Chapter 7.1–7.4

Outline

- Knowledge-based agents
- Wumpus world
- Logic in general—models and entailment
- Propositional (Boolean) logic
- Equivalence, validity, satisfiability
- Inference by exhaustive model checking

Knowledge bases

<table>
<thead>
<tr>
<th>Inference engine</th>
<th>domain-independent algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knowledge base</td>
<td>domain-specific content</td>
</tr>
</tbody>
</table>

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

Agents can be viewed at the knowledge level

i.e., what they know, regardless of how implemented

Or at the implementation level

i.e., data structures in KB and algorithms that manipulate them

A simple knowledge-based agent

```
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
```

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
- Deduce appropriate actions

⇒ sound and complete reasoning with partial information states

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

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

Sensors Breeze, Glitter, Smell

Wumpus world characterization

Observable??
Wumpus world characterization

Observable?? No—only local perception
Deterministic?? Yes—outcomes exactly specified
Episodic?? No—sequential at the level of actions
Static?? Yes—Wumpus and Pits do not move
Discrete?? Yes
Single-agent?? Yes—Wumpus is essentially a natural feature
Exploring a wumpus world

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; \[ x + 2 > y \] 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 \( x = 0, y = 6 \).

Other tight spots

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.

Note: brains process syntax (of some sort).

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 \( M(KB) \subseteq M(\alpha) \).

E.g., \( KB = \) Giants won and Reds won

\( \alpha = \) Giants won
Entailment in the wumpus world

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

Consider possible models for \(?s\) assuming only pits

3 Boolean choices \(\Rightarrow\) 8 possible models

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

\( \alpha_1 = \text{“[1,2] is safe”, } \text{KB} \models \alpha_1, \text{proved by model checking} \)

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

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

KB ⊨_i α means “sentence α can be derived from KB by procedure i”

Consequences of KB are a haystack; α is a needle.

Entailment = needle in haystack; inference = finding it

Soundness: i is sound if whenever KB ⊨_i α, it is also true that KB ⊨ α

Completeness: i is complete if whenever KB ⊨ α, it is also true that KB ⊨_i α

Preview: we will define a logic (first-order logic) which is expressive enough to say almost anything of interest, and for which there exists a sound and complete inference procedure

That is, the procedure will answer any question whose answer follows from what is known

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, ¬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 specifies true/false for each proposition symbol

E.g., P_{1,2} P_{2,2} P_{1,1} (3 symbols ⇒ 2^3 = 8 models)

true true false

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

S_1 → 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 ↔ S_2 is true iff S_1 is true and S_2 is true

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

¬P_{1,2} ∧ (P_{2,2} ∨ P_{1,1}) = ¬true ∧ (true ∨ false)

= false ∧ true = false

Wumpus world sentences: symbols

P_{i,j} has intended meaning “there is a pit in [i,j]”

B_{i,j} has intended meaning “there is a breeze in [i,j]”

This means we connect sensors and actuators to symbols, and write axioms, in such a way that this meaning is respected

R_1: ¬P_{1,1} (given)
R_2: ¬B_{1,1} (percept)
R_3: B_{2,1} (percept)

Wumpus world sentences: axioms

An axiom is just a sentence asserted to be true about the domain (typically general rather than specific to a particular situation)

E.g., “Pits cause breezes in adjacent squares”

R_4: B_{1,1} → (P_{1,2} ∨ P_{2,1})
R_5: B_{2,1} → (P_{1,1} ∨ P_{1,2} ∨ P_{2,1})

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

Notice that we need one such sentence for every square!

For shooting, movement, etc., we need axiom sets for every time step!!!
### Truth tables for inference

<table>
<thead>
<tr>
<th>$B_1$</th>
<th>$B_2$</th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
<th>$P_4$</th>
<th>$R_1$</th>
<th>$R_2$</th>
<th>$R_3$</th>
<th>$R_4$</th>
<th>$KB$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</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>true</td>
<td>false</td>
<td>false</td>
<td>false</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>true</td>
<td>false</td>
<td>false</td>
<td>false</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>true</td>
<td>false</td>
<td>false</td>
<td>false</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>true</td>
<td>false</td>
<td>false</td>
<td>false</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>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

Enumerate rows (different assignments to symbols),
if $KB$ is true in row, check that $\alpha$ is too

### Inference by enumeration

Depth-first enumeration of all models is sound and complete

```python
function TT-ENTAILS?(KB, $\alpha$) returns true or false
inputs: KB, the knowledge base, a sentence in propositional logic
$\alpha$, the query, a sentence in propositional logic
symbols -> a list of the proposition symbols in KB and $\alpha$
return TT-CHECK-ALL(KB, $\alpha$, symbols, [])

function TT-CHECK-ALL(KB, $\alpha$, symbols, model) returns true or false
if EMPTY?(symbols) then
  if PL-TRUE?(KB, model) then return PL-TRUE?($\alpha$, model)
  else return true
else do
  $P \leftarrow$ FIRST(symbols);
  rest -> REST(symbols)
  return TT-CHECK-ALL(KB, $\alpha$, rest, EXTEND(P, true, model)) and TT-CHECK-ALL(KB, $\alpha$, rest, EXTEND(P, false, model))
```

$O(2^n)$ for $n$ symbols; problem is co-NP-complete

### Logical equivalence

Two sentences are logically equivalent iff true in same models:

$\alpha \equiv \beta$ if and only if $\models \alpha \land \beta$ and $\models \beta \land \alpha$

- $(\alpha \land \beta) \equiv (\beta \land \alpha)$: commutativity of $\land$
- $(\alpha \lor \beta) \equiv (\beta \lor \alpha)$: commutativity of $\lor$
- $(\alpha \land (\beta \land \gamma)) \equiv (\alpha \land (\beta \land \gamma))$: associativity of $\land$
- $(\alpha \lor (\beta \lor \gamma)) \equiv (\alpha \lor (\beta \lor \gamma))$: associativity of $\lor$
- $(\neg(\alpha)) \equiv \alpha$: double-negation elimination
- $(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha)$: implication elimination
- $(\alpha \Rightarrow \beta) \equiv (\neg\alpha \lor \beta)$: contraposition
- $(\alpha \Rightarrow \beta) \equiv (\neg\alpha \lor \beta)$: De Morgan
- $(\alpha \land (\beta \land \gamma)) \equiv ((\alpha \land \beta) \land (\beta \land \gamma))$: biconditional elimination
- $(\neg(\alpha \land \beta)) \equiv (\neg\alpha \land \beta)$: De Morgan
- $(\alpha \lor (\beta \lor \gamma)) \equiv ((\alpha \lor \beta) \lor (\alpha \lor \gamma))$: distributivity of $\land$ over $\lor$
- $(\alpha \lor (\beta \lor \gamma)) \equiv ((\alpha \lor \beta) \lor (\alpha \lor \gamma))$: distributivity of $\lor$ over $\land$

### 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

### 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

Wumpus world requires the ability to represent partial and negated information, reason by cases, etc.

Propositional logic does all this (but lacks expressive power)

Inference by enumerating models: sound, complete, $O(2^n)$