Logical Agents

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

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
- 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-FACT-SENTENCE( percept, t ) )
action ← Ask( KB, MAKE-ACTION-QUERY( ) )
(← 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

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
  – ... it
  – Shooting uses up the only arrow
  – Grabbing picks up gold if in same square
  – Releasing drops the gold in same square

• Sensors: Stench, Breeze, Glitter, Bump (when walk into a wall), Scream (when wumpus is killed)

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

Wumpus world characterization

• Fully 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
Exploring a wumpus world

Exploring a wumpus world

Exploring a wumpus world

Exploring a wumpus world
Exploring a wumpus world

Exploring a wumpus world

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; $x^2+y > \emptyset$ is not a sentence
  - $x+2 \geq y$ is true if 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

- **Models** 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 \( m \)
  - Think of model as “possible world”
  - \( M(\alpha) \) is the set of all models of \( \alpha \)
  - Then \( KB \models \alpha \) iff \( M(KB) \subseteq M(\alpha) \)
  - E.g. \( KB = \text{Giants won and Reds won} \) \( \alpha = \text{Giants won} \)

Entailment in the wumpus world

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

Consider possible models for \( KB \) assuming only pits

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

Wumpus models
Wumpus models

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

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

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

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

• $\text{KB} \vdash \alpha = \text{sentence } \alpha \text{ can be derived from } \text{KB} \text{ by procedure } i$
• Soundness: $i$ is sound if whenever $\text{KB} \vdash \alpha$, it is also true that $\text{KB} \models \alpha$
• Completeness: $i$ is complete if whenever $\text{KB} \models \alpha$, it is also true that $\text{KB} \vdash \alpha$
• 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 by the KB.

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

<table>
<thead>
<tr>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>

With these symbols, 8 possible models, can be enumerated automatically.

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
- $S_1 \Leftrightarrow S_2$ is true iff $S_1 \Rightarrow S_2$ is true and $S_2 \Rightarrow S_1$ is true

i.e., $S_1 \Leftrightarrow S_2$ 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.,

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

Truth tables for connectives

<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>true</td>
<td>false</td>
<td>true</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>
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"
  $B_{1,1} \iff (P_{1,2} \lor P_{2,1})$
  $B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1})$

Truth tables for inference

<table>
<thead>
<tr>
<th>$B_{1,1}$</th>
<th>$B_{2,1}$</th>
<th>$P_{1,1}$</th>
<th>$P_{1,2}$</th>
<th>$P_{2,1}$</th>
<th>$P_{2,2}$</th>
<th>$P_{3,1}$</th>
<th>$KB$</th>
<th>$\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</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>true</td>
<td>true</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>true</td>
<td>true</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>true</td>
<td>true</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>true</td>
<td>true</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>true</td>
<td>true</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>true</td>
<td>true</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>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

Inference by enumeration

• Depth-first enumeration of all models is sound and complete

Logical equivalence

• Two sentences are logically equivalent iff true in same models: $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$

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

A sentence is **valid** if it is true in all models,
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**:
\( KB \models \alpha \) 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

Proof methods

- **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 algorithm
    Typically require transformation of sentences into a normal form

- **Model checking**
  - truth table enumeration (always exponential in \( n \))
  - improved backtracking, e.g., Davis-Putnam-Logemann-Loveland (DPLL)
  - heuristic search in model space (sound but incomplete)
    e.g., min-conflicts-like hill-climbing algorithms

Resolution

Conjunctive Normal Form (CNF)

- conjunction of disjunctions of literals
- clauses
- E.g., \( (A \lor \neg B) \land (B \lor \neg C \lor \neg D) \)

Resolution inference rule (for CNF):

\[
\begin{array}{c}
\xi \lor \ldots \lor \xi_i \lor \ldots \lor \xi_n \\
\m_1 \lor \ldots \lor \m_i \lor \ldots \lor \m_l \\
\end{array}
\]

where \( \xi \) and \( m_i \) are complementary literals.
E.g., \( \xi_1 \lor \xi_2 \lor \neg \xi_3 \lor \ldots \lor \neg \xi_n \)

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
Resolution algorithm

- Proof by contradiction, i.e., show $KB \land \neg \alpha$ unsatisfiable

```python
function PL-RESOLUTION(KB, \alpha) returns true or false

- clauses ← the set of clauses in the CNF representation of $KB \land \neg \alpha$
- new ← \{
- loop do
- for each $C_i, C_j$ in clauses do
- resolvents ← PL-RESOLVE($C_i, C_j$)
- if resolvents contains the empty clause then return true
- new ← new ∪ resolvents
- if new \subseteq clauses then return false
- clauses ← clauses ∪ new

Resolution example

- $KB = (B_{1,1} \iff (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1} \land \alpha = \neg P_{1,2}$

Forward and backward chaining

- Horn Form (restricted)
  - $KB = \text{conjunction of Horn clauses}$
  - Horn clause =
    - proposition symbol; or
    - (conjunction of symbols) $\Rightarrow$ symbol
  - Disjunction of literals of which at most one is positive
  - E.g., $C \land (B \Rightarrow A) \land (C \land D \Rightarrow B)$

- Modus Ponens (for Horn Form): complete for Horn KBs
  - \[ \alpha_1, \ldots, \alpha_m, \alpha_1 \land \ldots \land \alpha_m \Rightarrow \beta \]

- Can be used with forward chaining or backward chaining.
- These algorithms are very natural and run in linear time

Forward chaining

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

  \[ P \Rightarrow Q \]
  \[ L \land M \Rightarrow P \]
  \[ B \land L \Rightarrow M \]
  \[ A \land P \Rightarrow L \]
  \[ A \land B \Rightarrow L \]
  \[ A \]
  \[ B \]
Forward chaining algorithm

Forward chaining example

- Forward chaining is sound and complete for Horn KB

```plaintext
function FC-Entails(KB, q) returns true or false
local variables: count, a table, indexed by clause, initially the number of premises
inferred, a table, indexed by symbol, each entry initially false
agenda, a list of symbols, initially the symbols known to be true
while agenda is not empty do
  p ← Pop(agenda)
  unless inferred(p) do
    count[p] ← true
    for each Horn clause c in whose premise p appears do
      if count[c] = 0 then do
        if Head(c) = q then return true
        push(Head(c), agenda)
      end if
    end for
  end if
return false
```
Forward chaining example

Forward chaining example

Forward chaining example

Forward chaining example
Forward chaining example

Proof of completeness

- FC derives every atomic sentence that is entailed by KB
  1. FC reaches a fixed point where no new atomic sentences are derived
  2. Consider the final state as a model $m$, assigning true/false to symbols
  3. Every clause in the original KB is true in $m$
     $$a_1 \land \cdots \land a_k \Rightarrow b$$
  4. Hence $m$ is a model of KB
  5. If $KB \models q$, $q$ is true in every model of KB, including $m$

Backward chaining

Idea: work backwards from the query $q$:
  - to prove $q$ by BC, 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
  1. has already been proved true, or
  2. has already failed

Backward chaining example
Backward chaining example
Backward chaining example

Backward chaining example

Backward chaining example

Backward chaining example
Backward chaining example

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

Expressiveness limitation of propositional logic
- KB contains “physics” sentences for every single square
- For every time t and every location \([x,y]\),
  \[L_{t,x} \land \text{FacingRight} \land \text{Forward} \Rightarrow L_{t+1,x}\]
- Rapid proliferation of clauses

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.
- Resolution is complete for propositional logic
  Forward, backward chaining are linear-time, complete for Horn clauses
- Propositional logic lacks expressive power