7. Logical Agents

Russell & Norvig, Chapter 7.
Outline

- Knowledge-based agents
- Wumpus world
- Logic in general—models and entailment
- Propositional (Boolean) logic
- Equivalence, validity, satisfiability
- Inference rules and theorem proving
  - forward chaining
  - backward chaining
  - resolution
Models and Planning

World Model
- transition table
- dynamical system
- parametric model
- knowledge base

Planning
- state-based search
- simulation
- goals / utility
- logical inference

Perception

Action

Environment
Knowledge bases

Knowledge base = set of sentences in a formal language

Declarative approach to building an agent (or other system):
- tell it what it needs to know

Then it can Ask itself what to do
- answers should follow from the KB
A simple knowledge-based agent

The agent must be able to:

- represent states, actions, etc.
- incorporate new percepts
- update internal representations of the world
- deduce hidden properties of the world
- determine appropriate actions
Wumpus World PEAS description

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
  - uses up the only arrow
- Grabbing
  - picks up gold if in same square
- Releasing
  - drops the gold in same square
Wumpus World PEAS description

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

- **Actuators**
  - Left turn, Right turn, Forward, Grab, Release, Shoot

- **Sensors**
  - Breeze, Glitter, Smell
Exploring a wumpus world
Exploring a wumpus world
Other tight spots

Breeze in (1,2) and (2,1)  
⇒ no safe actions

Assuming pits uniformly distributed,  
pit more likely in (2,2) than in (3,1)  
Exercise: How much more likely?

Smell in (1,1) ⇒ cannot move  
Can use a strategy of coercion:
  ■ shoot straight ahead  
  ■ wumpus was there ⇒ dead ⇒ safe  
  ■ wumpus wasn’t there ⇒ safe
Logic in general

Logics are formal languages for representing information such that conclusions can be drawn.

Syntax defines the sentences in the language.

Semantics define the “meaning” of sentences; i.e. define truth of a sentence in a world.

e.g. the language of arithmetic

\( x + 2 \geq y \) is a sentence; \( x2 + y > \) is not a sentence.

\( x + 2 \geq y \) is true iff the number \( x + 2 \) is no less than the number \( y \).

\( x + 2 \geq y \) is true in a world where \( x = 7, y = 1 \).

\( x + 2 \geq y \) is false in a world where \( x = 0, y = 6 \).
Entailment

Entailment means that one thing follows from 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 won” entails “Either the Giants won or the Reds won”.

E.g. \( x + y = 4 \) entails \( 4 = x + y \)

Entailment is a relationship between sentences (i.e. syntax) that is based on semantics.
Models

Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated.

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 \( \alpha \).

Then \( KB \models \alpha \) if and only if \( M(KB) \subseteq M(\alpha) \).
Entailment in the wumpus world

Situation after detecting nothing in [1,1], moving right, breeze in [2,1]

Consider possible models for ? assuming only pits

3 Boolean choices $\Rightarrow$ 8 possible models
Wumpus models
Wumpus models

$KB = \text{wumpus-world rules} + \text{observations}$
Wumpus models

$KB = \text{wumpus-world rules + observations}$

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

\[ KB = \text{wumpus-world rules + observations} \]

\[ \alpha_2 = \text{“[2,2] is safe”, } KB \not\models \alpha_2 \]
Inference

$KB \vdash_i \alpha = \text{sentence } \alpha \text{ can be derived from } KB \text{ by procedure } i$

Consequences of $KB$ are a haystack; $\alpha$ is a needle.
Entailment = needle in haystack; inference = finding it

**Soundness**: $i$ is sound if
whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$

**Completeness**: $i$ is complete if
whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$
Propositional logic: Syntax

Propositional logic is the simplest logic—illustrates basic ideas.

The proposition symbols $P_1$, $P_2$ etc are sentences.

If $S$ is a sentence, $\neg S$ is a sentence (negation).

If $S_1$ and $S_2$ are sentences, $S_1 \land S_2$ is a sentence (conjunction).

If $S_1$ and $S_2$ are sentences, $S_1 \lor S_2$ is a sentence (disjunction).

If $S_1$ and $S_2$ are sentences, $S_1 \Rightarrow S_2$ is a sentence (implication).

If $S_1$ and $S_2$ are sentences, $S_1 \Leftrightarrow S_2$ is a sentence (biconditional).
Propositional logic: Semantics

Each model specifies true/false for each proposition symbol

E.g. $P_{1,2}$, $P_{2,2}$, $P_{3,1}$

TRUE   TRUE   FALSE

(With these symbols, 8 possible models, can be enumerated automatically.)
Propositional logic: Semantics

Rules for evaluating truth with respect to a model $m$:

- $\neg S$ is TRUE iff $S$ is FALSE
- $S_1 \land S_2$ is TRUE iff $S_1$ is TRUE and $S_2$ is TRUE
- $S_1 \lor S_2$ is TRUE iff $S_1$ is TRUE or $S_2$ is TRUE
- $S_1 \Rightarrow S_2$ 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
- $S_1 \iff S_2$ 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.

$\neg P_{1,2} \land (P_{2,2} \lor P_{3,1}) = TRUE \land (FALSE \lor TRUE) = TRUE \land TRUE = TRUE$
Truth tables for connectives

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>¬P</th>
<th>P ∧ Q</th>
<th>P ∨ Q</th>
<th>P ⇒ Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
Wumpus world 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 $[i, j]$.

\[
\neg P_{1,1} \\
\neg B_{1,1} \\
B_{2,1}
\]

“Pits cause breezes in adjacent squares”
Wumpus world 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 $[i, j]$.

\[ \neg P_{1,1} \]
\[ \neg B_{1,1} \]
\[ B_{2,1} \]

“Pits cause breezes in adjacent squares”

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

“A square is breezy if and only if there is an adjacent pit”
Logical equivalence

Two sentences are logically equivalent iff true in same models: $\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$.
Validity and satisfiability

A sentence is **valid** if it is true in all models,
e.g. TRUE, \( A \lor \neg A \), \( A \Rightarrow A \), \( (A \land (A \Rightarrow B)) \Rightarrow B \)

Validity is connected to inference via the **Deduction Theorem**:
\( KB \models \alpha \) if and only if \( (KB \Rightarrow \alpha) \) is valid

A sentence is **satisfiable** if it is true in some model
e.g. \( A \lor B \), \( C \)

A sentence is **unsatisfiable** if it is true in no models
e.g. \( A \land \neg A \)

Satisfiability is connected to inference via the following:
\( KB \models \alpha \) if and only if \( (KB \land \neg \alpha) \) is unsatisfiable
i.e. prove \( \alpha \) by **reductio ad absurdum**
Proof methods

Proof methods divide into (roughly) two kinds:

Application of inference rules

- Legitimate (sound) generation of new sentences from old
- **Proof** = a sequence of inference rule applications
  Can use inference rules as operators in a standard search algebra
- Typically require translation of sentences into a **normal form**

Model checking

- truth table enumeration (always exponential in $n$)
- improved backtracking, e.g. Davis–Putnam–Logemann–Loveland
- heuristic search in model space (sound but incomplete)
Forward and backward chaining

KB = conjunction of Horn clauses
Horn clause =

- proposition symbol; or

- (conjunction of symbols) ⇒ symbol

e.g. $C \land (B \Rightarrow A) \land (C \land D \Rightarrow B)$

**Modus Ponens** (for Horn Form): complete for Horn KBs

$$
\alpha_1, \ldots, \alpha_n, \quad \alpha_1 \land \cdots \land \alpha_n \Rightarrow \beta
$$

Can be used with forward chaining or backward chaining.
Forward chaining

Idea: fire any rule whose premises are satisfied in the $KB$, add its conclusion to the $KB$, until query is found.
Backward chaining

- Idea: work backwards from the query $q$:
  - check if $q$ is known already, or
  - prove by BC all premises of some rule concluding $q$

- Avoid loops: check if new subgoal is already on the goal stack

- Avoid repeated work: check if new subgoal
  - has already been proved true, or
  - has already failed
Forward vs. backward chaining

FC is **data-driven** – automatic, unconscious processing e.g. object recognition, routine decisions

May do lots of work that is irrelevant to the goal

BC is **goal-driven**, appropriate for problem-solving, e.g. Where are my keys? How do I get into a PhD program?

Complexity of BC can be **much less** than linear in size of KB
Resolution

Conjunctive Normal Form (CNF – universal)
conjunction of disjunctions of literals

\[ (A \lor \lnot B) \land (B \lor \lnot C \lor \lnot D) \]

Resolution inference rule (for CNF): complete for propositional logic

\[
\begin{align*}
\ell_1 \lor \cdots \lor \ell_k, & \quad m_1 \lor \cdots \lor m_n \\
\ell_1 \lor \cdots \lor \ell_{i-1} \lor \ell_{i+1} \lor \cdots \lor \ell_k \lor m_1 \lor \cdots \lor m_{j-1} \lor m_{j+1} \lor \cdots \lor m_n
\end{align*}
\]

where \( \ell_i \) and \( m_j \) are complementary literals. e.g.

\[
P_{1,3} \lor P_{2,2}, \quad \lnot P_{2,2}
\]

\[
P_{1,3}
\]

Resolution is sound and complete for propositional logic.
Conversion to CNF

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

1. Eliminate \( \iff \), replacing \( \alpha \iff \beta \) with \( (\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha) \).

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

2. Eliminate \( \Rightarrow \), replacing \( \alpha \Rightarrow \beta \) with \( \neg \alpha \lor \beta \).

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

3. Move \( \neg \) inwards using de Morgan’s rules and double-negation:

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

4. Apply distributivity law (\( \lor \) over \( \land \)) and flatten:

\[
(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1})
\]
Limitations of Propositional Logic

“A square is breezy if and only if there is an adjacent pit.”

This statement must be converted into a separate sentence for each square:

\[
\begin{align*}
B_{1,1} &\iff (P_{1,2} \lor P_{2,1}) \\
B_{2,1} &\iff (P_{1,1} \lor P_{2,2} \lor P_{3,1}) \\
&\vdots
\end{align*}
\]

What we really want is a way to express such a statement in one sentence for all squares, e.g.

\[
\text{Breezy}(i, j) \iff (\text{Pit}(i - 1, j) \lor \text{Pit}(i + 1, j) \lor \text{Pit}(i, j - 1) \lor \text{Pit}(i, j + 1))
\]

First-Order Logic will allow us to do this (next lecture).
Summary

Logical agents apply inference to a knowledge base to derive new information and make decisions.

Basic concepts of logic:

- **syntax**: formal structure of sentences
- **semantics**: truth of sentences wrt models
- **entailment**: necessary truth of one sentence given another
- **inference**: deriving sentences from other sentences
- **soundness**: derivations produce only entailed sentences
- **completeness**: derivations can produce all entailed sentences