Chapter 7: Logical Agents
“Thinking Rationally”

- Computational models of human “thought” processes
- Computational models of human behavior
- Computational systems that “think” rationally
- Computational systems that behave rationally
Knowledge Base

*Knowledge Base*: set of sentences represented in a knowledge representation language and represents assertions about the world.

*Inference rule*: when one ASKS questions of the KB, the answer should *follow* from what has been TELLed to the KB previously.
Generic KB-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
Abilities KB agent

Agent must be able to:

- Represent states and actions,
- Incorporate new percepts
- Update internal representation of the world
- Deduce hidden properties of the world
- Deduce appropriate actions
Description level

- The KB agent is similar to agents with internal state
- Agents can be described at different levels
  - Knowledge level
    - What they know, regardless of the actual implementation. (Declarative description)
  - Implementation level
    - Data structures in KB and algorithms that manipulate them e.g propositional logic and resolution.
A Typical Wumpus World
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

Sensors  Breeze, Glitter, Smell

Actuators  Left turn, Right turn,
           Forward, Grab, Release, Shoot
Exploring the Wumpus World

[1,1] The KB initially contains the rules of the environment. The first percept is [stench, breeze, glitter, bump, scream] = [none, none, none, none, none], move to safe cell e.g. [2,1]

[2,1] breeze which indicates that there is a pit in [2,2] or [3,1], return to [1,1], then to [1,2] to the other safe cell
Exploring the Wumpus World

[1,2] Stench in cell which means that wumpus is in [1,1], [1,3] or [2,2]  
YET … not in [1,1] since visited before  
YET … not in [2,2] or stench would have been detected in [2,1]  
THUS … wumpus is in [1,3]  
THUS [2,2] is safe because of lack of breeze in [1,2]  
THUS pit in [3,1]  
move to next safe cell [2,2]
Exploring the Wumpus World

[2,3] is safe since no breeze and no small in [2,2]
[2,2] move to [2,3]
[2,3] detect glitter, smell, breeze
   THUS pick up gold
   THUS pit in [3,3] or [2,4]
What is a logic?

- A formal language
  - Syntax – what expressions are legal (well-formed)
  - Semantics – what legal expressions mean
    - in logic the truth of each sentence with respect to each possible 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 in a world where $x=7$ and $y=1$
  - $X+2 \geq y$ is false in a world where $x=0$ and $y=6$
Entailment

- One thing follows from another
  \[ KB \models \alpha \]

- KB entails sentence \( \alpha \) if and only if \( \alpha \) is true in worlds where KB is true (i.e., \( \alpha \) is true in every model of KB)

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

- Entailment is a relationship between sentences 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.
- $m$ is a model of a sentence $\alpha$ if $\alpha$ is true in $m$
- $M(\alpha)$ is the set of all models of $\alpha$
Wumpus world model

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

Consider possible models for ?s assuming only pits

3 Boolean choices ⇒ 8 possible models
Wumpus world model
Wumpus world model

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

Rules = Squares adjacent to a pit have breeze

Observations = no breeze in [1,1] and breeze in [2,1]
Wumpus world model

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

$\alpha_1 = \text{"[1,2] is safe"}$, $KB \models \alpha_1$, proved by model checking

$\alpha = \text{true in every model of } KB$
Wumpus world model

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

$$\alpha_2 = \text{"[2,2] is safe", } KB \not\models \alpha_2$$
Logical inference

- The notion of entailment can be used for logic inference.
  - **Model checking (see wumpus example):** enumerate all possible models and check whether $\alpha$ is true.
- If an inference algorithm $i$ only derives entailed sentences it is called **sound** or **truth preserving**.
  - **Otherwise it just makes things up.**
    
    $i$ is sound if whenever $KB |- _i \alpha$, it is also true that $KB|= \alpha$
- Completeness: an inference algorithm can derive any sentence that is entailed.
  - $i$ is complete if whenever $KB|= \alpha$, it is also true that $KB|- _i \alpha$
Schematic perspective

If KB is true in the real world, then any sentence $\alpha$ derived from KB by a sound inference procedure is also true in the real world.
Propositional logic: Syntax

- Propositional logic is the simplest logic – illustrates basic ideas

- The proposition symbols $S_1$, $S_2$ 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, $S_1 \implies S_2$ is a sentence (implication)
  - If $S_1$ and $S_2$ are sentences, $S_1 \iff S_2$ is a sentence (biconditional)
Propositional logic: Semantics

Each **model** specifies true/false for **each proposition symbol**

E.g. Consider only 3 variables:

\[
\begin{array}{ccc}
P_{1,2} & P_{2,2} & P_{3,1} \\
\text{false} & \text{true} & \text{false}
\end{array}
\]

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

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
- \( S_1 \Rightarrow S_2 \) is true iff \( S_1 \) is false or \( S_2 \) is true
  
  i.e., \( S_1 \Leftarrow S_2 \) is false iff \( S_1 \) is true and \( S_2 \) is false
- \( S_1 \iff S_2 \) is true iff \( S_1 \Rightarrow S_2 \) is true and \( S_2 \Rightarrow S_1 \) is true

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

\[
\neg P_{1,2} \land (P_{2,2} \lor P_{3,1}) = true \land (true \lor false) = true \land true = true
\]
# 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>$P \implies Q$</th>
<th>$P \iff Q$</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>false</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 $P_{i,j}$ be true if there is a pit in $[i, j]$.
- Let $B_{i,j}$ be true if there is a breeze in $[i, j]$. □
- Current Knowledge Base (KB):
  - $\neg P_{1,1}$
  - $\neg B_{1,1}$
  - $B_{2,1}$ □

□ "Pits cause breezes in adjacent squares" □

- $B_{1,1} \leftrightarrow (P_{1,2} \lor P_{2,1})$
- $B_{2,1} \leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{3,1})$ □
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>$KB$</th>
<th>$\alpha_1$</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>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>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>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>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>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>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>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>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

$\alpha = \neg P[1,2]$
Inference by enumeration

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

```plaintext
function TT-ENTAILS?(KB, α) returns true or false
    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))
```

- 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 $\alpha \models \beta$ and $\beta \models \alpha$

  \[
  (\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 \iff \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
Validity and satisfiability

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

Validity is connected to inference via the Deduction Theorem:
   \( KB \models \alpha \) if and only if \( (KB \Rightarrow \alpha) \) is valid

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

A sentence is unsatisfiable if it is true in no model
   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
Proof methods

- Proof methods divided into (roughly) two kinds:
  - **Application of inference rules**
    - Legitimate (sound) generation of new sentences from old sentences
    - **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 $n$)
Resolution

- Conjunctive Normal Form (CNF)
  - clause = disjunctions of literals
  - CNF = conjunction of clauses
  - E.g., \((A \lor \neg B) \land (B \lor \neg C \lor \neg D)\)

- Resolution inference rule (for CNF):

\[
\begin{array}{c}
|_i \lor \ldots \lor |_k, \\
\hline
|_i \lor \ldots \lor |_{i-1} \lor |_{i+1} \lor \ldots \lor |_k \lor m_1 \lor \ldots \lor m_n \\
\hline
\end{array}
\]

where \(|_i\) and \(m_j\) are complementary literals; i.e., \(|_i = \neg m_j\)

- E.g., \(P_{1,3} \lor P_{2,2}, \neg P_{2,2}\)
  \[P_{1,3}\]

- Resolution is sound and complete for propositional logic
Resolution

Soundness of resolution inference rule:

\[ \neg (l_i \lor \ldots \lor l_{i-1} \lor l_{i+1} \lor \ldots \lor l_k) \Rightarrow l_i \]

\[ \neg m_j \Rightarrow (m_1 \lor \ldots \lor m_{j-1} \lor m_{j+1} \lor \ldots \lor m_n) \]

\[ \neg (l_i \lor \ldots \lor l_{i-1} \lor l_{i+1} \lor \ldots \lor l_k) \Rightarrow (m_1 \lor \ldots \lor m_{j-1} \lor m_{j+1} \lor \ldots \lor m_n) \]
Conversion to CNF

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

1. Eliminate \iff, replacing \alpha \iff \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 ((\neg P_{1,2} \lor \neg 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 (\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
        resolvents ← PL-Resolve($C_i, C_j$)
        if resolvents contains the empty clause then return true
        new ← new \cup resolvents
    if new \subseteq clauses then return false
    clauses ← clauses \cup new
```
Resolution example

- $KB = (B_{1,1} \iff (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1}$
- $\alpha = \neg P_{1,2}$
Forward and backward chaining

- **Horn Form** (restricted)
  - KB = conjunction of Horn clauses
  - Horn clause =
    - proposition symbol; or
    - (conjunction of symbols) ⇒ symbol
  - E.g., C ∧ (B ⇒ A) ∧ (C ∧ D ⇒ B) is a KB in Horn form

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

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

- 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

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

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
            end
            PUSH(HEAD[c], agenda)
    end
end

return false

Forward chaining is sound and complete for Horn KB
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
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 sub-goal is already on the goal stack

Avoid repeated work: check if new sub-goal

  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
Backward chaining example
Backward chaining example
Backward chaining example

[Diagram with nodes labeled Q, P, M, L, A, B]
Backward chaining example
Backward chaining example
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
Inference-based agents in the wumpus world

A wumpus-world agent using propositional logic:

- **Square [1,1] does not contain pit or wumpus:**
  \[ \neg P_{1,1} \]
  \[ \neg W_{1,1} \]

- **How breeze and stench arise for square [x,y]:**
  \[ B_{x,y} \iff (P_{x,y+1} \lor P_{x,y-1} \lor P_{x+1,y} \lor P_{x-1,y}) \]
  \[ S_{x,y} \iff (W_{x,y+1} \lor W_{x,y-1} \lor W_{x+1,y} \lor W_{x-1,y}) \]
  (Need a sentence for each possible value of \( x \) and \( y \).)

- **Exactly one wumpus** = *at least one wumpus* + *at most one wumpus*
  \[ W_{1,1} \lor W_{1,2} \lor \ldots \lor W_{4,4} \]
  \[ \neg W_{1,1} \lor \neg W_{1,2} \]
  \[ \neg W_{1,1} \lor \neg W_{1,3} \]
  \[ \ldots \]
  \[ \Rightarrow 64 \text{ distinct proposition symbols, 155 sentences} \]
function PL-WUMPUS-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, S_{x,y}) else TELL(KB, \neg S_{x,y})
if breeze then TELL(KB, B_{x,y}) else TELL(KB, \neg B_{x,y})
if glitter then action \leftarrow grab
else if plan is nonempty then action \leftarrow POP(plan)
else if for some fringe square [i,j], ASK(KB, \neg P_{i,j} \land \neg W_{i,j}) is true or
for some fringe square [i,j], ASK(KB, P_{i,j} \lor W_{i,j}) is false then do
plan \leftarrow A*-GRAPH-SEARCH(ROUTE-PB([x,y], orientation, [i,j],visited))
action \leftarrow POP(plan)
else action \leftarrow a randomly chosen move
return action
Expressiveness limitation of propositional logic

- KB contains "physics" sentences for every single square

- For every time $t$ and every location $[x,y]$,

\[ L_{x,y}^t \land FacingRight^t \land Forward^t \Rightarrow L_{x+1,y}^{t+1} \]

- 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 w.r.t. 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 and complete for Horn clauses

- Propositional logic lacks *expressive* power