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

Inference engine

Knowledge base

domain-specific content

domain-independent algorithms

Knowledge base = set of sentences in a formal language

Declarative approach to building an agent (or other system):

Tell, then, Ask - 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

Chapter 7

Local AGENTS
Wumpus world characterization

Observable

- No|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

Wumpus was there
File was there
Shoot straight ahead: can use a strategy of coercion

If dead
Smell in (1,1)

Cannot move
Can use a strategy of coercion:
Shoot straight ahead

If safe and no pits

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

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 + y|y is a sentence;
|x + y > y| is not a sentence

|x + y|y is true in a world where x = 7, y = 1;
|x + y|y is true in a world where x = 1, y = 7;
|x + y|y is true if the number x + y is no less than the number y

It is the language of arithmetic

Semantics define the "meaning" of sentences

Syntax deffnes the sentences in the language

Logics are formal languages for representing information and inferences
Situation after detecting nothing in $[1,1]$, moving right, breeze in $[2,1]$

Consider possible models for $\mathcal{A}$ assuming only pits

$\mathcal{A}$

$\mathcal{B}$

$\mathcal{C}$

$\mathcal{D}$

$3$ Boolean choices imply $8$ possible models. Consider possible models for $\mathcal{D}$
Inference

KB

\[ i \rightarrow \text{sentence can be derived from } KB \] by procedure \( i \)

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

Entailment = needle in haystack; inference = finding it

Soundness:

\[ i \] is sound if whenever \( KB \rightarrow i \), it is also true that \( KB \rightarrow j \)

Completeness:

\[ i \] is complete if whenever \( KB \rightarrow j \), it is also true that \( KB \rightarrow i \)

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 \( KB \).

Chapter 7 31

Propositional logic: Syntax

Propositional logic is the simplest logic—illustrates basic ideas

The proposition symbols \( P_1, P_2, \ldots \) 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)

Chapter 7 32

Propositional logic: Semantics

Each model specifies true/false for each proposition symbol

E.g.

\[
\begin{align*}
P_1 &:= \text{true} \\
P_2 &:= \text{false} \\
P_3 &:= \text{false} \\
\end{align*}
\]

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

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

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

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

\[
\begin{align*}
P_1 \land (P_2 \lor (P_3 \land P_1)) &:= \text{true} \\
\text{false} \lor \text{true} &:= \text{true} \\
\text{true} &:= \text{true}
\end{align*}
\]

Chapter 7 33

Truth tables for connectives

<table>
<thead>
<tr>
<th>( P )</th>
<th>( Q )</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>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</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>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>
</tr>
</tbody>
</table>

Chapter 7 34

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 \( [i,j] \).

\[ P_{1,1} \land (P_{2,1} \lor (P_{3,1} \land (P_{1,2} \land P_{2,1})))) \]

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

\[ B_{1,1} \land \neg B_{2,1} \]

The proposition symbols \( P \) fc are sentences

Propositional logic: Syntax

Wumpus world sentences

<table>
<thead>
<tr>
<th>( P )</th>
<th>( Q )</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>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</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>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>
</tr>
</tbody>
</table>

Chapter 7 35

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 \( [i,j] \).

\[ P_{1,1} \land (P_{2,1} \lor (P_{3,1} \land (P_{1,2} \land P_{2,1})))) \]

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

\[ B_{1,1} \land \neg B_{2,1} \]

Pro.
Truth tables for truth value assignment in all models.

1. If KB is true in a model, then A is also true.
2. If KB is false in a model, then ¬A is true.
3. If KB is true in all models, then A is a tautology.
4. If KB is false in some model, then ¬A is a tautology.
5. If KB is true in all models, then KB is valid.
6. If KB is true in some model, then A is a model.
7. If KB is false in all models, then ¬A is a model.
8. If KB is false in some model, then ¬A is a model.
Forward chaining

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

P \rightarrow Q
L \rightarrow M
P
B
L
A
Chapter 7 43

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
decrement count[c]
if count[c] = 0 then do
if Head[c] = q then return true
Push(Head[c], agenda)

return false

Chapter 7 44

Forward chaining example

Chapter 7 45

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
decrement count[c]
if count[c] = 0 then do
if Head[c] = q then return true
Push(Head[c], agenda)

return false

Chapter 7 46

Forward chaining example

Chapter 7 47

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
decrement count[c]
if count[c] = 0 then do
if Head[c] = q then return true
Push(Head[c], agenda)

return false

Chapter 7 48
Proof of completeness

1. **Forward Chaining Example**

   - **Assumption:** Every model of $KB$ is true in every model of $KB$, including $m$.

   - **Idea:** Construct any model of $KB$ by sound inference.

   - **Steps:**
     1. If $KB$ is a model of $KB$, then $KB$ is complete.
     2. To prove $KB$, suppose a clause $a_1 \land \cdots \land a_k$ is false in $KB$.

   - **Proof:**
     1. Consider the current state as model $m$.
     2. If clauses a fixed point when no new atomic sentences are derived,
     3. Every clause in the current KB is true in $m$. Check if new subgoals have already been solved.

   - **Avoid loops:** Check if new subgoals are already on the goal stack.

   - **Avoid repeated work:** Check if new subgoals are already on the goal stack.

   - **Idea:** Work backwards from the query to prove $q$ by BC.

   - **Backward Chaining Example**

     - **Assumption:**
       1. Every clause in the current KB is true in $m$.
       2. Consider the current state as model $m$.
       3. If clauses a fixed point when no new atomic sentences are derived.

     - **Stages:**
       1. Check if new subgoals have already been solved.
       2. Avoid loops: Check if new subgoals are already on the goal stack.

   - **Forward Chaining Example**

     - **Assumption:**
       1. Every clause in the current KB is true in $m$.
       2. Consider the current state as model $m$.
       3. If clauses a fixed point when no new atomic sentences are derived.

     - **Stages:**
       1. Check if new subgoals have already been solved.
       2. Avoid loops: Check if new subgoals are already on the goal stack.

   - **Idea:** Work backwards from the query to prove $q$ by BC.

   - **Backward Chaining Example**

     - **Assumption:**
       1. Every clause in the current KB is true in $m$.
       2. Consider the current state as model $m$.
       3. If clauses a fixed point when no new atomic sentences are derived.

     - **Stages:**
       1. Check if new subgoals have already been solved.
       2. Avoid loops: Check if new subgoals are already on the goal stack.
Backward chaining example

Complexity of BC can be much less than linear in size of K

e.g., Where are my keys? How do I get into a PhD program?

BC is goal-driven, appropriate for problem-solving.

May do lots of work that is irrelevant to the goal

FC is data-driven, e.g., object recognition, unconscious processing

Forward vs. backward chaining

Backward chaining example

Backward chaining example

Backward chaining example

Backward chaining example
Propositional logic is complete for propositional logic:

Resolution is sound and complete for propositional logic.

Resolution is a method for deriving sentences from other sentences.

Example: Let's derive the sentence \( P \land Q \lor R \lor S \) from the following set of clauses:

1. \( P \lor Q \)
2. \( Q \lor R \)
3. \( R \lor S \)

We can use the resolution rule to derive the goal sentence.

Resolution: Let's derive the sentence \( P \land Q \lor R \lor S \) from the following set of clauses:

1. \( P \lor Q \)
2. \( Q \lor R \)
3. \( R \lor S \)

We can use the resolution rule to derive the goal sentence.

Resolution: Let's derive the sentence \( P \land Q \lor R \lor S \) from the following set of clauses:

1. \( P \lor Q \)
2. \( Q \lor R \)
3. \( R \lor S \)

We can use the resolution rule to derive the goal sentence.