3. Knowledge Representation & Reasoning

Artificial Intelligence and Neural Network (AINN)

Part II

Dr. Udaya Raj Dhungana
Assist. Professor
Pokhara University, Nepal
Guest Faculty
Hochschule Darmstadt University of Applied Sciences, Germany
E-mail: udaya@pu.edu.np and udayas.epost@gmail.com

Overview

– Predicate logic
– Inference in predicate logic
– Knowledge Representation Using Rules
– Semantic Nets and Frames

Predicate Logic

- Drawbacks of propositional logic
  1. The assertion "$x > 1$", *where* ***x is a variable, is not a proposition***
     - because it is neither true nor false unless value of ***x is defined***.
  2. ***Consider***

     "All men are mortal.

     John is a man.

     Then John is mortal"

  - Fails to capture relationship between John and man or John and mortal.
  - In addition, we do not get any information about the objects involved.
    - For example, if asked a question : "who is a man?" we cannot get answer.

Predicate Logic

- Drawbacks of propositional logic

3. Consider

   "Not all integers are even."

   "Some integers are not even"

   - The Propositional logic treats statements independently.
   - There is no mechanism in propositional logic to find out whether these two statements are equivalent.

Predicate Logic

- The drawbacks of propositional logic are solved in predicate logic

- *Predicate logic*
  - *is powerful enough for expression and reasoning.*
  - is built upon the ideas of *propositional logic*.

Predicate Logic

- first-order logic (like natural language) assumes the world contains
  - Objects:
    - are terms
    - Terms are names of objects
    - E.g. people, houses, car, John …
  - Properties
    - Unary predicates on terms
    - Predicate represents a property of or relations between terms that can be true or false
    - Eg. Hight, red, area …
  - Relations:
    - N-ary predicates on terms
    - A *relation* takes terms as arguments, and results in a *sentence*, denoting a *claim*
    - *fatherOf("Barack Obama", "Sasha")*
    - *brother of*, *bigger than*, *part of* …
  - Functions:
    - Mapping from terms to other terms
    - N-ary function maps a tuple of n terms to another
    - A *function* takes terms as arguments, and results in another *term*, denoting an *object*
    - *getFather("Sasha") --> "Barack Obama"*
    - *plus (1, 2)* …

Predicate Logic

- Syntax:
  - Constants John, 2, CAR,...
  - Variables x, y, a, b,...
  - Predicates Brother(), IsBlue(),...
  - Functions Sqrt, sum,...
  - Connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$
  - Equality =
  - Quantifiers $\forall$ (universal quantifier) and $\exists$ (existential quantifier)

Predicate Logic

- Syntax:
  - Atomic sentence
    - *predicate* ($term_1$,...,$term_n$)
    - $term_1 = term_2$
  - Terms
    - *function* ($term_1$,...,$term_n$)
    - *constant* or
    - *variable*
  - Complex sentences are made from atomic sentences using connectives
    - $\neg S$
    - $S_1 \wedge S_2$
    - $S_1 \rightarrow S_2$

Predicate Logic

- Syntax
  - Universal quantifier (∀)
    - ∀x: means “for all” x
    - represent phrase “ for all”.
    - It says that something is true for all possible values of a variable.
    - Example

      “ John loves everyone”

      ∀x: loves(John , x)

Predicate Logic

- Syntax
  - Existential quantifier(∃)
    - Used to represent the fact " there exists some"
    - Example:

      1. John loves someone
      $\exists\, y: loves(John\ ,\ y)$

      2. some people like reading and hence they gain knowledge.
      $\exists\, x: \{\ [person(x) \wedge like\ (x\ ,\ reading)] \rightarrow gain(x,\ knowledge)\ \}$

Predicate Logic

- Syntax
  - Nesting of quantifiers
    - E.g. “ everybody loves somebody ”

$$\forall x : \exists y : \text{loves}(x, y)$$

Predicate Logic

- **Syntax**

  - Properties of quantifiers

    - $\forall x\ \forall y$ is the same as $\forall y\ \forall x$
    - $\exists x\ \exists y$ is the same as $\exists y\ \exists x$
    - $\exists x\ \forall y$ is not the same as $\forall y\ \exists x$

    Examples:

    - $\exists x\ \forall y$ Loves(x,y)
      - "There is a person who loves everyone in the world"

    - $\forall y\ \exists x$ Loves(x,y)
      - "Everyone in the world is loved by at least one person"

Predicate Logic

- **Syntax**
  - Properties of quantifiers
    - Quantifier duality: each can be expressed using the other

$\forall x\ Likes(x, IceCream) = \neg\exists x\ \neg Likes(x, IceCream)$

$\exists x\ Likes(x, Broccoli) = \neg\forall x\ \neg Likes(x, Broccoli)$

<u>In general</u>

$\forall x\ \neg P = \neg\exists x\ P$

$\neg\forall x\ P = \exists x\ \neg P$

$\forall x\ P = \neg\exists x\ \neg P$

$\exists x\ P = \neg\forall x\ \neg P$

$\forall x\ P(X) \wedge Q(X) = \forall x\ P(X) \wedge \forall x\ Q(X)$

$\exists x\ P(X) \wedge Q(X) = \exists x\ P(X) \wedge \exists x\ Q(X)$

Predicate Logic

- Representing Simple facts in predicate logic

1. Marcus was a man.

   man(Marcus)

2. Marcus was a Pompeian

   Pompeian(Marcus)

3. All Pompeian were Romans

   $\forall x : Pompeian(x) \rightarrow Roman\ (x)$

Predicate Logic

- Representing Simple facts in predicate logic

4. Caesar was a rular.

$$rular(Caesar)$$

5. All Romans were either loyal to Caesar or hated him.

$$\forall x : Roman(x) \rightarrow loyalto(x, Caesar) \vee hate(x, Caesar)$$

6. Everyone is loyal to someone

$$\forall x : \exists y : loyalto(x, y)$$

Predicate Logic

- Representing Simple facts in predicate logic

7. People only try to assassinate rulers they are not loyal to

$$\forall x : \forall y: \ person(x) \land ruler(y) \land try\text{-}assassinate(x, y) \rightarrow \neg loyalto(x, y)$$

8. Marcus tried to assassinate Caesar

$$try\text{-}assassinate(Marcus, Caesar)$$

9. All man are people

$$\forall x: man(x) \rightarrow person(x)$$

Predicate Logic

- Some more examples 1

1. all indoor games are easy.

   $\forall x : \text{indoor-game}(x) \rightarrow \text{easy}(x)$

2. god helps those who helps themselves

   $\forall x : \text{helps}(\text{god}, \text{helps}(x, x))$

3. Any person who is respected by **every person** is a king.

   $\exists x : \forall y: \text{person}(x) \wedge \text{person}(y) \wedge \text{respect}(y, x) \rightarrow \text{king}(x)$

Predicate Logic

- Practice 1

(represent these statement in predicate logic)

  - Every child loves Santa.
  - Everyone who loves Santa loves any reindeer.
  - Rudolph is a reindeer, and Rudolph has a red nose.
  - Anything which has a red nose is weird or is a clown.
  - No reindeer is a clown.
  - Scrooge does not love anything which is weird.
  - Scrooge is not a child.

Predicate Logic

- Practice 1 (Answer)
  - Every child loves Santa.
  - $\forall x\ (CHILD(x) \rightarrow LOVES(x,Santa))$

  - Everyone who loves Santa loves any reindeer.
  - $\forall x\ (LOVES(x,Santa) \rightarrow \forall y\ (REINDEER(y) \rightarrow LOVES(x,y)))$

  - Rudolph is a reindeer, and Rudolph has a red nose.
  - $REINDEER(Rudolph) \wedge REDNOSE(Rudolph)$

  - Anything which has a red nose is weird or is a clown.
  - $\forall x\ (REDNOSE(x) \rightarrow WEIRD(x) \vee CLOWN(x))$

  - No reindeer is a clown.
  - $\neg \exists x\ (REINDEER(x) \wedge CLOWN(x))$

  - Scrooge does not love anything which is weird.
  - $\forall x\ (WEIRD(x) \rightarrow \neg LOVES(Scrooge,x))$

  - Scrooge is not a child.
  - $\neg CHILD(Scrooge)$

Predicate Logic

- Practice 2

(represent these statement in predicate logic)

  - Anyone whom Mary loves is a football star.
  - Any student who does not pass does not play.
  - John is a student.
  - Any student who does not study does not pass.
  - Anyone who does not play is not a football star.
  - If John does not study, then Mary does not love John.

Predicate Logic

- Practice 2 (Answer)
  - Anyone whom Mary loves is a football star.
  - $\forall x\ (LOVES(Mary,x) \rightarrow STAR(x))$

  - Any student who does not pass does not play.
  - $\forall x\ (STUDENT(x) \wedge \neg PASS(x) \rightarrow \neg PLAY(x))$

  - John is a student.
  - $STUDENT(John)$

  - Any student who does not study does not pass.
  - $\forall x\ (STUDENT(x) \wedge \neg STUDY(x) \rightarrow \neg PASS(x))$

  - Anyone who does not play is not a football star.
  - $\forall x\ (\neg PLAY(x) \rightarrow \neg STAR(x))$

  - If John does not study, then Mary does not love John.
  - $\neg STUDY(John) \rightarrow \neg LOVES(Mary,John)$

Predicate Logic

- **Resolution**
  - Produces proof by refutation (Proof by contradiction)
  - To prove a statement,
    - resolution attempts to show the negation of the statement produces a contradiction with the known statements.
  - Requires sentences to be in Conjunctive Normal Form (CNF)

Predicate Logic

AND = Conjunctions
OR = Disjunction

- **Resolution**
  - Conjunctive Normal Form (CNF)
    - In CNF, statements are conjunctions (sequence of ANDs) of clauses with clauses of disjunctions (sequence of OR).
    - In other words, **a statement is a series of ORs connected by ANDs.**
    - Examples:

$$(P \vee Q) \wedge (\neg P \vee R)$$

Predicate Logic

- **Resolution**
  - Conversion to CNF Form:

1. Eliminate biconditionals (↔)and implications (→) using

   $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$

   $p \rightarrow q \equiv \neg p \vee q$

2. Move ¬ inwards:

   $\neg\forall x : p(x) \equiv \exists x : \neg p(x)$
   $\neg\exists x : p(x) \equiv \forall x : \neg p(x)$

   $\neg(\alpha \vee \beta) \equiv \neg\alpha \wedge \neg\beta$
   $\neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta$

   $\neg\neg\, \alpha \equiv \alpha$

Predicate Logic

- **Resolution**
  - Conversion to CNF Form:

3. Standardize variables so that each quantifier binds a unique variable. For instance

   $\forall x : P(x) \vee \forall x : Q(x)$ could be converted into

   $\forall x : P(x) \vee \forall y : Q(y)$

4. Move all quantifiers to the left without changing their relative order.

   $\forall x : P(x) \vee \forall y : Q(y) \equiv$

   $\forall x : \forall y : (P(x) \vee Q(y))$

Predicate Logic

- **Resolution**
  - Conversion to CNF Form:

5. Skolemize: each existential variable is replaced by a Skolem constant or Skolem function of the enclosing universally quantified variables.

   1. $\exists x\ Rich(x) \equiv Rich(G1)$ where G1 is a new Skolem constant.

   2. $\forall x\ [\exists y\ animal\ (y) \wedge \neg\ loves(x, y)\ ] \vee [\exists z\ loves(z, x)] \equiv$
      $\forall x\ [animal\ (F(x)) \wedge \neg\ loves(x, F(x))\ ] \vee [loves(G(x), x)]$
      where F and G are Skolem functions.

6. Drop universal quantifiers
   $[animal\ (F(x)) \wedge \neg\ loves(x, F(x))\ ] \vee [loves(G(x), x)]$

Predicate Logic

- **Resolution**
  - Conversion to CNF Form:

7. Distribute $\wedge$ over $\vee$

$$[\text{animal}(F(x)) \wedge \neg \text{loves}(x, F(x))] \vee [\text{loves}(G(x), x)] \equiv$$
$$[\text{animal}(F(x)) \vee \text{loves}(G(x), x)] \wedge [\neg \text{loves}(x, F(x)) \vee \text{loves}(G(x), x)]$$

- This sentence is now in CNF
- Consists of two clauses
  - $[\text{animal}(F(x)) \vee \text{loves}(G(x), x)]$
  - $[\neg \text{loves}(x, F(x)) \vee \text{loves}(G(x), x)]$
- Is quite unreadable (human seldom need look at CNF sentences)

Predicate Logic

- **Resolution**
  - Example: Conversion to CNF

"Everyone who loves all animal is loved by someone"

In predicate logic:

$\forall x\ [\forall y\ animal\ (y) \rightarrow loves(x, y)\ ] \rightarrow [\ \exists y\ loves(y, x)]$

1. Eliminate implications

   $\forall x\ [\forall y\ \neg animal\ (y) \vee loves(x, y)\ ] \rightarrow [\ \exists y\ loves(y, x)] \equiv$

   $\forall x\ [\neg\ [\forall y\ \neg animal\ (y) \vee loves(x, y)\ ]] \vee [\ \exists y\ loves(y, x)]$

2. Move $\neg$ inwards

   $\forall x\ [\neg \forall y\ \neg\ (\neg\ animal\ (y) \vee loves(x, y)\ )] \vee [\ \exists y\ loves(y, x)] \equiv$

   $\forall x\ [\exists y\ animal\ (y) \wedge \neg loves(x, y)\ ] \vee [\ \exists y\ loves(y, x)]$

3. Standardize variables

   $\forall x\ [\exists y\ animal\ (y) \wedge \neg loves(x, y)\ ] \vee [\ \exists z\ loves(z, x)]$

Predicate Logic

4. Skolemize (process of removing existential quantifiers)

$\forall x$ [animal (F(x)) $\wedge$ $\neg$loves(x, F(x))] $\vee$ loves(G(x), x)
Where F and G are skolem functions.

5. Drop universal quantifiers

[animal (F(x)) $\wedge$ $\neg$loves(x, F(x))] $\vee$ loves(G(x), x)

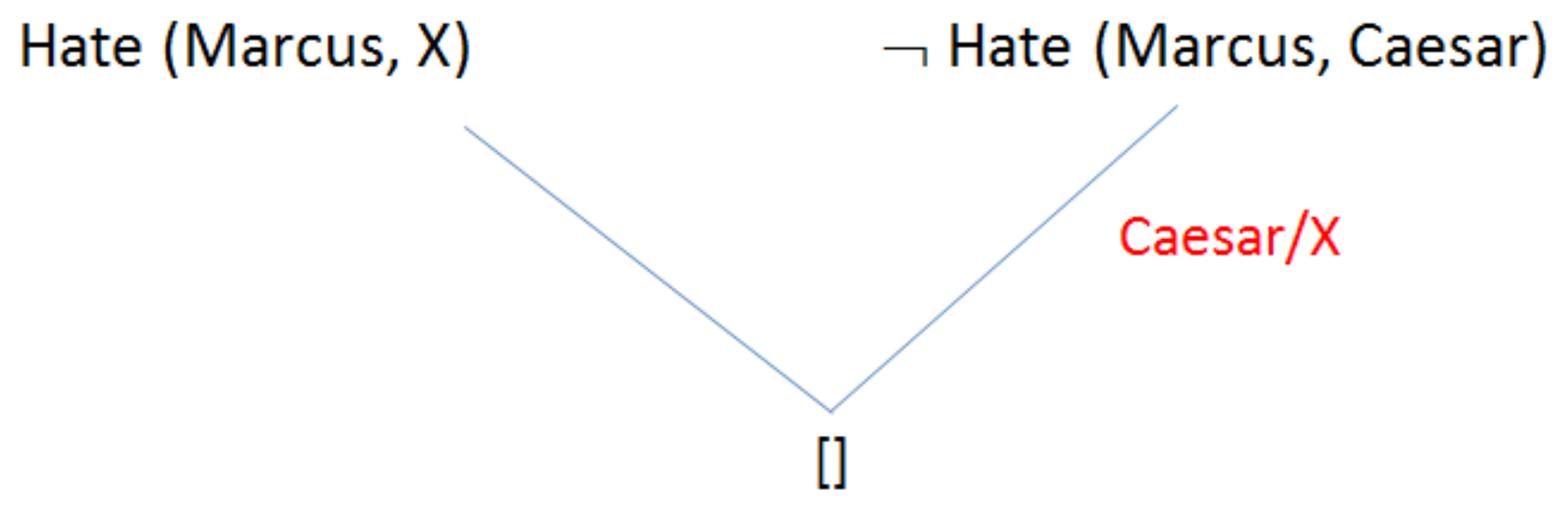
3. Distribute $\wedge$ over $\vee$

[animal (F(x)) $\vee$ loves(G(x), x)] $\wedge$ [$\neg$ loves(x, F(x)) $\vee$ loves(G(x), x)]

– This is in CNF and has two clauses
  [animal (F(x)) $\vee$ loves(G(x), x)]
  [$\neg$ loves(x, F(x)) $\vee$ loves(G(x), x)]

Predicate Logic

- **Unification**
  - It's a matching procedure that compares two literals and discovers whether there exists a set of substitutions that can make them identical.



  - Similarly,
    - Hate(X,Y) and Hate(john, Z) could be unified as:
      - John/X and y/z

Predicate Logic

- **Resolution**
  - Is rule of inference
  - Pre-processing steps:
    1. Convert the given English sentence into predicate sentence.
    2. Convert all of these sentences into CNF.

Predicate Logic

- **Resolution**
  - Assume that a set of given statements F and a statement to be proved P:

  - ALGORITHM: RESOLUTION IN PREDICATE LOGIC
    1. Convert all the statements of F to clause form
    2. Negate P and convert the result to clause form. Add it to the set of clauses obtained in step 1.
    3. Repeat until either a contradiction is found or no progress can be made:
       a. Select two clauses. Call these the parent clauses.
       b. Resolve them together. The resulting clause is called the resolvent.
       c. If the resolvent is the empty clause, then a contradiction has been found. (It means the assumption is wrong and the original clause is true.)
          If it is not then add it to the set of clauses available to the procedure.

Predicate Logic

- **Resolution**
  - Example: Let facts about Marcus:

1. Marcus was a man.
2. Marcus was a Pompeian.
3. All Pompeians were Romans.
4. Caesar was a ruler.
5. All Romans were either loyal to Caesar or hated him.
6. Every one is loyal to someone.
7. People only try to assassinate rulers they are not loyal to.
8. Marcus tried to assassinate Caesar.
9. All men are people

Predicate Logic

- **Resolution**
  - Example: given facts in predicate logic:

1. Man(Marcus)
2. Pompeian(Marcus)
3. $\forall x$: Pompeian(x) → Roman (x)
4. rular(Caesar)
5. $\forall x$: Roman(x) → loyalto(x, Caesar) ∨ hate(x, Caesar)
6. $\forall x$: $\exists y$: loyalto(x, y)
7. $\forall x$: $\forall y$: person(x) ∧ ruler(y) ∧ try-assassinate(x, y) → ¬loyalto(x, y)
8. try-assassinate(Marcus, Caesar)
9. $\forall x$: man(x) →person(x)

Predicate Logic

- **Resolution**
  - Example: given facts in predicate logic:

1. Man(Marcus)
2. Pompeian(Marcus)
3. $\forall x : Pompeian(x) \rightarrow Roman(x)$
4. rular(Caesar)
5. $\forall x : Roman(x) \rightarrow loyalto(x, Caesar) \vee hate(x, Caesar)$
6. $\forall x : \exists y : loyalto(x, y)$
7. $\forall x : \forall y: person(x) \wedge ruler(y) \wedge try\text{-}assassinate(x, y) \rightarrow \neg loyalto(x, y)$
8. try-assassinate(Marcus, Caesar)
9. $\forall x : man(x) \rightarrow person(x)$

Predicate Logic

- **Resolution**
  - Example: given facts in CNF:

1. Man(Marcus)
2. Pompeian(Marcus)
3. $\forall x$: Pompeian(x) $\rightarrow$ Roman (x) $\equiv$

   $\forall x$: $\neg$ Pompeian(x) $\vee$ Roman (x) $\equiv$

   $\neg$ Pompeian(x1) $\vee$ Roman (x1)
4. rular(Caesar)
5. $\forall x$: Roman(x) $\rightarrow$ loyalto(x, Caesar) $\vee$ hate(x, Caesar) $\equiv$

   $\forall x$: $\neg$ Roman(x) $\vee$ [loyalto(x, Caesar) $\vee$ hate(x, Caesar)] $\equiv$

   $\neg$ Roman(x2) $\vee$ loyalto(x2, Caesar) $\vee$ hate(x2, Caesar)

Predicate Logic

- **Resolution**
  - Example: given facts in CNF:

6. $\forall x : \exists y : loyalto(x, y) \equiv$

$\forall x : loyalto(x, f(x)) \equiv$

$\forall x3 : loyalto(x3, f(x3)) \equiv$

$loyalto(x3, f(x3))$

7. $\forall x : \forall y: \; person(x) \wedge ruler(y) \wedge try\text{-}assassinate(x, y) \rightarrow \neg loyalto(x, y) \equiv$

$\forall x : \forall y: \neg[person(x) \wedge ruler(y) \wedge try\text{-}assassinate(x, y)] \vee \neg loyalto(x, y) \equiv$

$\forall x4 : \forall y1: \neg person(x4) \vee \neg ruler(y1) \vee \neg try\text{-}assassinate(x4, y1) \vee \neg loyalto(x4, y1) \equiv$

$\neg person(x4) \vee \neg ruler(y1) \vee \neg try\text{-}assassinate(x4, y1) \vee \neg loyalto(x4, y1)$

Predicate Logic

- **Resolution**
  - Example: given facts in CNF:

8. try-assassinate(Marcus, Caesar)

9. $\forall x : man(x) \rightarrow person(x) \equiv$

$\forall x : \neg\ man(x) \vee person(x) \equiv$

$\neg\ man(x5) \vee person(x5)$

Predicate Logic

- **Resolution**
  - Example: given facts in CNF:

1. Man(Marcus)
2. Pompeian(Marcus)
3. ¬ Pompeian(x1) ∨ Roman (x1)
4. rular(Caesar)
5. ¬ Roman(x2) ∨ loyalto(x2, Caesar) ∨ hate(x2, Caesar)
6. loyalto(x3, f(x3))
7. ¬person(x4) ∨ ¬ruler(y1) ∨ ¬try-assassinate(x4, y1)∨ ¬loyalto(x4, y1)
8. try-assassinate(Marcus, Caesar)
9. ¬ man(x5) ∨ person(x5)

Predicate Logic

- **Resolution**
  - Prove that Marcus hate Caesar. That is

    hate(Marcus, Caesar)

  Negate: ¬ hate(Marcus, Caesar)

  which is already in CNF

Predicate Logic

- **Resolution**

¬ hate(Marcus, Caesar)

Predicate Logic

¬ Roman(x2) ∨ loyalto(x2, Caesar) ∨ hate(x2, Caesar)

¬ hate(Marcus, Caesar)

Predicate Logic

- **Resolution**

¬ Roman(x2) ∨ loyalto(x2, Caesar) ∨ hate(x2, Caesar)

¬ hate(Marcus, Caesar)

Marcus/X2

Predicate Logic

- **Resolution**

¬ Roman(x2) ∨ loyalto(x2, Caesar) ∨ hate(x2, Caesar)

¬ hate(Marcus, Caesar)

Marcus/X2

¬ Roman(Marcus) ∨ loyalto(Marcus, Caesar)

Predicate Logic

- **Resolution**

¬ Roman(x2) ∨ loyalto(x2, Caesar) ∨ hate(x2, Caesar)

¬ hate(Marcus, Caesar)

Marcus/X2

¬ Roman(Marcus) ∨ loyalto(Marcus, Caesar)

¬ Pompeian(x1) ∨ Roman (x1)

Marcus/x1

Predicate Logic

- **Resolution**

¬ Roman(x2) ∨ loyalto(x2, Caesar) ∨ hate(x2, Caesar)

¬ hate(Marcus, Caesar)

Marcus/X2

¬ Roman(Marcus) ∨ loyalto(Marcus, Caesar)

¬ Pompeian(x1) ∨ Roman (x1)

Marcus/x1

loyalto(Marcus, Caesar) ∨ ¬ Pompeian(Marcus)

Predicate Logic

- **Resolution**

¬ Roman(x2) ∨ loyalto(x2, Caesar) ∨ hate(x2, Caesar)

¬ hate(Marcus, Caesar)

Marcus/X2

¬ Roman(Marcus) ∨ loyalto(Marcus, Caesar)

¬ Pompeian(x1) ∨ Roman (x1)

Marcus/x1

Pompeian(Marcus)

loyalto(Marcus, Caesar) ∨ ¬ Pompeian(Marcus)

loyalto(Marcus, Caesar)

Predicate Logic

- **Resolution**

¬person(x4) ∨ ¬ruler(y1) ∨
¬try-assassinate(x4, y1)∨ ¬loyalto(x4, y1)

loyalto(Marcus, Caesar)

Marcus/x4, Caesar/y1

rular(Caesar)

¬person(Marcus) ∨ ¬ruler(Caesar)
∨
¬try-assassinate(Marcus, Caesar)

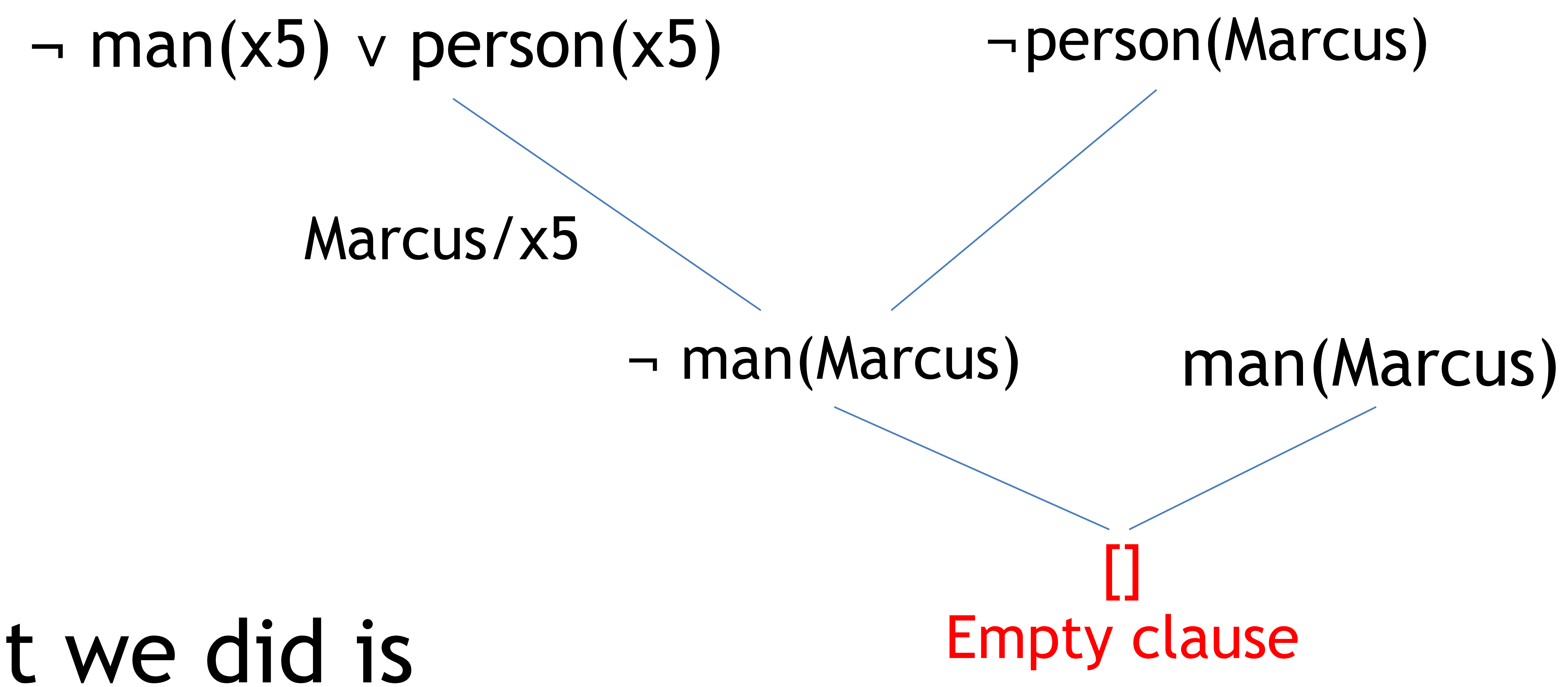
¬person(Marcus) ∨
¬try-assassinate(Marcus, Caesar)

try-assassinate(Marcus, Caesar)

¬person(Marcus)

Predicate Logic

- **Resolution**



- What we did is
  - Backward reason
  - Got contradiction, ie ¬ hate(Marcus, Caesar) was found wrong
    - **Therefore, “Marcus hates Caesar” is true.**

Predicate Logic

- **Resolution**
  - Question Answering
  - Resolution can also be used to answer the questions
  - Example: Assume the following facts:
    - John only likes easy courses.
    - All science courses are hard.
    - All the courses in Arts are easy.
    - A101 is an Art course.
    - S201 is a science course.
  - Now, using resolution answer the following:
    - Which course would John like?

Predicate Logic

- **Resolution**
  - Given facts in predicate logic
    - John only likes easy courses.
      $\forall x$: easyCourse(x) → likes(John, x)
    - All science courses are hard.
      $\forall x$: scienceCourse(x) → hardCourse(x)
    - All the courses in Arts are easy.
      $\forall x$: artCourse(x) → easyCourse(x)
    - A101 is an Art course.
      artCourse(A101)
    - A201 is a science course.
      scienceCourse(S201)

Predicate Logic

- **Resolution**
  - Given facts in CNF
    - $\neg$ easyCourse(x1) $\vee$ likes(John, x1)
    - $\neg$ scienceCourse(x2) $\vee$ hardCourse(x2)
    - $\neg$ artCourse(x3) $\vee$ easyCourse(x3)
    - artCourse(A101)
    - scienceCourse(S201)

Predicate Logic

- **Resolution**
  - Question clause is

    Likes(John, x)

  - For this, first prove that John like some course first.

  That is,

    likes(John, x)

Predicate Logic

- **Resolution**

¬ easyCourse(x1) v likes(John, x1)

¬ likes(John, x)

X1/x

¬ artCourse(x3) v easyCourse(x3)

¬ easyCourse(x1)

x3/x1

artCourse(A101)

¬ artCourse(x3)

A101/x3

[]

It means **John likes some course x.**
Which course would John like?

Predicate Logic

- **Resolution**

¬ easyCourse(x1) ∨ likes(John, x1)

¬ likes(John, x) ∨
likes(John, x)

X1/x

¬ artCourse(x3) ∨ easyCourse(x3)

likes(John, x1) ∨ ¬ easyCourse(x1)

x3/x1

artCourse(A101)

likes(John, x3) ∨ ¬ artCourse(x3)

A101/x3

likes(John, A101)

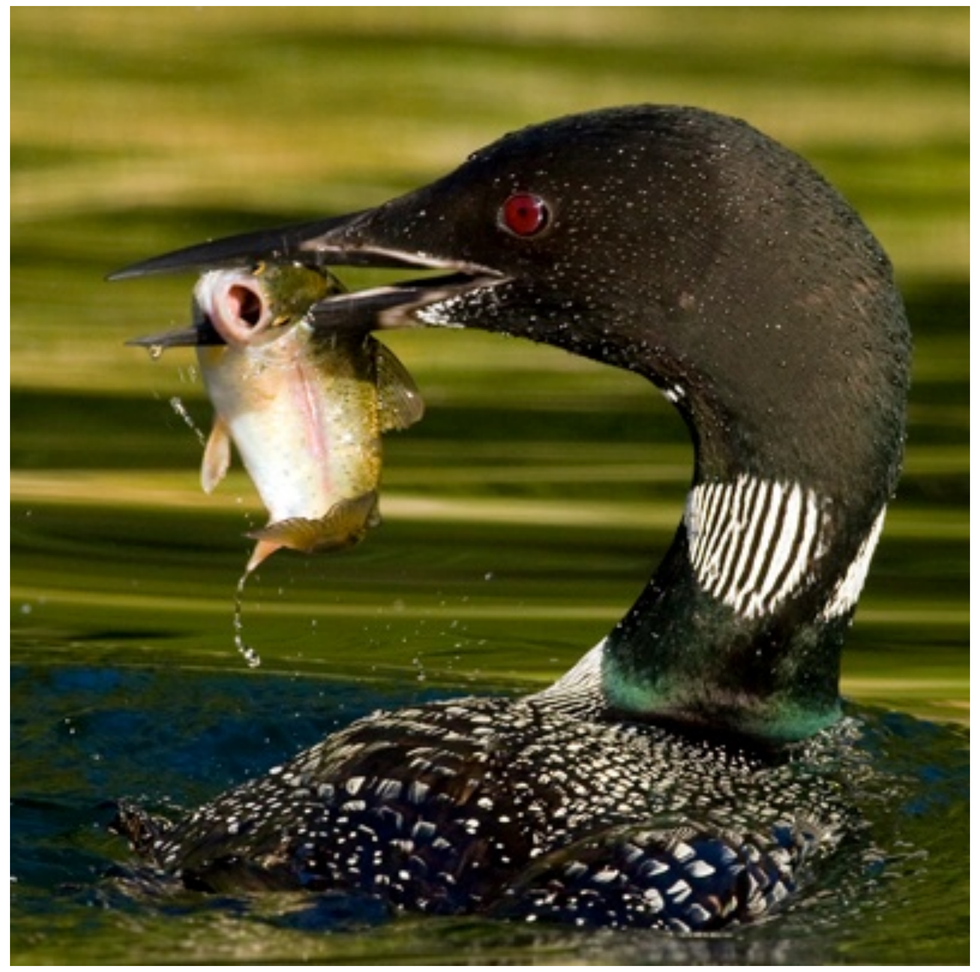
Therefore, John likes Art course A101

Predicate Logic

- **Resolution**
  - Consider the following axioms.
    - Every bird sleeps in some tree.
    - Every loon is a bird, and every loon is aquatic.
    - Every tree in which any aquatic bird sleeps is beside some lake.
    - Anything that sleeps in anything that is beside any lake eats fish.

  Using resolution prove that “Every loon eats fish”.

  Solution: http://www.sc.ehu.es/jiwlucap/Tema3-RA-2013-2014.pdf



Predicate Logic

- **Resolution**
  - Consider the following axioms.
    - John likes all kind of food.
    - Apples and chicken are food
    - Anything anyone eats and is not killed is food
    - Mary eats peanuts and is still alive
    - Bob eats everything that Mary eats
    - Everyone who is alive is not killed.
    - Everyone who is not killed is alive.

Prove by resolution that John likes peanuts using resolution.

Predicate Logic

- **Resolution**
  - facts in predicate logic.

    $\forall x$: food(x) → likes (John, x)

    food (Apple) ^ food (chicken)

    $\forall a$: $\forall b$: eats (a, b) ^ ~killed (a) → food (b)

    eats (Mary, Peanuts) ^ alive (Mary)

    $\forall c$: eats (Mary, c) → eats (Bob, c)

    $\forall d$: alive(d) → ~killed (d)

    $\forall e$: ~killed(e) → alive(e)

    Conclusion: likes (John, Peanuts)

Predicate Logic

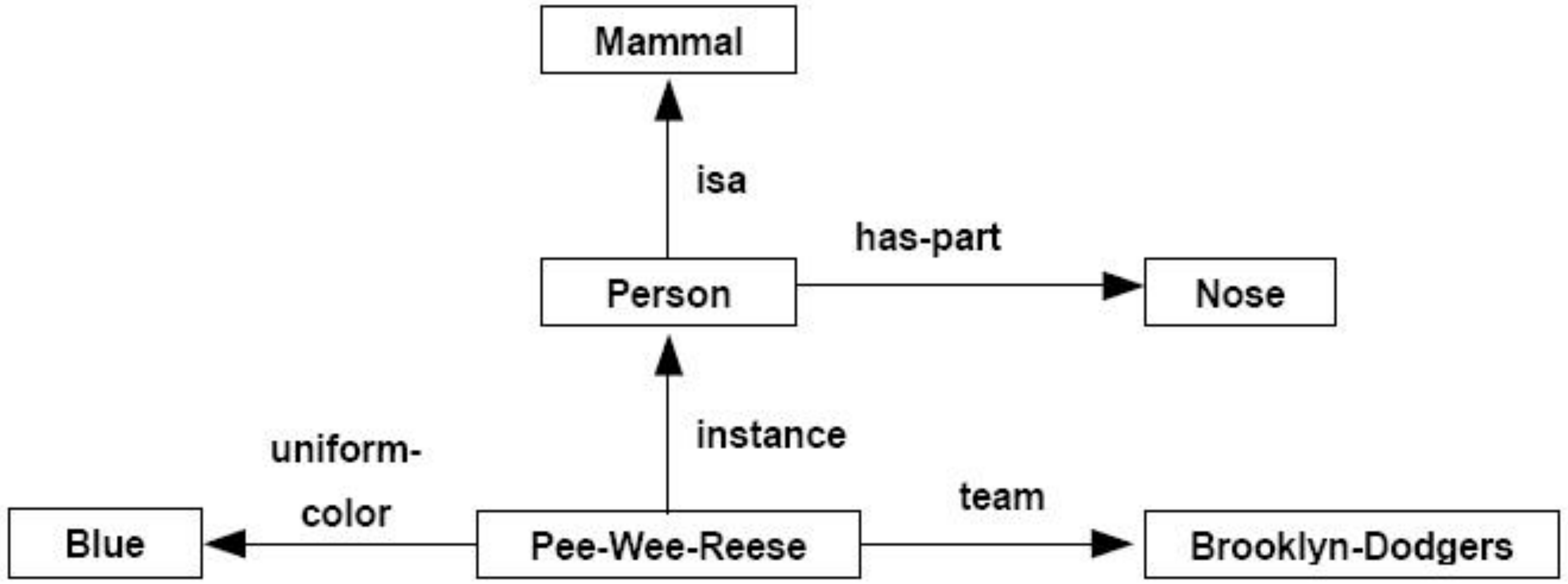
- **Resolution**
  - Facts in CNF.
    1. ~food(x) v likes(John, x)
    2. Food (apple)
    3. Food (chicken)
    4. ~ eats (a, b) v killed (a) v food (b)
    5. Eats (Mary, Peanuts)
    6. Alive(Mary)
    7. ~eats (Mary, c) V eats (Bob, c)
    8. ~alive (d) v ~ killed (d)
    9. Killed (e) v alive (e)
    - To prove: likes (John, Peanuts)

Semantic Nets

- is simple KR scheme
- uses a graph of labeled nodes and labeled directed arcs to encode knowledge
  - Nodes
    - objects, concepts, events
    - represents the information
  - Arcs – relationships between nodes
    - particularly *isa* arcs -
      - allow inheritance of properties.

Semantic Nets

- Example: ***semantic net***



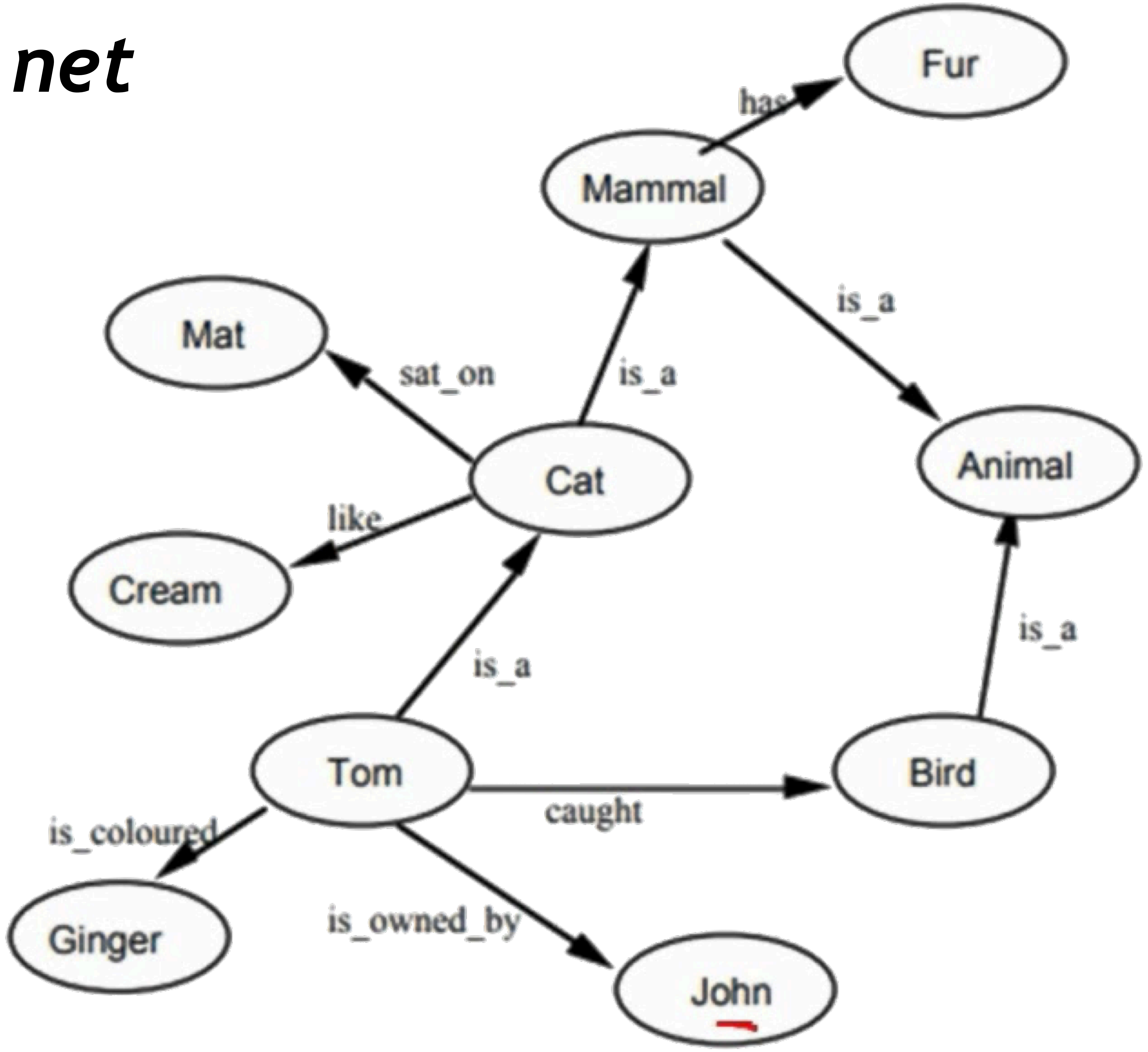
In logic representation

isa(Person, Mammal)
has-part(Person, Nose)
instance(Pee-Wee-Reese, Person)
team(Pee-Wee-Reese, Brooklyn-Dodgers)
uniform-color(Pee-Wee-Reese, Blue)

Semantic Nets

- Example: ***semantic net***

Tom is a cat.
Tom caught a bird.
Tom is owned by John.
Tom is ginger in colour.
Cats like cream.
The cat sat on the mat.
A cat is a mammal.
A bird is an animal.
All mammals are animals.
Mammals have fur.



Semantic Nets

- Benefits of ***semantic net***
  - Easy to visualize
  - Formal definitions of semantic networks have been developed.
  - Related knowledge is easily clustered.
  - Efficient in space requirements
    - Objects represented only once
    - Relationships handled by pointers

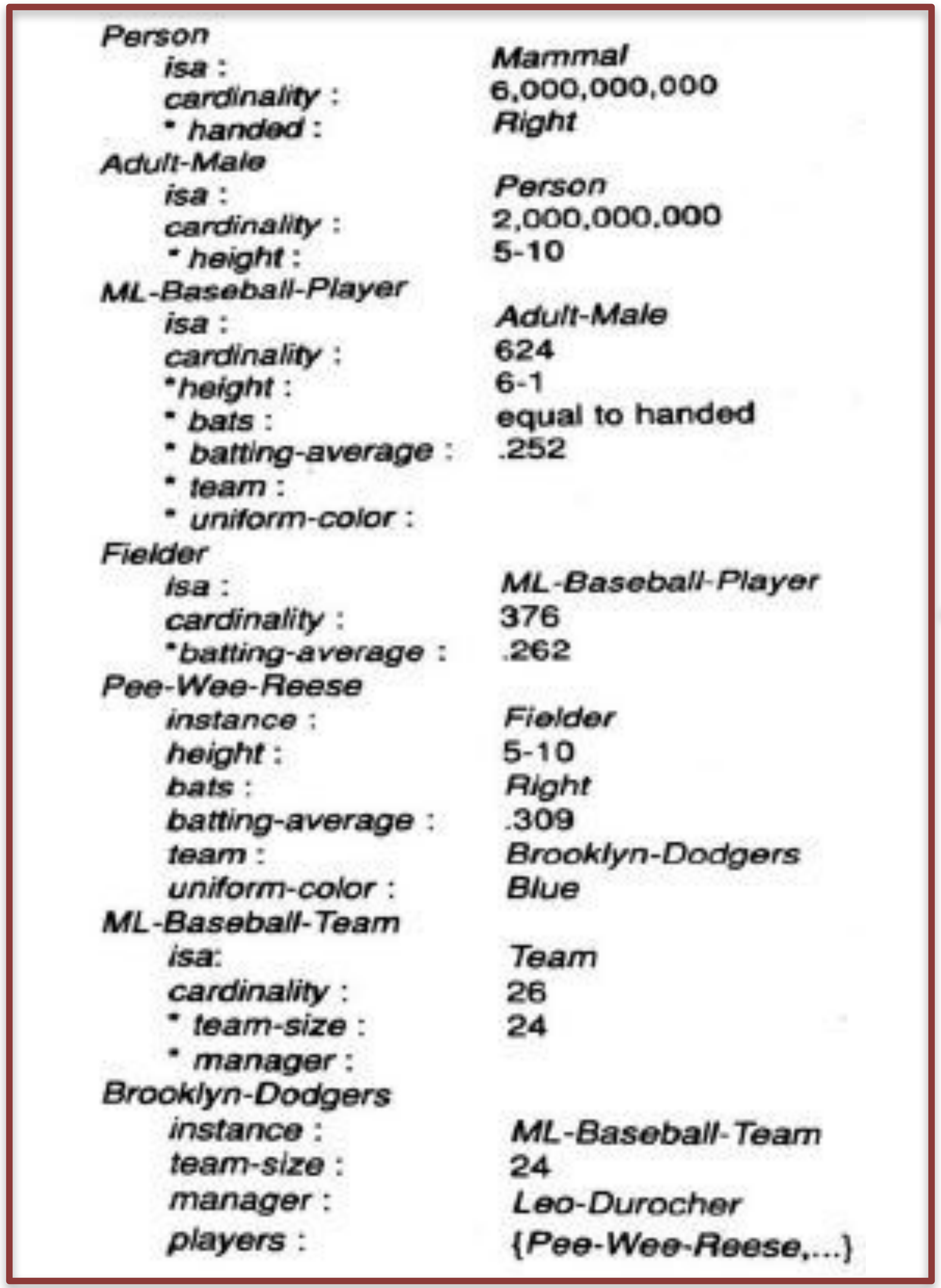
Frames

- A frame is a collection of attributes (usually called slots) and associated values(called filler) that describe some entity in the World.
- Three components of a frame
  - frame name
  - attributes (slots)
  - values (fillers: list of values, range, string, etc.)
- Example: Book

Book	
Title:	Artificial Intelligence, a modern approach
Author:	***Russell & Norvig***
Year:	2014

Frames

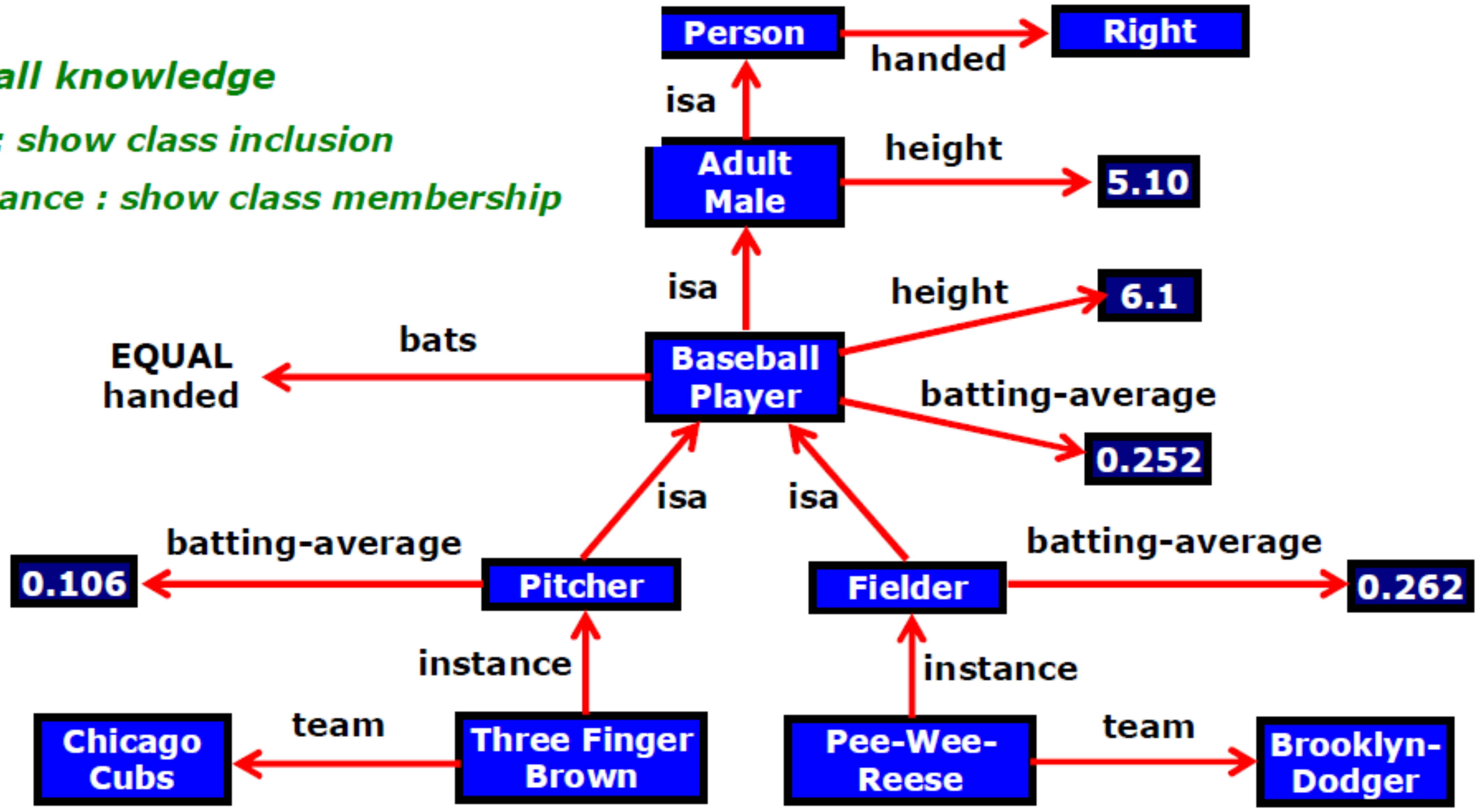
Frames



Person	
isa :	*Mammal*
cardinality :	6,000,000,000
* *handed :*	*Right*
Adult-Male	
isa :	*Person*
cardinality :	2,000,000,000
* *height :*	5-10
ML-Baseball-Player	
isa :	*Adult-Male*
cardinality :	624
* *height :*	6-1
* *bats :*	equal to handed
* *batting-average :*	.252
* *team :*	
* *uniform-color :*	
Fielder	
isa :	*ML-Baseball-Player*
cardinality :	376
* *batting-average :*	.262
Pee-Wee-Reese	
instance :	*Fielder*
height :	5-10
bats :	*Right*
batting-average :	.309
team :	*Brooklyn-Dodgers*
uniform-color :	*Blue*
ML-Baseball-Team	
isa:	*Team*
cardinality :	26
* *team-size :*	24
* *manager :*	
Brooklyn-Dodgers	
instance :	*ML-Baseball-Team*
team-size :	24
manager :	*Leo-Durocher*
players :	{*Pee-Wee-Reese*,...}

Baseball knowledge

- *isa : show class inclusion*
- *instance : show class membership*



Inheritable knowledge

Frames

- Benefits of frame
  - Makes programming easier by grouping related knowledge
  - Easily understood by non-developers
  - Expressive power
  - Easy to set up slots for new properties and relations
  - Easy to include default information and detect missing values

Questions

1. What are the drawbacks of propositional logic?
2. How resolution produce a proof for a statement? Convert the statement $(p \rightarrow q) \leftrightarrow (p \rightarrow r)$ to CNF?
3. Consider the following axioms.
   A. Every bird sleeps in some tree.
   B. Every loon is a bird, and every loon is aquatic.
   C. Every tree in which any aquatic bird sleeps is beside some lake.
   D. Anything that sleeps in anything that is beside any lake eats fish.

   Using resolution, prove that "Every loon eats fish".
4. What are semantic nets and frames? Give examples.

THANK YOU

End of Chapter