Int. J. Open Problems Compt. Math., Vol. 2, No. 2, June 2009

Reasoning in EHCPRs System

Sarika Jain [1], N.K. Jain [2] and C.K. Goel [3]

[1] Department of Computer Application, CCS University, Meerut, 250004, India
[2] Department of Electronics, Zakir Husain College, University of Delhi, New Delhi, India
[3] Department of Computer Application, CCS University, Meerut, 250004, India

Abstract

In this paper, efforts are made to exhibit that how defaults and various constraints along with other important information required for the definition and characterization of a natural kind are represented in an EHCPR. An Extended Hierarchical Censored Production Rules (EHCPRs) system is a knowledge representation system for reasoning with real life problems and a step towards a generalized representation system. There are a number of EHCPRs at various levels of hierarchy of knowledge structure in the system, which results in a tree of EHCPRs. The GROWTH algorithm as suggested for HCPRs system is implemented. Also, the reasoning thus facilitated due to such representation, i.e., the recognition and inheritance algorithms are implemented and demonstrated with various working sessions.

Keywords: *Constraints, Default Reasoning, Extended Hierarchical Censored Production Rules, Growth, Knowledge Representation, Recognition, Inheritance*

1 Introduction

An intelligent system is readily acceptable to all, if it is highly consistent, having minimum possible redundancy in representation and high degree of integrity in the stored knowledge. Efficient access to the stored knowledge as reflected in a very prompt response, to all types of queries possible and addressable, is also an essential requirement on any intelligent system. All these objectives, which are so important of an intelligent system, are shown to exhibit through an implemented system employing EHCPRs as knowledge representation scheme. The implemented system will be regarded as EHCPRs system, which has different learning and reasoning [1] capabilities. In this paper, various possible procedures for EHCPRs system are taken up through their fruitful implementation in object oriented programming language JAVA.

As the world is expanding, the knowledge base of a future intelligent system should be as general as possible. It should be open for expansion horizontally, i.e., changes for improvement in already acquired knowledge items, as well as vertically, i.e., introduction of altogether new knowledge items. A knowledge item here is referred to be the smallest unit of knowledge in the system. An EHCPR is one such unit of knowledge, which is suggested as a knowledge item for tackling real word problems of the so-called intelligent system. An object is regarded to be of natural kind if it has some defining properties with assigned values, which can't be changed in any case. These are the necessary and sufficient conditions to be satisfied by an object to qualify as a member of that natural kind. Along with the defining properties, there is another set of distinct characteristic properties relegated with objects regarded as natural kinds. The values assigned to these characteristic properties in contrast to defining properties are subject to various constraint and default. As an example, consider a Lemon as an example of a natural kind. Though the default characteristic color for a Lemon is Yellow or Lemon to be more specific; but a rotten Lemon will still be regarded as Lemon though the color has changed to Brown. The other allowed color for Lemon will be Green but Blue is never to be considered as a color for Lemon. So, vegetable Lemon will be allowed to have any color from the set of allowed colors ("may be regarded as constraints on colors of vegetable Lemon") of Green, Yellow, Lemon, or Brown, with default color value of Lemon.

The effort here is to represent various operators along with constraints and default values in the characteristics features of an object as an EHCPR and, reasoning thereof in the EHCPRs system. The need to include two sub operators namely ***Defaults*** and ***Constraints*** in Has_Part and Has_Property operators of an EHCPR to record the Default value(s) and Constraint(s) with characteristic parts and properties is explored in this paper. The suggested Constraints sub operator is the quantitative or qualitative description of the given object. It gives the range or set of relevant possible values to the characteristic parts or properties in the EHCPR. Also the most prominent default value of the characteristic property is given in the represented EHCPR of the object, satisfying the set of constraints on it allowed value. In this paper, the representation of defaults and constraints, along with recognition and inheritance algorithms is exhibited in the implemented EHCPRs system for animals in general and birds to be more specific. The characteristics of the implemented EHCPRs system are demonstrated through example sessions. The Growth algorithm as suggested for HCPRs system [2] is implemented. The implementation is carried out using programming language JAVA.

2 Background

The “rule + exception” models provide a realistic description of the real world [3]. Whereas, hierarchies give comprehensible knowledge structure that allows managing the complexity of the large knowledge bases and to view the knowledge partitioned at different levels of details. Moreover, it provides direction to the inference engine of the system on the different important aspects [4] based on the requirement at that particular instance of time. One of such knowledge representation scheme that combines rules, exceptions, and hierarchy is Hierarchical Censored Production Rules (HCPRs) System [5, 6].

The CPRs system and its extension HCPRs system have numerous applications in situations where decision must be taken in real time with uncertain information and with incomplete data [17, 18, 19, 20, 21, 22]. Various features of the HCPRs system, including the general control scheme (GCS) have already been discussed [1, 6]. Several extensions and generalizations of the system have been proposed incorporating Fuzzy Logic [13, 14, 15], DST [16], Genetic Algorithms [17], and Neural Networks [18].

The obvious problem for AI is how to characterize, to represent, and to compute with prototypes in psychology, or to concepts like natural kinds in philosophy, where default assumptions play a prominent role [19]. A concept in general is found to possess two types of features, namely, defining features and characteristic features [20]. Defining features of a concept must be true if an item (instance or individual) satisfies (or is a member of) that particular concept (class or category) and must be relegated with the If operator as a precondition part. On the other hand, the characteristic features usually hold true for an item that is a member of a particular category. The characteristic features are allowed not to hold for an item or individual that is an instance of the particular concept represented by the head of the EHCPR. In order to distinctly represent defining and characteristic features of a concept along with its instances, an Extended Hierarchical Censored Production Rules (EHCPRs) System is presented [20, 21, 22], as an attempt toward a generalized knowledge representation and reasoning system.

An EHCPR with all these operators takes the following general form:

```
A                           {decision/concept/object}       /* As Head of rule */

If B [b1, b2, …, bm]        {preconditions (AND conditions)}

Unless C [c1, c2, …, cn]    {censor conditions (OR conditions)}
```

Generality [G]	***{General Concept}***
Specificity S [a1, a2, …, ak]	***{Specific Concepts} /* mutually exclusive set*/***
Has_Part […]	***{default : structural or body parts}***
Has_Property […]	***{default : characteristic properties}***
Has_Instance […]	***{instances}***
: γ, δ	………….. (1)

In the above general form of an EHCPR, symbol “A” denotes the decision to be taken or concept to be satisfied as the case may be, when the rule satisfies. Symbol “B” with the **If** operator is the conjunction of premises, which should be satisfied to infer the decision “A” from the EHCPR. Any exception to the rule will be checked with the **Unless** operator. The symbol “C” denotes the set of disjunction of all the censor conditions to the rule. The **Generality** information “G” in an EHCPR is the clue about the just next general concept related to the concept “**A**” in the hierarchy of concepts. The S**pecificity** information “S” is the clue about the next set of more specific concepts (goals/ decisions) in a knowledge base: which are the most likely to be satisfied after successful execution of the present EHCPR. The characteristic features (structural parts and characteristic properties) of the concept are represented through **Has_Part** and **Has_Property** operators. All the known individuals or items that are satisfied as particular instances of the concept are relegated with the **Has_Instance** operator. To represent uniformly all the instances of the general concepts given as EHCPRs in the knowledge base, as a data item in the database of the EHCPRs system, the following general form is suggested:

Head /* particular instance of a concept / name of individual object

Instance_Of	***(a general concept)***
Has_Part	***(set of actual known parts)***
Has_Property	***(set of known true properties)*** ………………….. (2)

Now consider an EHCPR from the knowledge base for the general class of, say birds:

Bird
If: *“Bipedal”, “Warm Blooded”, “Lay Eggs”*
Unless: *Nil*
Generality: *Animal*
Specificity: *Crow, Ostrich, Parrot, Penguin, Sparrow, Kiwi*
Has_Part: *{Legs: 2, Wings: Yes, Beak: 1, Teeth: No}*
Has_Property: *(Fly: Yes, Habitat: Nest)*
Has_Instance: *(Titu, Mithu, Sweety)* ……………… (3)

And a data item in the database for a particular instance of EHCPR Bird, say Titu:

```
Titu
    Instance_Of  (Bird)
    Has_Part     (Legs: 1)
    Has_Property (Fly: No)                          …………………. (4)
```

Consider a sentence that ‘Titu is a bird having one leg and cannot fly’. The information ‘Titu is a bird’ is explicitly represented in the EHCPR of Bird using **Has_Instance** operator, and using **Instance_Of** operator in data item for Titu. The override or peculiar properties of Titu are kept with the **Has_Part** and **Has_Property** operators of the data item. Titu implicitly inherits other properties from the concept of Bird, Animal, and so on in the hierarchy, to which it is connected by the **Instance_Of** and then through the **Generality** operator. The information about subsumed classes of Bird (for example, Crow and Parrot) is represented with the **Specificity** operator. Defining properties are relegated with **If** operator though **Unless** operator records censor conditions to the rule. So, in this way Meta knowledge (i.e., knowledge about knowledge) is captured through various operators in the EHCPRs system of representation.

EHCPRs systems support professionals engaged in design, diagnosis, or evaluation of complex situations. They can be used either as interactive advisors or as automated tools for converting data into recommendations or other conclusions [23]. The EHCPRs system of representation incorporates temporal, spatial, default or fuzzy information in its knowledge structure naturally and efficiently [6].

3 Constraints and Defaults

The issues of representing natural kinds, default reasoning, and context sensitive reasoning are lacking in a HCPRs based system. Extended Hierarchical Censored Production Rules (EHCPRs) System [22] employs a general representation shown [20] to have merits of all four formalisms: logic, semantic networks, frames, and production rules. The characterization of any concept (EHCPR) might require suitable constraints imposed on the various defined attributes already available in the system in the form of EHCPRs (like Habitat; Cardinality of (Legs, Wings, Beak); Color; Flyness; Level of voice; Age; Position; Duration; Time; Location etc.). For example: To characterize a Human, the constraints on age of Human may be put {Infant, Adolescent, Young, Middle-Aged, Old}. But the Age of Stars, Sun, or Universe is millions of years. So in the absolute terms, Age in itself is not having any constraints but for different objects or system it will have different constraints on its value. *The Constraints as a Range or Set of values is quantitative or qualitative description of a given concept as EHCPRs.*

Every concept in the Has_Part and Has_Property operators will have an appropriate default value as and where possible in the hierarchy along with the constraints. For example: The Has_Property operator of the concept Human has a property Age with constraints on its value {Infant, Adolescent, Young, Middle-Aged, Old} and default, say Young. Though in the instances, the default value is allowed to override by the actual present value of the individual's age. So wherever values are entered they will be checked for type or range mismatched. If some one provides Red as value of Age, it will not be accepted and appropriate action has to be initiated by the system by going through a fixed procedure.

With the inclusion of the details of Default and Constraints, the **Has_Part** and **Has_Property** operators in an EHCPR takes the following modified form:

Has_Part {Part_Concept1: [Default], [Constraints], Part_Concept2: [Default], [Constraints]...
Part_Conceptp: [Default], [Constraints]}
Has_Property {Property_Concept1: [Default], [Constraints], Property_Concept2: [Default],
[Constraints]… Property_Conceptq: [Default], [Constraints]} ………… (5)

For example, after the inclusion of defaults and constraints as sub operators in the Has_Part and Has_Property operator of the EHCPR Bird, the Has_Part and Has_Propery operators of the EHCPR Bird takes the following modified form:

Bird
Has_Part*: {Legs: 2 [0, 1, 2], Wings: Yes [Yes], Beak: 1 [0, 1], Teeth: No [Yes, No]}*
Has_Property*: {Fly: Yes [Yes, No], Habitat: Nest [Nest, Sky, Tree, Ground], Voice: Sweet [Sweet, Harsh]}* ……………… (6)

4 Implementation Details

Every EHCPR in the knowledge base is an instance of the class EHCPR defined as follows:

```
class EHCPR
{
        String          concept;
        LinkedList      preConditions;
        EHCPR           generality;
        LinkedList      specificity;
        LinkedList      censors;
        LinkedList      hasPart;
        LinkedList      hasProperty;
        LinkedList      hasInstance;
}
```

……………… (7)

There is an initializer program InitializeKnowledgeBase.java, which initializes the knowledge base. The knowledge base is a linked list of EHCPRs trees as shown in Fig 1. The EHCPRs tree with object as root EHCPR has been displayed with great detail in Fig 2.

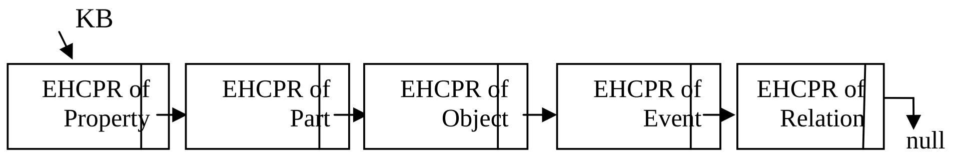
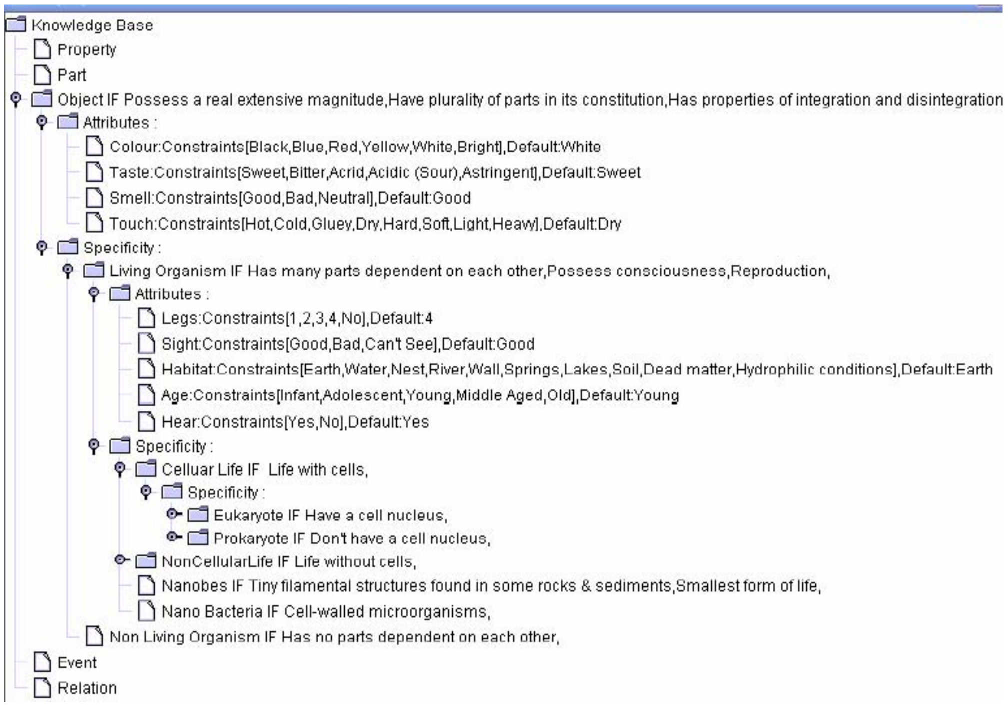
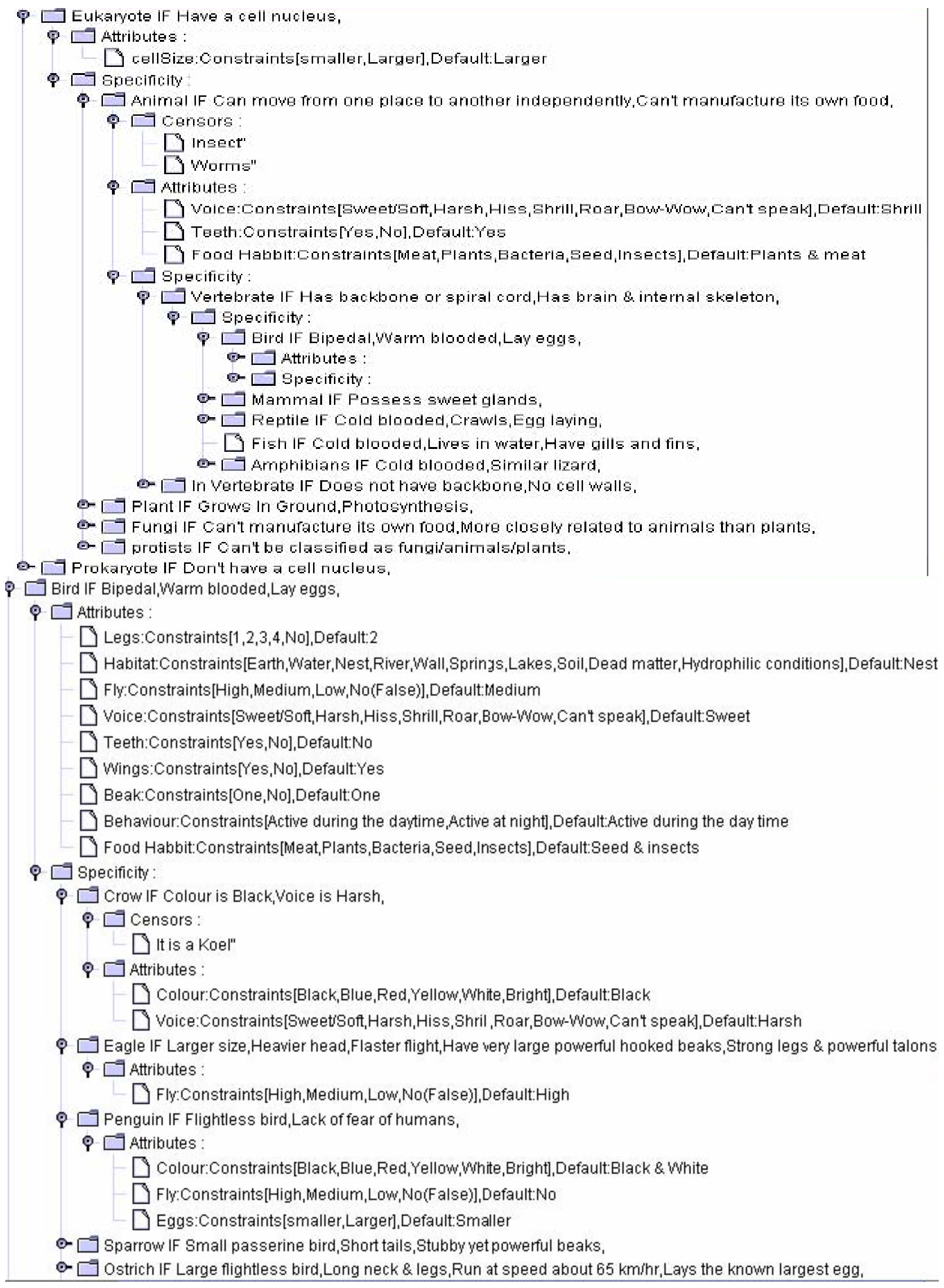


Figure 1: Knowledge Base

All the symbolic representation is available only once in the system. The multiple references to it at different EHCPRs and that too at different operators are filled or referred by employing suitable pointer to unique occurrence of that concept in the system. Our filling of operators is by reference and not by value. In network representation of displaying the knowledge base, it will be shown that operators are filled by reference/links to appropriate concept/EHCPR. All premises and censors are themselves EHCPRs.



```
Knowledge Base
  Property
  Part
  Object IF Possess a real extensive magnitude,Have plurality of parts in its constitution,Has properties of integration and disintegration
    Attributes :
      Colour:Constraints[Black,Blue,Red,Yellow,White,Bright],Default:White
      Taste:Constraints[Sweet,Bitter,Acrid,Acidic (Sour),Astringent],Default:Sweet
      Smell:Constraints[Good,Bad,Neutral],Default:Good
      Touch:Constraints[Hot,Cold,Gluey,Dry,Hard,Soft,Light,Heavy],Default:Dry
    Specificity :
      Living Organism IF Has many parts dependent on each other,Possess consciousness,Reproduction,
        Attributes :
          Legs:Constraints[1,2,3,4,No],Default:4
          Sight:Constraints[Good,Bad,Can't See],Default:Good
          Habitat:Constraints[Earth,Water,Nest,River,Wall,Springs,Lakes,Soil,Dead matter,Hydrophilic conditions],Default:Earth
          Age:Constraints[Infant,Adolescent,Young,Middle Aged,Old],Default:Young
          Hear:Constraints[Yes,No],Default:Yes
        Specificity :
          Celluar Life IF  Life with cells,
            Specificity :
              Eukaryote IF Have a cell nucleus,
              Prokaryote IF Don't have a cell nucleus,
          NonCellularLife IF Life without cells,
          Nanobes IF Tiny filamental structures found in some rocks & sediments,Smallest form of life,
          Nano Bacteria IF Cell-walled microorganisms,
      Non Living Organism IF Has no parts dependent on each other,
  Event
  Relation
```



```
Eukaryote IF Have a cell nucleus,
  Attributes :
    cellSize:Constraints[smaller,Larger],Default:Larger
  Specificity :
    Animal IF Can move from one place to another independently,Can't manufacture its own food,
      Censors :
        Insect"
        Worms"
      Attributes :
        Voice:Constraints[Sweet/Soft,Harsh,Hiss,Shrill,Roar,Bow-Wow,Can't speak],Default:Shrill
        Teeth:Constraints[Yes,No],Default:Yes
        Food Habbit:Constraints[Meat,Plants,Bacteria,Seed,Insects],Default:Plants & meat
      Specificity :
        Vertebrate IF Has backbone or spiral cord,Has brain & internal skeleton,
          Specificity :
            Bird IF Bipedal,Warm blooded,Lay eggs,
              Attributes :
              Specificity :
            Mammal IF Possess sweet glands,
            Reptile IF Cold blooded,Crawls,Egg laying,
            Fish IF Cold blooded,Lives in water,Have gills and fins,
            Amphibians IF Cold blooded,Similar lizard,
        In Vertebrate IF Does not have backbone,No cell walls,
    Plant IF Grows In Ground,Photosynthesis,
    Fungi IF Can't manufacture its own food,More closely related to animals than plants,
    protists IF Can't be classified as fungi/animals/plants,
Prokaryote IF Don't have a cell nucleus,
Bird IF Bipedal,Warm blooded,Lay eggs,
  Attributes :
    Legs:Constraints[1,2,3,4,No],Default:2
    Habitat:Constraints[Earth,Water,Nest,River,Wall,Springs,Lakes,Soil,Dead matter,Hydrophilic conditions],Default:Nest
    Fly:Constraints[High,Medium,Low,No(False)],Default:Medium
    Voice:Constraints[Sweet/Soft,Harsh,Hiss,Shrill,Roar,Bow-Wow,Can't speak],Default:Sweet
    Teeth:Constraints[Yes,No],Default:No
    Wings:Constraints[Yes,No],Default:Yes
    Beak:Constraints[One,No],Default:One
    Behaviour:Constraints[Active during the daytime,Active at night],Default:Active during the day time
    Food Habbit:Constraints[Meat,Plants,Bacteria,Seed,Insects],Default:Seed & insects
  Specificity :
    Crow IF Colour is Black,Voice is Harsh,
      Censors :
        It is a Koel"
      Attributes :
        Colour:Constraints[Black,Blue,Red,Yellow,White,Bright],Default:Black
        Voice:Constraints[Sweet/Soft,Harsh,Hiss,Shril ,Roar,Bow-Wow,Can't speak],Default:Harsh
    Eagle IF Larger size,Heavier head,Flaster flight,Have very large powerful hooked beaks,Strong legs & powerful talons
      Attributes :
        Fly:Constraints[High,Medium,Low,No(False)],Default:High
    Penguin IF Flightless bird,Lack of fear of humans,
      Attributes :
        Colour:Constraints[Black,Blue,Red,Yellow,White,Bright],Default:Black & White
        Fly:Constraints[High,Medium,Low,No(False)],Default:No
        Eggs:Constraints[smaller,Larger],Default:Smaller
    Sparrow IF Small passerine bird,Short tails,Stubby yet powerful beaks,
    Ostrich IF Large flightless bird,Long neck & legs,Run at speed about 65 km/hr,Lays the known largest egg,
```

Figure 2: The Detailed EHCPR of Object

4.1 Search and Growth

It has been shown that by simple and general snippets of code, the EHCPRs system of knowledge representation is able to acquire new pieces of knowledge and assimilate it properly in the already acquired knowledge base. There are a number of EHCPRs at various levels of hierarchy of knowledge structure in the system, which results in a tree of EHCPRs. This EHCPRs tree has the capability of continuous growth through new added EHCPRs to it at proper place. The EHCPRs tree will become richer in knowledge as time passes.

To reason around, we require a knowledge base and a database in first place. So Search and Growth algorithms as suggested in [2] for the HCPRs system have been implemented in programming language JAVA and presented here (removing the technical intricacies). They need to be modified for the EHCPRs system. The user is asked to enter the preconditions of the new HCPR(X), which is searched in the knowledge base, and if not found, it is entered in the knowledge base at its proper place.

Here every HCPR is assumed to be an instance of the class HCPR defined as follows:

```
class HCPR {
        String          concept;
        LinkedList      preConditions;
        HCPR            generality;
        LinkedList      specificity;
        LinkedList      censors;
 }
search () {
    preX = Linked List of premises of the HCPR to be searched
    Y     = Pointer to the Linked List of roots of HCPRs trees
    int i;
    Iterate in the Linked List of roots of HCPRs trees, i.e., Y
    while (there is a HCPR tree in Y) {
        currentHCPR = Pointer to next root HCPR in Y
        preY = Linked List of premises of currentHCPR
        i = subset (preX, preY)
        if (i == 0)
            return with output that HCPR has been found
            Y.remove (currentHCPR)
            Y.addFirst (currentHCPR)
        if (i == 1)
            return with output that the HCPR to be searched is in specificity list of currentHCPR
            Y.remove (currentHCPR)
            Y.addFirst (currentHCPR)
        if (i == 2)
            return with output that the HCPR to be searched is more general than currentHCPR
            Y.remove (currentHCPR)
            Y.addFirst (currentHCPR)
            if (i == 3)//HCPR to be searched is not related to currentHCPR at all so fetch next HCPRs
                          //tree
      }
}

subset (preX, preY) {
      Iterate in the premises lists preX and preY {
            while (there is an element left in both the lists) {// both preX and preY have more
premises
                  currentY = Nest Premise in linked list preY
                  currentX = Next Premise in linked list preX
                  if (currentY != currentX)
                        return (3)    // preX and prey are not at all related
            }
      }
      if (both preX and prey have no more premises)
            return (0) //   preY and preX are exactly same
      else if (preY has no more premises)
            return (1) //   Y is proper subset of X
      else if (preX has no more premises)
            return (2); //   X is proper subset of y
}

grow () {
      int i = search ()
      if (i == 0)
            return with output "the new HCPR is already present as currentHCPR. So No Growth
                                                Required"
      else if (i == 1)   // the new HCPR is in specificity list of currentHCPR
            iEqualsOne ()
      else if (i == 2)   // the new HCPR is more general than currentHCPR
            iEqualsTwo ()
      else if (i == 3) {//the new HCPR has to be added as root of new HCPRs tree in the linked list
            Y
            X = new HCPR ("New HCPR", preX, null, specX, null);
            Y.addFirst(X);
      }
}
iEqualsOne () {// the new HCPR (X) is in specificity list of currentHCPR
// preX = preX – preY Here release the premises in preX which are common to X andY. It is
//assumed that the common premises are at the beginning of the list always.
      Iterate in the premises list of Y (preY)
      while (there is a premise in preY) {
            preX.removeFirst ();
            Go to next premise in preX
       }
       Iterate in the specificity list of currentHCPR {
            while (there is some HCPR in specificity list of currentHCPR) {
                  currentHCPR = Next HCPR in specificity list
                  preY = Linked List of premises of currentHCPR
                  i = subset (preX, preY);
                  if (i == 0)
                        return with output that the new HCPR is already present as currentHCPR. So No
                        Growth Required
                  else if (i == 1)   // the new HCPR is in specificity list of currentHCPR
                        iEqualsOne ()
                        return;
                  else if (i == 2)   // the new HCPR is more general than currentHCPR
                        iEqualsTwo ()
                        return;
            }
            currentHCPR = currentHCPR.generality;
      }
       X = new HCPR ("New HCPR", preX, currentHCPR, specX, null);
       currentHCPR.specificity.addFirst(X);
}
iEqualsTwo () {    // the new HCPR is more general than currentHCPR
// preY = preY – preX Here release the premises in preY which are common to X andY. It is
//assumed that the common premises are at the beginning of the list always.
      Iterate in the premises list of X (preX)
      while (there is a premise in preX) {
            currentY.preConditions.removeFirst ();
            Go to next premise in preX
      }
      HCPR temp = currentY.generality;
      X = new HCPR ("New HCPR", preX, temp, specX, null);
      if (temp!=null)
            temp.specificity.remove (currentHCPR);
      specX.add (currentHCPR);
      currentHCPR.generality = X;
      if (temp != null)
            temp.specificity.addFirst(X)
      else {
            Y.remove (currentHCPR);
            Y.addFirst(X);
       }
}

fission (HCPR x) {
/*Say x is the recently added HCPR. The following functions are assumed to be available as
library functions
•  intersect(premisesSet1,premisesSet2) returns common premises of two sets
•  createName(x, y) generates a name for the newly created HCPR.
*/
Iterate in the linked list, x belongs to, starting from the second element as x is always the first
element in its list.
      while (there is some HCPR in the list) {
            currentY = Next HCPR in the list
            currentCommon (currentNumber) =intersect (X.preConditions, currentY.preConditions);
            if (maxNumber < currentNumber) {
                  common = currentCommon;
                  maxNumber = currentNumber;
                  y = currentY;
            }
```

```
    }
    if (maxNumber == 0)
        return; // no restructuring required
    if (y.preConditions.size () == maxNumber) {
    //premises of Y is subset of premises of X.Y is general concept of X
        preconditions of X = preconditions of X – preconditions of Y
        // Here release the premises of X which are common to X and Y
        x.generality = y;
        y.specificity.addFirst(x);
        parent.specificity.remove(x);
        return;
    }
    else if (x.preConditions.size () == maxNumber) {
    //premises of X is subset of premises of Y.X is general concept of Y
        preconditions of Y = preconditions of Y – preconditions of X
        // Here release the premises of Y which are common to X and Y
        y.generality = x;
        x.specificity.addFirst(y);
        parent.specificity.remove(y);
        return;
    }
    else {
    //X is neither a superset nor a subset of Y; a new HCPR is created with common as defining
    //properties of the new HCPR.
        newName = createName(x, y)
        // remove the common preconditions from X and Y and add them to new HCPR
        for (int i=0; i<maxNumber; i++) {
            HCPR temp = x.preConditions.removeFirst ()
            preNewHCPR.add (temp);
            y.preConditions.removeFirst ();
        }
        HCPR newHCPR = new HCPR (newName, preNewHCPR, parent, specNewHCPR);
        parent.specificity.remove(x);
        parent.specificity.remove(y);
        parent.specificity.addFirst (newHCPR);
        specNewHCPR.add(x);
        specNewHCPR.add(y);
        x.generality = newHCPR;
        y.generality = newHCPR;
    }
}
```

The search, growth, and fission algorithms have been implemented with required modifications, and other learning and maintenance algorithms are currently under implementation. These algorithms have been done for the HCPRs system and are under implementation for the EHCPRs system.

4.2 Constraints and Defaults

The first EHCPRs tree in the knowledge base has root EHCPR of Property. The Property EHCPR has in its specificity list all possible characteristic properties that an object can possess (shown in Fig 3a). The second EHCPRs tree in the knowledge base has root EHCPR of Part. The Part EHCPR has in its specificity list all possible structural parts that an object can possess (shown in Fig 3b). These parts and properties have been stored with possible constraints on their values.

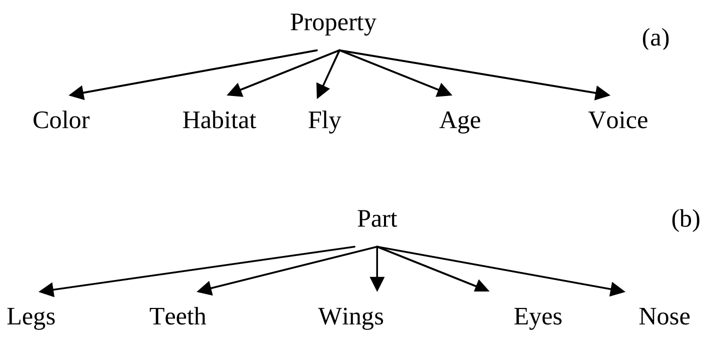


Figure 3: Parts and Properties of an Object

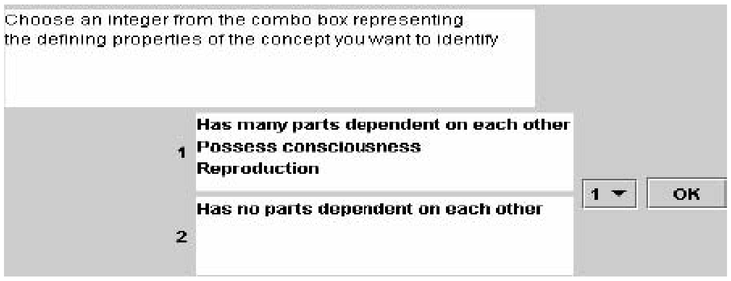
Parts and properties of an object are both characteristic attributes of an object. For the sake of implementation in this paper, we are not making any distinction between the two. From this point onwards, any part or property of an object will be called attribute in general. Every attribute has constraints on its value. All attributes are stored in the form of a linked list along with the possible constraints on their values. In the knowledge base, the third EHCPRs tree has root EHCPR of “Object” as shown in screenshots in Fig 2. The EHCPR of Object has linked list of four attributes: Color, Taste, Smell, Touch) in the hasAttributes part. These attributes have reference to the same storage where the EHCPRs of Color, Smell, Taste, and Touch are stored. The default for each attribute is stored here with the particular EHCPR, which may be overridden in the specific EHCPRs and with the actual value in the instances. The LivingOrganism EHCPR gets all these attributes of Object through inheritance and the additional attributes (Parts (Legs: 4, Age: Young), Properties (Sight: Good, Habitat: Earth, Hear: Yes)) are shown to have links through the LivingOrganism EHCPR, which is the most general EHCPR for these attributes. The Bird EHCPR has six added attributes {Parts (Wings: Yes, Beak: 1), Properties (Fly: Medium, Voice: Sweet, Behavior: Active during the day time, FoodHabbit: Seed and Insects)} and three override attributes {Parts (Legs: 2, Teeth: No), Properties (Habitat: Nest)}. The LivingOrganism EHCPR has link to attribute of Legs with default value 4. The Bird EHCPR has link to the same storage of attribute Legs but with default 2. The Reptile EHCPR does not have any link to the storage of attribute Legs, as in reptiles it is neither an added nor an override attribute. The Reptile EHCPR gets the attribute Legs through inheritance from the EHCPR of LivingOrganism. The Snake EHCPR can further override this default of Legs to 0 as King Cobra has no legs at all.

4.3 Recognition and Inheritance

Inheritance means getting the default and preconditions from the hierarchy. Recognition means matching and giving an unknown input concept a system name or classification. A child looks at a tortoise, but is not able to name the concept “Tortoise” as he/she is seeing it for the first time. The system asks him certain questions regarding the defining properties of the tortoise and keeps on giving him replies, which becomes more and more specific at each run. Refer Fig 4.

```
recognition () {
    Y = Pointer to the Linked List of roots of HCPRs trees
    Iterate in the Linked List of roots of HCPRs trees, i.e., Y
    while (there is a HCPR tree in Y) {
        currentHCPR = Pointer to next root HCPR in Y
        if (currentHCPR.concept == “Object”)
            break;
    }
    // Now currentHCPR is pointing to the HCPRs Tree with root HCPR of Object
    previousResult is an empty string
    attObject is empty string
    do {
        attributes (currentHCPR) // now attObject contains the string representation of the
                attributes of the concept “Object”
        OUTPUT: “The concept to be identified is a “+currentHCPR.concept + previousResult
                +” with attributes “+attObject
        previousResult = " which is a "+currentHCPR.concept +previousResult;
        flag = false
        Iterate in the specificity list of currentHCPR
        while (there is a some HCPR in the specificity list of currentHCPR) {
            temp = Pointer to next HCPR in the specificity list
            preTemp = temp.preConditions;
            answer = Ask the user if preTemp match the premises of the concept to be identified
            if (answer == YES)
                    currentHCPR = temp
                    flag = true
                    break
            else // if answer  == NO
                    continue
        }
    }while(flag)
}
attributes(HCPR obj){
    // attObject contains all the properties in all the general concepts of this obj up in the hierarchy
    if(obj.hasProperty != null){   // means some attributes are listed in the Has_Property operator
        of the concept obj
```

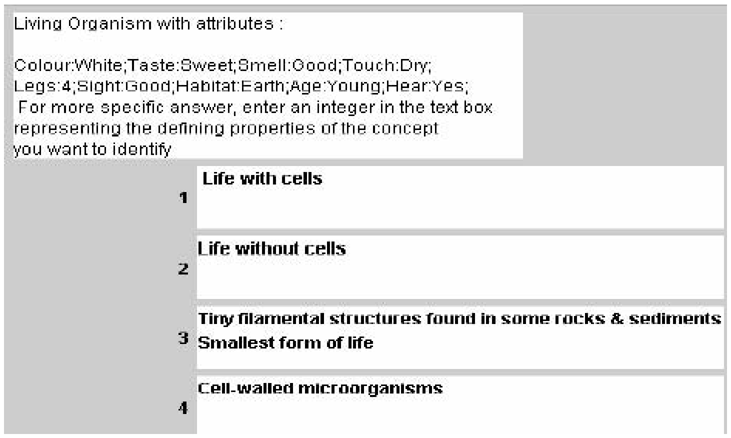
```
            Iterate in the Linked List of properties of obj
            while(there is a property in obj.hasProperty){
                  currentProp = Pointer to next property in obj.hasProperty
                  if (currentProp is already in attObject)// means it is an override property, so we need
                  to keep the new one from this point onwards
                               Remove the old default value and put the new default value
                  else //  it is an added property
                               add currentProp to attObject
            }
      }
}
```


Choose an integer from the combo box representing
the defining properties of the concept you want to identify

1 Has many parts dependent on each other
Possess consciousness
Reproduction

2 Has no parts dependent on each other

(a)


Living Organism with attributes :

Colour:White;Taste:Sweet;Smell:Good;Touch:Dry;
Legs:4;Sight:Good;Habitat:Earth;Age:Young;Hear:Yes;
For more specific answer, enter an integer in the text box
representing the defining properties of the concept
you want to identify

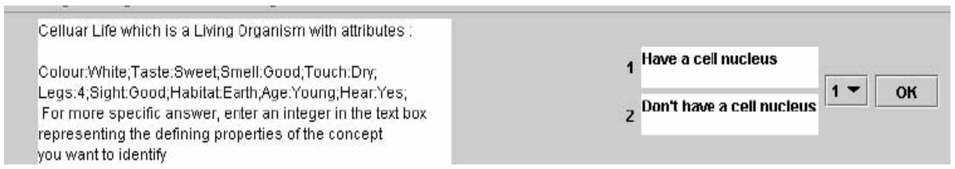
1 Life with cells

2 Life without cells

3 Tiny filamental structures found in some rocks & sediments
Smallest form of life

4 Cell-walled microorganisms

(b)

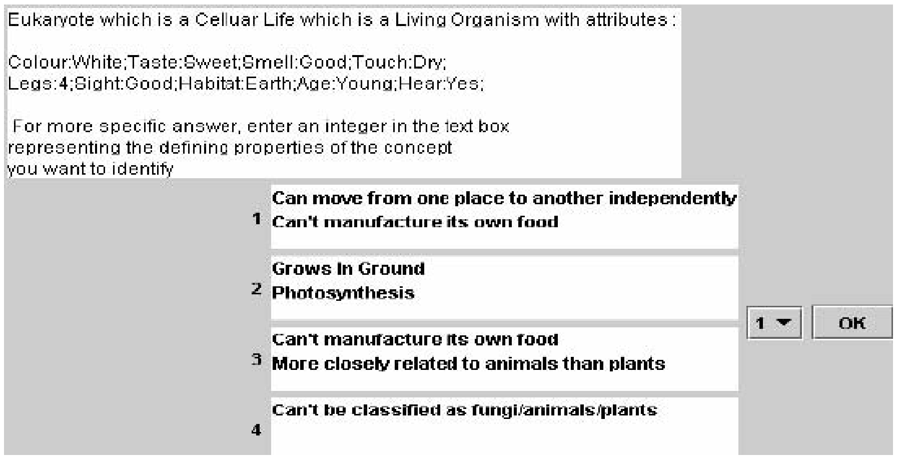

Celluar Life which is a Living Organism with attributes :

Colour:White;Taste:Sweet;Smell:Good;Touch:Dry;
Legs:4;Sight:Good;Habitat:Earth;Age:Young;Hear:Yes;
For more specific answer, enter an integer in the text box
representing the defining properties of the concept
you want to identify

1 Have a cell nucleus

2 Don't have a cell nucleus

(c)


Eukaryote which is a Celluar Life which is a Living Organism with attributes :

Colour:White;Taste:Sweet;Smell:Good;Touch:Dry;
Legs:4;Sight:Good;Habitat:Earth;Age:Young;Hear:Yes;

For more specific answer, enter an integer in the text box
representing the defining properties of the concept
you want to identify

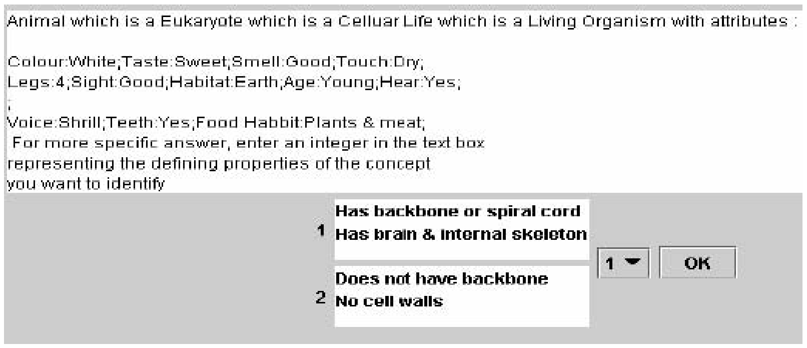
1 Can move from one place to another independently
Can't manufacture its own food

2 Grows in Ground
Photosynthesis

3 Can't manufacture its own food
More closely related to animals than plants

4 Can't be classified as fungi/animals/plants

(d)

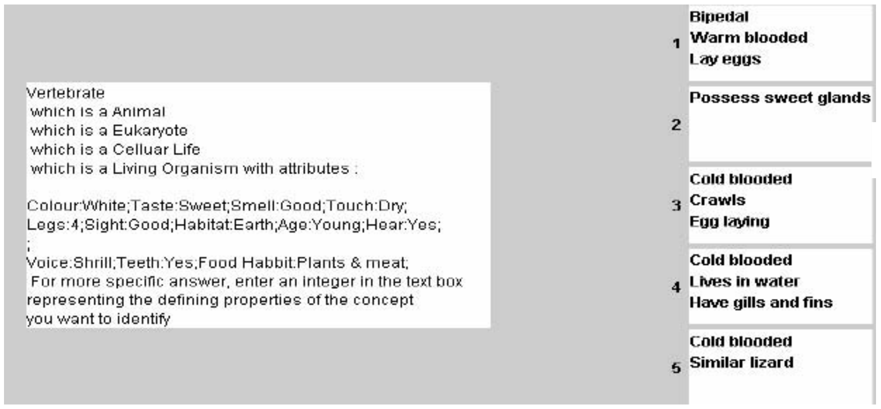

Animal which is a Eukaryote which is a Celluar Life which is a Living Organism with attributes :

Colour:White;Taste:Sweet;Smell:Good;Touch:Dry;
Legs:4;Sight:Good;Habitat:Earth;Age:Young;Hear:Yes;
;
Voice:Shrill;Teeth:Yes;Food Habbit:Plants & meat;
For more specific answer, enter an integer in the text box
representing the defining properties of the concept
you want to identify

1 Has backbone or spiral cord
Has brain & internal skeleton

2 Does not have backbone
No cell walls

(e)


Vertebrate
which is a Animal
which is a Eukaryote
which is a Celluar Life
which is a Living Organism with attributes :

Colour:White;Taste:Sweet;Smell:Good;Touch:Dry;
Legs:4;Sight:Good;Habitat:Earth;Age:Young;Hear:Yes;
;
Voice:Shrill;Teeth:Yes;Food Habbit:Plants & meat;
For more specific answer, enter an integer in the text box
representing the defining properties of the concept
you want to identify

1 Bipedal
Warm blooded
Lay eggs

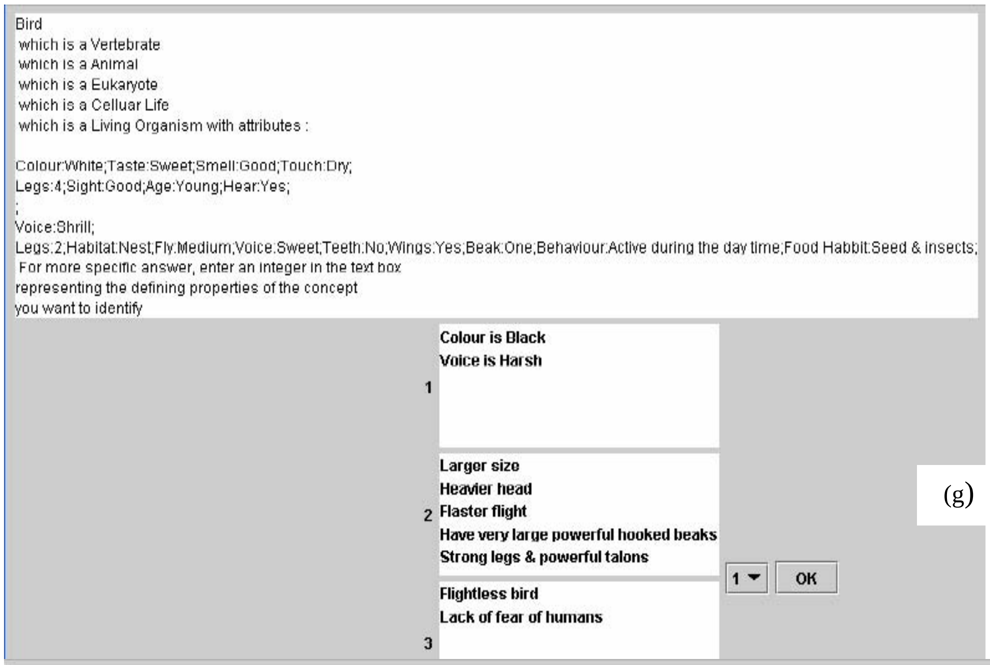
2 Possess sweet glands

3 Cold blooded
Crawls
Egg laying

4 Cold blooded
Lives in water
Have gills and fins

5 Cold blooded
Similar lizard

(f)



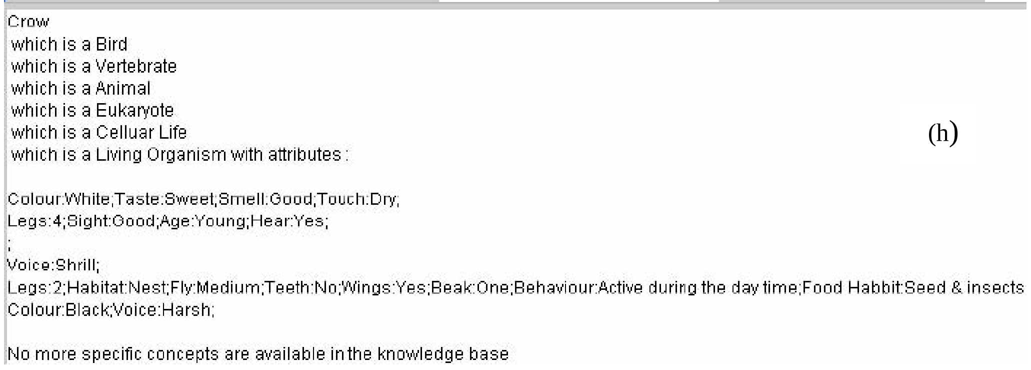
```
Bird
 which is a Vertebrate
 which is a Animal
 which is a Eukaryote
 which is a Celluar Life
 which is a Living Organism with attributes :

Colour:White;Taste:Sweet;Smell:Good;Touch:Dry;
Legs:4;Sight:Good;Age:Young;Hear:Yes;
;
Voice:Shrill;
Legs:2;Habitat:Nest;Fly:Medium;Voice:Sweet;Teeth:No;Wings:Yes;Beak:One;Behaviour:Active during the day time;Food Habbit:Seed & insects;
 For more specific answer, enter an integer in the text box
representing the defining properties of the concept
you want to identify
```

1
Colour is Black
Voice is Harsh

2
Larger size
Heavier head
Flaster flight
Have very large powerful hooked beaks
Strong legs & powerful talons

3
Flightless bird
Lack of fear of humans

(g)



```
Crow
 which is a Bird
 which is a Vertebrate
 which is a Animal
 which is a Eukaryote
 which is a Celluar Life
 which is a Living Organism with attributes :

Colour:White;Taste:Sweet;Smell:Good;Touch:Dry;
Legs:4;Sight:Good;Age:Young;Hear:Yes;
;
Voice:Shrill;
Legs:2;Habitat:Nest;Fly:Medium;Teeth:No;Wings:Yes;Beak:One;Behaviour:Active during the day time;Food Habbit:Seed & insects
Colour:Black;Voice:Harsh;

No more specific concepts are available in the knowledge base
```

(h)

Figure 4: Screenshots of Recognition of Crow

5 Open Problem

Different current projects in Artificial Intelligence, such as Thought Treasure [24], CYC [25], WordNet [26], OWL [27], etc., are successful to represent extensively large knowledge base and reasoning there forth. But they have used multiple representation schemes to represent different types of concepts. In place of employing multiple representation schemes, the EHCPRs system adopts a general representation by means of EHCPRs for any entity or concept, which is possible in the universe, whether it is real or imaginary. An EHCPR is aptly regarded as a unit of knowledge to represent any complex or simple concept employing the same general syntax and associated semantics. This general structure of an EHCPR facilitates common and hence general procedures of reasoning and learning irrespective of the domain where these EHCPRs are applied. The important feature that is lacking in the reasoning of all previous systems [24, 25, 26, 27] is that, these systems are not made for exhibiting variable precision with variable constraints on resources [such as time and memory]. Another important requirement on these systems is variable response for the same query, based on variable context of say time, location, user background, and even state of reasoning system (i.e., amount of knowledge & data and complexity of programs). An EHCPRs system is an open-ended evolving system and lots have been done and lots have to be done to make it as general as possible.

6 Discussion and Conclusion

This paper is an important and long-standing requirement in the subject area. The Extended Hierarchical Censored Production Rules (EHCPRs) system is a knowledge representation system for reasoning with real life problems and a step towards a generalized representation scheme. This work is first serious attempt towards implementing the EHCPRs system as a full-fledged system. The way in which the knowledge base has been managed provides (1) Minimum Redundancy (minimum or no duplicity of storage of any piece of knowledge), (2) Highest Consistency (always result in non contradictory results), and (3) Integrity (truthfulness) of stored knowledge and Facts. The knowledge base also provides ease of maintenance and adaptation in ever-changing and external real world situations and circumstances.

Defaults can be override with correct instances, say legs may be 1 in number for some PH category person. So cardinality is defined in EHCPR of Human and instances can select from allowed values. The advantage of this in management is that reasoning based on general information is available with EHCPRs without repetition. Though Growth algorithm is implemented with some modification but other learning and maintenance algorithms are currently under implementation.

References

[1] Sarika Jain, N.K. Jain, C.K. Goel, “Implementing General Control Scheme in EHCPRs System”, *Proceedings of Second National Conference on Mathematical Techniques : Emerging Paradigms for Electronics and IT Industries (MATEIT-2008)*, New Delhi, September 26-28, 2008.

[2] N.K. Jain and K.K. Bharadwaj, “Some Learning Techniques in Hierarchical Censored Production Rules (HCPRs) System”, *International Journal of Intelligent Systems*, vol. 13, John Wiley and Sons, Inc. North Holland, 1998.

[3] Yiyu Yao, Fei- YueWang and JueWang, "Rule + Exception Strategies for Knowledge Management and Discovery", *LNAI 3642*, Springer-Verlag, 2005, 69-78.

[4] B. Liu, M. Hu, and W. Hsu, "Multilevel Organization and Summarization of the Discovered Rules", *SIGKDD-2000*, Boston, USA, Aug 20-23, 2000, 208-217.

[5] K. K. Bharadwaj, and N.K. Jain, "Hierarchical Censored Production Rules (HCPRs) System", *Data and Knowledge Engineering*, vol 8, North Holland, 1992, 19-34.

[6] N.K. Jain, "Variable Precision Logic: A Hierarchical Censored Production Rules System", *M.Tech dissertation, School of Computers and Systems Sciences, Jawaharlal Nehru Univ.*, New Delhi, India, 1989.

[7] K.K. Bharadwaj, N.M. Hewahi, and M.A. Brandao, "Adaptive Hierarchical Censored Production Rule-based System: A genetic algorithm approach", *Lecture notes in Artificial Intelligence, No. 1159*, D.L.Borges and C.A. Kaestner, Eds., Advances in Artificial Intelligence, SBIA'96, Curitiba, Brazil, Proceedings, Springer-Verlag, 1996, 81-90.

[8] K.K. Bharadwaj and R. Varshneya, "Parallelization of Hierarchical Censored Production Rule-based System", *Information and Software Technology*, Elsevier Science, B.V. (UK), 37, 1995, 453-460.

[9] N.M. Hewahi, "Real Time Variable Certainty Systems", Proceedings of the 4th World Congress on Expert Systems, *Applications of Advanced Information Technology*, Mexico, vol 2, 1998, 692-697.

[10] N.M. Hewahi, "Real Time Variable Precision Logic Systems", in Flavio Moreira de Oliveira, (Ed.), Lecture Notes in Artificial Intelligence, *Advances in Artificial Intelligence* 1515, Springer Verlag, Germany, 1998, 201-208.

[11] N.M. Hewahi and K.K. Bharadwaj, "Bucket Brigade Algorithm for Hierarchical Censored Production Rules (HCPRs) System", *International Journal of Intelligent Systems*, vol. 11, 1996, 197-226, John Wiley and Sons, Inc. North Holland.

[12] K.K. Bharadwaj and N.K. Jain, "Towards Integrating Hierarchical Censored Production Rules (HCPRs) System and Neural Networks", SBIA'98, Lecture Notes in Artificial Intelligence, No. 1515, Berlin, Germany, Springer-Verlag, 1998, 121-130.

[13] Fadl M. Baa-Alwi and K.K. Bharadwaj, "Automated Discovery of

Hierarchical Ripple-Down Rules, Proceedings of the 23rd IASTED International Multi-Conference, *Artificial Intelligence and Applications"*, Innsbruck, Austria, 2005.

[14] Neerja and K.K. Bharadwaj, "Calculus of Fuzzy Hierarchical Censored Production Rules (FHCPRs) System", *International Journal of Intelligent Systems,* vol. 2 (1-25), John Wiley and Sons, Inc. North Holland, 1996.

[15] Fadl M. Baa-Alwi and K.K. Bharadwaj, "Discovery of Production Rules with Fuzzy Hierarchy", *Proceedings of World Academy of Science, Engineering and Technology*, vol 4, 2005, ISSN 1307-6884.

[16] K.K. Bharadwaj, Neerja, and G.C. Goel, "Hierarchical Censored Production Rules System employing Dampster Shafer Uncertainty Calculus", *Information and Software Technology, Elsevier Science B.V., 36*, 1994, *155-164.*

[17] Basheer M. Al-Maqaleh and K.K. Bharadwaj, "Genetic Programming Approach to Hierarchical Censored Production Rules Discovery", *Proceedings of the ENFORMATIKA-6*, Istanbul, Turkey, 2005, 271-274.

[18] J.D.S. da Silva and K.K. Bharadwaj, Integration of Hierarchical Censored Production Rules (HCPRs) System and Neural Networks", SBRN'98, *Proceedings of IEEE Computer Society,* Las Alamitos, California, USA, 1998, 73-88.

[19] R. Reiter, "Nonmonotonic Reasoning", *Annual Review of Computer Science*, 2, 1987, pgs 147- 186.

[20] N.K. Jain, K.K. Bharadwaj, Norian Marranghello, "Extended Hierarchical Censored Production Rules (EHCPRs) System: An Approach toward Generalized Knowledge Representation", *Journal of Intelligent Systems*, vol 9, 3-4, 1999, pgs 259-295.

[21] N.K. Jain, "Hierarchical Censored Production Rules (HCPRs) System: A Generalized Representation Scheme", *Ph.D. Thesis, School of Computers and Systems Sciences, Jawaharlal Nehru Univ.*, New Delhi, India, 1997.

[22] Sarika Jain, N.K. Jain, C.K. Goel, "A Generalized Knowledge Representation System and its Implementation: An Extended Hierarchical Censored Production Rules (EHCPRs) System", *MERI, Journal of Management and IT,* April 2007, vol. 1, pp. 88-101.

[23] Sarika Jain, N.K. Jain, C.K. Goel, 2009, "EHCPRs System and Needs of Management System", *Journal of IPEM*, vol3, Issue 1, pp. 1-17.

[24] Erik T. Muller, "Natural Language Processing with Thought Treasure, New York, *Signiform*, 1998.

[25] Douglas B. Lenat, CYC: A Large-Scale Investment in Knowledge Infrastructure, *Communications of the ACM*, 38(11) ,1995, pg. 33-48.

[26] Christiane Felbaum(Ed.), WordNet: An Electronic Lexical Database, C*ambridge, MA*, MIT Press, 1998.

[27] Peter Szolovits**,** Lowell B. Hawkinson**,** William A. Martin**, "**An Overview of OWL, A Language for Knowledge Representation**",** June, 1977, *Massachusetts Institute of Technology, Laboratory for Computer Science,* (formerly Project MAC), Cambridge Massachusetts, 02139.