LU-15: USING FIRST ORDER LOGIC

LU Objectives	
To explain assertions and queries in FOL	
LU Outcomes	CO : 3
Represent knowledge base using assertions	

Assertions and Queries

- Sentences are added to a knowledge base using TELL, exactly as in propositional logic. Such sentences are called **assertions.**
- **For Ex: we can assert that**

John is a king, Richard is a person, and all kings are persons:

TELL(KB, King(John)) .

TELL(KB, Person(Richard)) .

TELL(KB, $\forall$ x King(x) $\Rightarrow$ Person(x)) .

Assertions and Queries

- We can ask questions of the knowledge base using ASK.
- For Ex,

  ASK(KB, King(John))

  returns true.
- Questions asked with ASK are called **queries or goals**.

The kinship domain

- It is the domain of family relationships
- The objects in our domain are people
- There are two unary predicates, Male and Female.
- Kinship relations are parenthood, brotherhood, marriage, and so on. They are represented by binary predicates: Parent, Sibling, Brother , Sister , Child , Daughter, Son, Spouse, Wife, Husband, Grandparent , Grandchild , Cousin, Aunt, and Uncle
- We use functions for Mother and Father , because every person has exactly one of each of these

The kinship domain

- For example, one's mother is one's female parent:
  $\forall$ m, c Mother (c)=m $\Leftrightarrow$ Female(m) $\wedge$ Parent(m, c) .
- One's husband is one's male spouse:
  $\forall$ w, h Husband(h,w) $\Leftrightarrow$ Male(h) $\wedge$ Spouse(h,w) .
- Male and female are disjoint categories:
  $\forall$ x Male(x) $\Leftrightarrow$ $\neg$Female(x) .
- Parent and child are inverse relations:
  $\forall$ p, c Parent(p, c) $\Leftrightarrow$ Child (c, p) .
- A grandparent is a parent of one's parent:
  $\forall$ g, c Grandparent (g, c) $\Leftrightarrow$ $\exists$p Parent(g, p) $\wedge$ Parent(p, c)
- A sibling is another child of one's parents:
  $\forall$ x, y Sibling(x, y) $\Leftrightarrow$ x = y $\wedge$ $\exists$p Parent(p, x) $\wedge$ Parent(p, y)

Axioms and Theorems

- Each of the sentences in previous slide can be viewed as an **axiom of the kinship domain.**
- Axioms are commonly associated with purely mathematical domains. But they are needed in all domains.
- They provide the basic factual information from which useful conclusions can be derived. Our kinship axioms are also **definitions**

Axioms and Theorems

- The axioms define the Mother function and the Husband, Male, Parent, Grandparent, and Sibling predicates in terms of other predicates.
- Our definitions “bottom out” at a basic set of predicates (Child , Spouse, and Female) in terms of which the others are ultimately defined.
- There is not necessarily a unique set of primitive predicates;
- we could equally well have used Parent, Spouse, and Male. In some domains, there is no clearly identifiable basic set.

Axioms and Theorems

- Some logical sentences are **theorems—that is, they** are entailed by the axioms.
- For example, consider the assertion that siblinghood is symmetric:

  $\forall$ x, y Sibling(x, y) $\Leftrightarrow$ Sibling(y, x)
- In fact, it is a theorem that follows logically from the axiom that defines siblinghood.
- If we ASK the knowledge base this sentence, it should return true

Why need theorems?

- From a purely logical point of view, a knowledge base need contain only axioms and no theorems, because the **theorems do not increase the set of conclusions** that follow from the knowledge base.
- But from a practical point of view, theorems are essential to reduce the computational cost of deriving new sentences.
- Without theorems, a reasoning system has to start from first principles every time.

Numbers

- Numbers are perhaps the most vivid example of how a large theory can be built up from
- a tiny kernel of axioms.
- Consider the theory of **natural numbers or non-negative** integers.
- It uses a predicate NatNum that will be true of natural numbers;
- we need one constant symbol, 0; and we need one function symbol, S (successor).

Numbers

- Natural numbers are defined recursively:

  $NatNum(0)$.

  $\forall\, n\ NatNum(n) \Rightarrow NatNum(S(n))$.

- We also need axioms to constrain the successor function:

  $\forall\, n\ 0 \neq S(n)$.

  $\forall\, m, n\ m \neq n \Rightarrow S(m) \neq S(n)$.

- We can define addition in terms of the successor function:

  $\forall m\ NatNum(m) \Rightarrow +(0,m) = m$.

  $\forall\, m, n\ NatNum(m) \wedge NatNum(n) \Rightarrow +(S(m), n) = S(+(m, n))$

Numbers

- To make sentences about numbers easier to read, we allow the use of infix notation.
- We can also write $S(n)$ as $n+ 1$, so the second axiom becomes

  $$\forall\ m, n\ NatNum(m) \wedge NatNum(n) \Rightarrow (m + 1) + n = (m + n) + 1\ .$$

- This axiom reduces addition to repeated application of the successor function.
- The use of infix notation is an example of **syntactic sugar.**

Numbers

- Similarly we can define multiplication as repeated addition, exponentiation as repeated multiplication, integer division and remainders, prime numbers, and so on.
- Thus, the whole of number theory can be built up from one constant, one function, one predicate and four axioms

Sets

The set domain representations using FOL or Predicate Logic:

- The empty set is a constant written as {}.
- Unary Predicate : Set
- Binary Predicate: $x \in s$ ((x is a member of set s), $s_1 \subseteq s_2$ (set s1 is a subset of set s2)
- Binary function: $s_1 \cap s_2$ (the intersection of two sets), $s_1 \cup s_2$ (the union of two sets), $\{x|s_2\}$) (Set resulting from adjoining element x to s)

Sets - Axioms

1. The only sets are the empty set and those made by adjoining something to a set:

   $\forall$ s Set(s) $\Leftrightarrow$ (s={}) $\vee$ ($\exists$ x, s2 Set(s2) $\wedge$ s={x|s2}) .

2. The empty set has no elements adjoined into it. In other words, there is no way to decompose {} into a smaller set and an element:

   $\neg\exists$ x, s {x|s}={} .

3. Adjoining an element already in the set has no effect:

   $\forall$ x, s x$\in$ s $\Leftrightarrow$ s={x|s} .

Sets - Axioms

4. The only members of a set are the elements that were adjoined into it.

   $\forall$ x, s x$\in$ s $\Leftrightarrow$ $\exists$y, s2 (s={y|s2} $\wedge$ (x=y $\vee$ x$\in$ s2)) .

5. A set is a subset of another set if and only if all of the first set's members are members of the second set:

   $\forall$ s1, s2 s1 $\subseteq$ s2 $\Leftrightarrow$ ($\forall$x x$\in$ s1 $\Rightarrow$ x$\in$ s2) .

6. Two sets are equal if and only if each is a subset of the other:

   $\forall$ s1, s2 (s1 =s2) $\Leftrightarrow$ (s1 $\subseteq$ s2 $\wedge$ s2 $\subseteq$ s1)

Sets - Axioms

7. An object is in the intersection of two sets if and only if it is a member of both sets:

$$\forall\ x, s1, s2\ x \in (s1 \cap s2) \Leftrightarrow (x \in s1 \wedge x \in s2)\ .$$

8. An object is in the union of two sets if and only if it is a member of either set:

$$\forall\ x, s1, s2\ x \in (s1 \cup s2) \Leftrightarrow (x \in s1 \vee x \in s2)\ .$$

Lists

- **Lists are similar to sets.**
- The differences are LIST that lists are ordered and the same element can appear more than once in a list.
- Nil is the constant list with no elements;
- Cons, Append, First, and Rest are functions; and Find is the predicate that does for lists what Member does for sets.
- List? is a predicate that is true only of lists.
- The empty list is [ ].
- The term Cons(x, y), where y is a nonempty list, is written [x|y]. The term Cons(x, Nil) (i.e., the list containing the element x) is written as [x].
- A list of several elements, such as [A,B,C], corresponds to the nested term Cons(A, Cons(B, Cons(C, Nil))).

The wumpus world

- The first order axioms in this section are much more concise than prepositional logic , capturing in a natural world
- WKT the wumpus agent receives a percept vector with five elements. The corresponding first-order sentence stored in the knowledge base must include both the percept and the time at which it occurred; otherwise, the agent will get confused about when it saw what.

The wumpus world

- Typical percept sentence would be

  Percept ([Stench, Breeze, Glitter , None, None], 5) .
- Here, Percept is a binary predicate, and Stench and so on are constants placed in a list.
- The actions in the wumpus world can be represented by logical terms:

  Turn(Right), Turn(Left), Forward , Shoot , Grab, Climb .

The wumpus world

- To determine which is best, the agent program executes the query

  ASKVARS($\exists$ a BestAction(a, 5)) ,

  which returns a binding list such as {a/Grab}.
- The raw percept data implies certain facts about the current state.
- For Ex:

  $\forall$ t, s, g, m, c Percept ([s, Breeze, g,m, c], t) $\Rightarrow$ Breeze(t) ,

  $\forall$ t, s, b, m, c Percept ([s, b, Glitter,m, c], t) $\Rightarrow$ Glitter (t) ,
- These rules exhibit a trivial form of the reasoning process called **perception.**

The wumpus world

- Simple "reflex" behavior can also be implemented by quantified implication sentences.
- For example,

  $\forall$ t Glitter (t) $\Rightarrow$ BestAction(Grab, t) .
- Objects in the environment are squares, pits, and the wumpus
- Adjacency of any two squares can be defined as

  $\forall$ x, y, a, b Adjacent ([x, y], [a, b]) $\Leftrightarrow$

  (x = a $\wedge$ (y = b − 1 $\vee$ y = b + 1)) $\vee$ (y = b $\wedge$ (x = a − 1 $\vee$ x = a + 1)) .

The wumpus world

- A unary predicate Pit is true for squares containing pit
- Since there is exactly one wumpus, a constant Wumpus is just as good as a unary predicates.
- The agent’s location changes over time, so we write At(Agent, s, t) to mean that the agent is at square s at time t.
- We can then say that objects can only be at one location at a time:

$$\forall\ x, s1, s2, t\ At(x, s1, t) \land At(x, s2, t) \Rightarrow s1 = s2\ .$$

Diagnostic Rules - The wumpus world

- These rules generate hidden causes from observed effects. They help to reduce hidden facts in the world.
- Diagnostic Rule for finding pit
  1. *If square is breezy the agent can deduce some adjacent square must contain pit*

     $\forall$ s Breezy(s) $\Rightarrow$ $\exists$r Adjacent (r, s) $\wedge$ Pit(r) .
  2. *If square is not breezy the agent can deduce no adjacent square contains a pit*

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

Diagnostic Rules - The wumpus world

- Combining two rules we obtain the biconditional sentence

$$\forall\ s\ Breezy(s) \Leftrightarrow \exists r\ Adjacent\ (r, s) \land Pit(r)\ .$$

Causal Rules - The wumpus world

- Causal rules reflect the assumed direction of causality in the world. That is these rules are used to infer effect from cause.
- *Some hidden property of the world causes certain percepts to be generated.*
- *Ex: 1) a pit causes all adjacent squares to be breezy*

$$\forall\ r\ Pit(r) \Rightarrow [\forall s\ Adjacent\ (r, s) \Rightarrow Breezy(s)]$$

2) If all squares adjacent to a given square are pitless, the square will not be breezy

$$\forall s[\forall\ r\ Adjacent\ (r, s) \Rightarrow \neg Pit(r)] \Rightarrow \neg Breezy(s)]$$

Quantification- The wumpus world

- In first-order logic we can quantify over time. For example, the axiom for the arrow becomes

$$\forall\, t\ HaveArrow(t+1) \Leftrightarrow (HaveArrow(t) \wedge \neg Action(Shoot, t))\ .$$

.