# The Wumpus Game

<table>
<thead>
<tr>
<th></th>
<th>Breeze</th>
<th>Stench</th>
<th>Breeze</th>
<th>Gold</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stench</td>
<td></td>
<td>Breeze</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Breeze</td>
<td></td>
<td></td>
<td>Breeze</td>
<td></td>
</tr>
<tr>
<td>Stench</td>
<td></td>
<td>Breeze</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Start</td>
<td>Breeze</td>
<td></td>
<td></td>
<td>Breeze</td>
</tr>
</tbody>
</table>

The game starts with the player in a room with a breeze, a stench, and a gold. The player must navigate the cave, avoiding the stench and the wumpus, to reach the treasure.
The Wumpus Game

- **Stench** in the square containing the wumpus and in the directly adjacent squares
- **Breeze** in the squares directly adjacent to a pit
- **Glitter** in the square where the gold is
- **Bump** when an agent walks into a wall
- **Scream** when the wumpus is killed
The Wumpus Game

- Agent’s **percept**: [Stench, Breeze, Glitter, Bump, Scream]

- Agent’s **actions**:
  - Go forward, turn right 90°, turn left 90°
  - Grab to pick up an object in the same square
  - Fire 1 arrow in a straight line in the faced direction
  - Climb to leave the cave

- **Death** if entering a square of a pit or a live wumpus

- Agent’s **goal**: find and bring the gold back to the start
# The Wumpus Game

<table>
<thead>
<tr>
<th>Stench</th>
<th>Breeze</th>
<th>Stench</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breeze</td>
<td>OK</td>
<td>Breeze</td>
</tr>
<tr>
<td>OK</td>
<td>Breeze</td>
<td>OK</td>
</tr>
<tr>
<td>Start</td>
<td>Breeze</td>
<td>Breeze</td>
</tr>
</tbody>
</table>

- **A** = Agent
- **B** = Breeze
- **G** = Glitter/Gold
- **OK** = Safe square
- **P** = Pit
- **S** = Stench
- **V** = Visited
- **W** = Wumpus
The Wumpus Game

A = Agent
B = Breeze
G = Glitter/Gold
OK = Safe square
P = Pit
S = Stench
V = Visited
W = Wumpus
## The Wumpus Game

### Symbols
- \(A\) = Agent
- \(B\) = Breeze
- \(G\) = Glitter/Gold
- \(OK\) = Safe square
- \(P\) = Pit
- \(S\) = Stench
- \(V\) = Visited
- \(W\) = Wumpus

### Table

<table>
<thead>
<tr>
<th>Stench</th>
<th>Breeze</th>
<th>Stench</th>
<th>Breeze</th>
<th>Breeze</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A)</td>
<td>(\text{Breeze})</td>
<td>Breeze</td>
<td>(\text{Gold})</td>
<td>(\text{Breeze})</td>
</tr>
<tr>
<td>(\text{Stench})</td>
<td>(\text{OK})</td>
<td>Breeze</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\text{Start} \Rightarrow \text{V})</td>
<td>(\text{Breeze})</td>
<td>V</td>
<td>V</td>
<td>Breeze</td>
</tr>
</tbody>
</table>

### Diagram

- A = Agent
- B = Breeze
- G = Glitter/Gold
- OK = Safe square
- P = Pit
- S = Stench
- V = Visited
- W = Wumpus
The Wumpus Game

<table>
<thead>
<tr>
<th>Stench</th>
<th>Breeze</th>
<th>Breeze</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stench</td>
<td>A</td>
<td>Breeze</td>
</tr>
<tr>
<td>V</td>
<td>A</td>
<td>Breeze</td>
</tr>
</tbody>
</table>

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

Cao Hoang Tru  
CSE Faculty - HCMUT  
02 November 2009
## The Wumpus Game

<table>
<thead>
<tr>
<th>Stench</th>
<th>Breeze</th>
<th>Breeze</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>V</td>
<td>V</td>
</tr>
<tr>
<td>Start  =&gt;</td>
<td>Breeze</td>
<td>Breeze</td>
</tr>
</tbody>
</table>

- **A** = Agent
- **B** = Breeze
- **G** = Glitter/Gold
- **OK** = Safe square
- **P** = Pit
- **S** = Stench
- **V** = Visited
- **W** = Wumpus
Propositional Logic Syntax

• Logical **constants**: true, false

• Propositional **symbols**: P, Q, ...

• Logical **connectives**: ¬, ∧, ∨, ⇒, ⇔

• **Sentences (formulas)**:
  - Logical constants
  - Proposition symbols
  - If α is a sentence, then so are ¬α and (α)
  - If α and β are sentences, then so are α ∧ β, α ∨ β, α ⇒ β, and α ⇔ β
Propositional Logic Semantics

- **Interpretation**: propositional symbol $\rightarrow$ true/false
- The truth value of a sentence is defined by the truth table

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>Q</td>
<td>$\neg P$</td>
<td>$P \land Q$</td>
<td>$P \lor Q$</td>
<td>$P \Rightarrow Q$</td>
<td>$P \Leftrightarrow Q$</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>--------</td>
<td>------------</td>
<td>------------</td>
<td>----------------</td>
<td>----------------</td>
</tr>
<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>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>
Propositional Logic Semantics

- **Satisfiable**: true under an interpretation
- **Valid**: true under all interpretations

\[
\begin{array}{|c|c|c|c|c|}
\hline
P & Q & P \land \neg P & (P \lor Q) \land \neg Q & ((P \lor Q) \land \neg Q) \Rightarrow P \\
\hline
\text{false} & \text{false} & \text{false} & \text{false} & \text{true} \\
\text{false} & \text{true} & \text{false} & \text{false} & \text{true} \\
\text{true} & \text{false} & \text{false} & \text{true} & \text{true} \\
\text{true} & \text{true} & \text{false} & \text{false} & \text{true} \\
\hline
\end{array}
\]

unsatisfiable \hspace{1cm} satisfiable \hspace{1cm} valid
Propositional Logic Semantics

- **Model**: an interpretation under which the sentence is true
Propositional Logic Semantics

- **Entailment**: $\text{KB} \models \alpha$ iff every model of $\text{KB}$ is a model of $\alpha$

$\alpha$ is a logical consequence of $\text{KB}$

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$P$</td>
<td>$Q$</td>
<td>$P \implies Q$</td>
<td>$P \implies Q, P$</td>
</tr>
<tr>
<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>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

$\{P \implies Q, P\} \models Q$
Propositional Logic Semantics

- **Equivalence:** $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>P $\Rightarrow$ Q</th>
<th>$\neg P \lor Q$</th>
<th>$P \Rightarrow Q \equiv \neg P \lor Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td></td>
</tr>
</tbody>
</table>
Propositional Logic Semantics

• Theorems:
  – $\alpha \models \beta$ iff $\alpha \Rightarrow \beta$ is valid
    $$\text{KB} \models \alpha \text{ can be proved by validity of } \text{KB} \Rightarrow \alpha$$
  – $\alpha \models \beta$ iff $\alpha \land \neg \beta$ is unsatisfiable
    $$\text{KB} \models \alpha \text{ can be proved by refutation of } \text{KB} \land \neg \alpha$$
Inference Rules

• Rule $R$:

Premises: $\alpha_1, \alpha_2, \ldots, \alpha_n$

Conclusion: $\beta$

• Soundness: $R$ is sound iff $\{\alpha_1, \alpha_2, \ldots, \alpha_n\} \models \beta$
Inference Rules

- **Modus Ponens:**
  \[
  \alpha \Rightarrow \beta, \ \alpha \\
  \hline
  \beta
  \]

- **∧-Elimination:**
  \[
  \alpha_1 \land \alpha_2 \land \ldots \land \alpha_n \\
  \hline
  \alpha_i
  \]
Inference Rules

- **∧-Introduction:**

\[
\alpha_1, \alpha_2, \ldots, \alpha_n \quad \therefore \quad \alpha_1 \land \alpha_2 \land \ldots \land \alpha_n
\]

- **∨-Introduction:**

\[
\alpha_i \quad \therefore \quad \alpha_1 \lor \alpha_2 \lor \ldots \lor \alpha_n
\]
Inference Rules

- **Double-negation elimination:**
  \[ \neg
\neg \alpha \]

  \[ \alpha \]
Resolution

- **Unit resolution:**

\[
P_i \equiv \neg Q \\
\[P_1 \lor \ldots \lor P_i \lor \ldots \lor P_n, Q
\]

\[
P_1 \lor \ldots \lor P_{i-1} \lor P_{i+1} \lor \ldots \lor P_n
\]

(all \(P_i\) and \(Q\) are \textit{literals})

- **Full resolution:**

\[
P_i \equiv \neg Q_j \\
\[P_1 \lor \ldots \lor P_i \lor \ldots \lor P_n, Q_1 \lor \ldots \lor Q_j \lor \ldots \lor Q_m
\]

\[
P_1 \lor \ldots \lor P_{i-1} \lor P_{i+1} \lor \ldots \lor P_n \lor Q_1 \ldots \lor Q_{j-1} \lor Q_{j+1} \lor \ldots \lor Q_m
\]
Resolution

- **Conjunctive normal form (CNF):** conjunction of disjunctions of literals

\[(L_{11} \lor ... \lor L_{1k}) \land ... \land (L_{n1} \lor ... \lor L_{nm})\]

- **k-CNF:** each clause contains at most \(k\) literals

- Every sentence can be written in CNF
Resolution

- **Conjunctive normal form (CNF):** conjunction of disjunctions of literals

\[(P \land Q) \Rightarrow R \equiv \neg(P \land Q) \lor R \equiv \neg P \lor \neg Q \lor R\]

\[(S \lor T) \Rightarrow Q \equiv \neg(S \lor T) \lor Q \equiv (\neg S \lor Q) \land (\neg T \lor Q)\]
Inference Algorithms

• To derive/assert a sentence given a knowledge base (i.e., a set of sentences)

• Inference algorithm = Inference rules + a searching algorithm
Inference Algorithms

Algorithm PL-Resolution(KB, α)

**input:** KB and α are PL sentences

**output:** true (to assert KB |= α) or false (otherwise)

Clauses = set of clauses in the CNF representation of KB ∧ ¬α

New = {}

loop

for each Cᵢ, Cⱼ in Clauses do

    Resolvents = PL-Resolve(Cᵢ, Cⱼ)

    if Resolvents contains an empty clause then return true

    New = New ∪ Resolvents

if New ⊆ Clauses then return false

Clauses = Clauses ∪ New
Inference Algorithms

• Example:

\[ \text{KB} = \{ P, (P \land Q) \Rightarrow R, (S \lor T) \Rightarrow Q, T \} \]

\[ \alpha = R \]
Inference Algorithms

- **Soundness**: every deduced answer is correct
Inference Algorithms

• **Completeness**: every correct answer is deducible
Inference Algorithms

• Example: Google search
Inference Algorithms

Soundness

Completeness

Complexity
Inference Algorithms

- **Soundness**: every deduced answer is correct

  if PL-Resolution returns *true* then $\textbf{KB} \models \alpha$
Inference Algorithms

• Completeness: every correct answer is deducible

if $KB \models \alpha$ then PL-Resolution returns true
Inference Algorithms

- PL-Resolution is **sound** and **complete**
Inference Algorithms

- Modus Ponens with any searching algorithm is not complete

\[ KB = \{P \Rightarrow Q, Q \Rightarrow R\} \]
\[ \alpha = P \Rightarrow R \]
Inference Algorithms

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

\[ \neg P_1 \lor \neg P_2 \lor \ldots \lor \neg P_n \lor Q \]

or

\[ P_1 \land P_2 \land \ldots \land P_n \Rightarrow Q \]

• Forward or backward chaining can be used for inference on Horn clauses
Satisfiability Problem

- **SAT problem**: to determine if a sentence is satisfiable
  (to determine if the symbols in the sentence can be assigned true/false as to make it evaluate to true)

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

- SAT problem in propositional logic is \textbf{NP-complete}

This does not mean all instances of PL inference has the exponential complexity

Polynomial-time inference procedures for Horn clauses
Satisfiability Problem

• Many combinatorial problems in computer science can be reduced to SAT
Satisfiability Problem

- Backtracking algorithms
- Davis and Putnam (1960)
- Davis, Logemann, Loveland (1962)
Satisfiability Problem

- DPLL algorithm:
  - Recursive, depth-first enumeration of possible models
Satisfiability Problem

- **DPLL algorithm:**
  - *Early termination:* a clause is true if any literal is true

\[(A \lor B) \land (A \lor C)\] is true if A is true regardless of B and C.
Satisfiability Problem

- **DPLL algorithm:**
  - **Pure symbol:** has the same sign in all clauses

$$(A \lor \neg B) \land (\neg B \lor \neg C) \land (C \lor A)$$

A model would have the literals of pure symbols assigned true.

Satisfied clause removal makes new pure symbols.
Satisfiability Problem

- DPLL algorithm:
  - **Unit clause**: has just one literal

\[(A \lor B) \land \neg B\]

Unit clause satisfaction makes new unit clauses (unit propagation).
Statisfiability Problem

Algorithm DPLL(Clauses, Symbols, Model)

\[
\begin{align*}
\text{if } & \text{ every clause in Clauses is true in Model then return true} \\
\text{if } & \text{ some clause in Clauses is false in Model then return false} \\
(P, \text{value}) & = \text{PURE-SYMBOL(Clauses, Symbols, Model)} \\
\text{if } & \text{ P } \neq \text{ null then return } \text{DPLL(Clauses, Symbols - P, EXT(P, value, Model))} \\
(P, \text{value}) & = \text{UNIT-CLAUSE(Clauses, Symbols, Model)} \\
\text{if } & \text{ P } \neq \text{ null then return } \text{DPLL(Clauses, Symbols - P, EXT(P, value, Model))} \\
\text{choose a } P \text{ in Symbols} \\
\text{return } & \text{DPLL(Clauses, Symbols - P, EXT(P, true, Model)) or} \\
& \text{DPLL(Clauses, Symbols - P, EXT(P, false, Model))}
\end{align*}
\]
Satisfiability Problem

- DPLL algorithm:

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

- Local search algorithms

  Evaluation function counts the number of unsatisfied clauses
Satisfiability Problem

- Local search algorithms
  Evaluation function counts the number of unsatisfied clauses

- Randomness to escape local minima
  Flipping the truth value of one symbol at a time
Statisfiability Problem

Algorithm *WalkSAT*(Clauses, p, max_flips) /*1996*/

**input**: p is flipping probability, max_flips is number of flips allowed

**output**: a model or failure

Model = a random assignment of truth values to the symbols

for i = 1 to max_flips do
  if Model satisfies Clauses then return Model
  C = a randomly selected clause from Clauses that is false in Model
  with probability p flip the value in Model of a symbol from C
  else flip whichever symbol in C to maximizes no. of satisfied clauses

return failure
Satisfiability Problem

- **WalkSAT algorithm:**

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

**Initial model:**

- A = true
- B = false
- C = true
- D = true
- E = false
Satisfiability Problem

- The larger the ratio clause/symbol is, the less likely the clause is satisfiable.
Satisfiability Problem

- The probability of satisfiability drops sharply around the ratio of 4.3.
Satisfiability Problem

- That is actually when the problem is hard.
Satisfiability Problem

- WalkSAT is much faster than DPLL.
Inference Monotonicity

- Monotonicity: the set of entailed sentences can only increase when information is added to the KB

\[
\text{if } \textsf{KB}_1 \models \alpha \text{ then } (\textsf{KB}_1 \cup \textsf{KB}_2) \models \alpha
\]
Inference Monotonicity

• Propositional logic is monotone
An Agent for the Wumpus Game

- Knowledge Base: $X_{ij} = \text{column } i, \text{ row } j$

$$\neg S_{11} \quad \neg B_{11}$$
$$\neg S_{21} \quad B_{21}$$
$$S_{12} \quad \neg B_{12}$$
An Agent for the Wumpus Game

- Knowledge Base: $X_{ij} = \text{column } i, \text{ row } j$

\[ R_1: \neg S_{11} \Rightarrow \neg W_{11} \land \neg W_{12} \land \neg W_{21} \]
\[ R_2: \neg S_{21} \Rightarrow \neg W_{11} \land \neg W_{21} \land \neg W_{22} \land \neg W_{31} \]
\[ R_3: \neg S_{12} \Rightarrow \neg W_{11} \land \neg W_{12} \land \neg W_{22} \land \neg W_{13} \]
\[ R_4: \ S_{12} \Rightarrow W_{13} \lor W_{12} \lor W_{22} \lor W_{11} \]
Agent for the Wumpus Game

• Finding the wumpus:

Modus Ponens ($\neg S_{11} + R_1$): $\neg W_{11} \land \neg W_{12} \land \neg W_{21}$

$\land$-Elimination: $\neg W_{11} \neg W_{12} \neg W_{21}$

Modus Ponens ($\neg S_{21} + R_2$): $\neg W_{11} \land \neg W_{21} \land \neg W_{22} \land \neg W_{31}$

$\land$-Elimination: $\neg W_{11} \neg W_{21} \neg W_{22} \neg W_{31}$

Modus Ponens ($S_{12} + R_4$): $W_{13} \lor W_{12} \lor W_{22} \lor W_{11}$

Unit resolution ($\neg W_{11} + W_{13} \lor W_{12} \lor W_{22} \lor W_{11}$): $W_{13} \lor W_{12} \lor W_{22}$

Unit resolution ($\neg W_{22} + W_{13} \lor W_{12} \lor W_{22}$): $W_{13} \lor W_{12}$

Unit resolution ($\neg W_{12} + W_{13} \lor W_{12}$): $W_{13}$
Agent for the Wumpus Game

- Translating knowledge into action:

\[ A_{11} \land \text{East}_A \land W_{21} \Rightarrow \neg \text{Forward} \]
Agent for the Wumpus Game

• Problems with the propositional agent:
  
  – Too many propositions to handle:
    
    "Don’t go forward if the wumpus is in front of you!" requires $16 \times 4 = 64$ propositions.
  
  – Hard to deal with time and change
  
  – Not expressive enough to represent or answer a question like "What action should the agent take?", …
Homework

• In Russell & Norvig’s AIMA (2nd ed.): Exercises of Chapter 7.