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

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

- **Deterministic**  Yes – outcomes exactly specified
- **Static**  Yes – Wumpus and Pits do not move
- **Discrete**  Yes
- **Single-agent**  Yes – Wumpus is essentially a natural feature

- **Fully Observable**  No – only local perception
- **Episodic**  No – What was observed before (breezes, pits, etc) is very useful.

A New Kind of Problem!

Exploring the Wumpus World

1. The KB initially contains the rules of the environment.
2. **Location:** [1,1]  
   **Percept:** [¬Stench, ¬Breeze, ¬Glitter, ¬Bump]  
   **Action:** Move to safe cell e.g. 2,1

3. **Location:** [2,1]  
   **Percept:** [¬Stench, Breeze, ¬Glitter, ¬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

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]  
   **REMEMBER:** stench not detected in [2,1], thus not in [2,2]  
   **THUS:** Wumpus not in [1,1]  
   **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

- **Inference engine**  
  - domain-independent algorithms
  
- **Knowledge base**  
  - domain-specific content

**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-Agент(Percept) returns an action
    static: KB, a knowledge base
    in: a counter, initially 0, indicating time
    tell(KB, Make-Percept-Sentence(Percept, f))
    actions ← Add(KB, Make-Action-Query(Percept))
    tell(KB, Make-Action-Sentence(action, f))
    {← = 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 ≥ y is a legal sentence, x^2 + y is not a legal sentence
    - Semantics: x + 2 ≥ y is true in a world where x = 7 and y = 1
    - Semantics: x + 2 ≥ 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, ¬S is a sentence (negation)
  - If S_1 and S_2 are sentences, S_1 ∧ S_2 is a sentence (conjunction)
  - If S_1 and S_2 are sentences, S_1 ∨ S_2 is a sentence (disjunction)
  - If S_1 and S_2 are sentences, S_1 → S_2 is a sentence (implication)
  - If S_1 and S_2 are sentences, S_1 ↔ S_2 is a sentence (biconditional)

Propositional logic: Semantics

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

- E.g. P_1, false, true, false
- With these three symbols, 2^3 = 8 possible models (worlds), can be enumerated automatically.

Rules for evaluating truth with respect to a model m:

- ¬S is true iff S is false
- S_1 ∧ S_2 is true iff S_1 is true and S_2 is true
- S_1 ∨ S_2 is true iff S_1 is true or S_2 is true
- (if then) S_1 → S_2 is true iff S_1 is false or S_2 is true
- (if and only if) S_1 ↔ S_2 is true iff S_1 implies S_2 and S_2 implies S_1

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

- $$\neg (P_1 \lor (P_2 \land P_3)) = \text{true} \land (\text{false} \lor \text{true}) = \text{true} \land \text{true} = \text{true}$$

Wumpus world propositional sentences

Let P_i be true if there is a pit in [i, j].
Let B_i be true if there is a breeze in [i, j].

```
start: ¬P_1, ¬B_3

“Pits cause breezes in adjacent squares”

B_1 ↔ (P_1 ∨ P_2)
B_3 ↔ (P_1 ∨ P_3)
```

start: ¬P_1, ¬B_3
**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 α if α is true in m**

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

(Usually, logicians are interested in models of mathematical structures. AI-types are interested in models of commonsense objects and activities.)

---

**A key semantic relation: Entailment**

- **Entailment** means that the **truth** of one sentence follows from the truth of another:

$$KB \models \alpha$$

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 = $ Giants won and Reds lost
    - $\alpha = $ 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].

---

**Wumpus models I**

All possible models (exactly 8) in this reduced Wumpus world.

---

**Wumpus models II**

- 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

\[ \alpha_1 = "[1,2] is safe", \ KB \models \alpha_1, \text{ proved by model checking} \]

Deciding what to do by model checking II

\[ \alpha_2 = "[2,2] is safe", \ KB \not\models \alpha_2 \]

Truth tables for connectives

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

<table>
<thead>
<tr>
<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>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

OR: \(P \lor Q\) is true or both are true.
XOR: \(P \oplus Q\) is true but not both.
Implication is always true when the premises are False!

Inference Procedures

- \(KB \models \alpha\): sentence \(\alpha\) can be derived from \(KB\) by procedure \(i\).
- **Soundness**: \(i\) is sound if whenever \(KB \models \alpha\), 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 \(KB \models \alpha\).
  - 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 database 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: \(\alpha \equiv \beta \iff \alpha \Leftrightarrow \beta\).

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

You need to know these!
Validity and satisfiability

A sentence is **valid** if it is true in all models.
- e.g. `True`, `A ⊨ A`, `A ⊨ (A ∧ (A → B)) → B`

Validity is connected to inference via the Deduction Theorem:
- `KB ⊨ α` if and only if `(KB → α)` is valid

A sentence is **satisfiable** if it is true in some model
- e.g. `A ∨ B`, `C`

A sentence is **unsatisfiable** if it is false in all models
- e.g. `A ∧ ¬A`

Satisfiability is connected to inference via the following:
- `KB ⊨ α` if and only if `(KB ∧ ¬α)` is unsatisfiable
  (there is no model for which KB=true and α is false)