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

```plaintext
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())
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
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

Partially—only local perception

Deterministic

Yes—outcomes exactly specified

Episodic

<table>
<thead>
<tr>
<th>Wumpus world characterization</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Observable</strong>? Yes—local perception</td>
</tr>
<tr>
<td><strong>Deterministic</strong>? Yes—outcomes exactly specified</td>
</tr>
<tr>
<td><strong>Episodic</strong>? No—sequential at the level of actions</td>
</tr>
<tr>
<td><strong>Static</strong>? Yes—Wumpus and Pits do not move</td>
</tr>
<tr>
<td><strong>Discrete</strong>? Yes</td>
</tr>
<tr>
<td><strong>Single-agent</strong>? Yes—Wumpus is essentially a natural feature</td>
</tr>
</tbody>
</table>
Exploring a wumpus world

Percept variables: B=breeze, S=stench, G=glitter
State variables: P=pit, W=wumpus, OK

Percept variables: B=breeze, S=stench, G=glitter
State variables: P=pit, W=wumpus, OK
Percept variables: B=breeze, S=stench, G=glitter
State variables: P=pit, W=wumpus, OK
Other tight spots

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

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

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”, “the Reds won”, (...) entails “the Giants won or the Reds won” [logical or, not exclusive or] (also “the Gaints won and the Reds did not lose,” ...)

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

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

Note: brains process syntax (of some sort)

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 \( x = 0, y = 6 \)

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 \( KB \models \alpha \) iff \( M(KB) \subseteq M(\alpha) \) (ie \((M(KB) \cap M(\alpha)) = M(KB)\))

E.g. \( KB \): Giants won and Reds won

\( \alpha \): Giants won

[\( \alpha \neq “Gaints and Yankees won” \)]
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 ⇒ 8 possible models

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

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

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

**Inference**

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

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

Entailment = needle in haystack; inference = finding it

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

**Completeness:** $i$ is complete if whenever $KB \models \alpha$, it is also true that $KB \vdash \alpha$

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)

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

$\alpha_2 = \text{"2,2 is safe", } KB \not\models \alpha_2$
**Propositional logic: Semantics**

Each model specifies true/false for each proposition symbol

E.g. \( P_1, P_2, P_3 \)

\[
\begin{array}{ccc}
\text{true} & \text{true} & \text{false} \\
\end{array}
\]

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

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

\[
\begin{align*}
\neg S & \quad \text{is true iff } \quad S \quad \text{is false} \\
S_1 \land S_2 & \quad \text{is true iff } \quad S_1 \quad \text{and } \quad S_2 \quad \text{is true} \\
S_1 \lor S_2 & \quad \text{is true iff } \quad S_1 \quad \text{or } \quad S_2 \quad \text{is true} \\
S_1 \Rightarrow S_2 & \quad \text{is true iff } \quad S_1 \quad \text{is false or } \quad S_2 \quad \text{is true} \\
& \quad \text{i.e., is false iff } \quad S_1 \quad \text{and } \quad S_2 \quad \text{is false} \\
S_1 \Leftrightarrow S_2 & \quad \text{is true iff } \quad S_1 \quad \Rightarrow S_2 \quad \text{is true and } \quad S_2 \quad \Rightarrow S_1 \quad \text{is true}
\end{align*}
\]

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

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

**Truth tables for connectives**

<table>
<thead>
<tr>
<th></th>
<th></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>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</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>false</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 \([i, j]\).

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

“A square is breezy if and only if there is an adjacent pit”
Chapter 7

Truth tables for inference

| \( B_{1,1} \) | \( B_{2,1} \) | \( P_{1,1} \) | \( P_{1,2} \) | \( P_{2,1} \) | \( P_{2,2} \) | \( R_{1} \) | \( R_{2} \) | \( R_{3} \) | \( R_{4} \) | \( R_{5} \) | \( KB \) |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| false | false | false | false | false | false | true | true | true | true | false | false | false |
| false | false | false | false | false | false | true | true | true | true | false | false | false |
| false | true | false | false | false | false | true | true | true | true | false | false | true |
| false | true | false | false | false | false | true | true | true | true | false | false | true |
| false | true | false | false | false | false | true | true | true | true | false | false | true |
| true | true | false | false | false | false | true | true | true | true | false | false | true |
| true | true | false | false | false | false | true | true | true | true | false | false | true |
| true | true | false | false | false | false | true | true | true | true | false | false | true |
| true | true | false | false | false | false | true | true | true | true | false | false | true |

Enumerate rows/models (different assignments to symbols \( B_{1,1}...P_{3,1} \)). \( R_{1}, R_{2}, ... \) are (true) sentences in \( KB \) (what we know).

When all \( R_{1}, R_{2}, ... \) are true, \( KB \) is true.

There are only three rows. \( P_{1,2} \) (\( \alpha \)) is all false (i.e., infer no pit in \([1,2])\).

Inference by enumeration

Depth-first enumeration of all models is sound and complete

```
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 ← \text{FIRST}\((symbols)\); rest ← \text{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

Logical equivalence

Two sentences are logicall y equivalent iff true in same models:

\[ \alpha \equiv \beta \text{ if and only if } \alpha \vdash \beta \text{ and } \beta \vdash \alpha \]

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

Validity and satisfiability

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

\[ \text{e.g., } True, \ A \lor \neg A, \ A \Rightarrow A, \ (A \land (A \Rightarrow B)) \Rightarrow B \]

Validity is connected to inference via the Deduction Theorem:

\[ KB \vdash \alpha \text{ if and only if } (KB \Rightarrow \alpha) \text{ is valid} \]

A sentence is satisfiable if it is true in some models

\[ \text{e.g., } A \lor B, \ C \]

A sentence is unsatisfiable if it is true in no models

\[ \text{e.g., } A \land \neg A \]

Satisfiability is connected to inference via the following:

\[ KB \vdash \alpha \text{ if and only if } (KB \land \neg \alpha) \text{ is unsatisfiable} \]

i.e., prove \( \alpha \) by reductio ad absurdum (proof by contradiction)
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
  - 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)
  - e.g., min-conflicts-like hill-climbing algorithms

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 algorithm

```
function PL-FC-ENTAILS?(KB, q) returns true or false
  inputs: KB, the knowledge base, a set of propositional Horn clauses
          q, the query, a proposition symbol
  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 in KB
  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
          decrement count[c]
          if HEAD[c] = q then return true
        PUSH(HEAD[c], agenda)
      return false
```

Forward and backward chaining

**Horn Form** (restricted)
- \( KB = \text{conjunction} \) of Horn clauses
- Horn clause =
  - proposition symbol; or
  - (conjunction of symbols) \( \Rightarrow \) symbol
- E.g., \( C \land (B \Rightarrow A) \land (C \land D \Rightarrow B) \)

**Modus Ponens** (for Horn Form): complete for Horn KBs
- \( \frac{\alpha_1, \ldots, \alpha_n, \alpha_1 \land \cdots \land \alpha_n}{\beta} \Rightarrow \beta \)

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

Forward chaining example

Forward chaining example

Forward chaining example
Proof of completeness (FC with Horn clauses)

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 $m$
   - **Proof**: Suppose a clause $a_1 \land \ldots \land a_k \Rightarrow b$ is false in $m$
     - Then $a_1 \land \ldots \land a_k$ 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 $m$

**General idea**: construct any model of $KB$ by sound inference, check $\alpha$

Backward chaining example

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

Chapter 7 61

Backward chaining example

Chapter 7 62

Backward chaining example

Chapter 7 63

Backward chaining example

Chapter 7 64
Backward chaining example

Chapter 7 65

Forward vs. backward chaining

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

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

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.,

\[
\begin{align*}
P_{1,3} & \lor P_{2,2}, \quad \neg P_{2,2} \\
\hline
P_{1,3} & P_{1,3}
\end{align*}
\]

Resolution is sound and complete for propositional logic

Conversion to CNF

\(B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})\)

1. Eliminate \(\Leftrightarrow\), replacing \(\alpha \Leftrightarrow \beta\) 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:

\((\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1})\)
Resolution algorithm

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

```plaintext
function PL-RESOLUTION(KB, \alpha) returns true or false
  inputs: KB, the knowledge base, a sentence in propositional logic
           \alpha, the query, a sentence in propositional logic
  clauses ← the set of clauses in the CNF representation of $KB \land \neg \alpha$
  loop do
    new ← {} // updated from book
    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 ⊆ clauses then return false
    clauses ← clauses ∪ new
  end loop
  return false
```

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.

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

Resolution is complete for propositional logic

Propositional logic lacks expressive power

Resolution example

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