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
  – forward chaining
  – backward chaining
  – resolution
♦ Model checking
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(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
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

Actions Left turn, Right turn,
    Forward, Grab, Release, Shoot

Sensors Breeze, Glitter, Smell

Goals Get gold back to start without entering pit or wumpus square
Wumpus world characterization

Observable??
Wumpus world characterization

**Observable??**  No—only *local* perception

**Deterministic??**
Wumpus world characterization

**Observable** No—only *local* perception

**Deterministic** Yes—outcomes exactly specified

**Episodic**
Wumpus world characterization

**Observable**? No—only *local* perception

**Deterministic**? Yes—outcomes exactly specified

**Episodic**? No—sequential at the level of actions

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

A

OK

OK

OK

Δ
Exploring a wumpus world
Exploring a wumpus world

B

P?

A

OK

P?

A

OK

OK
Exploring a wumpus world
Exploring a wumpus world

- P
- B
- A
- S
- W
- P?
- OK
- OK
- OK
- OK

AIMA 3rd ed. - Chapter 7  17
Exploring a wumpus world

A

P

B

OK

OK

W

AIMA 3rd ed. - Chapter 7  18
Exploring a wumpus world
Exploring a wumpus world
Uncertainty

There may be situations where there are no guaranteed safe moves. In that case, we will consider the likelihoods of events (e.g. if we have no choice but go to one of the squares that may have a pit, decide on which one has a higher probability of having a pit; we will see this later).

But for now, we assume that we are dealing with problems that can be solved with logical thinking and there is no chance involved (i.e. no remaining uncertainty).
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
How can we get an agent do what we did? That is, how can it know whether it is OK to move to [2,1]?

Notice that an exhaustive search is not an option here because the agent doesn't want to die.

The agent needs a formal inference mechanism to derive valid conclusions even when it does not know what the interpretation is (the computer does not need to know the meaning of OK, or Wumpus or Pit... )

We will use a formal logic to extend the capacity of our agents by endowing them with the capacity for general logical reasoning.
Logic in general

Syntax defines the sentences in the language

Semantics define the “meaning” of sentences; i.e., define truth of a sentence in a world

Logic is the framework where a formal language represents information such that conclusions can be drawn.

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 \( x = 0, \ y = 6 \)
Knowledge bases - 6.1

Knowledge base is a set of representations of facts (sentences) about the world, expressed in knowledge representation language.**

<table>
<thead>
<tr>
<th>Inference engine</th>
<th>→</th>
<th>domain-independent algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knowledge base</td>
<td>←</td>
<td>domain-specific content</td>
</tr>
</tbody>
</table>

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

The **implementation level** may represent the knowledge in various forms:
   e.g. UcakSeferi(Istanbul,Samsun) can be represented as a string
      in a list of strings or as TRUE in a 2D table indexed by location pairs
      etc.

But we don’t care about the implementation details of the KB.
Entailment

We are interested in new sentences that we can infer from the knowledgebase and the new percepts. E.g. seeing that there is no breeze in [1,1], can we infer that there is no wumpus in [1,2]? We will use the word entailment which 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 \( x = 0 \) entails \( xy = 0 \).

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

We will consider possible worlds or models that define the truth settings of all sentences.

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 \( \alpha \)

Difference between world and model:
"model" is used interchangeably with "world" in the general sense; but a "model" of a particular sentence is a "world" in which that sentence is True)
Entailment in the wumpus world

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

Consider possible models for ?s assuming only pits:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>B</td>
<td>?</td>
</tr>
<tr>
<td>A</td>
<td></td>
<td>A</td>
</tr>
</tbody>
</table>

AIMA 3rd ed. - Chapter 7  30
Wumpus models

3 Boolean choices  \(\Rightarrow\)  8 possible models:
Models of KB:

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

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

\[ \alpha_2 = \text{“[2,2] is safe”}, \quad KB \not\models \alpha_2 \]
What we saw is summarized as: $KB \models \alpha$ if and only if $M(KB) \subseteq M(\alpha)$

Equivalently; $KB \models \alpha$ if and only if $\alpha$ is true everywhere $KB$ is true.

E.g. $KB =$ "GS won" and "BJK won" $\alpha =$ "GS won"
Inference

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

Consequences of $KB$ are a haystack; $\alpha$ is a needle.
Entailment = needle in haystack; inference = finding it

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

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

**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 $KB$.
Representations

We know of some knowledge representation languages:

◊ Programming languages

In a programming language you can say that ”there is a pit in [2,2]” by
world[2,2] = pit, but you cannot say ”either there is a pit in [2,1] or in [3,2]”
or that ”there is a wumpus in some square”

◊ Propositional logic

◊ First-order logic

◊ Natural languages are very expressive, but also too flexible/ambiguous
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)
Propositional logic: Semantics

Each model specifies true/false for each proposition symbol:

E.g. \( P_{1,2} \quad P_{2,2} \quad P_{3,1} \)
\[ \text{true} \quad \text{true} \quad \text{false} \]

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

\[ \neg S \quad \text{is true iff} \quad S \quad \text{is false} \]
\[ S_1 \land S_2 \quad \text{is true iff} \quad S_1 \quad \text{is true and} \quad S_2 \quad \text{is true} \]
\[ S_1 \lor S_2 \quad \text{is true iff} \quad S_1 \quad \text{is true or} \quad S_2 \quad \text{is true} \]
\[ S_1 \Rightarrow S_2 \quad \text{is true iff} \quad S_1 \quad \text{is false or} \quad S_2 \quad \text{is true} \]
\[ \text{i.e., is false iff} \quad S_1 \quad \text{is true and} \quad S_2 \quad \text{is false} \]
\[ S_1 \iff S_2 \quad \text{is true iff} \quad S_1 \quad \text{is true and} \quad S_2 \quad \text{is true} \]
### Truth tables for connectives

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

◊ **True** is true in every model

◊ **False** is false in every model

◊ Simple recursive process evaluates an arbitrarily complex sentence, e.g.,

$$\neg P_{1,2} \land (P_{2,2} \lor P_{3,1}) = false \land (false \lor true) = false \land true = false$$
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 $[i,j]$.

$\neg P_{1,1}$
$\neg B_{1,1}$
$B_{2,1}$

“Pits cause breezes in adjacent squares”
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 $[i,j]$.

$\neg P_{1,1}$
$\neg B_{1,1}$
$B_{2,1}$

“Pits cause breezes in adjacent squares”

$B_{1,1} \iff (P_{1,2} \lor P_{2,1})$
$B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1})$

“A square is breezy if and only if there is an adjacent pit”
Truth tables for inference

<table>
<thead>
<tr>
<th>(B_{1,1})</th>
<th>(B_{2,1})</th>
<th>(P_{1,1})</th>
<th>(P_{1,2})</th>
<th>(P_{2,1})</th>
<th>(P_{2,2})</th>
<th>(P_{3,1})</th>
<th>(R_1)</th>
<th>(R_2)</th>
<th>(R_3)</th>
<th>(R_4)</th>
<th>(R_5)</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>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</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>true</td>
<td>true</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>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<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>false</td>
<td>true</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>
<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>
<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 true too.
Inference by enumeration

Depth-first enumeration of all models is sound and complete

function TT-Entails?(KB, α) returns true or false
    inputs: KB, the knowledge base, a sentence in propositional logic
             α, the query, a sentence in propositional logic
    symbols ← a list of the proposition symbols in KB and α
    return TT-Check-All(KB, α, symbols, [])

function TT-Check-All(KB, α, symbols, model) returns true or false
    if Empty?(symbols) then
        if PL-True?(KB, model) then return PL-True?(α, model)
        else return true
    else do
        P ← First(symbols); rest ← Rest(symbols)
        return TT-Check-All(KB, α, rest, Extend(P, true, model)) and
             TT-Check-All(KB, α, rest, Extend(P, false, model))

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

Proof methods divide into (roughly) two kinds:

Model checking
  a) truth table enumeration (always exponential in $n$)
  b) heuristic search in model space (sound but incomplete)
     e.g., min-conflicts-like hill-climbing algorithms

Application of inference rules
  – Legitimate (sound) generation of new sentences from old
  – Proof = a sequence of inference rule applications
     Can use inference rules as operators in a standard search alg.
  – Typically require translation of sentences into a normal form
Logical equivalence

Two sentences are logically equivalent iff true in same models:

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

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

A sentence is **valid** if it is true in **all** models,
e.g., \textit{True}, \ A \lor \neg A, \ \ A \Rightarrow A, \ \ (A \land (A \Rightarrow B)) \Rightarrow B

Validity is connected to inference via the **Deduction Theorem**:
\[ KB \models \alpha \text{ if and only if } (KB \Rightarrow \alpha) \text{ is valid} \]

A sentence is **satisfiable** if it is true in **some** model
e.g., \ A \lor B, \ \ C

A sentence is **unsatisfiable** if it is true in **no** models
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**
Propositional inference: Enumeration method

Let $\alpha = A \lor B$ and $KB = (A \lor C) \land (B \lor \neg C)$

Is it the case that $KB \models \alpha$?
Check all possible models—$\alpha$ must be true wherever $KB$ is true

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
<th>$A \lor C$</th>
<th>$B \lor \neg C$</th>
<th>$KB$</th>
<th>$\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

AIMA 3rd ed. - Chapter 7  49
Propositional inference: Enumeration method

Let $\alpha = A \lor B$ and $KB = (A \lor C) \land (B \lor \neg C)$

Is it the case that $KB \models \alpha$?

Check all possible models: $\alpha$ must be true wherever $KB$ is true

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$B$</td>
<td>$C$</td>
<td>$A \lor C$</td>
<td>$B \lor \neg C$</td>
<td>$KB$</td>
<td>$\alpha$</td>
</tr>
<tr>
<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>False</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>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<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>False</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</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>
</tr>
</tbody>
</table>

Note that this is equivalent to proving that the sentence

$$((A \lor C) \land (B \lor \neg C) \Rightarrow A \lor B$$ is a valid sentence!
Complexity of inference

Truth table method: $2^N$ rows of table for any proof involving $N$ symbols

Linear space requirements though using Depth-First search.
An inference procedure that generates only entailed sentences is called **sound**

An inference procedure that generates all sentences that can be entailed is called **complete**

Formally:

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

**Soundness:** $i$ is sound if

whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$

**Completeness:** $i$ is complete if

whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$
Rules of Inference

Instead of enumeration method, we can also use inference rules that we know are sound. We list these rules as:

\[ \frac{\alpha}{\beta} \]

**Notation:** Whenever a sentence in the KB matches the pattern above the line (\(\alpha\)), the inference rule can conclude the sentence below the line (\(\beta\)).
Application: Propositional inference in the Wumpus world

Assume the following situation:

<table>
<thead>
<tr>
<th></th>
<th>1,1</th>
<th>1,2</th>
<th>1,3</th>
<th>1,4</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>OK</td>
<td>A</td>
<td>W!</td>
<td></td>
</tr>
<tr>
<td>OK</td>
<td></td>
<td>S</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>2,1</th>
<th>2,2</th>
<th>2,3</th>
<th>2,4</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>B</td>
<td>OK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>V</td>
<td>OK</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>3,1</th>
<th>3,2</th>
<th>3,3</th>
<th>3,4</th>
</tr>
</thead>
<tbody>
<tr>
<td>P!</td>
<td>OK</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>4,1</th>
<th>4,2</th>
<th>4,3</th>
<th>4,4</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A = Agent  
B = Breeze  
G = Glitter, Gold  
OK = Safe square  
P = Pit  
S = Stench  
V = Visited  
W = Wumpus

How does the agent know where to go?
Complete Set of Inference rules for propositional logic

Modus Ponens (or implication elimination)

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

And-elimination (from a conjunction, you can infer any of the conjuncts)

\[
\alpha_1 \land \cdots \land \alpha_n \\
\downarrow \\
\alpha_i
\]

Unit resolution (from a disjunction, if one of the disjuncts are false, you can infer the other one is true)

\[
\alpha \lor \beta, \quad \neg \beta \\
\downarrow \\
\alpha
\]

Resolution (for CNF): complete for propositional logic

\[
\alpha \lor \beta, \quad \neg \beta \lor \gamma \\
\downarrow \\
\alpha \lor \gamma
\]
Normal forms

Other approaches to inference use syntactic operations on sentences which are often expressed in standardized forms:

**Horn Form (restricted)**

conjunction of Horn clauses (clauses with \( \leq 1 \) positive literal)

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

Often written as set of implications:

\(B \Rightarrow A\) and \((C \land D) \Rightarrow B\)

**Disjunctive Normal Form (DNF—universal)**

disjunction of conjunctions of literals

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

**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 requires the Conjunctive Normal Form.

Every sentence can be written in CNF.
Resolution

Conjunctive Normal Form (CNF—universal)

*conjunction of disjunctions of literals* clauses

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

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

\[
\ell_1 \lor \cdots \lor \ell_k, \quad m_1 \lor \cdots \lor m_n
\]

\[
\ell_1 \lor \cdots \lor \ell_{i-1} \lor \ell_{i+1} \lor \cdots \lor \ell_k \lor m_1 \lor \cdots \lor m_{j-1} \lor m_{j+1} \lor \cdots \lor m_n
\]

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

\[
P_{1,3} \lor P_{2,2}, \quad \neg P_{2,2}
\]

\[
P_{1,3}
\]

Resolution is sound and complete for propositional logic
Proof by contradiction, i.e., show $KB \land \neg \alpha$ unsatisfiable

function PL-Resolution($KB, \alpha$) returns true or false

\begin{itemize}
  \item clauses $\leftarrow$ the set of clauses in the CNF representation of $KB \land \neg \alpha$
  \item new $\leftarrow \{ \}$
  \item loop do
    \begin{itemize}
      \item for each $C_i, C_j$ in clauses do
        \begin{itemize}
          \item resolvents $\leftarrow$ PL-Resolve($C_i, C_j$)
          \item if resolvents contains the empty clause then return true
          \item new $\leftarrow$ new $\cup$ resolvents
        \end{itemize}
      \item if new $\subseteq$ clauses then return false
    \end{itemize}
  \item clauses $\leftarrow$ clauses $\cup$ new
\end{itemize}
Resolution example

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

Add the negation of the sentence you want to prove (e.g. $\neg \neg P_{1,2}$) and see if you reach a contradiction:
Resolution example

\[ KB = (B_{1,1} \iff (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1} \quad \alpha = \neg P_{1,2} \]
Conversion to CNF

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

1. Eliminate \( \iff \), replacing \( \alpha \iff \beta \) with \((\alpha \implies \beta) \land (\beta \implies \alpha)\).

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

2. Eliminate \( \implies \), replacing \( \alpha \implies \beta \) 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:

\[(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1})\]
Restricted Forms: Horn Form

The completeness of resolution makes it a very important inference method. In many practical situations however, the full power of resolution is not needed.

Horn Form (restricted): A KB is said to be in Horn Form if it is a conjunction of Horn clauses.

Horn clause =

◊ proposition symbol; or
◊ (conjunction of symbols) ⇒ symbol

Alternative definition: a Horn clause is a disjunction of literals of which at most one is positive

Example KB: \(C \land (B \Rightarrow A) \land (C \land D \land E \Rightarrow B)\)

Terminology: premise ⇒ conclusion
Restricted Forms: Horn Form

The Horn form seems very restricted, but many real databases can be written in Horn form. How?

◊ Definite clauses: with exactly one positive literal

e.g.: \((C \land D \Rightarrow B)\)

◊ Facts: a definite clause with no negative literals simply asserts a given proposition

e.g.: \(C\)

◊ Integrity constraints: a Horn clause with no positive literals

e.g.: \(\neg(W_{11} \land W_{22})\)
How can you convert each sentence into an implication?

\((W_{11} \land W_{22}) \Rightarrow False\)
Restricted Forms: Horn Form

**Modus Ponens** is complete for Horn KBs (not in general)

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

Can be used with **forward chaining** or **backward chaining**.
These algorithms are very natural and run in *linear* time in the size of the KB.

They are also **sound** (easy to show) and **complete** (slightly more complicated argument).
Forward chaining

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

$P \Rightarrow Q$
$L \land M \Rightarrow P$
$B \land L \Rightarrow M$
$A \land P \Rightarrow L$
$A \land B \Rightarrow L$
$A$
$B$
function PL-FC-ENTAILS?(KB, q) returns true or false

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 to be true

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
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example

AIMA 3rd ed. - Chapter 7  73
Forward chaining example
Forward chaining example
Forward chaining example

A B
L M P
Q

0 0 0 0 0
Backward chaining

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

The algorithm is essentially identical to the AND-OR-GRAPH search given in Fig. 4.11.
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example

A
M
L
Q
P
B

AIMA 3rd ed. - Chapter 7
Backward chaining example
Backward chaining example
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
Satisfiability Problem

Satisfiability problem (is this sentence satisfiable?) is at the core of many general problems (e.g. constraint satisfaction problems).

Satisfiability can be checked by enumerating the possible models until one is found that satisfies the sentence using depth-first enumeration of possible models.

However, determining the satisfiability of sentences in propositional logic was the first problem proved to be NP-complete.
Davis-Putnam (or DPLL) Algorithm

♦ Recursive, depth-first enumeration of possible models,

♦ similar to TT-ENTAILS but with important improvements such as early termination...

DPLL algorithm is one of the fastest satisfiability algorithms, despite the fact that it is very old. It can solve hardware verification problems with a million variables.
Local search algorithms

We can use local search algorithms for satisfiability problems, as well.

Since each clause in the sentence needs to be satisfied, evaluation function should count the number of unsatisfied clauses (e.g. min-conflict heuristic).

WALKSAT algorithm is one of the most effective algorithms.
WALKSAT algorithm

- In each iteration, pick an unsatisfied clause and picks a symbol to flip.
- Two ways to decide how to choose the symbol:
  - randomly select the symbol from the clause and flip it with probability $p$
  - pick one that minimizes the number of unsatisfied clauses in the new state
Local search algorithms

In underconstrained situations, solutions are easy to find.

Ex. A 3-CNF with 5 symbols and 5 clauses:

\[ (\neg D \lor \neg B \lor C') \land (B \lor \neg A \lor \neg C') \land (\neg C \lor \neg B \lor E) \]
\[ \land (E \lor \neg D \lor B) \land (B \lor E \lor \neg C') \]

How many solutions?

What makes it a hard satisfiability problem?

What if WALKSAT returns failure?
Local search algorithms

In underconstrained situations, solutions are easy to find.

Ex. A 3-CNF with 5 symbols and 5 clauses:

\[ (\neg D \lor \neg B \lor C) \land (B \lor \neg A \lor \neg C') \land (\neg C' \lor \neg B \lor E) \]
\[ \land (E \lor \neg D \lor B) \land (B \lor E \lor \neg C') \]

*How many solutions?*

16 of the 32 possible worlds are models for this sentence!

*What makes it a hard satisfiability problem?*

How constrained it is (w.r.t number of constraints and number of variables)

*What if WALKSAT returns failure?*

We cannot tell if the sentence is unsatisfiable or just WALKSAT needs more time.
Shortcomings of PL

Lack of Expressive Efficiency:

We need to state the general rule about breeze and pits as a separate sentence for each square:

\[ B_{11} \Rightarrow P_{21} \lor P_{12} \]

\[ B_{21} \Rightarrow P_{11} \lor P_{22} \lor P_{31} \]
Shortcomings of PL - Fluents

**Fluents:** changing aspects of the world

E.g. Keeping track of location and orientation.

Let $L_{ij}$ be a location proposition. Then we would like to say something like:

$$L_{11} \land FacingRight \land Forward \implies L_{21}$$

But then we will have both $L_{11}$ and $L_{21}$ in the database!
Shortcomings of PL - Fluents

Solution: Associate propositions with time steps:

\[ L_0^{11}, FacingEast^0, HaveArrow^0 \ldots \]

After associating time with prepositions, we need axioms to keep track of these fluents:

\[ L_{xy}^t \Rightarrow (Breeze^t \iff Breeze_{xy}) \]

Effect axioms:

\[ L_0^{11} \land FacingEast^0 \land Forward^0 \Rightarrow (L_1^{21} \land \neg L_1^{11}) \]

We would need one such sentence for each possible time step and each possible square and orientation!
We still have other problems. The effect axioms fail to state what is *unchanged* as the result of an action.

E.g. That the agent still has an arrow.

One possible solution is to add **frame axioms** explicitly asserting all the propositions that remain the same:

\[
Forward^t \Rightarrow (HaveArrow^t \iff HaveArrow^{t+1})
\]

\[
Forward^t \Rightarrow (WumpusAlive^t \iff WumpusAlive^{t+1})
\]

...
Shortcomings of Propositional Logic - Representational Frame Problem

The proliferation of frame axioms is very inefficient and this problem was a significant problem for AI researchers.

**Solution:** Focusing from writing axioms about actions to writing axioms about fluents (**successor-state axioms**): \( HaveArrow^{t+1} \Leftrightarrow (HaveArrow^t \land \neg Shoot^t) \)

\[
L_{11}^{t+1} \Leftrightarrow (L_{11}^t \land (\neg Forward^t \lor Bump^{t+1})) \\
\lor (L_{12}^t \land (\neg FacingSouth^t \land Forward^t)) \\
\lor (L_{21}^t \land (\neg FacingWest^t \land Forward^t)).
\]
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. Propositional logic suffices for some of these tasks.

Truth table method is sound and complete for propositional logic.

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

Propositional logic lacks expressive power