LOGICAL AGENTS

Chapter 6, AIMA2e 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

<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

Wumpus World PEAS description

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

Environment
- Squares adjacent to wumpuses are smelly
- Squares adjacent to pits 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 in same square
- Releasing drops the gold in same square

Sensors Breeze, Glitter, Smell

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

A simple knowledge-based agent

```java
function KB-Agent(precept) returns an action
    static: KB, a knowledge base
    i, a counter, initially 0, indicating time
    Tell(KB, MAKE-Percept-Sentence(precept, 0))
    actions := A = INCLUDE(KB, MAKE-Action-Query())
    Tell(KB, MAKE-Action-Sentence(action, 0))
    for i = 0 to A do
        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 characterization

Observable??
### Wumpus world characterization

**Observable??** No—only local perception  
**Deterministic??**  
**Episodic??**

---

### 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??**

---

### 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

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>OK</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>OK</td>
<td>S</td>
</tr>
<tr>
<td>A</td>
<td>OK</td>
<td>W</td>
</tr>
</tbody>
</table>

#### 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; \( x + y > \) 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 \( 4 - x + y \).

#### Other Tight Spots

- Ring in (1.2) and (2.1) ⇒ no safe actions.
- Assuming pits uniformly distributed, (2.2) has pit w/ prob 0.86, vs. 0.31.

#### 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

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

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

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

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

\( KB \vdash \alpha \) — sentence \( \alpha \) can be derived from \( KB \) by procedure \( \vdash \)

Consequences of \( KB \) are a haystack; \( \alpha \) is a needle.

Entailment — needle in haystack; inference — finding it

Soundness: \( \vdash \) is sound if
whenever \( KB \vdash \alpha \), it is also true that \( KB \vdash \alpha \)

Completeness: \( \vdash \) is complete if
whenever \( KB \vdash \alpha \), it is also true that \( KB \vdash \alpha \)

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, \( S_1 \equiv S_2 \) is a sentence (biconditional)

Propositional logic: Semantics

Each model specifies true/false for each proposition symbol

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

\[
\begin{array}{ccc}
P_{1,2} & \text{true} & \text{false} \\
P_{2,2} & \text{true} & \text{false} \\
P_{1,1} & \text{false} & \text{true}
\end{array}
\]

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

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

\[
\begin{align*}
\neg S & \text{ is false if } S \text{ is true} \\
S_1 \land S_2 & \text{ is true if } S_1 \text{ is true and } S_2 \text{ is true} \\
S_1 \lor S_2 & \text{ is true if } S_1 \text{ is true or } S_2 \text{ is true} \\
S_1 \Rightarrow S_2 & \text{ is true if } S_1 \text{ is false or } S_2 \text{ is true} \\
S_1 \equiv S_2 & \text{ is true if } S_1 \text{ is true and } S_2 \text{ is false} \\
S_1 \lor S_2 & \text{ is true if } S_1 \text{ is true and } S_2 \text{ is false}
\end{align*}
\]

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

\[
\neg P_{1,2} \land (P_{2,2} \lor P_{1,1}) \equiv \text{true} \land (\text{false} \lor \text{true}) \equiv \text{false} \quad \text{true} \equiv \text{false}
\]

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 \implies q )</th>
<th>( p \iff 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>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</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>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</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 \([i, j]\).

\[
\neg P_{i,j}, \quad \neg B_{i,j}, \quad B_{i,j}
\]

"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 \([i, j]\).

\[
\neg P_{i,j}, \quad \neg B_{i,j}, \quad B_{i,j}
\]

"Pits cause breezes in adjacent squares"

\[
B_{i,j} \iff (P_{i,j} \lor P_{j,j}) \quad B_{i,j} \iff (P_{i,j} \lor P_{j,j} \lor P_{i,j})
\]

“A square is breezy if and only if there is an adjacent pit”
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>$P_5$</th>
<th>$KB$</th>
<th>$\alpha_1$</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>true</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>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>true</td>
<td>true</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>true</td>
<td>true</td>
<td>true</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>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>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>false</td>
<td>true</td>
</tr>
</tbody>
</table>

Validity and satisfiability

A sentence is valid if it is true in all models:

- e.g., $\text{True, } A \lor \neg A, \quad A \Rightarrow A, \quad (A \land (A \Rightarrow B)) \Rightarrow B$

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., $\{A \lor B, \quad \neg C\}$

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

Inference by enumeration

Depth-first enumeration of all models is sound and complete

```
function TT-ENTAILS([KB, α]) returns true or false
  symbols ← a list of the proposition symbols in KB and α
  return TT-CHECK-ALL(KB, α, symbols, [])
```

```
function TT-CHECK-ALL(KB, α, symbols, model) returns true or false
  if EMPTY?(symbols) then
    if PL-TRUE(KB, model) then return PL-TRUE(α, model)
  else return true
  end do
  P ← First(symbols);
  rest ← Rest(symbols)
  return TT-CHECK-ALL(KB, α, rest, Extend(P, true, model))
  and
  TT-CHECK-ALL(KB, α, rest, Extend(P, false, model))
```

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

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 alg.
  - 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
  - heuristics search in model space (sound but incomplete)
  - e.g., min-conflicts-like hill-climbing algorithms

Logical equivalence

Two sentences are logically equivalent iff true in same models:

- $\alpha = \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$

  $\alpha \land \beta = \beta \land \alpha$ — commutativity of $\land$

  $\alpha \lor \beta = \beta \lor \alpha$ — commutativity of $\lor$

  $\langle\alpha \land \beta \land \gamma \rangle = \langle \beta \land (\alpha \land \gamma) \rangle$ — associativity of $\land$

  $\langle\alpha \lor \beta \lor \gamma \rangle = \langle \beta \lor (\alpha \lor \gamma) \rangle$ — associativity of $\lor$

  $\neg(\neg \alpha) = \alpha$ — double-negation elimination

  $\neg(\alpha \lor \beta) = \neg \alpha \land \neg \beta$ — contraposition

  $\neg(\alpha \land \beta) = \neg \alpha \lor \neg \beta$ — implication elimination

  $\neg(\alpha \lor \beta) = \neg \alpha \land \neg \beta$ — biconditional elimination

  $\neg(\alpha \land \beta) = \neg \alpha \lor \neg \beta$ — De Morgan

  $\neg(\alpha \lor \beta) = \neg \alpha \land \neg \beta$ — De Morgan

  $\langle\alpha \lor (\beta \land \gamma)\rangle = \langle (\alpha \lor \beta) \land (\alpha \lor \gamma)\rangle$ — distributivity of $\lor$ over $\land$

  $\langle\alpha \land (\beta \lor \gamma)\rangle = \langle (\alpha \land \beta) \lor (\alpha \land \gamma)\rangle$ — distributivity of $\land$ over $\lor$

Forward and backward chaining

**Horn Form** (restricted)

- $KB$ — 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

  $\alpha_1 \land \cdots \land \alpha_m \land \alpha_{m+1} \Rightarrow \beta$

  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

\[
\begin{align*}
P & \Rightarrow Q \\
L \land M & \Rightarrow P \\
B \land L & \Rightarrow M \\
A \land P & \Rightarrow L \\
A \land B & \Rightarrow L \\
A & \\
B &
\end{align*}
\]

**Forward chaining example**

**Forward chaining algorithm**

```python
function PL-FC-Entail?(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
        inferred[p] := true
        for each Horn clause c in whose premise p appears do
            if count[c] = 0 then do
                if HasAn[c] = q then return true
                Push(HasAn[c], agenda)
            end if
        end for
    end if
end while
return false
```

**Forward chaining example**
**Forward chaining example**

- **Proof of completeness**

  FC derives every 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 $m$
     - **Proof:** Suppose a clause $\alpha_1 \land \ldots \land \alpha_k \Rightarrow b$ is false in $m$
     - Then $\alpha_1 \land \ldots \land \alpha_k$ is true in $m$ and $b$ is false in $m$.
     - Therefore the algorithm has not reached a fixed point!
  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

FC is data-driven, cf. 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
- classes

E.g., \((A \lor \neg B) \land (B \lor \neg C \lor \neg D)\)

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

\[ \ell_1 \lor \cdots \lor \ell_n, \quad m_1 \lor \cdots \lor m_n \]

\[ \ell_1 \land \cdots \land \ell_{n-1} \lor \ell_n \land \cdots \land \ell_{n+1} \lor \cdots \lor \ell_n \cdot \ell_{n+1} \lor \cdots \lor \ell_m \]

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

\[
\begin{array}{c|c|c|c|c|c}
P & q & r & s & -s & P_{2,3} \\
\hline
T & F & F & T & T & T \\
F & T & T & F & T & F \\
F & F & F & T & T & T \\
F & F & F & F & T & F \\
\end{array}
\]

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:
      \((\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor P_{2,1}) \lor B_{1,1})\)
  4. Apply distributivity law (\(\lor\) over \(\land\)) and flatten:
      \((\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \lor (\neg P_{2,1} \lor B_{1,1})\)

Resolution example

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

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: necessity 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.

Forward, backward chaining are linear-time, complete for Horn clauses

Resolution is complete for propositional logic

Propositional logic lacks expressive power

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 \cup resolvents
        if new \cup clauses then return false
        clauses = clauses \cup new
    return false
```