Lecture 7: Logical Agents and Propositional Logic
CS 580 (001) - Spring 2018

Amarda Shehu

Department of Computer Science
George Mason University, Fairfax, VA, USA

March 07, 2018
1. Outline of Today’s Class

2. Knowledge-based Agents

3. Wumpus World

4. Logic - Models and Entailment

5. Propositional (Boolean Logic)

6. Model Checking: Inference by Enumeration

7. Deductive Systems: Inference and Theorem Proving
   - Proof by Resolution
   - Forward Chaining
   - Backward Chaining

8. Inference-based Agent in Wumpus World
Knowledge base = set of sentences in a **formal** language

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

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

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

Or at the implementation level
i.e., data structures in KB and algorithms that manipulate them
A Simple Knowledge-based Agent

function KB-Agent (percept) returns an action

static: KB, a knowledge base
         t, a counter, initially 0, indicating time

Tell(KB, Make-Percept-Sentence(percept, t))
action ← Ask(KB, Make-Action-Query(t))
Tell(KB, Make-Action-Sentence(action, t))
t ← t + 1
return action

The agent must be able to:
- Represent states, actions, etc.
- Incorporate new percepts
- Update internal representations of the world
- Deduce hidden properties of the world
- Deduce appropriate actions
Performance measure
  gold +1000, death -1000
  -1 per step, -10 for using the arrow

Environment
  Squares adjacent to wumpus are smelly
  Squares adjacent to pit are breezy
  Glitter iff gold is in the same square
  Shooting kills wumpus if you are facing it
  Shooting uses up the only arrow
  Grabbing picks up gold if in same square
  Releasing drops the gold in same square

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

Sensors Breeze, Glitter, Smell
Wumpus World Characterization

**Observable**

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

Deterministic?? Yes—outcomes exactly specified
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
Wumpus World Characterization

Observable?? No—only local perception

Deterministic?? Yes—outcomes exactly specified

Episodic?? No—sequential at the level of actions

Static??
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
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?
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
**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??**
<table>
<thead>
<tr>
<th>Characterization</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>
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 diagram of a Wumpus World grid with labels and symbols indicating different states of the environment.]
Exploring a Wumpus World

[Diagram of a Wumpus World grid with labeled squares and arrows indicating movement and perceptions.]
Exploring a Wumpus World
Exploring a Wumpus World
Exploring a Wumpus World
Exploring a Wumpus World
Exploring a Wumpus World
Other Tight Spots

Breeze in (1,2) and (2,1)  
⇒  no safe actions

Assuming pits uniformly distributed, (2,2) has pit w/ higher probability (how much?)
Other Tight Spots

Breeze in (1,2) and (2,1) \implies no safe actions

Assuming pits uniformly distributed, (2,2) has pit w/ higher probability (how much?)

Smell in (1,1) \implies cannot move

Can use a strategy of coercion:
- shoot straight ahead
- wumpus was there \implies dead \implies safe
- wumpus wasn’t there \implies safe
Other Tight Spots

Breeze in (1,2) and (2,1) ⇒ no safe actions

Assuming pits uniformly distributed, (2,2) has pit w/ higher probability (how much?)

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
**Logic in General**

**Logics** are formal languages for representing information such that conclusions can be drawn.

**Syntax** determines how sentences are expressed in a particular logic/language.

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

E.g., the language of arithmetic:

- $x + 2 \geq y$ is a sentence; $x^2 + y >$ 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 $x = 0, y = 6$. 
## Types of Logic

Logics are characterized by what they commit to as primitives.

**Ontological commitment:** what exists—facts? objects? time? beliefs?

**Epistemological commitment:** what states of knowledge?

<table>
<thead>
<tr>
<th>Language</th>
<th>Ontological Commitment</th>
<th>Epistemological Commitment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propositional logic</td>
<td>facts</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>First-order logic</td>
<td>facts, objects, relations</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>Temporal logic</td>
<td>facts, objects, relations, times</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>Probability theory</td>
<td>facts</td>
<td>degree of belief</td>
</tr>
<tr>
<td>Fuzzy logic</td>
<td>facts + degree of truth</td>
<td>known interval value</td>
</tr>
</tbody>
</table>
First order of business: fundamental concepts of logical representation and reasoning
  independent of any logic’s particular form/type
  **Entailment**

Second order of business: Introduction to **propositional logic**
  Wumpus KB via propositional logic

Third order of business: Drawing conclusions
  **Inference** and theorem proving
Can use the term **model** in place of possible world

Logicians typically think in terms of **models**, which are formally-structured worlds with respect to which truth can be evaluated

Model = mathematical abstraction that fixes the truth/falsehood of every relevant sentence

Possible models are just all possible assignments of variables in the environment

We say that a model $m$ “satisfies” sentence $\alpha$ if $\alpha$ “is true in” $m$

Or: “$m$ is a model of $\alpha$”

$M(\alpha)$ is the set of all models of $\alpha$
Entailment means that one thing follows from another:

\[ KB \models \alpha \]

Knowledge base \( KB \) entails sentence \( \alpha \)
iff \( \alpha \) is true in all worlds/models where \( KB \) is true
\[ KB \models \alpha \text{ iff } M(KB) \subseteq M(\alpha) \]

E.g., KB containing “Giants won” and “Reds won” entails “Giants or Reds won”
\[ x + y = 4 \text{ entails } 4 = x + y \]

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

Note: brains process syntax (of some sort)
Quick Exercise

Given two sentences $\alpha$ and $\beta$, what does this mean:

$$\alpha \models \beta$$

$\alpha$ entails $\beta$
Quick Exercise

Given two sentences $\alpha$ and $\beta$, what does this mean:

$$ \alpha \models \beta $$

$\alpha$ entails $\beta$

$$ M(\alpha) \subseteq M(\beta) $$
Quick Exercise

Given two sentences $\alpha$ and $\beta$, what does this mean:

$$\alpha \models \beta$$

$\alpha$ entails $\beta$

$$M(\alpha) \subseteq M(\beta)$$

$\beta$ is satisfied in all models of $\alpha$
Quick Exercise

Given two sentences $\alpha$ and $\beta$, what does this mean:

$$\alpha \models \beta$$

$\alpha$ entails $\beta$

$$M(\alpha) \subseteq M(\beta)$$

$\beta$ is satisfied in all models of $\alpha$

$\beta$ may be satisfied in other models, as well
Given two sentences $\alpha$ and $\beta$, what does this mean:

$$\alpha \models \beta$$

$\alpha$ entails $\beta$

$M(\alpha) \subseteq M(\beta)$

$\beta$ is satisfied in all models of $\alpha$

$\beta$ may be satisfied in other models, as well

$\alpha$ is a stronger assertion than $\beta$
Given two sentences $\alpha$ and $\beta$, what does this mean:

$$\alpha \models \beta$$

$\alpha$ entails $\beta$

$M(\alpha) \subseteq M(\beta)$

$\beta$ is satisfied in all models of $\alpha$

$\beta$ may be satisfied in other models, as well

$\alpha$ is a stronger assertion than $\beta$
Hands On: Entailment in the Wumpus World

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

Consider possible models for ?s assuming only pits

3 Boolean choices $\implies$ 8 possible models
Wumpus Models
$KB = \text{wumpus-world rules} + \text{observations}$
$KB = \text{wumpus-world rules} + \text{observations}$

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

$\alpha_2 = \text{"[2,2] is safe"}, \ KB \not\models \alpha_2$
Entailment can be used to derive logical conclusions
  i.e.: carry out logical inference

A straightforward algorithm to carry out inference:
   Model checking

Model checking enumerates all possible models to check that $\alpha$ is true in all models
where $KB$ is true
  i.e.: $M(KB) \subseteq M(\alpha)$

To understand entailment and inference: haystack and needle analogy
  Consequences of $KB$ are a haystack; $\alpha$ is a needle.
  Entailment = needle in haystack
  inference = finding it

We need inference procedures to derive $\alpha$ from a given $KB$
$KB \vdash_i \alpha = $ sentence $\alpha$ can be derived from $KB$ by procedure $i$

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

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

**Soundness**: inference procedure $i$ is sound if whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$ (does not make stuff up)

**Completeness**: inference procedure $i$ is complete if whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$ (finds needle in haystack)

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 $KB$. Right now, we will venture into propositional logic; first-order logic is next.
Inference

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

**Soundness:** inference procedure $i$ is sound if whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$ (does not make stuff up)

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

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

**Soundness:** inference procedure $i$ is sound if
whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$ (does not make stuff up)

**Completeness:** inference procedure $i$ is complete if
whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$ (finds needle in haystack)
Inference

\[ KB \models_i \alpha = \text{sentence } \alpha \text{ can be derived from } KB \text{ by procedure } i \]

**Soundness**: inference procedure \( i \) is sound if whenever \( KB \models_i \alpha \), it is also true that \( KB \models \alpha \) \hspace{1cm} \text{(does not make stuff up)}

**Completeness**: inference procedure \( i \) is complete if whenever \( KB \models \alpha \), it is also true that \( KB \models_i \alpha \) \hspace{1cm} \text{(finds needle in haystack)}

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

**Soundness:** inference procedure \( i \) is sound if whenever \( KB \vdash_i \alpha \), it is also true that \( KB \models \alpha \) (does not make stuff up)

**Completeness:** inference procedure \( i \) is complete if whenever \( KB \models \alpha \), it is also true that \( KB \vdash_i \alpha \) (finds needle in haystack)

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 \( KB \).
Inference

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

**Soundness:** inference procedure \( i \) is sound if
whenever \( KB \vdash_i \alpha \), it is also true that \( KB \Vdash \alpha \)  
(does not make stuff up)

**Completeness:** inference procedure \( i \) is complete if
whenever \( KB \Vdash \alpha \), it is also true that \( KB \vdash_i \alpha \)  
(find needle in haystack)

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 \( KB \).

Right now, we will venture into propositional logic; first-order logic is next.
Inference

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

**Soundness:** inference procedure \( i \) is sound if

whenever \( KB \vdash_i \alpha \), it is also true that \( KB \models \alpha \) \hspace{1cm} (does not make stuff up)

**Completeness:** inference procedure \( i \) is complete if

whenever \( KB \models \alpha \), it is also true that \( KB \vdash_i \alpha \) \hspace{1cm} (finds needle in haystack)

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 \( KB \).

Right now, we will venture into propositional logic; first-order logic is next.
Propositional Logic: Syntax

Propositional logic is the simplest logic—illustrates basic ideas

Atomic sentences consist of a single proposition symbol
  E.g.: Proposition symbols $P_1$, $P_2$, etc. are atomic sentences

Each such symbol stands for a proposition that can be true or false
  E.g.: $W_{1,3}$ stands for proposition that wumpus is in [1,3]

Two propositions with fixed meaning: True and False

Complex sentences built over atomic ones via connectives:
  negation, conjunction, disjunction, implication, biconditional
If $S$ is a sentence, $\neg S$ is a sentence (negation)
  A (positive) literal is an atomic sentence
  A (negative) literal is a negated atomic sentence

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

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

If $S_1$ and $S_2$ are sentences, $S_1 \implies S_2$ is a sentence (implication/conditional)
  $S_1$ is called premise/antecedent
  $S_2$ is called conclusion or consequent

  implication also known as rule or if-then statement

If $S_1$ and $S_2$ are sentences, $S_1 \iff S_2$ is a sentence (biconditional)
BNF is an ambiguous formal grammar for propositional logic (pg. 1060 if unfamiliar):

Sentence $\rightarrow$ AtomicSentence $|$ ComplexSentence

AtomicSentence $\rightarrow$ True $|$ False $|$ $P$ $|$ $Q$ $|$ ... 

Complex Sentence $\rightarrow$ (Sentence) $|$ [Sentence] $|$ $\neg$Sentence $|$ Sentence $\land$ Sentence $|$ Sentence $\Rightarrow$ Sentence $|$ Sentence $\Leftrightarrow$ Sentence

We add operator precedence to disambiguate it

Operator precedence (from highest to lowest)
$\neg$, $\land$, $\lor$, $\Rightarrow$, $\Leftrightarrow$
Propositional Logic: Semantics

Each model specifies true/false for each proposition symbol

E.g. \( P_{1,2} P_{2,2} P_{3,1} \)
\[
\begin{array}{lll}
\text{true} & \text{true} & \text{false}
\end{array}
\]

This specific model: \( m_1 = \{ P_{1,2} = \text{true}, P_{2,2} = \text{true}, P_{3,1} = \text{false} \} \)

(With these 3 symbols, \( 2^3 = 8 \) possible models, feasible to enumerate.)

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

- \( \neg S \) is true iff \( S \) is false
- \( S_1 \land S_2 \) is true iff \( S_1 \) is true and \( S_2 \) is true
- \( S_1 \lor S_2 \) is true iff \( S_1 \) is true or \( S_2 \) is true
- \( S_1 \implies S_2 \) is true iff \( S_1 \) is false or \( S_2 \) is true
  - i.e., is false iff \( S_1 \) is true and \( S_2 \) is false
- \( S_1 \iff S_2 \) is true iff \( S_1 \implies S_2 \) is true and \( S_2 \implies S_1 \) is true

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

\[
\neg P_{1,2} \land ( P_{2,2} \lor P_{3,1} ) = \text{true} \land ( \text{false} \lor \text{true} ) = \text{true} \land \text{true} = \text{true}
\]
### 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>$P \equiv Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</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>false</td>
<td>false</td>
</tr>
</tbody>
</table>

*Note: The truth values are represented in green and red colors.*
Let $P_{i,j}$ be true if there is a pit in $[i,j]$  
Let $B_{i,j}$ be true if agent is in $[i,j]$ and perceives a breeze  
Let $W_{i,j}$ be true if there is a wumpus in $[i,j]$  
Let $S_{i,j}$ be true if agent is in $[i,j]$ and perceives a stench  
... you can define other atomic sentences

Percept sentences part of $KB$:
No pit, no breeze in $[1,1]$, but breeze perceived when in $[2,1]$

$$R_1 : \neg P_{1,1}$$  
$$R_4 : \neg B_{1,1}$$  
$$R_5 : B_{2,1}$$

Rules in $KB$:
“Pits cause breezes in adjacent squares” eqv. to “square is breezy iff adjacent pit”
Let $P_{i,j}$ be true if there is a pit in $[i,j]$
Let $B_{i,j}$ be true if agent is in $[i,j]$ and perceives a breeze
Let $W_{i,j}$ be true if there is a wumpus in $[i,j]$
Let $S_{i,j}$ be true if agent is in $[i,j]$ and perceives a stench
... you can define other atomic sentences

Percept sentences part of $KB$:
No pit, no breeze in $[1,1]$, but breeze perceived when in $[2,1]$

\[ R_1 : \neg P_{1,1} \]
\[ R_4 : \neg B_{1,1} \]
\[ R_5 : B_{2,1} \]

Rules in $KB$:
“Pits cause breezes in adjacent squares” eqv. to “square is breezy iff adjacent pit”

\[ R_2 : B_{1,1} \iff (P_{1,2} \lor P_{2,1}) \]
\[ R_3 : B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1}) \]
Truth Tables for Inference

| \( B_{1,1} \) | \( B_{2,1} \) | \( P_{1,1} \) | \( P_{1,2} \) | \( P_{2,1} \) | \( P_{2,2} \) | \( P_{3,1} \) | \( R_1 \) | \( R_2 \) | \( R_3 \) | \( R_4 \) | \( R_5 \) | \( KB \) |
|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| false       | false       | false       | false       | false       | false       | false       | true        | true        | true        | true        | false        | true        | false       |
| true        | false       | true        | false       | false       | false       | false       | true        | true        | true        | true        | false        | true        | false       |
| true        | true        | true        | false       | false       | false       | false       | true        | true        | true        | true        | false        | true        | true        |
| false       | true        | false       | false       | false       | false       | true        | true        | true        | true        | true        | false        | true        | false       |

Enumerate rows (different assignments to symbols); rows are possible models if \( KB \) is true in a row/model, check that \( \alpha \) is true; if not, entailment does not hold. If entailment not broken over all rows where \( KB \) is true, then else, \( \alpha \) entailed.
Depth-first enumeration of all models is sound and complete

```plaintext
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 divide into (roughly) two kinds:

**Model checking**
- truth table enumeration (always exponential in $n$)
- improved backtracking, e.g., Davis–Putnam–Logemann–Loveland
- backtracking with constraint propagation, backjumping
- heuristic search in model space (sound but incomplete)
  - e.g., min-conflicts-like hill-climbing algorithms

**Theorem Proving/Deductive Systems:** 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:
\[ \alpha \equiv \beta \text{ if and only if } \alpha \models \beta \text{ and } \beta \models \alpha \]

\[ (\alpha \land \beta) \equiv (\beta \land \alpha) \text{ commutativity of } \land \]
\[ (\alpha \lor \beta) \equiv (\beta \lor \alpha) \text{ commutativity of } \lor \]
\[ ((\alpha \land \beta) \land \gamma) \equiv (\alpha \land (\beta \land \gamma)) \text{ associativity of } \land \]
\[ ((\alpha \lor \beta) \lor \gamma) \equiv (\alpha \lor (\beta \lor \gamma)) \text{ associativity of } \lor \]
\[ \neg(\neg \alpha) \equiv \alpha \text{ double-negation elimination} \]
\[ (\alpha \implies \beta) \equiv (\neg \beta \implies \neg \alpha) \text{ contraposition} \]
\[ (\alpha \implies \beta) \equiv (\neg \alpha \lor \beta) \text{ implication elimination} \]
\[ (\alpha \equiv \beta) \equiv ((\alpha \implies \beta) \land (\beta \implies \alpha)) \text{ biconditional elimination} \]
\[ \neg(\alpha \land \beta) \equiv (\neg \alpha \lor \neg \beta) \text{ De Morgan} \]
\[ \neg(\alpha \lor \beta) \equiv (\neg \alpha \land \neg \beta) \text{ De Morgan} \]
\[ (\alpha \land (\beta \lor \gamma)) \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)) \text{ distributivity of } \land \text{ over } \lor \]
\[ (\alpha \lor (\beta \land \gamma)) \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) \text{ distributivity of } \lor \text{ over } \land \]
Validity and Satisfiability

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

Validity is connected to inference via the Deduction Theorem:
$KB \models \alpha$ if and only if $(KB \implies \alpha)$ 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:
$KB \models \alpha$ if and only if $(KB \land \neg \alpha)$ is unsatisfiable
i.e., prove $\alpha$ by reductio ad absurdum
Deductive Systems: Rules of Inference

- **Modus Ponens or Implication-Elimination**: (From an implication and the premise of the implication, you can infer the conclusion.)
  \[
  \alpha \Rightarrow \beta, \quad \alpha \quad \quad \Rightarrow \beta
  \]

- **And-Elimination**: (From a conjunction, you can infer any of the conjuncts.)
  \[
  \alpha_1 \land \alpha_2 \land \ldots \land \alpha_n \quad \Rightarrow \alpha_i
  \]

- **And-Introduction**: (From a list of sentences, you can infer their conjunction.)
  \[
  \alpha_1, \alpha_2, \ldots, \alpha_n \quad \Rightarrow \alpha_1 \land \alpha_2 \land \ldots \land \alpha_n
  \]

- **Or-Introduction**: (From a sentence, you can infer its disjunction with anything else at all.)
  \[
  \alpha_i \quad \Rightarrow \alpha_1 \lor \alpha_1 \lor \ldots \lor \alpha_n
  \]

- **Double-Negation Elimination**: (From a doubly negated sentence, you can infer a positive sentence.)
  \[
  \neg\neg\alpha \quad \Rightarrow \alpha
  \]

- **Unit Resolution**: (From a disjunction, if one of the disjuncts is false, then you can infer the other one is true.)
  \[
  \alpha \lor \beta, \quad \neg\beta \quad \Rightarrow \alpha
  \]

- **Resolution**: (This is the most difficult. Because \( \beta \) cannot be both true and false, one of the other disjuncts must be true in one of the premises. Or equivalently, implication is transitive.)
  \[
  \alpha \lor \beta, \quad \neg\beta \lor \gamma \quad \text{or equivalently} \quad \neg\alpha \Rightarrow \beta, \quad \beta \Rightarrow \gamma
  \]

*Figure 6.13* Seven inference rules for propositional logic. The unit resolution rule is a special case of the resolution rule, which in turn is a special case of the full resolution rule for first-order logic discussed in Chapter 9.
Inference by Resolution

Conjunctive Normal Form (CNF—universal)

conjunction of disjunctions of literals

conjunction of disjuncticlauses

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
Conversion to CNF

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

1. Eliminate \( \Leftrightarrow \), replacing \( \alpha \Leftrightarrow \beta \) 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:

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

Proof by contradiction/refutation, i.e., show $KB \land \neg \alpha$ unsatisfiable

function $\text{PL-Resolution}(KB, \alpha)$ returns true or false
inputs: $KB$, the knowledge base, a sentence in propositional logic  
$\alpha$, the query, a sentence in propositional logic

$\text{clauses} \leftarrow$ the set of clauses in the CNF representation of $KB \land \neg \alpha$

$new \leftarrow \emptyset$
loop do
  for each $C_i, C_j$ in $\text{clauses}$ do
    $\text{resolvents} \leftarrow \text{PL-Resolve}(C_i, C_j)$
    if $\text{resolvents}$ contains the empty clause then return true
    $new \leftarrow new \cup \text{resolvents}$
    if $new \subseteq \text{clauses}$ then return false
  $\text{clauses} \leftarrow \text{clauses} \cup new$
loop end

Can actually use any search algorithm, with clauses as states and resolution as operators. Goal state is list of clauses containing empty clause.
\[ KB = (B_{1,1} \iff (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1} \alpha = \neg P_{1,2} \]

Completeness of resolution algorithm follows from ground resolution theorem: If a set of clauses \( S \) is unsatisfiable, then the resolution closure \( RC(S) \) of those clauses contains an empty clause.

\( RC(S) \): set of all clauses derivable by repeated application of resolution rule to clauses in \( S \) or their derivatives.
Inference by resolution is complete, but sometimes an overkill

KB may contain restricted (rule-based) forms of sentences, such as:

**Definite clause:** disjunction of literals of which *exactly one* is positive.

\[(\neg L_{1,1} \lor B_{1,1})\] is
\[(P_{1,2} \lor P_{2,1})\] is not
\[(\neg L_{1,1} \lor \neg B_{1,1})\] is not

**Horn clause:** disjunction of literals of which *at most one* is positive.

Which of the above is a Horn clause?

Negated literals \(\neg A\) rewritten as \((A \implies False)\) (integrity constraints)

Inference with Horn clauses can be done through forward chaining and backward chaining

These are more efficient than the resolution algorithm, run in linear time
Definite Clauses and Horn Clauses

Inference by resolution is complete, but sometimes an overkill

KB may contain restricted (rule-based) forms of sentences, such as:

Definite clause: disjunction of literals of which \textit{exactly one} is positive.
\[
(\neg L_{1,1} \lor B_{1,1}) \text{ is not } (P_{1,2} \lor P_{2,1}) \text{ is not } (\neg L_{1,1} \lor \neg B_{1,1}) \text{ is not}
\]

Horn clause: disjunction of literals of which \textit{at most one} is positive.
Which of the above is a Horn clause?

Negated literals \( \neg A \) rewritten as \( A \implies False \) (integrity constraints)

Inference with Horn clauses can be done through forward chaining and backward chaining

These are more efficient than the resolution algorithm, run in linear time
Horn Form (restricted) KB (= conjunction of Horn clauses)
E.g., $C \land (B \implies A) \land (C \land D \implies B)$

Modus Ponens: complete for Horn KBs ($\alpha_1, \ldots, \alpha_n$ - premises, $\beta$ - sought conclusion)

\[
\frac{\alpha_1, \ldots, \alpha_n, \quad \alpha_1 \land \cdots \land \alpha_n \implies \beta}{\beta}
\]

Known as forward chaining inference rule; repeated applications until sentence of interest obtained – forward chaining algorithm

Modus Tollens - a form of Modus Ponens

\[
\frac{\neg \beta, \quad \alpha_1 \land \cdots \land \alpha_n \implies \beta}{\neg(\alpha_1 \land \cdots \land \alpha_n)}
\]

Known as backward chaining inference rule; repeated applications until all premises obtained – backward chaining algorithm

Both algorithms intuitive and run in linear time

Inference via forward or backward chaining forms basis of logic programming (Chapter 9)
**Forward Chaining**

**Idea:** Add literals in KB to facts (satisfied premises)  
apply each premise satisfied in KB (fire rules)  
add rule’s conclusion as new fact/premise to the KB  
(this is inference propagation via forward chaining)  
stop when query found as fact or no more inferences

\[
P \implies Q
\]

\[
L \land M \implies P
\]

\[
B \land L \implies M
\]

\[
A \land P \implies L
\]

\[
A \land B \implies L
\]

\[
A
\]

\[
B
\]

Figure: AND-OR tree
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, table indexed by clause, initial nr. of premises
                inferred, table indexed by symbol, entries initially false
                agenda, list of symbols, initial 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
Forward Chaining Example
Forward Chaining Example
Forward Chaining Example

Diagram of a deductive system with nodes and edges labeled with numbers.
Forward Chaining Example
Forward Chaining Example
Forward Chaining Example
Forward Chaining Example
Forward Chaining Example
Proof of Completeness

FC derives every atomic sentence that is entailed by $KB$
1. FC reaches a fixed point where no new atomic sentences are derived
2. Consider the final state as a model $m$, assigning true/false to symbols
3. Every clause in the original $KB$ is true in $m$

**Proof:** Suppose a clause $a_1 \land \ldots \land a_k \Rightarrow b$ is false in $m$

Then $a_1 \land \ldots \land a_k$ is true in $m$ and $b$ is false in $m$

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

**General idea:** construct any model of $KB$ by sound inference, check $\alpha$

FC is an example of a data-driven reasoning algorithm
start with what known, derive new conclusions, with no particular goal in mind
**Idea:** goal-driven reasoning – 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
Backward Chaining Example
Backward Chaining Example
Backward Chaining Example
Backward Chaining Example
Backward Chaining Example

Deductive Systems: Inference and Theorem Proving
Backward Chaining Example
Backward Chaining Example

[Diagram of a logic tree with nodes Q, P, M, L, A, B, and arrows showing the relationships between them.]
Backward Chaining Example
Backward Chaining Example
Backward Chaining Example
Forward versus 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, because only relevant facts are touched
A wumpus-world agent using propositional logic:

\[ \neg P_{1,1} \]
\[ \neg W_{1,1} \]
\[ B_{x,y} \iff (P_{x,y+1} \lor P_{x,y-1} \lor P_{x+1,y} \lor P_{x-1,y}) \]
\[ S_{x,y} \iff (W_{x,y+1} \lor W_{x,y-1} \lor W_{x+1,y} \lor W_{x-1,y}) \]
\[ W_{1,1} \lor W_{1,2} \lor \ldots \lor W_{4,4} \]
\[ \neg W_{1,1} \lor \neg W_{1,2} \]
\[ \neg W_{1,1} \lor \neg W_{1,3} \]
\[ \ldots \]

64 distinct proposition symbols, 155 sentences
function PL-WUMPUSS-AGENT( percept) returns an action
inputs: percept, a list, [stench, breeze, glitter]
static: KB, initially containing the “physics” of the wumpus world
          x, y, orientation, the agent’s position (init. [1,1]) and orient. (init. right)
          visited, an array indicating which squares have been visited, initially false
          action, the agent’s most recent action, initially null
          plan, an action sequence, initially empty

update x, y, orientation, visited based on action
if stench then TELL(KB, Sx,y) else TELL(KB, ¬ Sx,y)
if breeze then TELL(KB, Bx,y) else TELL(KB, ¬ Bx,y)
if glitter then action ← grab
else if plan is nonempty then action ← POP(plan)
else if for some fringe square [i,j], ASK(KB, (¬ Pi,j ∨ ¬ Wi,j)) is true or
        for some fringe square [i,j], ASK(KB, (Pi,j ∨ Wi,j)) is false then do
        plan ← A*-GRAPH-SEARCH(Route-PB([x,y], orientation, [i,j], visited))
        action ← POP(plan)
else action ← a randomly chosen move
return action
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**: 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.
Forward, backward chaining are linear-time, complete for Horn clauses.

Resolution is complete for propositional logic.

Propositional logic does not scale to environments of unbounded size, as it lacks expressive power to deal concisely with time, space, and universal patterns of relationships among objects.