Logical Agents

(AIMA - Chapter 7)
Outline

1. Wumpus world
2. Logic-based agents
3. Propositional logic
   • Syntax, semantics, inference, validity, equivalence and satisfiability

Next Time:
• **Automated Propositional Theorem Provers**
  • Resolution
  • A Practical Method: Walksat
1. Automating “Hunt the Wumpus”: A different kind of problem
The Wumpus World

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Stench</td>
<td>Breeze</td>
<td>PIT</td>
</tr>
<tr>
<td>2</td>
<td>Stench</td>
<td>Breeze</td>
<td>PIT</td>
</tr>
<tr>
<td>1</td>
<td>START</td>
<td>Breeze</td>
<td>PIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### PEAS description

- **Performance measure**
  - gold: +1000, death: -1000
  - -1 per step

- **Environment**
  - Squares adjacent to wumpus are *smelly*
  - Squares adjacent to pit are *breezy*
  - *Glitter* iff gold is in the same square
  - Gold is picked up by reflex, can’t be dropped
  - You *bump* if you walk into a wall

- **Actuators:** Move <dir>
- **Sensors:** Stench, Breeze, Glitter, Bump
Full PEAS description

- **Performance measure**
  - gold: +1000, death: -1000
  - -1 per step, -10 for using the arrow

- **Environment**
  - Squares adjacent to wumpus are *smelly*
  - Squares adjacent to pit are *breezy*
  - *Glitter* iff gold is in the same square
  - Shooting kills wumpus if you are facing it. It *screams*
  - Shooting uses up the only arrow
  - Grabbing picks up gold if in same square
  - Releasing drops the gold in same square
  - You *bump* if you walk into a wall

- **Actuators**: Face <direction>, Move, Grab, Release, Shoot
- **Sensors**: Stench, Breeze, Glitter, Bump, Scream
## Wumpus world characterization

<table>
<thead>
<tr>
<th>Characterization</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Deterministic</strong></td>
<td>Yes – outcomes exactly specified</td>
</tr>
<tr>
<td><strong>Static</strong></td>
<td>Yes – Wumpus and Pits do not move</td>
</tr>
<tr>
<td><strong>Discrete</strong></td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Single-agent</strong></td>
<td>Yes – Wumpus is essentially a natural feature</td>
</tr>
<tr>
<td><strong>Fully Observable</strong></td>
<td><strong>No</strong> – only <em>local</em> perception</td>
</tr>
<tr>
<td><strong>Episodic</strong></td>
<td><strong>No</strong> – What was observed before <em>(breezes, pits, etc)</em> is very useful.</td>
</tr>
</tbody>
</table>

*A New Kind of Problem!*
Exploring the Wumpus World

1. The KB initially contains the rules of the environment.

2. **Location:** [1,1]  
   **Percept:** \([-\text{Stench}, -\text{Breeze}, -\text{Glitter}, -\text{Bump}]\)  
   **Action:** Move to safe cell e.g. 2,1

3. **Location:** [2,1]  
   **Percept:** \([-\text{Stench}, \text{Breeze}, -\text{Glitter}, -\text{Bump}]\)  
   **INFER:** Breeze indicates that there is a pit in [2,2] or [3,1]  
   **Action:** Return to [1,1] to try next safe cell
Exploring the Wumpus World

<table>
<thead>
<tr>
<th></th>
<th>1.4</th>
<th>2.4</th>
<th>3.4</th>
<th>4.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3</td>
<td>W!</td>
<td>2.3</td>
<td>3.3</td>
<td>4.3</td>
</tr>
<tr>
<td>1.2</td>
<td>A</td>
<td>2.2</td>
<td>3.2</td>
<td>4.2</td>
</tr>
<tr>
<td>1.1</td>
<td>v</td>
<td>2.1</td>
<td>3.1 R!</td>
<td>4.1</td>
</tr>
</tbody>
</table>

A = Agent  
B = Breeze  
G = Glitter, Gold  
OK = Safe square  
P = Pit  
S = Stench  
V = Visited  
W = Wumpus

4. **Location**: [1,2] (after going through [1,1])  
**Percept**: [Stench, ¬Breeze, ¬Glitter, ¬Bump]  
**INFER**: Wumpus is in [1,1] or [2,2] or [1,3]  
**INFER** ... stench not detected in [2,1], thus not in [2,2]  
**REMEMBER**....Wumpus not in [1,1]  
**THUS** ... Wumpus is in [1,3]  
**THEREFORE** [2,2] is safe because of lack of breeze in [1,2]  
**Action**: Move to [2,2]  
**REMEMBER**: Pit in [2,2] or [3,1]  
**THEREFORE**: Pit in [3,1]!
2. Logic-based Agents
Logical Agents

- Most useful in **non-episodic, partially observable** environments

- **Logic (Knowledge-Based) agents** combine
  1. A *knowledge base (KB)*: a list of facts that are known to the agent.
  2. Current percepts to *infer hidden* aspects of the current state using *Rules of inference*

- **Logic** provides a good formal language for both
  - Facts encoded as *axioms*
  - Rules of inference
The Knowledge Base

A set of *sentences*
- in a formal *knowledge representation language*
- that *encodes assertions* about the world

---

**Declarative** approach to building an agent:
- *Tell* it what it needs to know – add fact to KB.
- *Ask* it what to do – compute using *inference rules* also in KB
Generic KB-Based Agent Pseudocode

```plaintext
function KB-AGENT(percept) returns an action
    static: KB, a knowledge base
             t, a counter, initially 0, indicating time
    TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
    action ← ASK(KB, MAKE-ACTION-QUERY(t))
    TELL(KB, MAKE-ACTION-SENTENCE(action, t))
    t ← t + 1
    return action
```

- **Agent must be able to:**
  1. Represent states and actions,
  2. Incorporate new percepts
  3. Update internal representation of the world
  4. *Deduce hidden properties of the world*
  5. Deduce appropriate actions – requires some *logic of action*
3. Propositional Logic

- Propositional logic is the simplest logic – illustrates basic ideas
- Inference in propositional logic is also *tractable* with reasonable constraints – therefore very useful
LOGIC: What is a logic?

- A formal language with associated
  - **Syntax** – what expressions are *legal* (well-formed)
  - **Semantics** – what legal expressions *mean*

  — in logic, meaning is defined by the *truth* of each sentence *with respect to a set of possible worlds*
    - **Possible world (simplified):** an assignment of values to all logical variables

- e.g. the language of arithmetic

  - **Syntax:** $X + 2 \geq y$ is a legal sentence, $x^2 + y$ is not a legal sentence
  - **Semantics:** $X + 2 \geq y$ is true in a world where $x = 7$ and $y = 1$
  - **Semantics:** $X + 2 \geq y$ is false in a world where $x = 0$ and $y = 6$
Propositional logic: *Syntax*

Recursively defined:

**Base case:**

- The atomic proposition symbols $P_1, P_2$ etc are sentences

**Recursion:**

- If $S$ is a sentence, $\neg S$ is a sentence  \((\text{negation})\)
- If $S_1$ and $S_2$ are sentences, $S_1 \land S_2$ is a sentence \((\text{conjunction})\)
- If $S_1$ and $S_2$ are sentences, $S_1 \lor S_2$ is a sentence \((\text{disjunction})\)
- If $S_1$ and $S_2$ are sentences, $S_1 \Rightarrow S_2$ is a sentence \((\text{implication})\)
- If $S_1$ and $S_2$ are sentences, $S_1 \Leftrightarrow S_2$ is a sentence \((\text{biconditional})\)
Propositional logic: **Semantics**

Each model/world specifies true or false for each proposition symbol

E.g. \( P_{1,2} \), \( P_{2,2} \), \( P_{3,1} \)

<table>
<thead>
<tr>
<th>( P_{1,2} )</th>
<th>( P_{2,2} )</th>
<th>( P_{3,1} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>

With these three symbols, \( 2^3 \) possible models (worlds), can be enumerated automatically.

Rules for evaluating truth with respect to a model \( m \):

- **(not)** \( \neg S \) is true iff \( S \) is false
- **(and)** \( S_1 \land S_2 \) is true iff \( S_1 \) is true and \( S_2 \) is true
- **(or)** \( S_1 \lor S_2 \) is true iff \( S_1 \) is true or \( S_2 \) is true
- **(if..then)** \( S_1 \Rightarrow S_2 \) is true iff \( S_1 \) is false or \( S_2 \) is true
  - i.e., is false iff \( S_1 \) is true and \( S_2 \) is false
- **(if and only if)** \( S_1 \Leftrightarrow S_2 \) is true iff \( S_1 \Rightarrow S_2 \) is true and \( S_2 \Rightarrow S_1 \) is true

Simple recursive process evaluates an arbitrary sentence, e.g.,

\[
\neg P_{1,2} \land (P_{2,2} \lor P_{3,1}) = true \land (true \lor false) = true \land true = true
\]
Wumpus world propositional sentences

Let $P_{i,j}$ be true if there is a pit in $[i, j]$.
Let $B_{i,j}$ be true if there is a breeze in $[i, j]$.

```
start:  ¬P_{1,1}
        ¬B_{1,1}
        B_{2,1}
```

"Pits cause breezes in adjacent squares"

\[
B_{1,1} \iff (P_{1,2} \lor P_{2,1})
\]

(and many more very similar)

\[
B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1})
\]

(and many more very similar)
Model Theory

- Logicians often think in terms of *models*
  - Formally structured “worlds” with respect to which truth can be evaluated
    
    [“world” means (loosely) *possible* state of this world where some things that are true in this world may be false, and vice versa.]

- We say $m$ *is a model of* a sentence $\alpha$ if $\alpha$ is true in $m$

- $M(\alpha)$ is the set of all models of $\alpha$

(Usually, logicians are interested in models of mathematical structures. AI-types are interested in models of common-sense objects and activities.)
A key semantic relation: **Entailment**

- **Entailment** means that the *truth* of one sentence **follows** from the *truth of* another:
  \[ \text{KB} \models \alpha \]
  Knowledge base **KB entails** sentence **\( \alpha \)** if and only if
  \( \alpha \) is true in **all worlds** where **KB** is true
  - e.g., the KB containing *the Giants won and the Reds lost* entails *The Giants won*
  - e.g., the KB containing \( x+y = 4 \) entails \( 4 = x+y \)

- **Entailment is a relationship between sentences based on semantics**
Models II

- **Review:**
  - \( m \) is a model of a sentence \( \alpha \) if \( \alpha \) is true in \( m \)
  - \( M(\alpha) \) is the set of all models of \( \alpha \)

- **Entailment:**
  - \( KB \models \alpha \) iff \( M(KB) \subseteq M(\alpha) \)

- **Example:**

  \[ KB = \text{Giants won and Reds lost} \]

  \[ \alpha = \text{Giants won} \]
Entailment in the wumpus world

- Consider possible models for $KB$ assuming only pits and a reduced Wumpus world with only 5 squares and pits

- Situation after
  A. detecting nothing in [1,1],
  B. moving right, breeze in [2,1]:

```
? | ?
---|
A | B | ?
```
Wumpus models I

All possible models (exactly 8) in this reduced Wumpus world.
• In Red: all possible wumpus-worlds consistent with the observations on slide 22 and the “physics” of the Wumpus world.
Deciding what to do by *model checking* I

$\alpha_1 = "[1,2] is safe",\ KB \models \alpha_1$, proved by *model checking*
Deciding what to do by *model checking II*

\[ \alpha_2 = \text{"[2,2] is safe"}, \ KB \not\models \alpha_2 \]
Truth tables for connectives

- Truth tables enumerate all possible propositional models.
- Thus, truth tables are a simple form of model checking.

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$P$</td>
<td>$Q$</td>
<td>$\neg P$</td>
<td>$P \land Q$</td>
<td>$P \lor Q$</td>
<td>$P \Rightarrow Q$</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>true</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>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

**OR:** $P$ or $Q$ is true or both are true.

**XOR:** $P$ or $Q$ is true but not both.

**Implication is always true when the premises are False!**
Inference Procedures

- \( KB \models_i \alpha \): sentence \( \alpha \) can be derived from \( KB \) by procedure \( i \)

- **Soundness**: \( i \) is sound if whenever \( KB \models_i \alpha \), it is also true that \( KB \models \alpha \)
  - No false inferences
  - (but all true statements might not be derived)

- **Completeness**: \( i \) is complete if whenever \( KB \models \alpha \), it is also true that \( KB \models_i \alpha \)
  - All true sentences can be derived,
  - (but some false statements might be derived)

- **Desirable**: sound and complete
Inference by enumeration

- **Enumeration of all models (truth tables)** is sound and complete.

- For $n$ propositional symbols, time complexity is $O(2^n)$

- We need a smarter way to do inference!

- One approach: *infer* new logical sentences from the data-base and see if they match a query.
Logical equivalence

- To manipulate logical sentences we need some rewrite rules.
- Two sentences are **logically equivalent** iff they are true in same models: \( \alpha \equiv \beta \) iff \( \alpha \models \beta \) and \( \beta \models \alpha \)

\[
\begin{align*}
(\alpha \land \beta) & \equiv (\beta \land \alpha) \quad \text{commutativity of } \land \\
(\alpha \lor \beta) & \equiv (\beta \lor \alpha) \quad \text{commutativity of } \lor \\
((\alpha \land \beta) \land \gamma) & \equiv (\alpha \land (\beta \land \gamma)) \quad \text{associativity of } \land \\
((\alpha \lor \beta) \lor \gamma) & \equiv (\alpha \lor (\beta \lor \gamma)) \quad \text{associativity of } \lor \\
\neg(\neg \alpha) & \equiv \alpha \quad \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) & \equiv (\neg \beta \Rightarrow \neg \alpha) \quad \text{contraposition} \\
(\alpha \Rightarrow \beta) & \equiv (\neg \alpha \lor \beta) \quad \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) & \equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination} \\
(\neg(\alpha \land \beta) & \equiv (\neg \alpha \lor \neg \beta) \quad \text{de Morgan} \\
(\neg(\alpha \lor \beta) & \equiv (\neg \alpha \land \neg \beta) \quad \text{de Morgan} \\
(\alpha \land (\beta \lor \gamma)) & \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)) \quad \text{distributivity of } \land \text{ over } \lor \\
(\alpha \lor (\beta \land \gamma)) & \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) \quad \text{distributivity of } \lor \text{ over } \land
\end{align*}
\]
Validity and satisfiability

A sentence is *valid* if it is true in all models,

\[ \text{e.g. } \text{True, } A \lor \neg A, \ A \Rightarrow A, \ (A \land (A \Rightarrow B)) \Rightarrow B \]

*Validity* is connected to *inference* via the Deduction Theorem:

\[ KB \models \alpha \text{ if and only if } (KB \Rightarrow \alpha) \text{ is valid} \]

A sentence is *satisfiable* if it is true in some model

\[ \text{e.g. } A \lor B, \ C \]

A sentence is *unsatisfiable* if it is false in all models

\[ \text{e.g. } A \land \neg A \]

Satisfiability is connected to inference via the following:

\[ KB \models \alpha \text{ if and only if } (KB \land \neg \alpha) \text{ is unsatisfiable} \]

(there is no model for which KB=true and \( \alpha \) is false)