Module 5

Knowledge Representation and Logic – (Propositional Logic)

5.1 Instructional Objective

- Students should understand the importance of knowledge representation in intelligent agents
- Students should understand the use of formal logic as a knowledge representation language
- The student should be familiar with the following concepts of logic
    - syntax
    - semantics
    - validity
    - satisfiability
    - interpretation and models
    - entailment
- Students should understand each of the above concepts in propositional logic
- Students should learn different inference mechanisms in propositional logic

At the end of this lesson the student should be able to do the following:

- Represent a natural language description as statements in logic
- Deduct new sentences by applying inference rules.

Lesson 11

Propositional Logic

5.2 Knowledge Representation and Reasoning

Intelligent agents should have capacity for:

- **Perceiving**, that is, acquiring information from environment,
- **Knowledge Representation**, that is, representing its understanding of the world,
- **Reasoning**, that is, inferring the implications of what it knows and of the choices it has, and
- **Acting**, that is, choosing what it want to do and carry it out.

Representation of knowledge and the reasoning process are central to the entire field of artificial intelligence. The primary component of a knowledge-based agent is its knowledge-base. A knowledge-base is a set of sentences. Each sentence is expressed in a language called the knowledge representation language. Sentences represent some assertions about the world. There must mechanisms to derive new sentences from old ones. This process is known as inferencing or reasoning. Inference must obey the primary requirement that the new sentences should follow logically from the previous ones.

Logic is the primary vehicle for representing and reasoning about knowledge. Specifically, we will be dealing with formal logic. The advantage of using formal logic as a language of AI is that it is precise and definite. This allows programs to be written which are declarative - they describe what is true and not how to solve problems. This also allows for automated reasoning techniques for general purpose inferencing.

This, however, leads to some severe limitations. Clearly, a large portion of the reasoning carried out by humans depends on handling knowledge that is uncertain. Logic cannot represent this uncertainty well. Similarly, natural language reasoning requires inferring hidden state, namely, the intention of the speaker. When we say, "One of the wheel of the car is flat.", we know that it has three wheels left. Humans can cope with virtually infinite variety of utterances using a finite store of commonsense knowledge. Formal logic has difficulty with this kind of ambiguity.

A logic consists of two parts, a language and a method of reasoning. The logical language, in turn, has two aspects, syntax and semantics. Thus, to specify or define a particular logic, one needs to specify three things:

Syntax: The atomic symbols of the logical language, and the rules for constructing well-formed, non-atomic expressions (symbol structures) of the logic. Syntax specifies the symbols in the language and how they can be combined to form sentences. Hence facts about the world are represented as sentences in logic.

Semantics: The meanings of the atomic symbols of the logic, and the rules for determining the meanings of non-atomic expressions of the logic. It specifies what facts in the world a sentence refers to. Hence, also specifies how you assign a truth value to a sentence based on its meaning in the world. A **fact** is a claim about the world, and may be true or false.

Syntactic Inference Method: The rules for determining a subset of logical expressions, called theorems of the logic. It refers to mechanical method for computing (deriving) new (true) sentences from existing sentences.

Facts are claims about the world that are True or False, whereas a **representation** is an expression (sentence) in some language that can be encoded in a computer program and stands for the objects and relations in the world. We need to ensure that the representation is consistent with reality, so that the following figure holds:

```
                             entails
Representation:    Sentences --------------> Sentences
                    |                          |
                    |                          |
                    | Semantics                | Semantics
                    | refer to                 | refer to
                    |                          |
                   \/      follows            \/
World:           Facts ------------------> Facts
```

There are a number of logical systems with different syntax and semantics. We list below a few of them.

- Propositional logic

  All objects described are fixed or unique

  "John is a student" student(john)

  Here John refers to one unique person.

- First order predicate logic

  Objects described can be unique or variables to stand for a unique object

  "All students are poor"

  ForAll(S) [student(S) -> poor(S)]

  Here S can be replaced by many different unique students.

  This makes programs much more compact:

  eg. ForAll(A,B)[brother(A,B) -> brother (B,A)]

  replaces half the possible statements about brothers

- Temporal

  Represents truth over time.

- Modal

  Represents doubt

- Higher order logics

  Allows variable to represent many relations between objects

- Non-monotonic

  Represents defaults

Propositional is one of the simplest systems of logic.

5.3 Propositional Logic

In propositional logic (PL) an user defines a set of propositional symbols, like *P* and *Q*. User defines the semantics of each of these symbols. For example,

  - P means "It is hot"
  - Q means "It is humid"
  - R means "It is raining"
  -

- A **sentence** (also called a formula or well-formed formula or wff) is defined as:
  1. A symbol
  2. If S is a sentence, then ~S is a sentence, where "~" is the "not" logical operator
  3. If S and T are sentences, then (S v T), (S ^ T), (S => T), and (S <=> T) are sentences, where the four logical connectives correspond to "or," "and," "implies," and "if and only if," respectively
  4. A finite number of applications of (1)-(3)

- Examples of PL sentences:
  - (P ^ Q) => R (here meaning "If it is hot and humid, then it is raining")
  - Q => P (here meaning "If it is humid, then it is hot")
  - Q (here meaning "It is humid.")

- Given the truth values of all of the constituent symbols in a sentence, that sentence can be "evaluated" to determine its truth value (True or False). This is called an **interpretation** of the sentence.

- A **model** is an interpretation (i.e., an assignment of truth values to symbols) of a set of sentences such that each sentence is True. A model is just a formal mathematical structure that "stands in" for the world.

- A **valid** sentence (also called a **tautology**) is a sentence that is True under *all* interpretations. Hence, no matter what the world is actually like or what the semantics is, the sentence is True. For example "It's raining or it's not raining."

- An **inconsistent** sentence (also called **unsatisfiable** or a **contradiction**) is a sentence that is False under *all* interpretations. Hence the world is never like what it describes. For example, "It's raining and it's not raining."

- Sentence P **entails** sentence Q, written P |= Q, means that whenever P is True, so is Q. In other words, all models of P are also models of Q

Entailment ($\models$): Given 2 *sentences* p and q we say p *entails* q, written $p \models q$, if q holds in every model that p holds.

Example: Entailment

Show that: $p \wedge (p \Rightarrow q) \models q$

Proof: For any model M in which $p \wedge (p \Rightarrow q)$ holds then we know that *p* holds in M and $p \Rightarrow q$ holds in M. Since *p* holds in M then since $p \Rightarrow q$ holds in M, *q* must hold in M. Therefore *q* holds in every model that $p \wedge (p \Rightarrow q)$ holds and so $p \wedge (p \Rightarrow q) \models q$.

As we have noted models affect equivalence and so we repeat the definition again and give an example of a proof of equivalence.

Equivalence: Two sentences are *equivalent* if they hold in exactly the same models.

Example: Equivalence

Show that: $p \Rightarrow q \equiv \neg p \vee q$

Proof: We need to provide two proofs as above for
$p \Rightarrow q \models \neg p \vee q$

- For any model M in which $p \Rightarrow q$ holds then we know that either p holds in M and so q holds in M, or p does not hold in M and so $\neg p$ holds in M. Since either q holds in M or $\neg p$ holds in M, then $\neg p \vee q$ holds in M.

and $\neg p \vee q \models p \Rightarrow q$

- For any model M in which $\neg p \vee q$ holds then we know that either $\neg p$ holds in M or q holds in M. If $\neg p$ holds in M then $p \Rightarrow q$ holds in M. Otherwise, if q holds in M then $p \Rightarrow q$ holds in M. Therefore $p \Rightarrow q$ holds in M.

Knowledge based programming relies on concluding new knowledge from existing knowledge. Entailment is a required justification; i.e. if $p_1, \ldots, p_n$ is known then there is justification to conclude q if

$$p_1 \wedge \ldots \wedge p_n \models q$$

In some circumstances we insist on this strong form of justification; i.e. we cannot conclude q unless the entailment holds. Reasoning like this is the equivalent for knowledge based programs of running a piece of conventional software.

Note: Entailment ($\models$) is concerned with *truth* and is determined by considering the truth of the sentences in all models.

5.4 Propositional Logic Inference

Let KB = { S1, S2,..., SM } be the set of all sentences in our Knowledge Base, where each Si is a sentence in Propositional Logic. Let { X1, X2, ..., XN } be the set of all the symbols (i.e., variables) that are contained in all of the M sentences in KB. Say we want to know if a goal (aka query, conclusion, or theorem) sentence G follows from KB.

5.4.1 Model Checking

Since the computer doesn't know the interpretation of these sentences in the world, we don't know whether the constituent symbols represent facts in the world that are True or False. So, instead, consider *all* possible combinations of truth values for all the symbols, hence enumerating all logically distinct cases:

```
X1 X2 ... XN | S1 S2 ... SM | S1 ^ S2 ^...^ SM | G | (S1 ^...^ SM) => G
-------------|--------------|------------------|---|--------------------
F  F  ... F  |              |                  |   |
F  F  ... T  |              |                  |   |
...          |              |                  |   |
T  T  ... T  |              |                  |   |
```

- There are 2^N rows in the table.
- Each row corresponds to an equivalence class of worlds that, under a given interpretation, have the truth values for the N symbols assigned in that row.
- The **models** of KB are the rows where the third-to-last column is *true*, i.e., where all of the sentences in KB are *true*.
- A sentence R is **valid** if and only if it is true under all possible interpretations, i.e., if the entire column associated with R contains all *true* values.
- Since we don't know the semantics and therefore whether each symbol is True or False, to determine if a sentence G is **entailed** by KB, we must determine if **all** models of KB are also models of G. That is, whenever KB is true, G is true too. In other words, whenever the third-to-last column has a T, the same row in the second-to-last column also has a T. But this is logically equivalent to saying that the sentence (KB => G) is valid (by definition of the "implies" connective). In other words, if the last column of the table above contains only *True*, then **KB entails G**; or conclusion G logically follows from the premises in KB, no matter what the interpretations (i.e., semantics) associated with all of the sentences!
- The truth table method of inference is **complete** for PL (Propositional Logic) because we can always enumerate all 2^*n* rows for the *n* propositional symbols that occur. But this is exponential in *n*. In general, it has been shown that the problem of checking if a set of sentences in PL is satisfiable is NP-complete. (The truth table method of inference is *not* complete for FOL (First-Order Logic).)

Example

Using the "weather" sentences from above, let KB = (((P ^ Q) => R) ^ (Q => P) ^ Q) corresponding to the three facts we know about the weather: (1) "If it is hot and humid, then it is raining," (2) "If it is humid, then it is hot," and (3) "It is humid." Now let's ask the query "Is it raining?" That is, is the query sentence R entailed by KB? Using the truth-table approach to answering this query we have:

P	Q	R	(P ^ Q) => R	Q => P	Q	KB	R	KB => R
T	T	T	T	T	T	T	T	T
T	T	F	F	T	T	F	F	T
T	F	T	T	T	F	F	T	T
T	F	F	T	T	F	F	F	T
F	T	T	T	F	T	F	T	T
F	T	F	T	F	T	F	F	T
F	F	T	T	T	F	F	T	T
F	F	F	T	T	F	F	F	T

Hence, in this problem there is only one model of KB, when P, Q, and R are all True. And in this case R is also True, so R is entailed by KB. Also, you can see that the last column is all True values, so the sentence KB => R is valid.

Instead of an exponential length proof by truth table construction, is there a faster way to implement the inference process? Yes, using a **proof procedure** or **inference procedure** that uses **sound rules of inference** to deduce (i.e., derive) new sentences that are true in all cases where the premises are true. For example, consider the following:

P	Q	P	P => Q	P ^ (P => Q)	Q	(P ^ (P => Q)) => Q
F	F	F	T	F	F	T
F	T	F	T	F	T	T
T	F	T	F	F	F	T
T	T	T	T	T	T	T

Since whenever P and P => Q are both true (last row only), Q is true too, Q is said to be **derived** from these two premise sentences. We write this as KB |- Q. This local pattern referencing only two of the M sentences in KB is called the **Modus Ponens** inference rule. The truth table shows that this inference rule is **sound**. It specifies how to make one kind of step in deriving a conclusion sentence from a KB.

Therefore, given the sentences in KB, construct a **proof** that a given conclusion sentence can be derived from KB by applying a sequence of sound inferences using either sentences in KB or sentences derived earlier in the proof, until the conclusion sentence is derived. This method is called the **Natural Deduction** procedure. (Note: This step-by-step, local proof process also relies on the **monotonicity** property of PL and FOL. That is, adding a new sentence to KB does not affect what can be entailed from the original KB and does not invalidate old sentences.)