CHAPTER 7

LOGICAL AGENTS
Outline

Knowledge-based agents

Wumpus world

Logic in general—models and entailment

Propositional (Boolean) logic

Equivalence, validity, satisfiability

Inference rules and theorem proving

-resolution
-backward chaining
-forward chaining

Chapter 7
Declarative approach to building an agent (or other system):

\[ \text{Knowledge base} = \text{set of sentences in a formal language} \]

<table>
<thead>
<tr>
<th>Knowledge base</th>
<th>Inference engine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knowledge base</td>
<td>Domain-independent algorithms</td>
</tr>
<tr>
<td>Specific content</td>
<td>Domain-specific content</td>
</tr>
</tbody>
</table>

Agents can be viewed at the knowledge level, i.e., what they know, regardless of how implemented.

Tell it what it needs to know.

Then it can ask itself what to do—answers should follow from the KB.

Or at the implementation level, i.e., data structures in KB and algorithms that manipulate them.
A simple knowledge-based agent

```
return action
I + 1 → I

TELL(KB, MAKE-ACTION-SENTENCE(action, t))

ASK(KB, MAKE-ACTION-QUERY(t))

TELL(KB, MAKE-PRECEPT-SENTENCE(percept, t))
```

A counter, initially 0, indicating time

Static: KB, a knowledge base

Function KB-Agent\[\text{return}\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{percept}(\text{perce...
WumpusWorld PEAS description

Performances measure
-1 per step, -10 for using the arrow
1000 gold, -1000 death

Environment
- Squares adjacent to pits are breezy
- Squares adjacent to wumpus are smelly

Actuators
- Left turn, Right turn, Left turn, Right turn

Sensors
- Breeze, Smell, Gold

Forward, Grab, Release, Shoot

Shooting uses up the only arrow
Shooting kills wumpus and picks up gold if in same square
Grabbing picks up gold if in same square
Releasing drops the gold in the same square

Grabbing picks up gold if in same square
Shooting kills wumpus if you are facing it
Climbing up the arrow

1 2 3 4
START

Chapter 75
Chapter 7

Wumpus World Characterization

Deterministic?

Observable?

No—only local perception
<table>
<thead>
<tr>
<th>Characterization</th>
<th>Observable</th>
<th>Deterministic</th>
<th>Episodic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

Wumpus World Characterization
Wumpus world characterization

Observable??  No—only local perception

Deterministic??  Yes—outcomes exactly specified

Episodic??  No—sequential at the level of actions

Static??
<table>
<thead>
<tr>
<th>Characterization</th>
<th>Deterministic</th>
<th>Discrete</th>
<th>Observable</th>
<th>Static</th>
<th>Episodic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes — Wumpus and Pits do not move</td>
<td>Yes — outcomes exactly specified</td>
<td>No — only local perception</td>
<td>Yes — sequential at the level of actions</td>
<td>Yes — Wumpus world characterization</td>
<td></td>
</tr>
<tr>
<td>Characterization</td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Observable</td>
<td>No only local perception</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deterministic</td>
<td>Yes outcomes exactly specified</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Episodic</td>
<td>No sequential at the level of actions</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Static</td>
<td>Yes—Wumpus and Pits do not move</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single-agent</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Discrete</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wumpus World Characterization</td>
<td>Wumpus and Pits do not move</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deterministic</td>
<td>Yes—outcomes exactly specified</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Episodic</td>
<td>No—sequential at the level of actions</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Static</td>
<td>Yes—Wumpus and Pits do not move</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single-agent</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Discrete</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Chapter 711**
<table>
<thead>
<tr>
<th>Characterization</th>
<th>Observable</th>
<th>Deterministic</th>
<th>Episodic</th>
<th>Static</th>
<th>Discrete</th>
<th>Single-agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wumpus world</td>
<td>No—only local perception</td>
<td>Yes—outcomes exactly specified</td>
<td>No—sequential at the level of actions</td>
<td>Yes—Wumpus and Pits do not move</td>
<td>Yes</td>
<td>Yes—Wumpus is essentially a natural feature</td>
</tr>
</tbody>
</table>
Exploring a wumpus world

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td>OK</td>
<td>A</td>
</tr>
<tr>
<td>OK</td>
<td>OK</td>
<td></td>
</tr>
</tbody>
</table>
Exploring a Wumpus World
Exploring a Wumpus World
Exploring a Wumpus World
Exploring a Wumpus World
Exploring a Wumpus World
Smell in (1,1) cannot move

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

Assuming pits uniformly distributed,

no safe actions

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

Assuming wumpus wasn’t there dead

wumpus wasn’t there safe

Can use a strategy of coercion:

Cannot move

Safe

Wumpus was there safe

Wumpus was there dead

Shoot straight ahead

 autres sports
Logics are formal languages for representing information such that conclusions can be drawn. Syntax defines the sentences in the language and Semantics defines the "meaning" of sentences. E.g., the language of arithmetic defines the truth of a sentence in a world, i.e., $x + 2 > y$ is true in a world where $x = 7$, $y = 1$, and $x + 2 < y$ is not a sentence. Logical conclusions are drawn.
Entailment means that one thing follows from another.

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

$\forall \phi \in \mathcal{L}, \mathcal{L} \subseteq \{\forall, \exists, \land, \lor, \neg, \rightarrow, \leftrightarrow, \vdash\}$

$\mathcal{L}$ is the set of all well-formed formulas in a given language.

Knowledge base $\mathcal{KB}$ entails sentence $\phi$ if and only if $\phi$ is true in all worlds where $\mathcal{KB}$ is true.

$\mathcal{KB} \models \phi$
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.: \( KB = \) Giants won and Reds won

Then \( KB \models \alpha \) if and only if \( M \models (KB) \supset M \models \alpha \)

\( M(\alpha) \) is the set of all models of \( \alpha \).
Entailment in the Wumpus World

Consider possible models for $s$ assuming only pits

3 Boolean choices $\Rightarrow$ 8 possible models

Situation after detecting nothing in [1,1], moving right, breeze in [2,1]

Entailment in the Wumpus World
$KB = \text{wumpus-world rules + observations}$

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

\[ KB = [1, 2] \text{ is safe} \]

\[ \models [1, 2] \text{ is safe}, \ KB \models \alpha_1 \]

\[ \alpha_1 = \text{“1,2 is safe”} \text{, proved by model checking} \]

Wumpus models
KB = wumpus-world rules + observations

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

$\mathcal{O}_2 = \text{"[2,2] is safe", } KB \not\models \mathcal{O}_2$
Inference

KB

What is known by the KB.

That is, the procedure will answer any question whose answer follows from complete inference procedure.

Consequences of KB are haystack; is a needle.

Entailment = needle in haystack; inference = finding it.

Complteness: is complete if

\[ KB \vdash \alpha \]

Soundness: is sound if

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

Preview: we will define a logic (first-order logic) which is expressive enough

Sentence \( \alpha \) can be derived from KB by procedure.

Chapter 731
Propositional logic is the simplest logic—illustrates basic ideas.

The proposition symbols $p_1, p_2, \ldots$ etc are sentences.

P

Propositional Logic: Syntax
Propositional logic: Semantics

Each model specifies true/false for each proposition symbol.

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

Rules for evaluating truth with respect to a model $m$: 

(E.g. $P_1$, $P_2$, $P_3$, $P_4$)
<table>
<thead>
<tr>
<th>$P$</th>
<th>$\neg P$</th>
<th>$P \land Q$</th>
<th>$P \lor Q$</th>
<th>$P \iff Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</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>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</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 \([i,j]\).
A square is breezy if and only if there is an adjacent pit.

\[
\begin{align*}
( P_{i+1}^1 \wedge P_{i}^2 \wedge & P_{i-1}^3 ) \iff B_{i+1}^2 \\
( P_{i+2}^1 \wedge P_{i+1}^2 \wedge & P_{i}^3 ) \iff B_{i+1}^1
\end{align*}
\]

"Pits cause breezes in adjacent squares."

"A square is breezy if there is a breeze in an adjacent pit."

[\mathcal{W}m\text{umps World sentences]
If \( \text{KB} \) is true in row, check that \( \alpha \) is too.

Enumerate rows (different assignments to symbols),

<table>
<thead>
<tr>
<th>( \text{KB} )</th>
<th>( R_1 )</th>
<th>( R_2 )</th>
<th>( R_3 )</th>
<th>( \ldots )</th>
<th>( R_n )</th>
<th>( P_1 )</th>
<th>( P_2 )</th>
<th>( P_3 )</th>
<th>( \ldots )</th>
<th>( P_m )</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>false</td>
<td>false</td>
<td>false</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>false</td>
<td>false</td>
<td>false</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>false</td>
<td>false</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>false</td>
</tr>
<tr>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</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>false</td>
</tr>
<tr>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</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>false</td>
<td>false</td>
<td>false</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>false</td>
<td>false</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>false</td>
</tr>
<tr>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</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>false</td>
<td>false</td>
<td>false</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>false</td>
<td>false</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>false</td>
</tr>
<tr>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</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>false</td>
<td>false</td>
<td>false</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>false</td>
<td>false</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>false</td>
</tr>
<tr>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</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>false</td>
<td>false</td>
<td>false</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>false</td>
<td>false</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>false</td>
</tr>
<tr>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
</tr>
</tbody>
</table>

Truth tables for inference
Inference by enumeration

Depth-first enumeration of all models is sound and complete.

function \textsc{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 \textsc{TT-CHECK-ALL}(KB, \alpha, symbols, [])

function \textsc{TT-CHECK-ALL}(KB, \alpha, symbols, []) returns true or false

if \textsc{EMPTY}?(symbols) then
  if \textsc{PL-TRUE}?(KB, model) then return \textsc{PL-TRUE}?(\alpha, model)
  else return true
else do
  first — \textsc{FIRST}(symbols)
  rest — \textsc{REST}(symbols)
  \( P \leftarrow \textsc{TT-CHECK-ALL}(KB, \alpha, \text{rest, EXTEND}(P, true, model)) \) and
  \( \textsc{TT-CHECK-ALL}(KB, \alpha, \text{rest, EXTEND}(P, false, model)) \)
  return true
else return false

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

Chapter 7
### Logical Equivalence

Two sentences are logically equivalent if and only if they are true in the same models.

**De Morgan's Laws**
- \( \lnot (p \land q) \equiv (\lnot p) \lor (\lnot q) \)
- \( \lnot (p \lor q) \equiv (\lnot p) \land (\lnot q) \)

**De Morgan's Laws (Double-Negation)**
- \( \lnot \lnot p \equiv p \)

**Associativity**
- \( (p \land (q \land r)) \equiv (p \land q) \land r \)
- \( (p \lor (q \lor r)) \equiv (p \lor q) \lor r \)

**Commutativity**
- \( (p \land q) \equiv (q \land p) \)
- \( (p \lor q) \equiv (q \lor p) \)

**Distributivity**
- \( (p \land (q \lor r)) \equiv (p \land q) \lor (p \land r) \)
- \( (p \lor (q \land r)) \equiv (p \lor q) \land (p \lor r) \)

**Implication**
- \( p \rightarrow q \equiv \lnot p \lor q \)

**Contraposition**
- \( \lnot q \rightarrow \lnot p \equiv p \rightarrow q \)

**Double-Negation Elimination**
- \( \lnot \lnot p \equiv p \)

**Biconditional**
- \( (p \leftrightarrow q) \equiv (p \rightarrow q) \land (q \rightarrow p) \)

**Identity**
- \( p \equiv p \)

**Logical Equivalence**

\( p \iff q \equiv q \iff p \)

Two sentences are logically equivalent if they are true in the same models.
Validity and Satisfiability

A sentence is valid if it is true in all models, e.g.,\[ \text{True}, \quad A, \quad (A \land B) \land \neg C \]

Validity is connected to inference via the Deduction Theorem:

\[ KB \models \phi \iff KB \models \phi \land \neg \phi \]

Satisfiability is connected to inference via the following:

\[ KB \models \phi \iff KB \models \phi \land \neg \phi \]

A sentence is satisfiable if it is true in some model, e.g., \[ A \land B, \quad C \]

A sentence is unsatisfiable if it is true in no models, e.g., \[ \neg A \land \neg B \]

Satisfiability is connected to inference via the following:

\[ KB \models \phi \iff KB \models \phi \land \neg \phi \]

Validity is connected to inference via the Deduction Theorem:

\[ B \models (B \land (A \land B) \land \neg A) \iff \text{True}, \quad \neg A \land \neg A \]

A sentence is valid if it is true in all models.
Proof methods divide into (roughly) two kinds:

- Application of inference rules
  - Legitimate (sound) generation of new sentences from old rules
  - Improper backtracking, e.g., Davis-Putnam-Logemann-Loveland
- Model checking
  - Truth table enumeration (always exponential in $n$)
  - Heuristic search in model space (sound but incomplete)
    - e.g., min-conflicts-like hill-climbing algorithms
  - Improved backtracking, e.g., Davis-Putnam-Logemann-Loveland
  - Min-con, hill-climbing algorithms

Chapter 7
Forward and backward chaining

Horn Form

KB = conjunction of Horn clauses

Horn clause: or

E.g., \( \forall \text{ symbol} \subseteq \text{ proposition symbol} \)

Can be used with forward chaining or backward chaining.

Modus Ponens (for Horn Form): complete for Horn KBs

Chapter 7
Forward chaining

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

\[
\begin{align*}
T & \leftarrow B \land \forall \\
W & \leftarrow T \land B \\
d & \leftarrow \forall \land \forall \\
\emptyset & \leftarrow d
\end{align*}
\]
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
    \(p \leftarrow \text{POP}(\text{agenda})\)
    unless inferred[\(p\)] do
        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)
        end for
    end unless
end while

return false
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Chapter 750

Forward chaining example
Forward chaining example
Forward chaining example
Proof of completeness

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 $m$.
4. Therefore, the algorithm has not reached a fixed point.
5. If $KB \models b$, $b$ is true in every model of $KB$, including $m$.
6. Hence $m$ is a model of $KB$.

Proof: Suppose a clause $a_1 \lor \cdots \lor a_k$ is false in $m$ and $q$ is false in $m$.
Then $a_1 \lor \cdots \lor a_k$ is true in $m$ and $q$ is false in $m$.

General idea: Construct any model of $KB$ by sound inference, check $a$. If $KB \models b$, $b$ is true in every model of $KB$, including $m$. Hence $m$ is a model of $KB$.
Idea: work backwards from the query \( q \) to prove \( q \) by BC, check if new subgoal is already on the goal stack.

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

Avoid repeated work: check if new subgoal has already been proved true, or has already failed.

To prove by BC, all premises of some rule concluding \( b \) check if \( b \) is known already, or to prove by BC, \( b \) from the query: \( b \)
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Chapter 759

Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Forward vs. Backward Chains

**FC** is data-driven, automatic, unconscious processing.

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

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

E.g., object recognition, routine decisions.

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

BC is much less than linear in size of KB.

Forward vs. Backward Chains
Resolution is sound and complete for propositional logic.

\[
\frac{P_{1.3} \land P_{2.7}}{P_{1.3} \land P_{2.7}}
\]

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

\[
\begin{align*}
&\left( P_1 \lor P_2 \lor P_3 \right) \land \left( P_4 \lor P_5 \lor P_6 \right) \\
&\left( P_7 \lor P_8 \lor P_9 \right) \land \left( P_{10} \lor P_{11} \lor P_{12} \right)
\end{align*}
\]

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

E.g.,

\[
\begin{align*}
&\left( \neg P_2 \land \neg \neg P_2 \right) \lor \left( \neg P_4 \land \neg \neg P_4 \right) \\
&\left( \neg P_6 \land \neg \neg P_6 \right) \lor \left( \neg P_8 \land \neg \neg P_8 \right) \\
&\left( \neg P_{10} \land \neg \neg P_{10} \right) \lor \left( \neg P_{12} \land \neg \neg P_{12} \right)
\end{align*}
\]

Resolution is sound and complete for propositional logic.
Conversion to CNF

\[(B_1 \land B_1') \lor (B_1' \land B_1) \lor (B_1' \land B_2') \lor (B_1' \land B_2) \lor (B_1' \land B_2') \lor (B_1' \land B_2) \]

4. Apply distributivity law (\land over \lor) and flatten:

\[\neg B_1 \land (B_1' \land B_2' \lor B_1' \lor B_2') \lor (B_1' \land B_2') \lor (B_1' \land B_2) \]

3. Move ∧ inwards using de Morgan’s rules and double-negation:

\[\neg B_1 \land (B_1' \land B_2' \lor B_1' \land B_2) \lor (B_1' \land B_2') \lor (B_1' \land B_2) \]

2. Eliminate ‘ replacing \( \neg a \) with \( a' \).  

\[\neg B_1 \land (B_1' \land B_2' \lor B_1' \land B_2) \lor (B_1' \land B_2') \lor (B_1' \land B_2) \]

1. Eliminate \( \alpha \) replacing \( \alpha' \) with \( \alpha \).  

\[\neg B_1 \land (B_1' \land B_2' \lor B_1' \land B_2) \lor (B_1' \land B_2') \lor (B_1' \land B_2) \]

\[\neg B_1 \land B_2 \lor \neg B_1 \land B_2' \lor B_1' \land B_2 \lor B_1' \land B_2' \]

\[B_1' \lor B_2' \lor \neg B_1 \land \neg B_2' \lor \neg B_1 \land B_2 \]

\[\neg B_1 \land \neg B_2' \lor \neg B_1 \land B_2 \]

\[\neg B_1 \land (B_2' \lor B_2) \]

\[\neg B_1 \land B_2 \]

\[\neg B_1 \land B_2 \lor \neg B_1 \land B_2' \lor \neg B_1 \land B_2 \lor \neg B_1 \land B_2' \]

\[\neg B_1 \land \neg B_2' \lor \neg B_1 \land B_2 \]

\[\neg B_1 \land (B_2' \lor B_2) \]

\[\neg B_1 \land B_2 \]

\[\neg B_1 \land B_2 \lor \neg B_1 \land B_2' \lor \neg B_1 \land B_2 \lor \neg B_1 \land B_2' \]

\[\neg B_1 \land \neg B_2' \lor \neg B_1 \land B_2 \]

\[\neg B_1 \land (B_2' \lor B_2) \]

\[\neg B_1 \land B_2 \]

\[\neg B_1 \land B_2 \lor \neg B_1 \land B_2' \lor \neg B_1 \land B_2 \lor \neg B_1 \land B_2' \]

\[\neg B_1 \land \neg B_2' \lor \neg B_1 \land B_2 \]

\[\neg B_1 \land (B_2' \lor B_2) \]

\[\neg B_1 \land B_2 \]
Proof by contradiction, i.e., show $KB \lor \neg a$ unsatisfiable
Resolution example
Logical agents apply inference to a knowledge base to derive new information and make decisions.

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, reason by cases, etc.

Propositional logic lacks expressive power. Resolution is complete for propositional logic.

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

Forward, backward chaining are linear-time, complete for Horn clauses. Resolution is complete for propositional logic.

Propositional logic lacks expressive power.