CHAPTER 7

LOGICAL AGENTS
Knowledge-based agents

Wumpus world

Logic in general—models and entailment

Equivalence, validity, satisfiability

Inference rules and theorem proving

Resolution

Forward chaining

Backward chaining

Propositional (Boolean) logic

Outline
Knowledgebases

- Knowledgebase = set of sentences in a formal language
- Knowledgebase
  - Domain-specific content
  - Domain-independent algorithms

Inference engine

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

- Tell it what it needs to know
- Ask itself what to do—answers should follow from the KB

Or at the implementation level:

- Tell, i.e., what they know, regardless of how implemented
- Ask, i.e., what it needs to know, regardless of how it knows
A simple knowledge-based agent

\textbf{KB-Agent} \((\text{percept})\) returns an action:

\begin{align*}
\text{return} & \quad \text{action} \\
& \quad t + 1 \rightarrow t \\
& \quad \text{Tell}(\text{KB'}, \text{Make-Action-Sentence}(\text{action}, t)) \\
& \quad \text{Ask}(\text{KB'}, \text{Make-Action-Query}(t)) \\
& \quad \text{Tell}(\text{KB'}, \text{Make-Percept-Sentence}(\text{percept}, t)) \\
\end{align*}

\(t\), a counter, initially 0, indicating time

\textbf{Function KB-Agent} \textbf{returns} an action

\textbf{Static: KB'}, a knowledge base

\textbf{Represent states, actions, etc.}

\textbf{Deduce appropriate actions}

\textbf{Deduce hidden properties of the world}

\textbf{Update internal representations of the world}

\textbf{Incorporate new percepts}

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
Sensors: Breeze, Glitter, Smell

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

Performance measure

-1 per step
-Gold +1000, death -1000

Environment

Breeze
Smell

1
2
3
4

1
2
3
4

PIT

Start

Gold

Wumpus World PEAS description
Wumpus world characterization

Observable??
Wumpus world characterization

Deterministic? No—only local perception

Observable? No
<table>
<thead>
<tr>
<th>Observable</th>
<th>No</th>
<th>only local perception</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic</td>
<td>Yes</td>
<td>outcomes exactly specified</td>
</tr>
<tr>
<td>Episodic</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Wumpus world characterization
Chapter 7

Wumpus world characterization

<table>
<thead>
<tr>
<th>Static?</th>
<th>Yes—sequential at the level of actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Episodic?</td>
<td>No—only local perception</td>
</tr>
<tr>
<td>Deterministic?</td>
<td>Yes—outcomes exactly specified</td>
</tr>
<tr>
<td>Observable?</td>
<td>Yes—local perception</td>
</tr>
</tbody>
</table>

Observed
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??
<table>
<thead>
<tr>
<th>Characterization</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-agent</td>
<td>Yes</td>
</tr>
<tr>
<td>Discrete</td>
<td>Yes</td>
</tr>
<tr>
<td>Static</td>
<td>Yes</td>
</tr>
<tr>
<td>Episodic</td>
<td>No</td>
</tr>
<tr>
<td>Deterministic</td>
<td>Yes</td>
</tr>
<tr>
<td>Observable</td>
<td>No</td>
</tr>
<tr>
<td>Wumpus World Characterization</td>
<td></td>
</tr>
<tr>
<td>Feature</td>
<td>Wumpus world characterization</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>Observable</td>
<td>No—only local perception</td>
</tr>
<tr>
<td>Deterministic</td>
<td>Yes—outcomes exactly specified</td>
</tr>
<tr>
<td>Episodic</td>
<td>No—sequential at the level of actions</td>
</tr>
<tr>
<td>Static</td>
<td>Yes—Wumpus and Pits do not move</td>
</tr>
<tr>
<td>Discrete</td>
<td>Yes</td>
</tr>
<tr>
<td>Single-agent</td>
<td>Yes—Wumpus is essentially a natural feature</td>
</tr>
</tbody>
</table>
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
other tight spots

Assuming pits uniformly distributed,

no safe actions

Breeze in (1,2) and (2,1)

(2,2) has pit w/ prob 0.86, vs. 0.31

Assuming wumpus wasn’t there, safe

wumpus was there, dead

Can use a strategy of coercion:

cannot move

Smell in (1,1)

(1,2) and (2,1)

Breeze

The wumpus wasn’t there, vs. 0.31
Logics are formal languages for representing information such that conclusions can be drawn. Syntax defines the sentences in the language. Semantics defines the "meaning" of sentences. i.e., define truth of a sentence in a world. E.g., the language of arithmetic

\[
\begin{align*}
6 = y & \text{ is false in a world where } x = 0, \\
1 = y & \text{ is true in a world where } x = 7, \\
\end{align*}
\]

\[
\begin{align*}
x + y \geq 2 & \text{ is true if the number } x + y \text{ is no less than the number } y, \\
x + y \geq 2 & \text{ is a sentence; } x + y + 2 \geq x \text{ is not a sentence.}
\end{align*}
\]

Chapter 7
Entailment is a relationship between sentences (i.e., syntax) that is based on semantics.

Entailment means that one thing follows from another.

\[ KB \vdash \alpha \]

\[ KB \models \alpha \]

Note: brains process syntax (of some sort) that is based on semantics.

Example:

If two sentences are true in all worlds where \( KB \) is true, then \( KB \) entails \( \alpha \).

\[ KB \models \alpha \]

Example:


\[ E.g., \text{the Giants won or the Reds won} \]

\[ E.g., \text{the Giants won and the Reds won} \]

\[ x + y = 4 \] entails \[ 4 = x + y \]
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 \( m \).

E.g. \( KB = \text{Giants won and Reds won} \)

Then \( KB \models \alpha \) if and only if \( m \in M (KB) \)

\( M (\alpha) \) is the set of all models of \( \alpha \).

\[ M (\alpha) = \{ m : m \models \alpha \} \]
Situation after detecting nothing in [1,1], moving right, breeze in [2,1].

Consider possible models for $s$ assuming only pits.

3 Boolean choices $\iff$ 8 possible models

$A \iff B \iff C$
Wumpus models
$KB = \text{wumpus-world rules + observations}$
$\alpha_1 = \{1, 2\} \text{ is safe}, \ KB \models \alpha_1$, proved by model checking

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

Wumpus models
\[ \alpha \not\models \Box i \land \Box j, \Box k \Rightarrow \Box l \]

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

\[ KB = \text{wumpus models} \]
Inference

That is, the procedure will answer any question whose answer follows from what is known by the KB. Complete inference procedure:

Consequences of KB are a haystack; entailment = needle in haystack; inference = finding it.

Soundness: is sound if

Whenever $\forall \alpha \in \text{KB} \vdash \alpha$, it is also true that $\alpha$.

Completeness: is complete if

Whenever $\forall \alpha \in \text{KB} \models \alpha$, it is also true that $\alpha$.

Entailment = needle in haystack; inference = finding it.

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.

Inference
Propositional Logic: Syntax

Propositional logic is the simplest logic that illustrates basic ideas.

Propositional symbols $P_1$, $P_2$, etc. are sentences.

If $S_1$ and $S_2$ are sentences, then $S_1 \land S_2$ is a sentence (conjunction).

If $S_1$ and $S_2$ are sentences, then $S_1 \lor S_2$ is a sentence (disjunction).

If $S_1$ and $S_2$ are sentences, then $S_1 \rightarrow S_2$ is a sentence (implication).

If $S_1$ and $S_2$ are sentences, then $S_1 \equiv S_2$ is a sentence (biconditional).

The proposition symbols $P_1$, $P_2$, etc. are sentences.
Propositional Logic: Semantics

Each model specifies true/false for each proposition symbol.

Propositional Logic: Semantics
### Truth Tables for Connectives

<table>
<thead>
<tr>
<th>$\neg p$</th>
<th>$\neg \neg p$</th>
<th>$\neg \neg p$</th>
<th>$\neg \neg p$</th>
<th>$\neg \neg p$</th>
<th>$\neg \neg p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

**Truth Tables for Connectives**
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 $[i,j]$.

"Pits cause breezes in adjacent squares."

\[
\begin{align*}
&\neg B_{1,1} \\
&\neg B_{1,1} \\
&\neg P_{1,1} \\
&\neg P_{1,1} \\
&\neg P_{1,2} \\
&\neg B_{1,1} \\
\end{align*}
\]
Let $P_{i;j}$ be true if there is a pit in $(i,j)$.

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

\[ P_{i;j} \land (P_{i+1;j} \lor P_{i-1;j} \lor P_{i;j+1} \lor P_{i;j-1}) \iff B_{i+1;j} \lor B_{i-1;j} \lor B_{i;j+1} \lor B_{i;j-1} \]

Pits cause breezes in adjacent squares

\[ B_{i+1;j} \iff \neg P_{i+1;j} \land \neg P_{i-1;j} \land \neg P_{i;j+1} \land \neg P_{i;j-1} \]

Let $B_{i;j}$ be true if there is a breeze in $(i,j)$.
If KB is true in row, check that a is too.

Enumerate rows (different assignments to symbols),

<table>
<thead>
<tr>
<th>KB</th>
<th>B1</th>
<th>B2</th>
<th>P1</th>
<th>P2</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>false</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>false</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>false</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>false</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>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</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>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</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>

Truth tables for inference
Inference by enumeration

Depth-first enumeration of all models is sound and complete

\[ O(2^n) \] for \( n \) symbols; problem is co-NP-complete

```
function TT-ENTAILS(KB, q) returns true or false
inputs: KB, the knowledge base; q, a sentence in propositional logic

function TT-CHECK-ALL(KB, q, rest, model)
returns true or false
inputs: KB, the knowledge base; q, a sentence in propositional logic; rest, a list of the proposition symbols in KB and a symbol

if PL-TRUE?(KB, model) then return PL-TRUE?(q, model)
else return true else do
    if EMPTY?(symbols) then return true or false
    first = FIRST(symbols); rest = REST(symbols)
    return TT-CHECK-ALL(KB, q, rest, Extend(first; true; model)) and TT-CHECK-ALL(KB, q, rest, Extend(first; false; model))
```

Chapter 7
\[
\begin{align*}
\lor \text{ distributes over } \land & \quad \left( (\lor \land \varphi) \lor (\lor \land \psi) \right) \equiv \left( (\lor \land \psi) \lor (\lor \land \varphi) \right) \\
\land \text{ distributes over } \lor & \quad \left( (\land \lor \varphi) \land (\land \lor \psi) \right) \equiv \left( (\land \lor \psi) \land (\land \lor \varphi) \right) \\
\text{De Morgan} & \quad (\lor \land \varphi) \equiv (\lor \land \varphi) \\
\text{De Morgan} & \quad (\land \lor \varphi) \equiv (\land \lor \varphi) \\
\text{Biconditional elimination} & \quad (\varphi \iff \psi) \lor (\psi \iff \varphi) \equiv (\psi \iff \varphi) \\
\text{Implication elimination} & \quad (\varphi \land \psi) \equiv (\varphi \iff \psi) \\
\text{Contraposition} & \quad (\varphi \iff \psi) \equiv (\varphi \iff \psi) \\
\text{Double-negation elimination} & \quad \varphi \equiv (\neg \neg \varphi) \\
\land \text{ associativity} & \quad \left( (\land \land \varphi) \land \varphi \right) \equiv \left( (\land \land \varphi) \land \varphi \right) \\
\lor \text{ associativity} & \quad \left( (\lor \lor \varphi) \lor \varphi \right) \equiv \left( (\lor \lor \varphi) \lor \varphi \right) \\
\land \text{ commutativity} & \quad (\varphi \land \psi) \equiv (\psi \land \varphi) \\
\lor \text{ commutativity} & \quad (\varphi \lor \psi) \equiv (\psi \lor \varphi)
\end{align*}
\]

\[\varphi \equiv \psi \text{ if and only if } \varphi \iff \psi \equiv \varphi\]

**Logical equivalence**

Two sentences are logically equivalent iff true in same models.
Validity and satisfiability

Validity

A sentence is valid if it is true in all models, e.g., True, \(A\), \((A \land (A \lor B))\).

Validity is connected to inference via the Deduction Theorem:

\[ KB \models \alpha \iff KB \models \neg \neg \alpha \]

Validity and satisfiability

Satisfiability

A sentence is satisfiable if it is true in some model, e.g., \(A \lor \neg A\). A sentence is unsatisfiable if it is true in no models, e.g., \(C \land \neg C\).

Satisfiability is connected to inference via the following:

\[ KB \models \alpha \iff KB \models \neg \neg \alpha \]

Validity and satisfiability

A sentence is valid if it is true in all models, e.g., True, \(A\), \((A \land (A \lor B))\).
Proof methods divide into (roughly) two kinds:

1. Application of inference rules

   - Legitimate (sound) generation of new sentences from old sentences
   - Typically require translation of sentences into a normal form
   - Can use inference rules as operators in a standard search alg.

   Proof = a sequence of inference rule applications

2. Model checking

   - Typically require translation of model space (sound but incomplete)
   - Improved backtracking, e.g., Davis-Putnam-Logemann-Loveland
   - Truth table enumeration (always exponential in $n$)

   - Min-conflicts-like hill-climbing algorithms
   - Heuristic search in model space (sound but incomplete)

   - e.g., Davis-Putnam-Logemann-Loveland

Chapter 7
These algorithms are very natural and run in \textit{linear} time. Can be used with forward chaining or backward chaining:

\[
\begin{align*}
\emptyset' & \subseteq a_1 \lor \ldots \lor a_n \\
\emptyset' & \subseteq q_1 \land q_2 \land \ldots \land q_n
\end{align*}
\]

\textbf{Modus Ponens (for Horn Form): complete for Horn KBs}

\[
\begin{align*}
\emptyset' & \subseteq (A \land B) \\
\emptyset' & \subseteq (D \land C) \\
\emptyset' & \subseteq (A \lor C) \\
\emptyset' & \subseteq (B \lor D)
\end{align*}
\]

\textbf{E.g.}: 

\textbf{Example of symbol}: 

\textbf{Example of proposition symbol or Horn clause}: 

\textbf{Horn Form (restricted)}: 

\textbf{Forward and backward chaining}
Forward chaining

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

\[
\begin{align*}
A & \leftarrow M \\
L & \leftarrow A \\
M & \leftarrow L \\
P & \leftarrow M \\
B & \leftarrow P \\
\forall & \leftarrow B \land \forall \\
T & \leftarrow B \land \forall \\
W & \leftarrow T \land B \\
d & \leftarrow W \land T \\
\emptyset & \leftarrow d
\end{align*}
\]
Forward chaining algorithm

Function PL-FCENTAILS(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

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

For each Horn clause c in whose premise appears q do
    inferred[c] = true
    unless inferred[d] do
        POP(agenda)

While agenda is not empty do
    Pop(agenda)
    if inferred[Head[c]] = true then return true
    Decrement count[c]
    unless count[c] = 0 then inferred[c] = true
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Proof of completeness

General idea: construct any model of KB by sound inference, check a

1. FC derives every atomic sentence that is entailed by KB

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

4. Hence m is a model of KB

5. If KB |- b, b is true in every model of KB, including m

Therefore the algorithm has not reached a fixed point!

Then a1 \lor \ldots \lor ak is true in m and q is false in m

Proof: Suppose a clause a1 \lor \ldots \lor ak \iff q is false in m

Then a1 \lor \ldots \lor ak is true in m and q is false in m
Idea: work backwards from the query \( q \) to prove that it is already known or to prove by BC all premises of some rule concluding \( q \). Check if new subgoal has already been proved true, or has already failed. Avoid loops: check if new subgoal is already on the goal stack. Avoid repeated work: check if new subgoal is already on the goal stack.
Backward chaining example
Backward chaining example
Backward chaining example
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, cf. automatic, unconscious processing.

May do lots of work that is irrelevant to the goal.

e.g., object recognition, routine decisions.

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.

Chapter 7
Resolution is sound and complete for propositional logic.

\[ \frac{p_3 \lor p_2, p_2 \land p_3}{p_3} \]

where \( i \) and \( m \) are complementary literals. E.g.,

\[ \begin{array}{c}
\text{Resolution} \\
\text{Inference rule (for CNF): complete for propositional logic.}
\end{array} \]

E.g.,

\[ (\neg A \land \neg B) \lor (A \land B) \]

\[ \text{Clauses} \]

\[ \text{Conjunction of disjunctions of literals} \]

\[ \text{Conjunctive Normal Form (CNF—universal)} \]
Conversion to CNF

I. Eliminate \( \equiv \) replacing \( \equiv \) with \( \equiv \)

\[(B_{1}^{1} \lor (P_{1}^{1} \land P_{2}^{1})) \lor ((P_{1}^{1} \land P_{2}^{1}) \equiv (P_{1}^{2} \land P_{2}^{2})).\]

II. Eliminate \( \equiv \) replacing \( \equiv \) with \( \equiv \)

\[(B_{1}^{1} \equiv (P_{1}^{1} \lor P_{2}^{1})) \lor (((P_{1}^{1} \lor P_{2}^{1}) \equiv (P_{1}^{2} \lor P_{2}^{2}))).\]

III. Move \( \equiv \) inwards using de Morgan's rules and double-negation:

\[(B_{1}^{1} \equiv ((P_{1}^{1} \land P_{2}^{1}) \lor P_{1}^{1})) \lor (((P_{1}^{1} \land P_{2}^{1}) \lor P_{1}^{1}) \equiv (P_{1}^{2} \land P_{2}^{2}))).\]

IV. Apply distributivity law (\& over \lor) and flatten:

\[\neg B_{1}^{1} \lor (P_{1}^{1} \land P_{2}^{1}) \lor (P_{1}^{2} \land P_{2}^{1}) \lor (P_{1}^{1} \lor P_{2}^{1}) \lor (P_{1}^{2} \lor P_{2}^{1}) \lor (P_{1}^{1} \lor P_{2}^{1}) \lor (P_{1}^{1} \land P_{2}^{1}).\]
Resolution Algorithm

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

**Resolution Algorithm**
Resolution example

\[ \neg d_1 \land d_2 \land d_3 \land d_4 \land d_5 \land d_6 \land d_7 \land d_8 \land d_9 \land d_{10} = \neg (\neg d_1 \land \neg d_2 \land \neg d_3 \land \neg d_4 \land \neg d_5 \land \neg d_6 \land \neg d_7 \land \neg d_8 \land \neg d_9 \land \neg d_{10}) \]

\( KB \)
Propositional logic lacks expressive power.

Resolution is complete for propositional logic.

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

Wumpus world requires the ability to represent partial and negated information.

- **completeness**: derivations can produce all entailed sentences
- **soundness**: derivations produce only entailed sentences
- **inference**: deriving sentences from other sentences
- **entailment**: necessary truth of one sentence given another
- **semantics**: truth of sentences w.r.t. models
- **syntax**: formal structure of sentences

Basic concepts of logic:

to derive new information and make decisions

Logical agents apply inference to a knowledge base.