Isa Hierarchy and chaining

The ISA Hierarchy

Knowledge about objects, their attributes and their values need not be as simple as in the relational knowledge. It may be necessary sometimes to augment the basic representation of knowledge with some inference mechanisms. The inference mechanisms operate on the structure of the representation. One of the most useful forms of inference is property inheritance. In property inheritance, elements of specific classes inherit attributes and values from general classes in which they are included.

We can represent "Mithu is a parrot" as

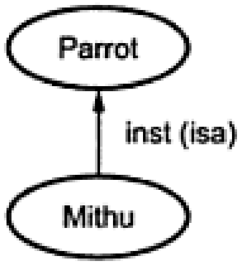

Fig. 4.2 The ISA hierachy

We can further have the same property for different parrots along with their properties like "Mithu is a parrot of green colour" and "Sithu is a parrot of green color".

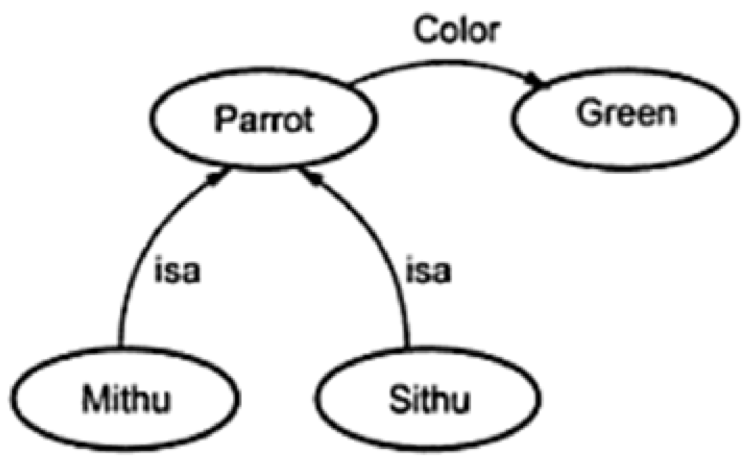

Fig. Inheritence of properties

We can further ellaborate this property inheritance. It could be said as "parrot belongs to a category of flying birds". It has properties of other birds like feathers, head, eyes etc.

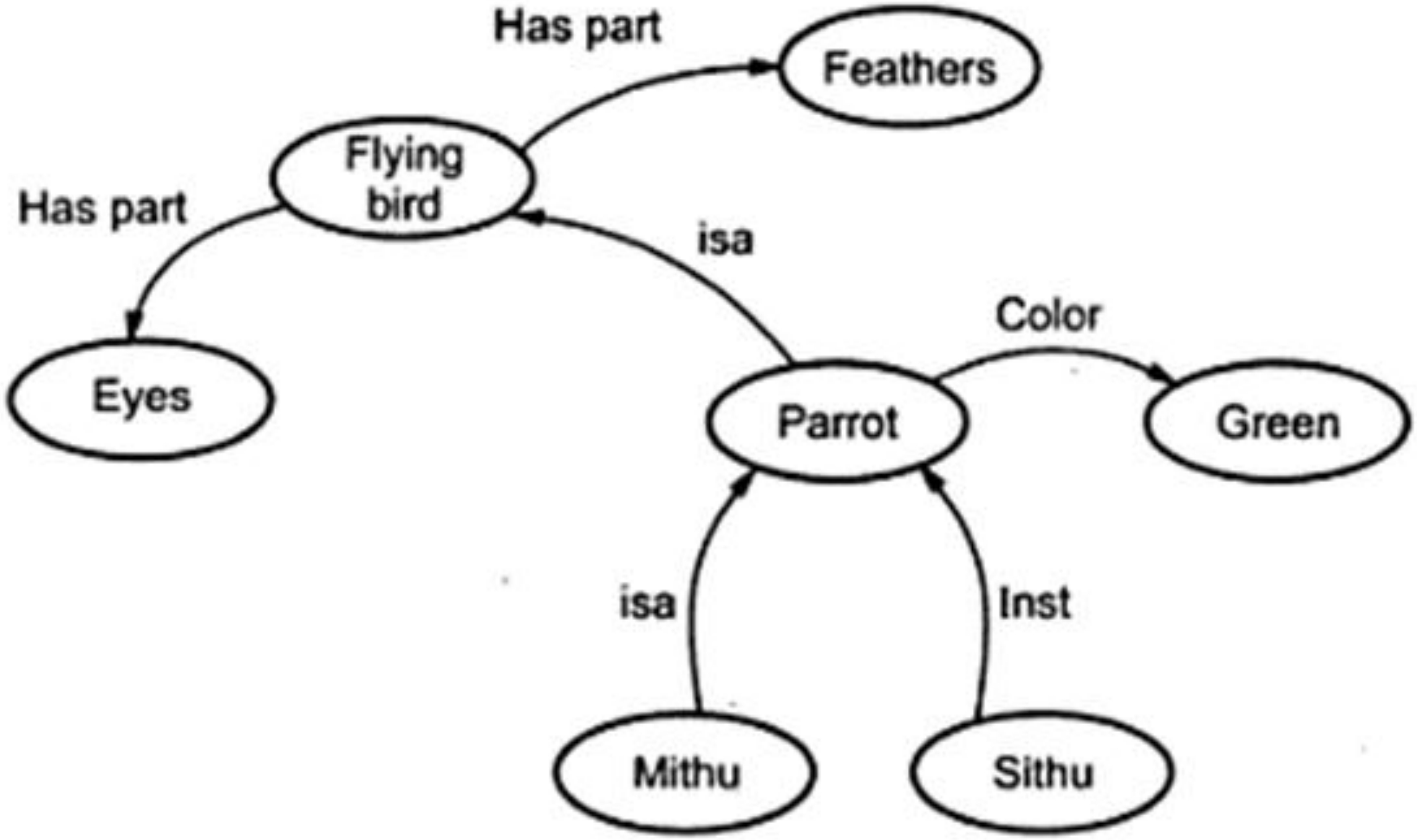

Fig. An ISA hierarchy ellaborated

The relationship which results mainly due to ISA hierarchy is class membership and class inclusion.

Represent the following facts in predicate calculus.

a) Shreekant was a man.

b) Shreekant is Hindu.

c) All Hindus are Indians.

d) Shivaji is ruler.

e) All Indians were loyal to Shivaji.

1. man (Shreekant)
2. Hindu (Shreekant)
3. $\forall x$: Hindu (x) $\rightarrow$ Indian (x)
4. ruler (Shivaji)
5. $\forall x$: Indian (x) $\rightarrow$ loyalto (x, Shivaji)

or

1. Instance (Shreekant, man)
2. Instance (Shreekant, Hindu)
3. $\forall x$: Instance (x, Hindu) $\rightarrow$ Instance (x, Indian)
4. Instance (Shivaji, ruler)
5. $\forall x$: Instance (x, Indian) $\rightarrow$ loyalto (x, Shivaji)

Algorithm to retrieve a value for an attribute of instance object:

1. Find the object in the knowledge base.
2. If there is a value for the attribute report it.
3. Otherwise look for a value of instance if none fail.
4. Otherwise go to that node and find a value for the attribute and then report it.

Otherwise search through using *isa* until a value is found for the attribute.

Computable Functions and Predicates

gt (10, 2)	lt (1, 10)
gt (15, 1)	lt (2, 15)
greater than	**less than**

gt (2+4,1)

1. Chaminda was a witch.
2. Chaminda was a Ravansenapati.
3. Chaminda was born in 20 A.D.
4. All witches are mortal.
5. All Ravansenapatis died when the war started in 39 A.D.
6. No mortal lives longer than 120 years.
7. It is now 2005.
8. Alive means not dead.
9. If someone dies, then he is dead at all the later times.

1. Chaminda was a witch.

   witch (Chaminda)

2. Chaminda was a Ravansenapati.

   Ravansenapati (Chaminda)

3. Chaminda was born in 20 A.D.

   born (Chaminda, 20)

4. All witches are mortal.

   $\forall x : \ witch(x) \rightarrow mortal(x)$

5. All Ravansenapatis died when the war started in 39 A.D.

   $Started(war, 39) \wedge \forall x : [Ravansenapati(x) \rightarrow died(x,39)]$

6. No mortal lives longer than 120 years.

   $\forall x : \forall t_1 : \forall t_2 : mortal(x) \wedge born(x, t_1) \wedge gt(t_2 - t_1, 120) \rightarrow dead(x, t_2)$

7. It is now 2005.

   now = 2005

8. Alive means not dead.

   $\forall x : \forall t : [alive(x,t) \rightarrow \neg dead(x,t) \wedge [\ dead(x,t) \rightarrow \neg alive(x,t)]]$

9. If someone dies, then he is dead at all the later times.

   $\forall x : \forall t_1 : \forall t_2 : died(x, t_1) \wedge gt(t_2, t_1) \rightarrow dead(x, t_2)$

$\neg$ alive (Chaminda, now)

(9, Substitution)

dead (Chaminda, now)

(10, Substitution)

died (Chaminda, t_1) ^ gt (now, t_1)

(5, Substitution)

Ravansenapati (Chaminda) ^ gt (now, 39)

2

gt (now, 39)

(8, Substitution)

gt (2005, 39)

(Compute gt)

nil

(a) One method to prove "Chaminda is dead"

$\neg$ alive (Chaminda, now)

↑ (9, Substitution)

dead (Chaminda, now)

↑ (7, Substitution)

mortal (Chaminda) ^

born (Chaminda, t_1) ^

gt (now – t_1, 120)

↑ (4, Substitution)

witch (Chaminda) ^

gt (now – t_1, 120)

↑ (1)

born (Chaminda, t_1) ^

gt (now – t_1, 120)

↑ (3)

gt (now – 20, 120)

↑ (8)

gt (2005 – 20, 120)

↑ (Compute minus)

gt (1985, 120)

↑ (Compute gt)

nil

(b) Second way to prove "Chaminda is dea

Conversion into Clause Form

1. Eliminate all <=> connectives by replacing each instance of the form $(P \Leftrightarrow Q)$ by the equivalent expression $((P \Rightarrow Q) \wedge (Q \Rightarrow P))$
2. Eliminate all => connectives by replacing each instance of the form $(P \Rightarrow Q)$ by $(\neg P \vee Q)$
3. Reduce the scope of each negation symbol to a single predicate by applying equivalences such as converting

   $\neg\neg P$ to P;

   $\neg(P \vee Q)$ to $\neg P \wedge \neg Q$;

   $\neg(P \wedge Q)$ to $\neg P \vee \neg Q$;

   $\neg(\forall x) P$ to $(\exists x) \neg P$, and

   $\neg(\exists x) P$ to $(\forall x) \neg P$

4 Standardize variables: rename all variables so that each quantifier has its own unique variable name.

For example, convert $(\forall x)\ P(x)$ to $(\forall y)\ P(y)$ if there is another place where variable x is already used.

5. Eliminate existential quantification by introducing **Skolem functions**.

For example, convert $(\exists\ x)\ P(x)$ to $P(c)$ where c is a brand new constant symbol that is not used in any other sentence. c is called **a Skolem constant**. More generally, if the existential quantifier is within the scope of a universal quantified variable, then introduce a Skolem function that depend on the universally quantified variable. For example,

$(\forall x)\ (\exists\ y)\ P(x,y)$ is converted to $(\forall x)\ P(x, f(x))$. f is called **a Skolem function**, and must be a brand new function name that does not occur in any other sentence in the entire KB.

Example: $(\forall x)\ (\exists y)$ loves $(x,\ y)$ is converted to

$(\forall x)$ loves$(x,\ f(x))$ where in this case $f(x)$ specifies the person that x loves. (If we knew that everyone loved their mother, then f could stand for the mother-of function.

6. Remove universal quantification symbols by first moving them all to the left end and making the scope of each the entire sentence, and then just dropping the "prefix" part.
7. Distribute "and" over "or" to get a conjunction of disjunctions called **conjunctive normal form**.

   Convert $(P \wedge Q) \vee R$ to $(P \vee R) \wedge (Q \vee R)$,

   convert $(P \vee Q) \vee R$ to $(P \vee Q \vee R)$.
8. Split each conjunct into a separate **clause**, which is just a disjunction ("or") of negated and un-negated predicates, called **literals**. Standardize variables apart again so that each clause contains variable names that do not occur in any other clause.

$\forall x$:[Roman(x) $\wedge$ know(x,Marcus) $\rightarrow$ [hate(x, Caesar) $\vee$ ($\forall$ y: $\exists$z: hate(y, z)$\rightarrow$ thinkcrazy (x,y))]

1. **Eliminate $\leftarrow\rightarrow$**

Nothing to do here

2. **Eliminate $\rightarrow$**

$\forall x$: ¬ [Roman(x) $\wedge$ know(x,Marcus)]$\vee$[hate(x, Caesar) $\vee$ ($\forall$ y: ¬ ($\exists$z: hate(y, z)) $\vee$ thinkcrazy (x,y))].

3. **Reduce Scope of ¬ to single term.**

$\forall x$:[¬Roman(x) $\vee$ ¬ know(x,Marcus)]$\vee$[hate(x, Caesar) $\vee$ ($\forall$ y: $\forall$z : ¬hate(y, z)) $\vee$ thinkcrazy (x,y))]

4. **Standardize variables so that quanifiers bind a unique value**

For eg ∀ x: p(x) ∨ ∀ x : Q(x)

Would be converted to

∀ x: p(x) ∨ ∀ y : Q(y)

∀ This step is in preparation for the next.

At this point formula is known as prenex normal form. It consists of prefix of quantifiers followed by a matrix which is quantifier free

5. **Move all quantifiers to the left of the formula**

∀x :∀ y:∀z : [¬Roman(x) ∨ ¬ know(x,Marcus)]∨[hate(x, Caesar) ∨ (¬ hate(y, z)) ∨ thinkcrazy (x,y))]

6. Eliminate Existential qualifier

∀x : ∃ y: father-of (y,x) converted to

∀x : father-of (y,x)

∃ y: President(y)

President(S1)

Solemnization

7. **Drop the Prefix**

[¬Roman(x) ∨ ¬ know(x,Marcus)]∨[hate(x, Caesar) ∨ (¬ hate(y, z)) ∨ thinkcrazy (x,y))]

8. **Convert the matrix into conjunction of disjuncts**

In our case there is no and so explore the associative property of or

¬Roman(x) ∨ ¬ know(x,Marcus) ∨ hate(x, Caesar) ∨ ¬ hate(y, z) ∨ thinkcrazy (x,y)

9. **Create a separate clause corresponding to each conjunct.**

10. **Standardise apart the variables in set of clauses.**

Convert the sentence

$(\forall x)\ (P(x)\ \Rightarrow\ ((\forall y)\ (P(y)\ \Rightarrow\ P(f(x,y)))\ \wedge\ \neg\ (\forall y)\ (Q(x,y)\ \Rightarrow\ P(y))))$ to clause form

1. Eliminate <=>

   Nothing to do here.

2. Eliminate =>

   $(\forall x)\ (\neg P(x) \vee ((\forall y)\ (\neg P(y) \vee P(f(x,y))) \wedge\ \neg(\forall y)\ (\neg Q(x,y) \vee\ P(y))))$

3. Reduce scope of negation

   $(\forall x)\ (\neg P(x) \vee ((\forall y)\ (P(y) \vee P(f(x,y))) \wedge (\exists\ y)\ (Q(x,y) \wedge\ \neg P(y))))$

4. Standardize variables

   $(\forall x)\ (\neg P(x) \vee ((\forall y)\ (\neg P(y) \vee\ P(f(x,y))) \wedge (\exists z)\ (Q(x,z) \wedge\ \neg\ P(z))))$

5. Eliminate existential quantification

   $\forall x)\ (\neg P(x) \vee ((\forall y)\ (\neg\ P(y) \vee P(f(x,y))) \wedge (Q(x,g(x)) \wedge\ \neg\ P(g(x)))))$

6. Drop universal quantification symbols

   $(\neg P(x) \vee ((\neg P(y) \vee P(f(x,y))) \vee (Q(x,g(x)) \wedge\ \neg P(g(x)))))$

7. Convert to conjunction of disjunctions

$(\neg P(x) \vee \neg P(y) \vee P(f(x,y))) \vee (\sim P(x) \vee Q(x,g(x))) \wedge (\sim P(x) \vee \neg P(g(x)))$

8. Create separate clauses

- $\neg P(x) \vee \neg P(y) \vee P(f(x,y))$
- $\neg P(x) \vee Q(x,g(x))$
- $\neg P(x) \vee \neg P(g(x))$

9. Standardize variables

- $\neg P(x) \vee \neg P(y) \vee P(f(x,y))$
- $\neg P(z) \vee Q(z,g(z))$
- $\neg P(w) \vee \neg P(g(w))$

Consider the following sentences:

- Shree is a mega star.
- Mega stars are rich.
- Rich people have fast cars.
- Fast cars consume a lot of petrol.

and try to draw the conclusion: Shree's *car consumes a lot of petrol*. So we can translate *Shree* is a mega star mega_star(Shree)

Mega stars are rich *into:*

$$\forall\, x : \text{mega\_star}(x) \Rightarrow \text{rich}(x)$$

Rich people have fast cars, *the third axiom is more difficult:*

- Is *cars* a relation and therefore *car(x,y)* says that case y is x's car. **OR**
- Is *cars* a function? So we may have *car_of(x)*.

Assume *cars* is a relation then axiom 3 may be written:

$$\forall\, x,y : \text{car}(x,y) \Rightarrow \text{rich}(y) \Rightarrow \text{fast}(x)$$

The fourth axiom is a general statement about *fast cars*. Let *consume(x)* mean that car x consumes a lot of petrol. Then we may write:

$$\forall\, x : [\text{fast}(x) \wedge \exists\, y : \text{car}(x,y) \Rightarrow \text{consume}(x)$$