Lecture #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
  - forward chaining
  - backward chaining
  - resolution

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

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
  - Shooting uses up the only arrow
  - Grabbing picks up gold if in same square
  - Releasing drops the gold in same square
- Sensors: Stench, Breeze, Glitter, Bump, Scream
- Actuators: Left turn, Right turn, Forward, Grab, Release, Shoot

Wumpus World Characterization

- Fully 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

Exploring a Wumpus World (II)

Exploring a Wumpus World (III)

Exploring a Wumpus World (IV)

Exploring a Wumpus World (V)

Exploring a Wumpus World (VI)
Logic in general
- Logics are formal languages for representing information such that conclusions can be drawn
- Syntax defines the sentences in the language
- Semantics define the "meaning" of sentences;
  - i.e., define truth of a sentence in a world
- E.g., the language of arithmetic
  - \( x + 2 \geq y \) 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 \( x = 0, y = 6 \)

Entailment
- Entailment 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 “the Giants won” and “the Reds won” entails “Either the Giants won or the Reds won”
  - E.g., \( x + y = 4 \) entails \( 4 = x + y \)
  - Entailment is a relationship between sentences (i.e., syntax) that is based on semantics

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 \( m \)
- \( M(\alpha) \) is the set of all models of \( \alpha \)
- Then \( KB \models \alpha \) iff \( M(KB) \subseteq M(\alpha) \)
  - E.g. \( KB = \) Giants won and Reds won \( \alpha = \) Giants won

Entailment in the Wumpus World
Situation after detecting nothing in [1,1], moving right, breeze in [2,1]
Consider possible models for \( KB \) assuming only pits
3 Boolean choices \( \Rightarrow 8 \) possible models
Wumpus Models

- \( KB = \) wumpus-world rules + observations
- \( \alpha_1 = \) "[1,2] is safe", \( KB \vdash \alpha_1 \), proved by model checking

Wumpus Models (II)

- \( KB = \) wumpus-world rules + observation

Wumpus Models (III)

- \( KB = \) wumpus-world rules + observations
- \( \alpha_1 = \) "[1,2] is safe", \( KB \vdash \alpha_1 \), proved by model checking

Wumpus Models (IV)

- \( KB = \) wumpus-world rules + observations

Wumpus Models (V)

- \( KB = \) wumpus-world rules + observations
- \( \alpha_2 = \) "[2,2] is safe", \( KB \nvdash \alpha_2 \)

Inference

- \( KB \vdash \alpha = \) sentence \( \alpha \) can be derived from \( KB \) by procedure \( i \)
- Soundness: \( i \) is sound if whenever \( KB \vdash \alpha \), it is also true that \( KB \models \alpha \)
- Completeness: \( i \) is complete if whenever \( KB \models \alpha \), it is also true that \( KB \vdash \alpha \)
- 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 \( KB \).
Propositional logic: Syntax

- Propositional logic is the simplest logic – illustrates basic ideas
- The proposition symbols P₁, P₂ etc are sentences
  - If S is a sentence, then ~S is a sentence (negation)
  - If S₁ and S₂ are sentences, S₁ ∧ S₂ is a sentence (conjunction)
  - If S₁ and S₂ are sentences, S₁ ∨ S₂ is a sentence (disjunction)
  - If S₁ and S₂ are sentences, S₁ ⇒ S₂ is a sentence (implication)
  - If S₁ and S₂ are sentences, S₁ ⇔ S₂ is a sentence (biconditional)

Propositional Logic: Semantics

Each model specifies true/false for each proposition symbol

E.g.  P₁,₂  P₂,₂  P₃,₁  false true false

With these symbols, 8 possible models, can be enumerated automatically.

Rules for evaluating truth with respect to a model m:
- ~S is true iff S is false
- S₁ ∧ S₂ is true iff S₁ is true and S₂ is true
- S₁ ∨ S₂ is true iff S₁ is true or S₂ is true
- S₁ ⇒ S₂ is true iff S₁ is false or S₂ is true
- S₁ ⇔ S₂ is true iff S₁ ⇒ S₂ is true and S₂ ⇒ S₁ is true

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

¬P₁,₂ ∧ (P₂,₂ ∨ P₃,₁) = true ∧ (true ∨ false) = true ∧ true = true

Truth tables for connectives

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>~P</th>
<th>P ∧ Q</th>
<th>P ∨ Q</th>
<th>P ⇒ Q</th>
<th>P ⇔ Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>false</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>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

Wumpus World Sentences

Let Pᵢ,j be true if there is a pit in [i, j].
Let Bᵢ,j be true if there is a breeze in [i, j].

¬P₁,₁ {no pit in start square}
¬B₁,₁ {no breeze detected in square 1,1}
B₂,₁ {breeze detected in square 2,1}

"Pits cause breezes in adjacent squares"

B₁,₁ ⇔ (P₁,₂ ∨ P₂,₁)
B₂,₁ ⇔ (P₁,₁ ∨ P₂,₂ ∨ P₃,₁)

Truth Tables for Inference

<table>
<thead>
<tr>
<th>B₁,₁</th>
<th>B₂,₁</th>
<th>P₁,₁</th>
<th>P₁,₂</th>
<th>P₂,₁</th>
<th>P₂,₂</th>
<th>P₃,₁</th>
<th>KB</th>
<th>α₁</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>true</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>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</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>false</td>
<td>false</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>false</td>
<td>false</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>false</td>
<td>false</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>false</td>
<td>false</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>false</td>
<td>false</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>false</td>
<td>false</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>false</td>
<td>false</td>
</tr>
</tbody>
</table>

Inference by Enumeration

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

```
function TT-EXTEND([KB, α]) returns true or false
    symbols ← a list of the propositional 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 model ∈ symbols then return TT-TRUE([KB, α, model])
    else return true
    function TT-TRUE([KB, α, model])
        for i from 1 to 1 do
            return false
        return true
```

```
for n symbols, time complexity is O(2^n), space complexity is O(n)
```
Logical equivalence

- Two sentences are logically equivalent} iff true in same models: \( \alpha \equiv \beta \) iff \( \alpha \Rightarrow \beta \) and \( \beta \Rightarrow \alpha \)
- \( (\alpha \land \beta) \equiv (\beta \land \alpha) \) commutativity of \( \land \)
- \( (\alpha \lor \beta) \equiv (\beta \lor \alpha) \) commutativity of \( \lor \)
- \( ((\alpha \land \beta) \land \gamma) \equiv (\alpha \land (\beta \land \gamma)) \) associativity of \( \land \)
- \( ((\alpha \lor \beta) \lor \gamma) \equiv (\alpha \lor (\beta \lor \gamma)) \) associativity of \( \lor \)
- \( \neg(\neg \alpha) \equiv \alpha \) double-negation elimination
- \( (\alpha \Rightarrow \beta) \equiv (\neg \beta \Rightarrow \neg \alpha) \) contraposition
- \( (\alpha \Rightarrow \beta) \equiv (\neg \alpha \lor \beta) \) implication elimination
- \( (\neg \alpha \land \neg \beta) \equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) \) biconditional elimination
- \( (\alpha \Rightarrow \beta) \equiv (\neg \alpha \lor \neg \beta) \) de Morgan
- \( (\alpha \lor (\beta \land \gamma)) \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) \) distributivity of \( \lor \) over \( \land \)

Validity and Satisfiability

A sentence is valid if it is true in all models.
- e.g., 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:

\[ KB \models \alpha \text{ if and only if } (KB \land \neg \alpha) \text{ is unsatisfiable} \]

Proof Methods

- Proof methods divide into (roughly) two kinds:
- 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 algorithm
  - Typically require transformation of sentences into a normal form
- Model checking
  - truth table enumeration (always exponential in \( n \))
  - improved backtracking, e.g., Davis–Putnam-Logemann-Loveland (DPLL)
  - heuristic search in model space (sound but incomplete)
    e.g., min-conflicts-like hill-climbing algorithms

Resolution

Conjunctive Normal Form (CNF)

- 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):

\[ \begin{align*}
\neg(l_1 \lor \ldots \lor l_{i-1} \lor l_{i+1} \lor \ldots \lor l_k) \Rightarrow m_1 \lor \ldots \lor m_n \Rightarrow (\neg m_1 \lor \ldots \lor m_{i-1} \lor m_{i+1} \lor \ldots \lor m_n)
\end{align*} \]

Resolution is sound and complete for propositional logic

Conversion to Conjunctive Normal Form

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

1. Eliminate \( \equiv \), replacing \( \alpha \equiv \beta \) with \( (\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha) \).
\[ (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1}) \]

2. Eliminate \( \Rightarrow \), replacing \( \alpha \Rightarrow \beta \) with \( \neg \alpha \lor \beta \).
\[ (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land ((P_{1,2} \lor P_{2,1}) \lor B_{1,1}) \]

3. Move \( \neg \) inwards using de Morgan's rules and double-negation:
\[ (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land ((\neg P_{1,2} \lor \neg P_{2,1}) \lor B_{1,1}) \]

4. Apply distributivity law (\( \land \) over \( \lor \)) 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, i.e., show \( KB \land \neg \alpha \) unsatisfiable

```plaintext
Function PL-RESOLUTION(KB, \alpha) returns true or false

clauses ← the set of clauses in the CNF representation of \( KB \land \neg \alpha \)
new ← \{
loop do
  for each \( C_i, C_j \) in clauses do
    resolvent ← PL-RESOLUTION(C_i, C_j)
    if resolvent contains the empty clause then return true
    new ← new \cup resolvent
  if new \subseteq clauses then return false
  clauses ← clauses \cup new
```

Resolution Example

- \( KB = (B_{1,1} \iff (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1} \land \neg P_{1,2} \)

Forward and Backward Chaining

- Horn Form (restricted)
  - \( KB = \) conjunction of Horn clauses
  - Horn clause =
    - proposition symbol; or
    - (conjunction of symbols) \( \Rightarrow \) symbol
  - E.g., \( C \land (B \Rightarrow A) \land (C \land D \Rightarrow B) \)
- Modus Ponens (for Horn Form): complete for Horn KBs

\[
\alpha_1 \land \ldots \land \alpha_n \land \beta \Rightarrow \beta
\]

- Can be used with forward chaining or backward chaining.
- These algorithms are very natural and run in linear time.

Forward Chaining

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

\[
P \Rightarrow Q \land \neg Q \land \neg P
\]

Rules

- \( P \Rightarrow Q \)
- \( L \land M \Rightarrow P \)
- \( B \land L \Rightarrow M \)
- \( A \land P \Rightarrow L \)
- \( A \land B \Rightarrow L \)
- \( A \)
- \( B \)

Forward Chaining Algorithm

```plaintext
Function PL-FC-INITIALIZE(KB) returns true or false
local Variables: count, a table, indexed by clause, initially the number of premises
informed, 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
  \( P \leftarrow \) Pop(agenda)
  unless informed(P) do
    informed(P) ← true
    for each Horn clause \( \psi \) in whose premise \( P \) appears do
      decrement count[\psi]
    if count[\psi] = 0 then do
      if head[\psi] \( = \) \( Q \) then return true
      Push(\psi, agenda)
  return false
```

- Forward chaining is sound and complete for Horn KB
Forward Chaining Example

\[
\begin{align*}
P &\Rightarrow Q \\
L \land M &\Rightarrow P \\
B \land L &\Rightarrow M \\
A \land P &\Rightarrow L \\
A \land B &\Rightarrow L \\
A &\land B
\end{align*}
\]

Connectors
Arc = AND
No arc = OR

Numbers in red indicate number of propositions needed to prove result

A is true, so B or P needed to prove L

Forward Chaining Example (II)

Forward Chaining Example (III)

Forward Chaining Example (IV)

Forward Chaining Example (V)

Forward Chaining Example (VI)
Forward Chaining Example (VII)

\[ P \implies Q \]
\[ L \land M \implies P \]
\[ B \land L \implies M \]
\[ A \land P \implies L \]
\[ A \land B \implies L \]
\[ A \]
\[ B \]

Forward Chaining Example (VIII)

\[ P \implies Q \]
\[ L \land M \implies P \]
\[ B \land L \implies M \]
\[ A \land P \implies L \]
\[ A \land B \implies L \]
\[ A \]
\[ B \]

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 \( m \)
  4. Hence \( m \) is a model of \( KB \)
  5. If \( KB \models q \), \( q \) is true in every model of \( KB \), including \( m \)

Backward Chaining Examples

**Backward Chaining Example**

\[ P \implies Q \]
\[ L \land M \implies P \]
\[ B \land L \implies M \]
\[ A \land P \implies L \]
\[ A \land B \implies L \]
\[ A \]
\[ B \]

We want to prove \( Q \)

**Backward Chaining Example (II)**

\[ P \implies Q \]
\[ L \land M \implies P \]
\[ B \land L \implies M \]
\[ A \land P \implies L \]
\[ A \land B \implies L \]
\[ A \]
\[ B \]

Q is True if \( P \) is True, and

Try to prove \( P \)
Artificial Intelligence

Backward Chaining Example (III)

P ⇒ Q
L ∧ M ⇒ P
B ∧ L ⇒ M
A ∧ P ⇒ L
A ∧ B ⇒ L
A
B

P is True if L and M are True,
Try to prove L and M

Backward Chaining Example (VI)

P ⇒ Q
L ∧ M ⇒ P
B ∧ L ⇒ M
A ∧ P ⇒ L
A ∧ B ⇒ L
A
B

Backward Chaining Example (V)

P ⇒ Q
L ∧ M ⇒ P
B ∧ L ⇒ M
A ∧ P ⇒ L
A ∧ B ⇒ L
A
B

Backward Chaining Example (VI)

P ⇒ Q
L ∧ M ⇒ P
B ∧ L ⇒ M
A ∧ P ⇒ L
A ∧ B ⇒ L
A
B

Backward Chaining Example (VII)

P ⇒ Q
L ∧ M ⇒ P
B ∧ L ⇒ M
A ∧ P ⇒ L
A ∧ B ⇒ L
A
B

Backward Chaining Example (VIII)

P ⇒ Q
L ∧ M ⇒ P
B ∧ L ⇒ M
A ∧ P ⇒ L
A ∧ B ⇒ L
A
B
Backward Chaining Example (IX)

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

Backward Chaining Example (X)

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

Forward vs. Backward Chaining

- FC is data-driven, 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

Efficient Propositional Inference

Two families of efficient algorithms for propositional inference:
- Complete backtracking search algorithms
  - DPLL algorithm (Davis, Putnam, Logemann, Loveland)
- Incomplete local search algorithms
  - WalkSAT algorithm

The DPLL algorithm

Determine if an input propositional logic sentence (in CNF) is satisfiable.

Improvements over truth table enumeration:
1. Early termination
   A clause is true if any literal is true.
   A sentence is false if any clause is false.
2. Pure symbol heuristic
   Pure symbol: always appears with the same "sign" in all clauses.
   e.g., In the three clauses (A \lor \neg B), (\neg B \lor \neg C), (C \lor A),  
   A and B are pure, C is impure.
   Make a pure symbol literal true.
3. Unit clause heuristic
   Unit clause: only one literal in the clause.
   The only literal in a unit clause must be true.

The DPLL algorithm

FUNCTION DPLL-SATISFIABLE(x) returns true or false
inputs: x a sentence in propositional logic
clauses -- the set of clauses in the CNF representation of x
symbols -- a list of the proposition symbols in x
returns: DPLL(clauses, symbols, x)

Function DPLL(clauses, symbols, model) returns true or false

if every clause in clauses is true in model then return true
if some clause in clauses is false in model then return false
P = value(FIND PURE SYMBOL(symbols, clauses, model))
if P is non-null then return DPLL(clauses, symbols -- P, [P = value(model)])
else
return DPLL(clauses, rest, [P = true(model)]) or
DPLL(clauses, rest, [P = false(model)])
The **WalkSAT algorithm**

- Incomplete, local search algorithm
- Evaluation function: The min-conflict heuristic of minimizing the number of unsatisfied clauses
- Balance between greediness and randomness

**Hard Satisfiability Problems**

- Consider random 3-CNF sentences. e.g.,
  \[ \neg D \lor \neg B \lor C \land (B \lor \neg A \lor \neg C) \land \]
  \[ (\neg C \lor \neg B \lor E) \land (E \lor \neg D \lor B) \land \]
  \[ (B \lor E \lor \neg C) \]

- \( m \) = number of clauses
- \( n \) = number of symbols
- Hard problems seem to cluster near \( m/n \) = 4.3 (critical point)

**Inference-Based Agents in the Wumpus World**

A wumpus-world agent using propositional logic:

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

\( \Rightarrow \) 64 distinct proposition symbols, 155 sentences
Artificial Intelligence

### Expressiveness Limitation of Propositional Logic

- KB contains "physics" sentences for every single square
- For every time $t$ and every location $[x,y]$,
  \[ L_{x,y} \land \text{FacingRight} \land \text{Forward} \Rightarrow L_{x+1,y} \]
- Rapid proliferation of clauses

---

#### 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.
- Resolution is complete for propositional logic
- Forward, backward chaining are linear-time, complete for Horn clauses
- Propositional logic lacks expressive power