Propositional logic

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

Game search
Game search problem

- **Game problem formulation:**
  - **Initial state:** initial board position + info whose move it is
  - **Operators:** legal moves a player can make
  - **Goal (terminal test):** determines when the game is over
  - **Utility (payoff) function:** measures the outcome of the game and its desirability

- **Search objective:**
  - find the sequence of player’s decisions (moves) maximizing its utility (payoff)
  - Consider the opponent’s moves and their utility

---

### Game problem formulation (Tic-tac-toe)

#### Objectives:
- **Player 1:** maximize outcome
- **Player 2:** minimize outcome

#### Operators

#### Initial state

#### Terminal (goal) states

#### Utility:
-1 0 1
Minimax algorithm

How to deal with the contingency problem?

- Assuming that the opponent is rational and always optimizes its behavior (opposite to us) we consider the best opponent’s response
- Then the **minimax algorithm** determines the best move

![Minimax Algorithm Diagram]

---

Minimax algorithm. Example

![Minimax Algorithm Example Diagram]
Complexity of the minimax algorithm

- We need to explore the complete game tree before making the decision
- Impossible for large games
  - Chess: 35 operators, game can have 50 or more moves

![Game Tree Diagram]

Complexity:
\[ O(b^m) \]

Solution to the complexity problem

Two solutions:

1. **Dynamic pruning of redundant branches** of the search tree
   - identify provably suboptimal branch of the search tree even before it is fully explored
   - Cutoff the suboptimal branch
   **Procedure:** Alpha-Beta pruning

2. **Early cutoff of the search tree**
   - uses imperfect minimax value estimate of non-terminal states.
Alpha beta pruning. Example

MAX

MIN

MAX

4 3 6 2 2 1 9 5 3 1 5 4 7 5

4 = 6 ≥ 4 = 5
3 = 5 ≥ 2 = 5
6 ≥ 6 = 2
2 = 5
2 = 2
1 = 5
9 ≤ 2
5 = 5
3 = 5
1 = 5
5 ≥ 7

nodes that were never explored !!!
Search tree cutoffs

- Idea:
  - Cutoff the search tree before the terminal state is reached
  - Use imperfect estimate of the minimax value at leaves
    - Heuristic evaluation function
  - Select one move - repeat before every move

Heuristic evaluation functions

- Gives a **heuristic estimate** of the value for a sub-tree
- **Example of a heuristic functions**:
  - Material advantage in chess, checkers
    - Gives a value to every piece on the board, its position and combines them
  - More general **feature-based evaluation function**
    - Typically a linear evaluation function:
      \[ f(s) = f_1(s)w_1 + f_2(s)w_2 + \ldots + f_k(s)w_k \]
      \[ f_i(s) \] - a feature of a state \( s \)
      \[ w_i \] - feature weight
Evaluation functions

- **Even better heuristic estimate** of the value for a sub-tree
- Restricted set of moves to be considered under the cutoff level
  - reduces branching and improves the evaluation function
  - Example: consider only the capture moves in chess

![Heuristic estimates diagram]

Knowledge representation:

Propositional logic
Knowledge-based agent

- **Knowledge base (KB):**
  - A set of sentences that describe facts about the world in some formal (representational) language
  - Domain specific

- **Inference engine:**
  - A set of procedures that work upon the representational language and can infer new facts or answer KB queries
  - Domain independent

---

Example: MYCIN

- MYCIN: an expert system for diagnosis of bacterial infections

  - **Knowledge base** represents
    - Facts about a specific patient case
    - Rules describing relations between entities in the bacterial infection domain

  ```
  If 1. The stain of the organism is gram-positive, and  
     2. The morphology of the organism is coccus, and  
     3. The growth conformation of the organism is chains  
  Then the identity of the organism is streptococcus
  ```

  - **Inference engine:**
    - manipulates the facts and known relations to answer diagnostic queries (consistent with findings and rules)
Knowledge representation

- The objective of knowledge representation is to express the knowledge about the world in a computer-tractable form.

- Key aspects of knowledge representation languages:
  - **Syntax:** describes how sentences are formed in the language.
  - **Semantics:** describes the meaning of sentences, what is it the sentence refers to in the real world.
  - **Computational aspect:** describes how sentences and objects are manipulated in concordance with semantical conventions.

Many KB systems rely on some variant of logic.

Logic

A formal language for expressing knowledge and ways of reasoning.

Logic is defined by:

- **A set of sentences**
  - A sentence is constructed from a set of primitives according to syntax rules.

- **A set of interpretations**
  - An interpretation gives a semantic to primitives. It associates primitives with values.

- **The valuation (meaning) function** \( V \)
  - Assigns a value (typically the truth value) to a given sentence under some interpretation:

\[
V : \text{sentence} \times \text{interpretation} \rightarrow \{ \text{True, False} \}
\]
Example of logic

Language of numerical constraints:

• A sentence:
  \[ x + 3 \leq z \]
  \[ x, z \] - variable symbols (primitives in the language)

• An interpretation:
  \[ I: \ x = 5, z = 2 \]
  Variables mapped to specific real numbers

• Valuation (meaning) function \( V \):
  \[ V ( x + 3 \leq z, I ) \] is \textit{False} for \[ I: \ x = 5, z = 2 \]
  is \textit{True} for \[ I: \ x = 5, z = 10 \]

Types of logic

• Different types of logics possible:
  – Propositional logic
  – First-order logic
  – Temporal logic
  – Numerical constraints logic
  – Map-coloring logic

In the following:

• \textbf{Propositional logic}.
  – Formal language for making logical inferences
  – Foundations of \textit{propositional logic}: \textbf{George Boole} (1854)
Propositional logic. Syntax

• Propositional logic \( P \):  
  – defines a language for symbolic reasoning

First step: define Syntax+interpretation+semantics of \( P \)

Syntax:
• Symbols (alphabet) in \( P \):
  – Constants: True, False
  – A set of propositional variables (propositional symbols):
    Examples: \( P, Q, R, \ldots \) or statements like:
    
    - Light in the room is on,
    - It rains outside, etc.
  – A set of connectives:
    \( \neg, \land, \lor, \Rightarrow, \Leftrightarrow \)

Sentences in the propositional logic:

• Atomic sentences:
  – Constructed from constants and propositional symbols
  – True, False are (atomic) sentences
  – \( P, Q \) or Light in the room is on, It rains outside are
    (atomic) sentences

• Composite sentences:
  – Constructed from valid sentences via connectives
  – If \( A, B \) are sentences then
    \( \neg A, \quad (A \land B), \quad (A \lor B), \quad (A \Rightarrow B), \quad (A \Leftrightarrow B) \)
    or
    \( (A \lor B) \land (A \lor \neg B) \)
    are sentences
Propositional logic. Semantics.

The semantic gives the meaning to sentences.

In the propositional logic the semantics is defined by:

1. Interpretation of propositional symbols and constants
   - Semantics of atomic sentences
2. Through the meaning of connectives
   - Meaning (semantics) of composite sentences

Semantic: propositional symbols

A propositional symbol (an atomic sentence) can stand for an arbitrary fact (statement) about the world

Examples: “Light in the room is on”, “It rains outside”, etc.

- An interpretation maps symbols to one of the two values: True (T), or False (F), depending on whether the symbol is satisfied in the world
  - I: Light in the room is on -> True, It rains outside -> False
  - I’: Light in the room is on -> False, It rains outside -> False
- The meaning (value) of the propositional symbol for a specific interpretation is given by its interpretation
  \[ V(\text{Light in the room is on}, I) = \text{True} \]
  \[ V(\text{Light in the room is on}, I') = \text{False} \]
Semantics: constants

- **The meaning (truth) of constants:**
  - True and False constants are always (under any interpretation) assigned the corresponding *True, False* value.

\[
\begin{align*}
V(\text{True}, I) &= \text{True} \\
V(\text{False}, I) &= \text{False}
\end{align*}
\]

For any interpretation \( I \)

Semantics: composite sentences.

- **The meaning (truth value) of complex propositional sentences.**
  - Determined using the following rules for combining sentences:

<table>
<thead>
<tr>
<th></th>
<th>( P )</th>
<th>( Q )</th>
<th>( \neg P )</th>
<th>( P \land Q )</th>
<th>( P \lor Q )</th>
<th>( P \Rightarrow Q )</th>
<th>( P \Leftrightarrow 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>
<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>
<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>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>
Contradiction and Tautology

Some composite sentences may always (under any interpretation) evaluate to a single truth value:

- **Contradiction** (always *False*)
  \[ P \land \neg P \]

- **Tautology** (always *True*)
  \[ P \lor \neg P \]

\[
\begin{align*}
\neg (P \lor Q) & \iff (\neg P \land \neg Q) \\
\neg (P \land Q) & \iff (\neg P \lor \neg Q)
\end{align*}
\]

DeMorgan’s Laws

---

Model, validity and satisfiability

- A **model (in logic)**: An interpretation is a model for a set of sentences if it assigns true to each sentence in the set.
- A sentence is **satisfiable** if it has a model;
  - There is at least one interpretation under which the sentence can evaluate to True.
- A sentence is **valid** if it is *True* in all interpretations
  - i.e., if its negation is **not satisfiable** (leads to contradiction)

<table>
<thead>
<tr>
<th></th>
<th>( P )</th>
<th>( Q )</th>
<th>( P \lor Q )</th>
<th>( (P \lor Q) \land \neg Q )</th>
<th>( ((P \lor Q) \land \neg Q) \Rightarrow P )</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td></td>
</tr>
</tbody>
</table>

---
Model, validity and satisfiability

- A **model (in logic)**: An interpretation is a model for a set of sentences if it assigns true to each sentence in the set.
- A sentence is **satisfiable** if it has a model;
  - There is at least one interpretation under which the sentence can evaluate to True.
- A sentence is **valid** if it is *True* in all interpretations
  - i.e., if its negation is not satisfiable (leads to contradiction)

<table>
<thead>
<tr>
<th>Satisfiable sentence</th>
<th>Valid sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P\land\neg(Q\lor\neg R))</td>
<td>((P\land\neg Q)\lor\neg (P\land\neg Q)\Rightarrow P)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(P) (Q)</th>
<th>(P\lor Q)</th>
<th>((P\lor Q)\land\neg Q)</th>
<th>((P\lor Q)\land\neg Q\Rightarrow P)</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
</tbody>
</table>

Entailment

- The KB agent with reasoning capabilities should be able to generate new sentences (conclusions) that are true given the existing true sentences (stored in its knowledge base)
- **Entailment** reflects the relation of one fact in the world following from the others

- Entailment \(KB\models \alpha\)
- Knowledge base KB entails sentence \(\alpha\) if and only if \(\alpha\) is true in all worlds where KB is true
Sound and complete inference.

- **Inference** is a process by which conclusions are reached.

- Assume an inference procedure \( I \)
  
  \[ KB \models_i \alpha \]
  
  a sentence \( \alpha \) can be derived from KB by \( i \)

- **Soundness**: An inference procedure is sound
  
  If \( KB \models_i \alpha \) then it is true that \( KB \models \alpha \)

- **Completeness**: An inference procedure is complete
  
  If \( KB \models \alpha \) then it is true that \( KB \models_i \alpha \)

Logical inference

**Logical inference problem:**

- Given a knowledge base KB (a set of sentences) and a sentence \( \alpha \), does a KB semantically entail \( \alpha \)?
  
  \[ KB \models \alpha \ ? \]

  In other words: In all interpretations in which sentences in the KB are true, is also \( \alpha \) true?

- Sentence \( \alpha \) is also called a theorem

- **Logical inference problem for the propositional logic is decidable.**
  
  - There is a procedure that can answer the logical inference problem in a finite number of steps