CENG 466
Artificial Intelligence

Lecture 8

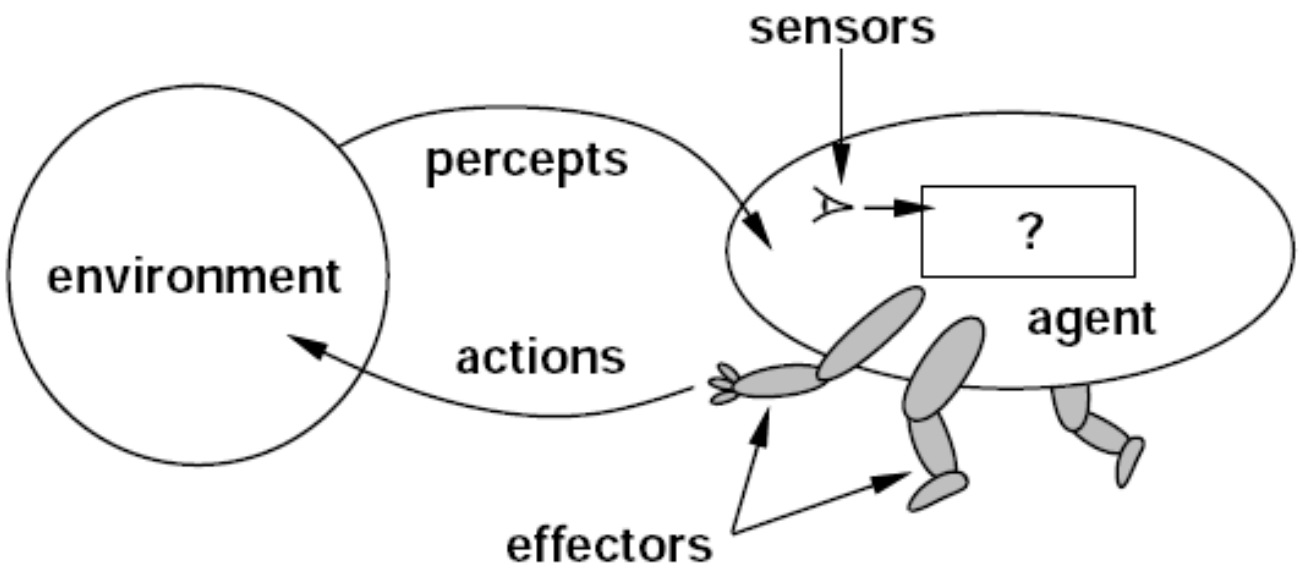
Knowledge Representation

Topics

- Knowledge Base
- Knowledge Representation
- Propositional Logic
- Predicate Logic
- Example

Intelligent Agents

- An agent is something that perceives and acts in an environment
- An ideal agent always takes actions that maximizes its performance
- An agent adopts a goal and searches the best path to reach that goal



Intelligent Agents

- An agent that has goals and searches for solutions to these goals.
- Therefore, an agent should have some knowledge about its goal, its environment, and its actions.
- In addition, the agent should be capable of general logical reasoning.

Knowledge-based Agents

- knowledge-based agents are agents which
  - can be seen as *knowing* about their world,
  - and *reasoning* about their possible action.

Knowledge Base (I)

- The central component of a knowledge-based agent is its **knowledge base**, or KB.
- Informally, a knowledge base is a set of facts about the world.
- Each fact in a KB is called a **sentence**.

Knowledge Base (II)

- The sentences are expressed in a language called a **knowledge representation language**.
- To add new sentences to the knowledge base we use TELL.
- To query what is known we use ASK.

Knowledge Representation

- The goal of **knowledge representation** is to express knowledge in computer-usable form.
- This format is used to help agents perform well.
- A knowledge representation language is defined by two aspects:
  - Syntax
  - Semantics

Knowledge Representation Syntax

- **The syntax** of a language describes the possible configurations that make sentences.

- A syntax is a set of rules that define the patterns of the sentences in a language.

Knowledge Representation Semantics

- The **semantics** determines the facts in the world to which the sentences refer.

- Semantics is the meaning or the concept that we want to store in a KB.

Knowledge Representation Methods

- Propositional Logic
- Predicate Logic

Propositional Logic

- In propositional logic, facts are written as expressions or propositions.
- Each proposition is given a label (symbol)
- For example, *D* might be "the wumpus is dead."
- Each proposition might be true or false.
- Propositions can be combined using Boolean connectives to generate sentences with more complex meanings.

Syntax

- **The syntax** of prepositional logic is made of
  - The logical constants *True* and *False*,
  - Proposition symbols such as *P* and *Q* ,
  - *T*he logical connectives ∧, ∨, <=>, =>, and ¬,
  - Parentheses, ().

Syntax Rules of Propositional Logic (I)

- The sentences are made by using the following rules:
- The logical constants *True* and *False* are sentences by themselves.
- A prepositional symbol such as *P* or *Q* is a sentence by itself.
- A sentence can be formed by combining simpler sentences using logical connectives, example: P V Q.
- Putting parentheses around a sentence gives a sentence, for example, (P V Q).
- A sentence using ∧, such as $P \wedge Q$, is called a **conjunction.**
- A sentence using V, such as P V *Q* , is called a **disjunction.**

Syntax Rules of Propositional Logic (II)

- (implies). A sentence such as P => *R* is called an **implication** also known as **if-then** statements.
- (equivalent). The sentence *(P ⇔ Q)* is an **equivalence.**
- (not). A sentence such as ¬*P* is called the **negation** of P.
- ¬ is a unary operator.

Semantics

- The **semantics** of prepositional logic is defined by interpretation of the propositions and constants.
- A proposition can mean whatever we want
- Example:
  - *P* might be the fact that “Paris is the capital of France”

Semantics of Complex Sentences

- A complex sentence has a meaning derived from the meaning of its parts.
- Each connective (AND, OR) is like a function.
- A table is used to determine the value of a sentence.
- Such a table is called a **truth table**.

Truth Table for AND Connectives

P	Q	P and Q
T	T	T
T	F	F
F	T	F
F	F	F

Predicate Logic

- Predicate logic or first-order logic assumes that the world is made up of objects.
- Objects have some properties that distinguish them from other objects.

Relations Between Objects

- Various **relations** hold among the objects.
- Some of these relations are **functions** in which there is only one "value" for a given "input."
- Examples:
- Objects: people, houses, numbers, theories, football games, wars.
- Relations: brother of, bigger than, inside, part of, owns
- Properties: red, round, prime, multistory.
- Functions: father of, best friend, one more than

- Predicate logic claims that facts are referring to objects and their properties or relations.
- Examples 1:
  - "One plus two equals three"
  - Objects: one, two, three,
  - Relation: equals;
  - Function: plus.
- Example 2:
  - "Squares neighboring the wumpus are smelly."
  - Objects: wumpus, square;
  - Property: smelly;
  - Relation: neighboring.

Predicate Logic View

- In fact the world is not really made up of objects and relations, but dividing up the world that way helps us reason about it.

- This view, together with the logical connectives from propositional logic, enables us to represent general rules.

Syntax

- Predicate logic has sentences, but it also has **terms**, which represent objects.
- Constant symbols, variables, and function symbols are used to build terms.
- Quantifiers and predicate symbols are used to build sentences.

Constants

- **Constant symbols:** *A*, *B*, ***C***, *John* ...
- Constants specify which object in the world is referred to.
- Each constant symbol names exactly one object, but not all objects need to have names, and some can have several names.

Predicates

- **Predicate symbols**: *Brother*,...
- A predicate specifies a relation in the model.
- For example, the *Brother* symbol might refer to the relation of brotherhood.
- *Brother* is a binary predicate symbol, and accordingly brotherhood is a relation that holds between pairs of objects.

Functions

- **Function symbols:** *FatherOf*, *LeftLegOf*...
- Some relations *are functional—which means* that, any given object is related to exactly one other object by the relation.
- For example: any person has only one person that is his or her father.

Term

- A **term** is a logical expression that refers to an object.
- For example, in English we might use the expression "King John's left leg" rather than giving a name to his leg.
- This is what function symbols are for: instead of using a constant symbol, we use *LeftLegOf (John)*.
- In the general case, a complex term is formed by a function followed by a list of terms as arguments to the function.

Simple Sentences

- A simple or atomic sentence is formed from a predicate symbol followed by a list of terms in parenthesis.
- Example,
  - *Brother(Richard, John)*
- Atomic sentences can have arguments that are complex terms:
  - *Married(FatherOf (Richard), MotherOf (John))*
- *An atomic sentence is true if the relation shown by the predicate holds between the objects given as the arguments*

Complex Sentences

- **Logical connectives are used** to construct more complex sentences.
- Example:
  - *Brother(Richard, John)* ∧ *OlderThan(John, Richard)* is true only when John is the brother of Richard and John is older than Richard.
  - *Older(John*, 30) ∨ *Younger(*Richard, 30) is true only when John is older than 30 or Richard is younger than 30 (or both !).

Quantifiers

- **Universal quantification (∀)**
- This is an advantage of predicate logic over propositional logic.
- It says that a property is true about all objects of a set.
- Example: "All balls are round“
  - ∀X, Ball(X) => Round(X)

Existential Quantifier

- **Existential quantifier** makes a statement about *some* object in the universe without naming it.
- For example, Spot has a sister who is a cat, we write
  - $\exists x, \mathit{Sister}(x, \mathit{Spot}) \wedge \mathit{Cat}(x)$
- $\exists$ is read "There exists ...".

Equality

- We can use the **equality symbol** to make statements to refer to the same object.
- For example,

  Father(John) = Henry

  says that the object referred to by *Father(John)*

  and the object referred to by *Henry* are the same.

Example (I)

- **The kinship domain**
- Facts such as:
  - "Elizabeth is the mother of Charles"
  - "Charles is the father of William,"
- Rules such as:
  - "If x is the mother of y and y is a parent of z, then x is a grandmother of z."
- Objects in this domain are people.
- Relations such as:
  - Parenthood,
  - Brotherhood, Marriage.
- Unary predicates, *Male* and *Female*.
- Binary predicates: *Parent*, *Sibling*, *Brother*, *Sister*, *Child*, *Daughter*, *Son*, *Spouse*, *Wife*, *Husband*, *Grandparent*, *Grandchild*, *Cousin*, *Aunt*, *Uncle*.
- Functions: *Mother* and *Father,* because every person has exactly one mother and one father.

Example (II)

- 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, $Sibling(x,y) \Leftrightarrow x \neq y \wedge \exists$p, $Parent\ (p, x) \wedge Parent\ (p,y)$

Asking Questions and Getting Answers

- To add the kinship sentences to a knowledge base *KB*, call
  - *TELL*(*KB*, (∀ *m*, *c*, *Mother*(*c*) = *m* ⇔ *Female*(*m*) ∧ *Parent*(*m*, *c*)))
  - *TELL*(*KB*, (*Female*(*Maxi*) ∧ *Parent*(*Maxi*, *Spot*) ∧ *Parent*(*Spot*, *Boots*)))
- Then we can ask:
  - *ASK*(*KB*, *Grandparent*(*Maxi*, *Boots*))

Questions?