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

<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

- Exploring a wumpus world

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

- Exploring a wumpus world

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

Chapter 7 19

Other tight spots

- Breeze in (1,2) and (2,1) ⇒ no safe actions
- Assuming pits uniformly distributed, (2,2) has pit w/ prob 0.86, vs. 0.31
- 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

Chapter 7 20

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 $x = 0$, $y = 6$

Chapter 7 21

Logic in general

- Logic in general

<table>
<thead>
<tr>
<th></th>
<th>Logic in general</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>OK</td>
</tr>
<tr>
<td>A</td>
<td>OK</td>
</tr>
<tr>
<td>S</td>
<td>OK</td>
</tr>
<tr>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>

Chapter 7 22

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 $x = x + y$

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

- Note: brains process syntax (of some sort)

Chapter 7 23

Models

- Models

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

Chapter 7 24

Models

- Models
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

\( KB = \) wumpus-world rules + observations
\( \alpha_1 = \) “[1,2] is safe”, \( KB \models \alpha_1 \), proved by model checking

\( KB = \) wumpus-world rules + observations
\( \alpha_2 = \) “[2,2] is safe”, \( KB \not\models \alpha_2 \)
Inference

KB ⊨ 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 ⊨ α, it is also true that KB |= α

Completeness: i is complete if whenever KB |= α, it is also true that KB ⊨ 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 P₁, P₂ etc are sentences

If S is a sentence, ¬S is a sentence (negation)
If S₁ and S₂ are sentences, S₁ ∧ S₂ is a sentence (conjunction)
If S₁ and S₂ are sentences, S₁ ∨ S₂ is a sentence (disjunction)
If S₁ and S₂ are sentences, S₁ → S₂ is a sentence (implication)
If S₁ and S₂ are sentences, S₁ ↔ S₂ is a sentence (biconditional)

Propositional logic: Semantics

Each model specifies true/false for each proposition symbol

E.g. P₁₂ P₁₂ P₁₁ (3 symbols ⇒ 2³ = 8 models)

true true false

Rules for evaluating truth with respect to a model m:

¬S is true iff S is false
S₁ ∧ S₂ is true iff S₁ is true and S₂ is true
S₁ ∨ S₂ is true iff S₁ is true or S₂ is true
S₁ → S₂ is true iff S₁ is false or S₂ is true (i.e., S₁ ↔ S₂ is true iff S₁ is true and S₂ is false)
S₁ ↔ S₂ is true iff S₁ ⇒ S₂ is true and S₂ ⇒ S₁ is true

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

¬P₁₂ ∧ (P₁₂ ∨ P₁₁) = ¬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

R₅: ¬P₁₁ (given)
R₆: ¬B₁₁ (percept)
R₇: B₂₂ (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"

R₄: B₁₁ ⇔ (P₁₂ ∨ P₁₁)
R₅: B₂₂ ⇔ (P₁₁ ∨ P₂₂ ∨ P₃₁)

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>false</td>
<td>false</td>
<td>false</td>
<td>false</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>false</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
    $P$ = 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:

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

<table>
<thead>
<tr>
<th>$\alpha \land \beta$</th>
<th>$\alpha \lor \beta$</th>
<th>$\neg (\alpha \land \beta)$</th>
<th>$\neg (\alpha \lor \beta)$</th>
<th>$\alpha \Rightarrow \beta$</th>
<th>$\neg \neg \alpha$</th>
<th>$\alpha \iff \beta$</th>
<th>$\neg \neg (\alpha \Rightarrow \beta)$</th>
<th>$\neg \neg (\alpha \iff \beta)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>commutativity of $\land$</td>
<td>commutativity of $\lor$</td>
<td>associativity of $\land$</td>
<td>associativity of $\lor$</td>
<td>implication elimination</td>
<td>double-negation elimination</td>
<td>contraposition</td>
<td>biconditional elimination</td>
<td>distribution of $\land$ over $\lor$</td>
</tr>
</tbody>
</table>

### Validity and satisfiability

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

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

Validity is connected to inference via the Deduction Theorem:

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

A sentence is satisfiable if it is true in some model

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

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:

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

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 “actions” 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
  - heuristic search in model space (sound but incomplete)
  - e.g., min-conflicts-like hill-climbing algorithms

### Forward and backward chaining

Horn Form (restricted)

\[ KB = \text{conjunction of Horn clauses} \]

Horn clause =

- \( \diamond \) proposition symbol; or
- \( \diamond \) (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

\[ \alpha_1, \ldots, \alpha_n, \quad \alpha_1 \land \cdots \land \alpha_n \Rightarrow \beta \]

\[ \beta \]

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

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
    \( p \leftarrow \text{POP}(\text{agenda}) \)
    unless inferred\( [p] \) do
        inferred\( [p] \leftarrow \text{true} \)
        for each Horn clause \( c \) in whose premise \( p \) appears do
            decrement count\( [c] \)
            if count\( [c] \) = 0 then do
                if Head\( [c] \) = \( q \) then return true
                \( \text{PUSH}(\text{Head}[c], \text{agenda}) \)
            end
        end
    end
return false

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 $m$

   \[ \text{Proof: Suppose a clause } a_1 \land \ldots \land a_k \Rightarrow b \text{ is false in } m \]

   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 $m$

   \[ \text{General idea: construct any model of } KB \text{ by sound inference, check } \alpha \]

Backward chaining

Idea: work backwards from the query $q$:

   \[ \text{to prove } q \text{ by BC}, \]

   \[ \text{check if } q \text{ is known already, or} \]

   \[ \text{prove by BC all premises of some rule concluding } q \]

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

Avoid repeated work: check if new subgoal

   \[ 1) \text{ has already been proved true, or} \]

   \[ 2) \text{ has already failed} \]
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
clauses
E.g., \((A \lor B) \land (B \lor \neg C \lor \neg D)\)

Resolution inference rule (for CNF):

\[
\ell_1 \lor \cdots \lor \ell_i \lor m_1 \lor \cdots \lor m_1 \lor m_1 \lor \cdots \lor m_n
\]

where \(\ell_i\) and \(m_1\) are complementary literals. E.g.,

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

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 \((\lor \text{ over } \land)\) 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

function PL-Resolve(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 \(KB \land \neg \alpha\)

new— { }

loop do
for each \(C_s, C_f\) in clauses do
resolvents— PL-Resolve(C_s, C_f)
if resolvents contains the empty clause then return true
new— new \cup resolvents
if new \subseteq clauses then return false

clauses— clauses \cup new

Resolution example

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

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

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.