A knowledge-based agent has at its core a knowledge base.

A knowledge base is a set of facts expressed in a suitable representation about some domain (these facts are called sentences).

Sentences are expressed in a (formal) knowledge representation language.

**Question:** How do we write down knowledge about a domain/problem? Once we have written down knowledge, can we automate or mechanise reasoning to deduce new facts?

**References:**
Why do we need formal languages?

- Consider an English sentence like:
  “The boy saw a girl with a telescope”

  Also:
  “Our shoes are guaranteed to give you a fit” (lexical ambiguity)
  “I heard about him at school” (structural ambiguity)
  “As he uttered the all-important word he dropped his voice, but she just managed to catch it” (ambiguity of cross reference)

- Natural languages exhibit ambiguity

- Not only does ambiguity make it difficult for us to understand what is the intended meaning of certain phrases and sentences but also makes it very difficult to make inferences

- Symbolic logic is a syntactically unambiguous knowledge representation language (originally developed in an attempt to formalize mathematical reasoning)

Syntax vs. Semantics

**Syntax** Describes the legal sentences in a knowledge representation language (e.g. in the language of arithmetic expressions \(x < 4\))

**Semantics** Refers to the meaning of sentences. Relates sentences (and sentence fragments) to aspects of the world the sentence is about. Semantics refers to a sentence’s relationship to the “real world” or to some model of the world. Semantic properties of sentences include truth and falsity (e.g. \(x < 4\) is true when \(x\) is a strictly smaller number than 4 and false otherwise). Semantic properties of names and descriptions include referents.

Note: The meaning of a sentence is not intrinsic to that sentence. An interpretation is required to determine sentence meanings. Interpretations are provided by agreement amongst a linguistic community or by fiat (e.g. use the symbol ‘cat’ to refer to the class of cats).

Propositions

- Propositions are entities (facts or non-facts) that can be true or false

- Propositions are expressed using ordinary declarative sentences
  - e.g. the sentence “The sky is blue” expresses the proposition that the sky is blue. Is this proposition true?

- Other propositions:
  - “Socrates is bald” (assume ‘Socrates’, ‘bald’ are well defined)
  - “The car is red” (assume ‘the car’ is identified)
  - “Socrates is bald and the car is red” (complex proposition)

- Often use single letters to represent propositions in reasoning:
  - \(P\): Socrates is bald
  - (reasoning is independent of propositional substructure!)

- This is referred to as a scheme of abbreviation

Logical Arguments

- All humans have 2 eyes
- Jane is a human
- Therefore Jane has 2 eyes

- All humans have 4 eyes
- Jane is a human
- Therefore Jane has 4 eyes

- **Both** are (logically) correct valid arguments

- Which statements are true/false?
Logical Arguments

All humans have 2 eyes
Jane has 2 eyes
Therefore Jane is human
No human has 4 eyes
Jane has 2 eyes
Therefore Jane is not human

■ Both are (logically) incorrect invalid arguments
■ Which statements are true/false?

Propositional Logic

■ Use letters to stand for “basic” propositions; combine them into more complex sentences using operators for not, and, or, implies, iff

■ Propositional connectives:
  ▶ negation
  ▶ conjunction
  ▶ disjunction
  ▶ implication
  ▶ bi-implication

From English to Propositional Formulae

■ “it is not the case that the sky is blue”: ¬B  
(alternatively “the sky is not blue”)
■ “the sky is blue and the grass is green”: B ∧ G
■ “either the sky is blue or the grass is green”: B ∨ G
■ “if the sky is blue, then the grass is not green”: B → ¬G
■ “the sky is blue if and only if the grass is green”: B ↔ G
■ “if the sky is blue, then if the grass is not green, the plants will not grow”: B → (¬G → ¬P)

Improving Readability

■ (P → (Q → (¬R))) vs. P → (Q → ¬R)
■ Rules for omitting brackets
  ▶ omit brackets where possible
  ▶ precedence from highest to lowest is: ¬, ∧, ∨, →, ↔
  ▶ all binary operators are left associative (so P → Q → R abbreviates (P → Q) → R)
■ Questions:
  ▶ Is (P ∨ Q) ∨ R the same as P ∨ (Q ∨ R)?
  ▶ Is (P → Q) → R the same as P → (Q → R)?
Semantics

- The semantics of the connectives can be given by **truth tables**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>¬P</th>
<th>P ∧ Q</th>
<th>P ∨ Q</th>
<th>P → Q</th>
<th>P ↔ Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
</tbody>
</table>

- One row for each possible assignment of True/False to propositional variables

- **Important:** Above P and Q can be any sentence, including complex sentences

Semantics — Complex Sentences

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>¬R</th>
<th>R ∧ S</th>
<th>¬R ∨ S</th>
<th>(R ∧ S) → (¬R ∨ S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>

Terminology

- A sentence is **valid** if it is True under all possible assignments of True/False to its propositional variables (e.g. \( P \lor \neg P \))
- Valid sentences are also referred to as **tautologies**
- A sentence is **satisfiable** if and only if there is some assignment of True/False to its propositional variables for which the sentence is True
- A sentence is **unsatisfiable** if and only if it is not satisfiable (e.g. \( P \land \neg P \))

Material Implication

- The only time \( P \rightarrow Q \) evaluates to False is when \( P \) is True and \( Q \) is False (i.e. \( P \rightarrow Q \) is equivalent to \( \neg P \lor Q \))
- This is known as **material implication**
- English usage often suggests a causal connection between antecedent \( (P) \) and consequent \( (Q) \) — this is not reflected in the truth table
- So \( (P \land \neg P) \rightarrow \text{anything} \) is a tautology!
Entailment

- $S \models P$ — whenever all the formulae in the set $S$ are True, $P$ is True
- This is a semantic notion; it concerns the notion of truth
- We can determine whether $S \models P$ by constructing a truth table for $S$ and $P$ (gives a syntactic counterpart to entailment) — $S \models P$ if, in any row of the truth table where all the formulae in $S$ are True, $P$ is also True
- A tautology is just the special case when $S$ is the empty set

Entailment — Tautologies

<table>
<thead>
<tr>
<th>$R$</th>
<th>$S$</th>
<th>$\neg R$</th>
<th>$R \land S$</th>
<th>$\neg R \lor S$</th>
<th>$(R \land S) \rightarrow (\neg R \lor S)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>

Therefore $\models (R \land S) \rightarrow (\neg R \lor S)$

Conclusion

- Due to the ambiguity in natural languages there is a need to specify knowledge through the use of formal languages
- Not only will these formal languages give us a way to remove ambiguity but they will also help to provide methods for automating inference
- Propositional logic is a first move in this direction
- In the next lecture we look at automating inference using the resolution rule on which Prolog is based
- We shall also investigate first-order predicate logic
- Keep in mind that there are a large number of logics and knowledge representation schemes that we do not study

<table>
<thead>
<tr>
<th>$P$</th>
<th>$Q$</th>
<th>$P \rightarrow Q$</th>
<th>$Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>

Therefore $\{P, P \rightarrow Q\} \models Q$
- since in the only row where both $P$ and $P \rightarrow Q$ are True (row 1), $Q$ is also True (here $S$ on the previous slide is the set $\{P, P \rightarrow Q\}$)

Note: The column for $P \rightarrow Q$ is calculated from that for $P$ and $Q$ using the truth table definition, and $Q$ is used again to check the entailment.