = \{A€HBc \mid P \models A\}$.

This theorem states that the minimal model contains exactly the atoms logically implied by P. Now consider Example 1. For this case,

$M_P = \{R_1(a_l), R_1(a_2), R_2(a_1,a_3), R_2(a_2,a_4), R_3(a_1,a_5, a_7), R_3(a_2, a_5, a_4), R_3(a_8, a_l, a_7)\}$

In general, for a relational database, *Mp*represents exactly the rows in the tables. Now consider the queries 1.1-1.3. Assuming that *Mp*is the intended meaning of the database, the answers are constants whose substitutions make the queries true in *M* . For instance, considering 1.1, *Mp*k $R_1(a_l)$ and *Mp* $R_1(a_2)$, but for any other constant c e C, it is not the case that M_pk $R_1(c)$. Hence the answers are *al* and *as*.

Next, consider Example 2. In this case,

$M_p = \{R_1(a_l, a_3), R_1(a_3, a_4) R_1(a_4,a_6), R_2(a_l, a_3), R_2(a_3, a_4), R_2(a_4, a_6), R_2(a_1,a_4), R_2(a_1,a_6), R_2(a_3, a_6), R_3(a_1, a_5, a_3), R_3(a_1, a_5, a_4), R_3(a_1, a_5, a_6)).$

So, for 2.2, $M_p \models R_3(a_l, a_2,a_3)$ & $R_3(a_l, a_5, a_4)$ & $R_3(a_1, a_5, a_6)$, but for any other constants c_1, c_2 it is not the case that $M_p \models R_3(a_l, c_1, c_2)$. That is how the three answers are obtained.

17.2 *Fixpoint semantics for Horn databases*

The second type of semantics is called fixpoint semantics. This type of semantics involves the building of the intended Herbrand model in a step-by-step process using Herbrand interpretations. The starting point is the empty set. At each step the rules of the database imply the addition of new atoms to the existing Herbrand interpretation. For instance, if $A_1, ..., A_n$ are in a Herbrand interpretation of P and $A \leftarrow A_1, ..., A_n$ is a clause in P, then A should be added to the Herbrand interpretation to obtain a new Herbrand interpretation. When no more additions are needed, a fixpoint is reached. Such a fixpoint is a Herbrand model and the goal of fixpoint semantics is to compute the smallest fixpoint as the intended Herbrand model.

The general concepts involve the mathematical theory of lattices. *A lattice* is a set with a partial ordering (essentially a less than or equal to: ≤) relation. For a lattice L and a set X ⊆ L, a € L is called an*upper bound* of X if x ≤a for all x € X. A*least upper bound a* is anupper bound such thataa' for all upper bounds a'. If the least upper bound of X exists, it must be unique and is denoted by*lub(X).* The notions of*lower bound* and *greatest lower bound* is defined in a similar but opposite manner. A lattice L is called *complete* if lub(X) and glb(X) *exist* for every subset X of L.

The set of all subsets of a set S is called the*power set* of S and is written as 2^s. For the applicationto database semantics the set S should be thought of asHB_c, the Herbrand base. Then 2^s is the set of all Herbrand interpretations. Under the subset relation $2^s$is known to be a complete lattice with lub(X) = u S_i {Si € X} and gib(X) = ∩ Si {Si E X}. The top element is S and the bottom element is *Ø*(the empty set).-The crucial aspect of the theory involves properties of transformations between Herbrand interpretations; these are mappings from 2^s to 2^s in the present setup. A mapping T: 2^s -▸ 2^s is*monotonic* if for elements I, J of the lattice I⊆ J implies T(I)⊆ T(J), and *continuous* if T(lub(X)) = lub(T(X)) for every directed subset X of 2^s. (A subset of L is *directed* if it contains an upper bound for every finite subset.)

The powers of a monotonic mapping T are defined as follows:

T ↑ 0 = Ø (The 0th power of T is the empty set.)

T ↑T i + 1 = T(T ↑T i) (The next power of T is obtained by applying T to the previous power.)

T ↑T *ω*= lub{T↑ T i ii<*ω)* (The Goth power of T is obtained by taking the lub of all finite powers.)

An element*I* €$2^s$is called*a fixpoint* of T if T(I) = I, that is, T does not change I. The *least fixpoint* of T is written as *lfp(T),* and is defined as the fixpoint which is a subset of every fixpoint. In general, lfp(T) need not exist. The following two results are well-known about lattices (see Lloyd, 1987)):

(1) If T is monotonic, then lfp(T) exists.

(2)If T is continuous, then lfp(T) = T Tω

As mentioned earlier, the connection between monotonic mappings on a power set and deductivedatabases is obtained by letting S =HB_c, the Herbrand base, in which case 2^s is the set of Herbrandinterpretations. The mapping is usually referred to asT_P, where P is the program, which, in this case, is the set of non basic axioms of P.

For a Herbrand interpretation I,

$T_P(I) = \{A €HB_c I\ A \leftarrow A_l, ..., A_n$ is a ground instance of a clause in P and $\{A_1, ..., A_{,,}\} \subseteq$I}.

$T_p(I)$ contains all immediate consequences of the rules of P applied to I. The following theorem is a crucial **result.**

Theorem 3 (van Emden and Kowalski, 1976). If the clauses of P are Horn then T_P is continuous.

Theorem 3 and Result (2) above imply that lfp(T_P) =*Tp*↑ω. Fixpoint semantics picks lfp(T_p) as the meaning of P. The following computations yield T_P↑ω for the two examples given in the previous section.

For Example 1,

T_P↑0=Ø

T_P↑1 = T_P(T_P↑ 0) = {$R_1(a_l)$, $R_1(a_2)$, $R_2(a_1, a_3)$, $R_2(a_2, a_4)$, $R_3(a_l,a_5, a_7)$,$R_3(a_2, a_3, a_4)$, $R_3(a_6, a_l, a_7)$}

T_P↑ n = T_P↑ 1 for all n≥ 1

T_P↑ω=lub{T_P↑ i I i <ω} = T_P↑1.

For Example 2,

T_P↑O=Ø

$T_P\uparrow 1 = T_P(T_P\uparrow 0) = \{R_1(a_1, a_3), R_1(a_3, a_4), R_1(a_4, a_6), R_3(a_l, a_2, a_3)\}$
$T_P\uparrow 2 = T_P(T_P\uparrow 1) = T_P\uparrow 1 \cup \{\{R_2(a_l,a_3), R_2(a_3,a_4), R_2(a_4,a_6)\}$
$T_P\uparrow 3 = T_P(T_P\uparrow 2) = T_P\uparrow 2 \cup (R_2(a_i, a_4), R_2(a_3, a_6), R_3(a_l, a_3, a_4), R_3(a_3, a_4, a_6)\}$
$T_P\uparrow 4 = T_P(T_P\uparrow 3) = T_P\uparrow 3 \cup \{R_2(a_l, a_6), R_3(a_l, a_3, a_6)\}$
$T_P\uparrow n = T_P\uparrow 4$ for all n ≥
$T_P\uparrow \omega = \mathrm{lub}\{T_P\uparrow i \mid i < \omega\} = T_P\uparrow 4$.
Note that in both cases, $T_p\uparrow\omega = M_P$. In fact, the following theorem shows the equivalence of the declarative and fixpoint semantics.

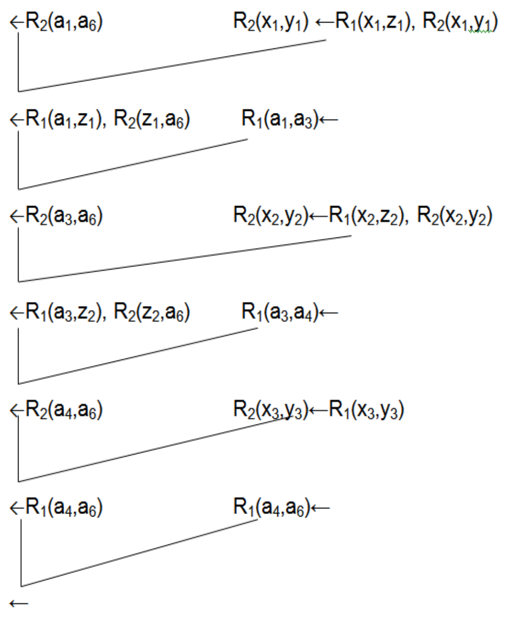


Figure 1

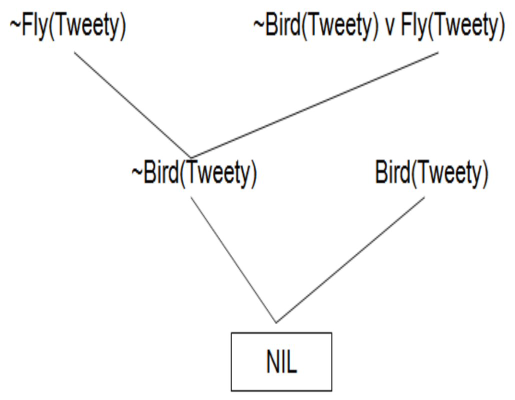
Theorem 4 (van Emden and Kowalski, 1976). For a Horn database, T_pT *to* = *M* .

17.3 *Procedural semantics for Horn databases*

The third type of semantics, procedural semantics, refers to a computational method for obtaining the meaning of a deductive database. For Horn databases SLD-resolution is used as procedural semantics. In an SLD-refutation, a Horn database P and a goal (clause of the form $\leftarrow B_1, \ldots, B_m$) are given. An SLD-derivation starts with the goal clause as the top clause that is resolved in a linear manner with clauses in P (each clause of P is given new variables not previously used in the derivation). An atom in the present goal,*B;*, is selected, and a clause in P is chosen whose head canunify with*B;* by a substitution; the new goal is obtained by replacing*B,* with the body of the clauseafter applying the substitution required for the unification. An SLD-refutation is an SLD-derivation that ends with the empty clause: ← (also written as []).
The success set of P, *Succ(P),* is defined as the set of all A e *HBc*such that P ∪{ ← A} has an SLD-refutation. In Example 1 it is easy to see that A can be in Succ(P) if and only if it is an axiom. Now consider Example 2 and take the atom $R_2(a_l, a_6)$. Figure 1 is an SLD-resolution which shows that $R_2(a_l, a_6)$ €*Succ(P)*.
The following theorem shows the connection between thedeclarative and procedural semantics.
Theorem 5 (van Emden and Kowalski, 1976). For a Horn database (using SLD-refutation) *Mp = Succ(P).*
Theorem 6 follows from Theorems 4 and 5.
Theorem 6 (van Emden and Kowalski, 1976). For a deductive Horn database $M_p = T_p\uparrow\omega$= Succ(P).

18. ILLUSTRATION 1

Let the given theory be
A= { Fly(x) ← Bird(x),
Bird (Tweety) ←
Bird(x) ← Penguin(x),
Give _egg(x)← Bird(x)
Penguin (Fred) ←
Fly(x) ←Penguin(x), }
Can Tweety fly? Write the success set.
Solution: ----
From the given problem, we have
~ Bird(x) v Fly(x)
~ Penguin(x) v Bird(x)
~ Bird(x) v Gives_egg(x)
~ Penguin(x) v Fly(x)
Bird(Tweety)
Penguin(Fred)
The ground atoms are
{ Bird(Tweety)
Fly(Tweety)
Penguin(Tweety)
Gives_egg(Tweety)
~ Fly(Tweety)
Penguin(Fred)
Bird(Fred)
Gives_egg(Fred)
~ Fly(Fred)
Fly(Fred) }
Using resolution principle, we have



Therefore Tweety can Fly
Success set :- { Bird(x), Fly(x), ~ Fly(x) }
Note : Success Set is a subset of ground atom. Constant function- Tweety, Fred.

19.ILLUSTRATION 2

Let the given theory be
A= { Fly(x) ← Bird(x),
Bird (Tweety) ←
Bird(x) ← Penguin(x),
Give _egg(x)← Bird(x)

Penguin (Fred) ←
Fly(x) ←Penguin(x), }
Find the belief set
using least fix point semantics,
Using Declarative semantics
Using procedural semantics
Solution: ----

19.1 ***Using least fix point semantics to generate belief set***
Now here basically two facts
1 .Bird (Tweety)
2. Penguin (Fred)
And four types of rules:--
1. Bird(x) ←Penguin(x)
2. Gives_egg(x)← Bird(x)
3. Fly(x) ← Penguin(x)
4. Fly(x) ← Bird(x)

The powers of a monotonic mapping T are defined as follows:

Tp 0 = Ø
Tp ↑ 1 = Tp (Tp 0) = {Bird (Tweety), Penguin (Fred)}
Tp ↑ 2 = Tp (Tp ↑ 1) = (Tp ↑ 1) U { Fly (Tweety), Give _egg (Tweety) }
Tp ↑ n = Tp ↑ 2 for all n>= 2
Tp ↑ w = lub{ Tp ↑ i | i < w } = Tp ↑ 2

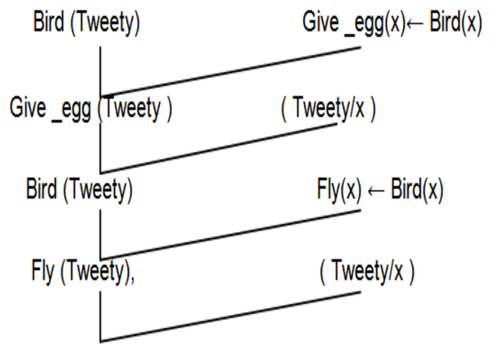
Now the belief set is
{Bird (Tweety), Penguin (Fred) Fly (Tweety), Give _egg (Tweety) }

19.2 ***Using Declarative semantics to generate belief set***
Declarative semantics is based on interpretation. Now in case of declarative semantics we use herbrand model for a horn database. But hebrand model is too big. In general all the possible facts are true. the idea is to look for small subsets of the herbrand base that are herbrand model in order to make the least no of assumption concerning what is true in the database .
Now according to theorem -1(van emden & kowaski 1976) the Intersection of every set of hebrand model (denoted by MP) for a horn database is a hebrand model
So MP obtain intended meaning of the deductive database .
Now according to theorem -2 (van emden & kowaski 1976)

MP ={ A € HBc | p ╞ A }

States that minimum model contains exactly the atoms logically implied by P. Where P are all the axioms.
Now for the above example all the atoms are – [Bird (Tweety), Penguin (Fred),Fly (Tweety),Give _egg (Tweety)]
So according to this theorem and using declarative semantics we can conclude that belief states for this example is-
[Bird (Tweety), Penguin (Fred) Fly (Tweety), Give _egg (Tweety)]

19.3 ***Using procedural semantics to generate belief set***
Procedural semantics refer to the method for obtaining meaning of a deductive database. Resolution principal is used in this process .For horn database SLD resolution is used as procedural semantics. In an SLD refutation a horn database P and a clause (of the form M1, M2…………Mn) are given .An SLD refutation that ends with empty clause: (may written as [ ]) The success set of P, Succ(p) is the set of all A € HBc such that P U { A}Has an SLD refutation
Now according to theorem -5, 6 (van emden & kowaski 1976) ---for a horn database (using SLD refutation) MP = Succ(p) and for a deductive horn database MP =Succ(p) = Tp ↑ w
These rules ensure that after using procedural semantics procedure outcome belief set is same as declarative and fix point semantics for the same example.



Now there is no need to apply SLD refutation to reach our desired goals and also there is a two constant clause [enguin (Fred) , Bird (Tweety),] which have no use in any other SLD refutation to fulfill our purpose . so this two clause normally included as a constant term in a belief set .
So after apply *SLD refutation* or using *procedural semantics* we get new belief set which is - [Bird (Tweety), Penguin (Fred) Fly (Tweety), Give _egg (Tweety)
So output belief set is - [Bird (Tweety), Penguin (Fred) Fly (Tweety), Give _egg (Tweety)]
Theorem 7 (van Emden and Kowalski, 1976). For a Horn database the declarative, fixpoint, and procedural semantics provide identical answers to queries.
i.e. For the Illustration 2 using the declarative, fixpoint, and procedural semantics provide identical answers to generate belief set and the belief set is - [Bird (Tweety), Penguin (Fred) Fly (Tweety), Give _egg (Tweety)]

20. CONCLUSION

Thus for a Horn database the declarative, fixpoint, and procedural semantics provide identical answers to queries.
i.e. For the Illustration 2 using the declarative, fixpoint, and procedural semantics provide identical answers to generate belief set. We can conclude that belief sets for this illustration 2 is
[Bird (Tweety), Penguin (Fred) Fly (Tweety), Give _egg (Tweety)]

21.APPLICATION / USES

My thesis can be applied on the following real life problem
1. Railway maintain punctuality
2. Airlines regularly operate their flight
3. Letters are delivered in time
4. Telephones work properly and no crosstalk

5. In short, human life in itself is governed by principles of nonmonotonicity.

The consistent belief set is shown to contain the same information as those obtained by the other well established techniques. Though, unlike those, the belief set constructed in this work is capable to handle the fuzziness of real world. I've also proposed a procedural semantics for the logic program containing propositions as well as dispositions and proved its soundness and completeness with respect to the unique consistent belief set.

REFERENCES:

1. G. Brewka, Non-monotonic Reasoning Logical Foundations of Commonsense, Cambridge University Press, 1991.
2. M. Gelfond, V. Lifschitz, The Stable Model Semantics for the Logic Programming, In fifth Int'l Conf. Symp. On Logic 3. Programming, pp 1070-1080, Seattle, 1988.
4. J. McCarthy, Circumscription-A Form of Nonmonotonic Reasoning, artificial Intelligence, Vol 13, pp 27-39,1980.
5. D. McDermott, J. Doyle, Non-Monotonic Logic 1, A.I. Memo 1979, Massachusetts Institute of Technology, Artificial Intelligence Laboratory 1979.
6. D. McDermott, Nonmonotonic Logic II, J. ACM, Vol 29, No 1, pp 33-57, 1982.
7. R. Moore, Semantical Considerations of on Nonmonotonic Logic, Artificial Intelligence, Vol 25, pp 75-94, 1985.
8. R. Reiter, A Logic for Default Reasoning, Artificial Intelligence, Vol 13, pp 81-132, 1980.
9. A. Tarski, A Lattice-theoretical Fixpoint Theorem and Its Application, Pacific J. Math. 5 (1955), 285-309.
10. T. Weigert, J. J. P. Tsai , A Computationally Tractable Nonmonotonic Logic, IEEE Transactions on Knowledge and Data Engineering, Vol 6, No 1, February 1994