COSC1125/1127 Artificial Intelligence

Week 5: Knowledge Representation II (Predicate Calculus)

[RN2] Sec 7.1-7.6 Chapters 8-9
[RN3] Sec 7.1-7.6 Chapters 8-9

Predicate Calculus

Predicate calculus (like natural language) assumes the world contains:

- **Objects**

  e.g. people, houses, numbers, theories, colors, baseball games, wars, centuries..

- **Relations**
  - **unary relations** (or **properties**):

    e.g. red, round, prime, bogus, multistoried…
  - **n-ary relations**:

    e.g. brother of, bigger than, inside, has color, part of, occurred after, owns, comes between …

- **Functions**

  e.g. father of, best friend, third inning of, one more than, beginning of…

Basic Elements of PC

Basic elements are:

- **Constants**

  represent individual objects, such as *People*, *House*, *Wumpus* …

- **Variables**

  represent classes or sets of objects, such as *a, b, x, y* …

- **Connectives**

  $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$

- **Predicates**

  represent relations (include properties) of objects e.g. *Red, HasColor, Inside…*

- **Functions**

  FatherOf, MotherOf …

- **Quantifiers**

  $\forall \; \exists$

- **Equality** $=$

Predicates

A predicate is either **true** or **false.**

The semantics of a predicate depends on an interpretation (similar to propositional logic). For example:

Smelly(Wumpus)	The wumpus is smelly
Dead(Elvis)	Elvis is dead
Likes(John, Mary)	John likes Mary
Income(Inadequate)	Income is inadequate
Plays(John, football, Tues)	John plays football on Tuesday
Likes(x, Mary)	Somebody likes Mary
Breeze(x, y)	There is a breeze in some square
Brother(Richard, John)	Richard is John's brother

Functions

A function returns a value **other than** true, false. The value is some object.

For example:

FatherOf(John) returns *Bill* if Bill is the father of John.

Plus(2,3) returns *5*

- The act of replacing a function with its value is called **evaluation**.

Question, what is returned by *FatherOf(FatherOf(John))* ?

- Function can be used in place of a variable or a constant, e.g.

Friends(FatherOf(John), FatherOf(Susie)) Father of John and father of Susie are friends.

Predicate Functions

Sentences in Predicate Calculus

There are two kinds of sentences, atomic sentences and complex sentences

– An **atomic sentence** is a predicate, e.g.

Brother(Richard, John)

Atomic sentences can have complex arguments, e.g.

Married(Father(Richard), Mother(John))

>(Length(LeftLegOf(Richard)), Length(LeftLegOf(John)))

– **Complex sentences** are made from atomic sentences using connectives, e.g.

Brother(Richard, John) $\Rightarrow$ *Sibling(John, Richard)*

$\neg$ *Brother(Father(Richard), John)*

Brother(Richard, John) $\wedge$ *Brother(John, Richard)*

Quantifiers

Predicate Calculus allows sets of objects. To express properties of entire set of objects, **quantifiers** can be used instead of enumerating the objects by name.

Two quantifiers in PC: universal quantifier $\forall$ and existential quantifier $\exists$

General form of **Universal Quantifier**

$\forall$ <variables> <sentence>

It means the sentence is true for **every** possible value of the variables, e.g

$\forall x\ P(x)$	– for all x, P is true.
$\forall x\ Red(x)$	– all objects are red.
$\forall x\ Likes(x, AI)$	– everyone likes AI.
$\forall x\ Study(x, AI) \Rightarrow Smart(x)$	– everyone studying AI is smart.

Quantifiers…

General form of **Existential Quantifier**

$\exists$ <variables> <sentence>

It means the sentence is true for **some** possible value of the variables, e.g

$\exists\, x\, P(x)$

means there exists an x such that P is true.

or P is true for one or more substitutions for x.

$\exists\, x\, Red(x)$ – at least one object is red.

$\exists\, x\, Likes(x, Apple)$ – someone likes apple.

$\exists\, x\, Study(x, Laws) \wedge Smart(x)$ – someone studying laws is smart.

Next, we show some common mistakes in using quantifiers.

Common Mistakes

Typically, $\Rightarrow$ is the main connective with $\forall$, while $\wedge$ is the main connective with $\exists$

Common mistakes:

using $\wedge$ as the main connective with $\forall$, using $\Rightarrow$ as the main connective with $\exists$

$\forall x\ Study(x, AI) \wedge Smart(x)$

- It does not mean "everyone studying AI is smart"
- It means "everyone studies AI and everyone is smart".

$\exists x\ Study(x, Laws) \Rightarrow Smart(x)$

- This sentence can be true if there is someone who does not study laws and not smart.
- Because *Study(x, Laws)* is false, *Smart(x)* is false, false $\Rightarrow$ false is true!

Multiple Quantifiers

Multiple quantifiers are used to express more complex sentences, such as

$\forall x \, \forall y \; Brother(x, y) \Rightarrow Sibling(x, y)$ means “All brothers are siblings”

Consecutive quantifiers of the same type can be written as one quantifier.

$\forall \boldsymbol{x, y} \; Brother(x, y) \Rightarrow Sibling(x, y)$ is same to $\forall x \, \forall y \; Brother(x, y) \Rightarrow Sibling(x, y)$

If there are mixtures, the order of quantifiers is very important.

$\forall x \, \exists y \; Loves(x, y)$

– means “Everybody has someone he/she loves”

$\exists y \, \forall x \; Loves(x, y)$

– means “There is someone who is loved by everyone”

Parentheses can make it clearer. $\forall x \, (\exists y \; Loves(x, y))$ and $\exists y \, (\forall x \; Loves(x, y))$

Some Equivalences

1. $\forall x \ \neg P \ \equiv \ \neg\exists x \ P$

E.g. $\forall x \ \neg Likes(x, Parsnips)$ is same as $\neg\exists x \ Likes(x, Parsnips)$

(Everyone dislikes parsnips or no one likes parsnips)

2. $\forall x \ P \ \equiv \ \neg\exists x \ \neg P$

E.g. $\forall x \ Likes(x, IceCream)$ is same as $\neg\exists x \ \neg Likes(x, IceCream)$

(Everyone likes ice cream or no one dislikes ice cream)

3. $\neg\forall x \ P \ \equiv \ \exists x \ \neg P$

E.g. $\neg\forall x \ Likes(x, Curry)$ & $\exists x \ \neg Likes(x, Curry)$

(Not everyone likes curry or someone dislikes curry)

4. $\neg\forall x \neg P \ \equiv \ \exists x \ P$

E.g. $\neg\forall x \neg Likes(x, Garlic)$ & $\exists x \ Likes(x, Garlic)$

(Not everyone dislikes garlic or someone likes garlic)

Equality

In Predicate Calculus, the equality symbol = means the objects referred by both side are the same, e.g.

$$CapitalOf(Japan) = Tokyo$$

To express "Richard has only one child":

$$\forall x, y\ Child(x, Richard) \wedge Child(y, Richard) \wedge \boldsymbol{(x = y)}$$

Could it be expressed as the follow?

$$\forall x, y\ Child(x, Richard) \wedge Child(y, Richard)$$

To express "Richard has at least two children":

$$\exists x, y\ Child(x, Richard) \wedge Child(y, Richard) \wedge \boldsymbol{\neg(x = y)}$$

Could it be expressed as the follow?

$$\exists x, y\ Child(x, Richard) \wedge Child(y, Richard)$$

Knowledge Representation in PC

A **domain** in Knowledge Representation (KR) is some part of the world which we wish to express some knowledge.

A set of constants, predicates and functions needs to be determined to fit the domain. (*We assume they have been determined for the early and later examples of predicates/functions)*

For example the Wumpus domain

$$\forall s\ Breezy(s) \Rightarrow \exists r\ Adjacent(r,s) \wedge Pit(r)$$

Predicates *Breezy, Adjacent* etc. are determined for this domain.

The use of constants, predicates and functions must be consistent within a domain. E.g. don't use *Adjacent* sometime but *NextTo* or *Adjac* next time.

There is currently no general set of predicates and functions that can be used in any domain. Whether we can find such a set remains a major research topic.

Kinship Domain

Express some knowledge in the domain of family relationship, or kinship.

$\forall x\ Male(x) \Leftrightarrow \neg Female(x)$

Male and female are disjoint.

$\forall x, y\ Parent(x, y) \Leftrightarrow Child(y, x)$

Parent and child are inverse relations.

$\forall x, y\ Mother(x, y) \Leftrightarrow Female(x) \wedge Parent(x, y)$

One's mother is one's female parent.

$\forall x, y\ Husband(x, y) \Leftrightarrow Male(x) \wedge Spouse(x, y)$

One's husband is one's male spouse.

$\forall x, y\ GrandParent(x, y) \Leftrightarrow \exists z\ Parent(x, z) \wedge Parent(z, y)$

A grandparent is a parent of one's parent.

$\forall x, y\ Sibling(x, y) \Leftrightarrow \neg(x = y) \wedge \exists z\ Parent(z, x) \wedge Parent(z, y)$

A sibling is another child of one's parent.

$\forall x, y\ Sibling(x,y) \Leftrightarrow \neg(x=y) \wedge \exists m,f\ \neg(m=f) \wedge Parent(m,x) \wedge Parent(m,y) \wedge Parent(f,x) \wedge Parent(f,y)$

Definition of full sibling (same father and mother)!!

Kinship Domain

To the kinship knowledge base complete some facts are needed, such as

- *Parent(Mary, John)*
- *Parent(George, John)*
- *Spouse(Mary, George)*
- *Male(George)*
- *Female(Mary)*
- *Parent (John, Henry)*
- *……*

Then we would answer queries like

- *Who is the husband of Mary?*
- *Is George Henry's grandparent?*

Fun with PC Sentences

Predicate Calculus is quite expressive, and can be fun

There is a big dog next door

$\exists x\ (\ Big(x)\ \wedge\ Dog(x)\ \wedge\ NextDoor(x)\ )$

Jim ate a big cake

$\exists x\ (\ Big(x)\ \wedge\ Cake(x)\ \wedge\ ate(Jim, x)\ )$

Everybody loves Raymond

$\forall x\ (\ Person(x) \Rightarrow Loves(x, Raymond)\ )$

The lord of the rings

$\exists x\ \forall y\ Lord(x, y)\ \wedge\ Ring(y)$

Mission Impossible

$\neg\exists x, y\ Mission(x)\ \wedge\ Solution(x, y)$

Fun with PC Sentences…

Try to write the followings in PC sentences:

- *There is someone who is loved by everybody*
- *All cats are lazy*
- *Some students are clever*
- *No student is rich*
- *Every man loves a woman*
- *Everything is bitter or sweet*
- *Everything is bitter or everything is sweet*
- *Martin has a new bicycle*
- *Lynn gets a present from John, but she doesn't get anything from Peter*

Next, we will look at inference in Predicate Calculus.

Inference in Predicate Calculus

Can we prove "Socrates is mortal" based on the knowledge:

1. *$\forall x$ Man(x) $\Rightarrow$ Mortal (x)*
2. *Man(Socrates)*

We can **instantiate** sentence 1, then get

3. *Man(Socrates) $\Rightarrow$ Mortal (Scorates)*

Man(x) $\Rightarrow$ *Mortal (x)* is true for all *x*, so is true for any instance of *x* such as *Socrates*, *John*, *Richard* …… [**Universal Instantiation**]

Apply Modus Ponens $\dfrac{\alpha \Rightarrow \beta, \alpha}{\beta}$ on sentences 3 & 2, then we get

4. *Mortal(Socrates)*

So Socrates is mortal.

Inference in Predicate Calculus

Inference in PC is an extension of inference in propositional logic.

We can reduce PC Inference to propositional inference [Propositionalization]
(Need to deal with *substitutions* and *quantifiers*)

The basic idea

- A variable of sentences with existential quantifier can be substituted by one instantiation.
- A variable of sentences with universal quantifier can be substituted by the set of *all possible* instantiations.

Substitution is written as { <variable> / <instantiation> }.

E.g. $\forall x \; Man(x) \Rightarrow Mortal(x)$

Substitutions are: $\{x/Socrates\}$, $\{x/John\}$, $\{x/Richard\}$ ......

Generalized Modus Ponens

Modus Ponens can be extended for inference in PC – Generalized Modus Ponens.

$$\frac{p'_1, p'_2, \ldots, p'_n, (p_1 \wedge p_2 \wedge \ldots \wedge p_n) \Rightarrow q}{SUBST(\theta, q)}$$

where θ is a substitution such that for all i, $SUBST(\theta, p'_i) = SUBST(\theta, p_i)$

$SUBST(\theta, \alpha)$ denotes the result of applying the substitution θ to sentence α.

There are $n+1$ premises to this rule: the n atomic sentences p'_i and the one implication.

It is basically modus ponens + to find a substitution for the variables in sentences.

Generalized Modus Ponens…

Example of Generalized Modus Ponens (GMP) – to prove Socrates is mortal

1. $\forall x \; Man(x) \Rightarrow Mortal(x)$ *2.* $Man(Socrates)$

p'_1 is $Man(Socrates)$, p_1 is $Man(x)$, q is $Mortal(x)$

We can find a substitution θ of x to make $SUBST(\theta, p'_1) = SUBST(\theta, p_1)$

$$\downarrow$$

$$SUBST(\theta, Man(Socrates)) = SUBST(\theta, Man(x))$$

The right substitution θ is $\{x/Socrates\}$,

so $SUBST(\theta, q) = SUBST(\theta, Mortal(x)) = Mortal(Socrates)$

Now we can have the rule

$$\frac{Man(Socrates) \quad Man(x) \Rightarrow Mortal(x)}{Mortal(Socrates)}$$

Therefore, Socrates is mortal.

Unification

Finding substitutions is a key component in PC inference. The process is called **unification.**

A unification algorithm takes two sentences and returns a substitution if one exists

$$UNIFY(p,q) = \theta \quad \text{where } SUBST(\theta, p) = SUBST(\theta, q)$$

Examples:

	Sentence 1	Sentence 2	θ
1	*p(John)*	*p(x)*	$\{x/John\}$
2	*p(John)*	*p(Richard)*	Failure
3	*p(John, x)*	*p(y, Mary)*	$\{y/John,\ x/Mary\}$
4	*p(x, x)*	*p(John,y)*	$\{x/John,\ y/John\}$
5	*p(x, x)*	*p(John, Richard)*	Failure
6	*p(x)*	*p(y)*	$\{x/y\}$

Unification…

As the result of a unification

- Some variables are replaced with a constant (e.g. 1, 3 of previous examples)
- Some (two or more) variables are made identical (e.g. example 4, 6)

When a variable would need to be bound to two distinct values, unification returns failure (e.g. example 5)

Once a variable is given a value via unification, that value cannot be replaced.

There could be more than one substitution returned. For example, θ for *p(x)* and *p(y)* is $\{x/y\}$. However $\{x/John,\ y/John\}$, $\{x/Mary,\ y/Mary\}$ … can also let $p(x) = p(y)$

Unification should only return the most general one, which makes the least commitment to constants. It is called **most general unifier** (MGU).

The algorithm for computing MGUs can be found in the text book (p. 278).

Example of Inference in PC

American law says that is a crime for Americans to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is an American. Prove that Colonel West is a criminal.

The approach is:

1. decide predicates and functions
2. construct a knowledge base
3. search for a proof
   - By forward-chaining

     starts with known facts and infer new ones until reach the goal
   - Or by backward-chaining

     starts with the goal, and work backwards until hit known facts

Example...

Based on the story and common sense, a knowledge base can be set as:

1. $\forall x,y,z\ American(x) \land Weapon(y) \land Sells(x,y,z) \land Hostile(z) \Rightarrow Criminal(x)$

   ... it is a crime for an American to sell weapons to hostile nations:

2. $Owns(Nono,M1)$
3. $Missile(M1)$

   Nono ... has some missiles,

4. $\forall x\ Missile(x) \land Owns(Nono,x) \Rightarrow Sells(West,x,Nono)$

   ... all of its missiles were sold to it by Colonel West

5. $\forall x\ Missile(x) \Rightarrow Weapon(x)$

   Missiles are weapons

6. $\forall x\ Enemy(x,America) \Rightarrow Hostile(x)$

   An enemy of America counts as "hostile"

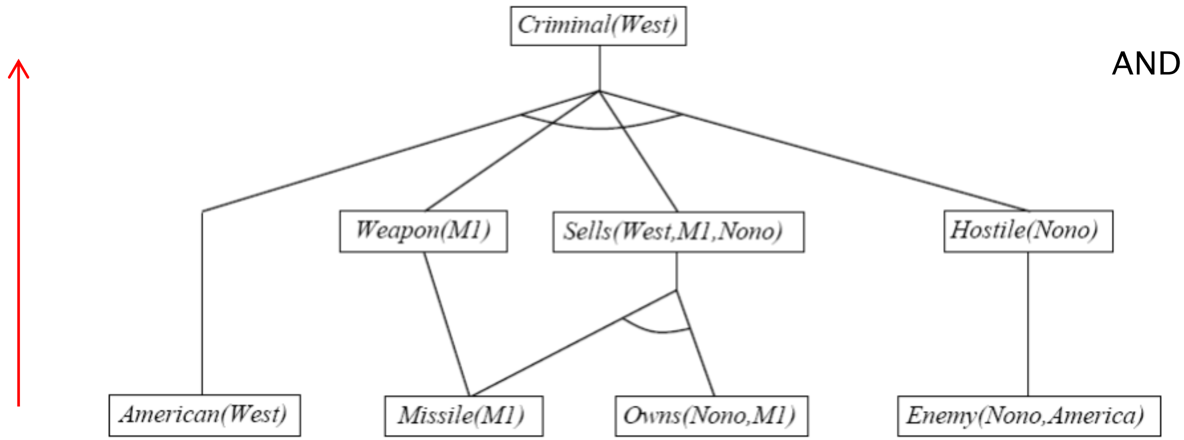
7. $American(West)$

   West, who is American ...

8. $Enemy(Nono,America)$

   The country Nono, an enemy of America ...

Proof by Forward Chaining



- Start from known facts:

  2. *Owns(Nono,M1)* 3. *Missile(M1)* 7. *American(West)* 8. *Enemy(Nono,America)*

- Infer new facts by applying sentences 4, 5, 6:

  Missile(M1) $\wedge$ *Owns(Nono,M1)* $\Rightarrow$ *Sells(West,M1,Nono)* 9.

  Missile(M1) $\Rightarrow$ *Weapon(M1)* 10.

  Enemy(Nono,America) $\Rightarrow$ *Hostile(Nono)* 11.

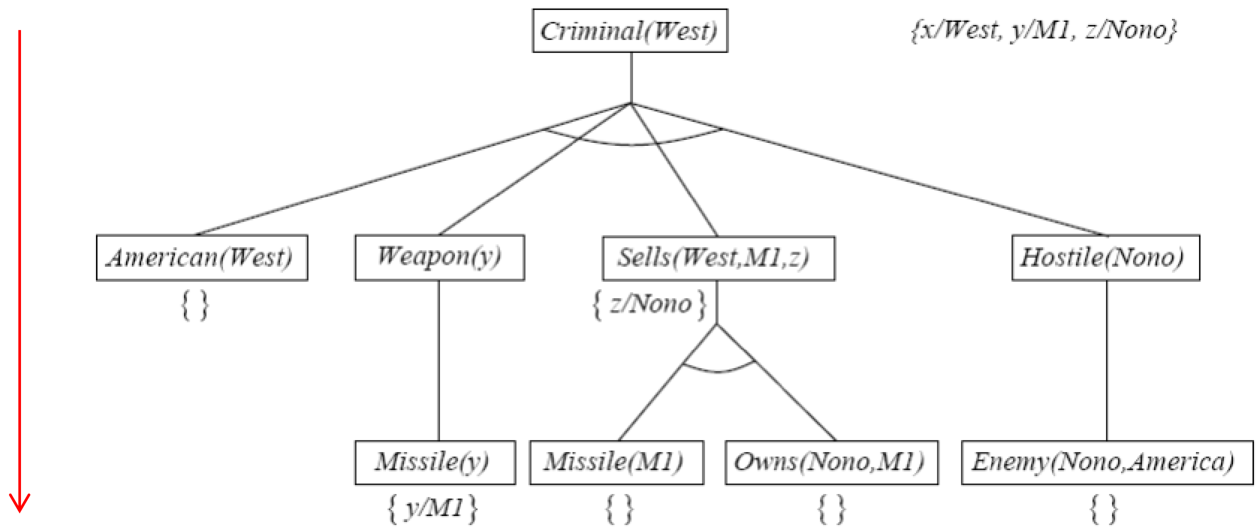
- Infer the conclusion by apply 1 & 7, 9,10, 11

  American(West) $\wedge$ *Weapon(M1)* $\wedge$ *Sells(West,M1,Nono)* $\wedge$ *Hostile(Nono)* $\Rightarrow$ *Criminal(West)*

Proof by Backward Chaining

Backward chaining starts from the goal – *Criminal(West)*

The backward chaining process can be represented as a proof tree. The tree should be read **depth first, left to right.**



The arc across branches mean that each branch must be proved.

A proof branch terminates at an atomic sentence which has no variables.

Proof by Backward Chaining…

The goal is *Criminal(West).*

This can be established by rule 1 under substitution $\{x/West\}$

American(West) $\wedge$ *Weapon(y)* $\wedge$ *Sells(West,y,z)* $\wedge$ *Hostile(z)* $\Rightarrow$ *Criminal(West)*

Now each of the following (one branch in the proof tree) must be established:

- *American(West)*
- *Weapon(y)*
- *Sells(West, y, z)*
- *Hostile(z)*

Some of these are in the knowledge base e.g. *American(West)*, and others requires further backward chaining e.g. *Weapon(y)* .

To establish *Weapon(y),* we need establish *Missile(y)* (rule 5)

(see next)

Proof by Backward Chaining…

Missile(y) can be established under substitution $\{y/M1\}$ – already in the knowledge base.

Next we will establish *Sells(West, M1, z)*. Note y in *Sells(West, y, z)* is already $M1$.

Sells(West, M1, z) can be established under substitution $\{z/Nono\}$ by *Missile(M1)* and *Owns(Nono,M1)* – which are already in the knowledge base.

Now we need to establish *Hostile(Nono)*.

It is true because *Enemy(x, America)* $\Rightarrow$ *Hostile(x)* and *Enemy(Nono,America)* (GMP)

Each predicate required to prove *Criminal(West)* are proved.

So Colonel West is a criminal.

Search for a Proof Tree

```
function pattern_search(current_goal)
    if current_goal is in CLOSED
    then return fail
    else add current_goal to CLOSED

    While there remain unifying facts or rule do
       case: current_goal unifies with a fact
             return success
       case: current_goal is a conjunction
                for each conjunction
                   pattern_search( conjunction )
                if pattern_search succeeded for all
                then return success
                else return fail
       case: current_goal unifies with implication (p in q=>p)
                apply goal unifying substitution to premise q
                pattern_search( premise )
                if pattern_search succeeds
                then return success
                else return fail
    End while
    Return fail
```

Drawbacks of Logic

Logical consequence (for First Order Logic) is only semi-decidable:

given a query, there is no possibility of an algorithm which is guaranteed to return the correct answer every time.

Logical proofs can be computationally prohibitive:

even problems in Propositional Logic are NP-complete.

Logic does not deal well with uncertainty and default rules

Tomorrow maybe a sunny day.

Birds fly.

Logic does not provide enough “structure” to the knowledge base creation process.

Summary

- Knowledge base and Inference engine
- Logic – syntax, semantic, inference mechanism, WFF
- **Propositional Logic**
  - constants, propositional symbols, connectives, sentences, interpretation
  - Implication, Equivalence, Validity, Satisfiability,
  - Inference, Modus Ponens, And-Elimination
  - Monotonicity, Horn Clauses, AND-OR Graph, Forward/Backward Chaining

Summary

Predicate Calculus (First-order Logic)

- constants, variables, predicates, functions, quantifiers, connectives, equality
- atomic sentences and complex sentences
- Universal quantifiers, existential quantifiers, multiple quantifiers
- Knowledge representation in PC, domain
- Inference, Generalized Modus Ponens, Unification, Most general unifier
- Forward/Backward Chaining, Proof tree

Acknowledgement: the slides were developed based on notes from Russell & Norvig, by several computer science staff members over the years.