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

Inference engine ← domain-independent algorithms
Knowledge base ← domain-specific content

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
function KB-Agent (percept) returns an action

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

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

The agent must be able to:
Represent states, actions, etc.
Incorporate new percepts
Update internal representations of the world
Deduce hidden properties of the world
Deduce appropriate actions
Wumpus World PEAS description

Performance measure
- gold +1000, death -1000
- -1 per step, -10 for using the arrow

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

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

Sensors
- Breeze, Glitter, Smell

Chapter 7
Wumpus world characterization

Observable??
Wumpus world characterization

Observable?? No—only local perception

Deterministic??
Wumpus world characterization

Observable? No—only local perception

Deterministic? Yes—outcomes exactly specified

Episodic?
Wumpus world characterization

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

**Deterministic??** Yes—outcomes exactly specified

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

**Static??**
Wumpus world characterization

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

**Deterministic??** Yes—outcomes exactly specified

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

**Static??** Yes—Wumpus and Pits do not move

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

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

---

[Diagram of a wumpus world grid with marked OK and P?]
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/ prob 0.86, vs. 0.31

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 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; $x^2 + y > \quad$ 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 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

Note: brains process syntax (of some sort)
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 \( KB \models \alpha \) if and only if \( M(KB) \subseteq M(\alpha) \).

E.g. \( KB = \) Giants won and Reds won

\( \alpha = \) Giants won
Situation after detecting nothing in [1,1], moving right, breeze in [2,1]

Consider possible models for $?s assuming only pits

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

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

Question \( \alpha_2 = \text{"[2,2] is safe"}, \) \( KB \not\models \alpha_2 \)  
Answer: no, not safe
Inference

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

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

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

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

Preview: we will define a logic (first-order logic) which is expressive enough to say almost anything of interest, and for which there exists a sound and complete inference procedure.

That is, the procedure will answer any question whose answer follows from what is known by the \( KB \).
Propositional logic: Syntax

Propositional logic is the simplest logic—illustrates basic ideas

The proposition symbols $P_1$, $P_2$ etc are sentences

If $S$ is a sentence, $\neg S$ is a sentence (negation)

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

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

If $S_1$ and $S_2$ are sentences, $S_1 \Rightarrow S_2$ is a sentence (implication)

If $S_1$ and $S_2$ are sentences, $S_1 \Leftrightarrow S_2$ is a sentence (biconditional)
Propositional logic: Semantics

Each model specifies true/false for each proposition symbol

E.g. $P_{1,2} \quad P_{2,2} \quad P_{3,1}$

true  true false

(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., 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.,

$\neg P_{1,2} \land (P_{2,2} \lor P_{3,1}) = true \land (false \lor true) = true \land true = true$
## 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>$P \Rightarrow Q$</th>
<th>$P \Leftrightarrow Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>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>
Let $P_{i,j}$ be true if there is a pit in $[i, j]$.
Let $B_{i,j}$ be true if there is a breeze in $[i, j]$.

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

“Pits cause breezes in adjacent squares”
Wumpus world sentences

Let $P_{i,j}$ be true if there is a pit in $[i, j]$.
Let $B_{i,j}$ be true if there is a breeze in $[i, j]$.

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

Rules R1 through R5

“Pits cause breezes in adjacent squares”

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

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

<table>
<thead>
<tr>
<th>$B_{1,1}$</th>
<th>$B_{2,1}$</th>
<th>$P_{1,1}$</th>
<th>$P_{1,2}$</th>
<th>$P_{2,1}$</th>
<th>$P_{2,2}$</th>
<th>$P_{3,1}$</th>
<th>$R_1$</th>
<th>$R_2$</th>
<th>$R_3$</th>
<th>$R_4$</th>
<th>$R_5$</th>
<th>$KB$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>

Enumerate rows (different assignments to symbols),
if $KB$ is true in row, check that $\alpha$ is too
Inference by enumeration

Depth-first enumeration of all models is sound and complete

<table>
<thead>
<tr>
<th>function TT-ENTAILS((KB, \alpha)) returns true or false</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputs: (KB), the knowledge base, a sentence in propositional logic</td>
</tr>
<tr>
<td>(\alpha), the query, a sentence in propositional logic</td>
</tr>
<tr>
<td>(symbols) ← a list of the proposition symbols in (KB) and (\alpha)</td>
</tr>
<tr>
<td>return TT-CHECK-ALL((KB, \alpha, symbols, []))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>function TT-CHECK-ALL((KB, \alpha, symbols, model)) returns true or false</th>
</tr>
</thead>
<tbody>
<tr>
<td>if EMPTY((symbols)) then</td>
</tr>
<tr>
<td>if PL-TRUE((KB, model)) then return PL-TRUE((\alpha, model))</td>
</tr>
<tr>
<td>else return true</td>
</tr>
<tr>
<td>else do</td>
</tr>
<tr>
<td>(P ← FIRST(symbols)); (rest ← REST(symbols))</td>
</tr>
<tr>
<td>return TT-CHECK-ALL((KB, \alpha, rest, EXTEND(P, true, model))) and</td>
</tr>
<tr>
<td>TT-CHECK-ALL((KB, \alpha, rest, EXTEND(P, false, model)))</td>
</tr>
</tbody>
</table>

“and” : must check all branches

\(O(2^n)\) for \(n\) symbols; problem is co-NP-complete
## Logical equivalence

Two sentences are **logically equivalent** iff true in same models:

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

<table>
<thead>
<tr>
<th>Logical Equivalence</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>((\alpha \land \beta) \equiv (\beta \land \alpha))</td>
<td>commutativity of (\land)</td>
</tr>
<tr>
<td>((\alpha \lor \beta) \equiv (\beta \lor \alpha))</td>
<td>commutativity of (\lor)</td>
</tr>
<tr>
<td>(((\alpha \land \beta) \land \gamma) \equiv (\alpha \land (\beta \land \gamma)))</td>
<td>associativity of (\land)</td>
</tr>
<tr>
<td>(((\alpha \lor \beta) \lor \gamma) \equiv (\alpha \lor (\beta \lor \gamma)))</td>
<td>associativity of (\lor)</td>
</tr>
<tr>
<td>(\neg(\neg \alpha) \equiv \alpha)</td>
<td>double-negation elimination</td>
</tr>
<tr>
<td>((\alpha \Rightarrow \beta) \equiv (\neg \beta \Rightarrow \neg \alpha))</td>
<td>contraposition</td>
</tr>
<tr>
<td>((\alpha \Rightarrow \beta) \equiv (\neg \alpha \lor \beta))</td>
<td>implication elimination</td>
</tr>
<tr>
<td>((\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)))</td>
<td>biconditional elimination</td>
</tr>
<tr>
<td>(\neg(\alpha \land \beta) \equiv (\neg \alpha \lor \neg \beta))</td>
<td>De Morgan</td>
</tr>
<tr>
<td>(\neg(\alpha \lor \beta) \equiv (\neg \alpha \land \neg \beta))</td>
<td>De Morgan</td>
</tr>
<tr>
<td>((\alpha \land (\beta \lor \gamma)) \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)))</td>
<td>distributivity of (\land) over (\lor)</td>
</tr>
<tr>
<td>((\alpha \lor (\beta \land \gamma)) \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)))</td>
<td>distributivity of (\lor) over (\land)</td>
</tr>
</tbody>
</table>
Validity and satisfiability

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

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

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

A sentence is unsatisfiable if it is true in no models
e.g., \( A \land \neg A \)

Satisfiability is connected to inference via the following:
\[ KB \models \alpha \text{ if and only if } (KB \land \neg \alpha) \text{ is unsatisfiable} \]

i.e., prove \( \alpha \) by reductio ad absurdum
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 alg.
- Typically require translation of sentences into a normal form

Model checking
truth table enumeration (always exponential in \( n \))
improved backtracking, e.g., Davis–Putnam–Logemann–Loveland
heuristic search in model space (sound but incomplete)
  - e.g., min-conflicts-like hill-climbing algorithms
Forward and backward chaining

**Horn Form** (restricted)

\[ \text{KB} = \text{conjunction of Horn clauses} \]

Horn clause =

- \( \diamond \) proposition symbol; or
- \( \diamond \) (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

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

\( \beta_1 \land \beta_2, \text{ok Horn form} \)
\( \beta_1 \lor \beta_2, \text{not in Horn form} \)

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

\[ P \implies Q \]
\[ L \land M \implies P \]
\[ B \land L \implies M \]
\[ A \land P \implies L \]
\[ A \land B \implies L \]
\[ A \]
\[ B \]
Forward chaining algorithm

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, 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 in \(KB\)

while agenda is not empty do
  \(p \leftarrow \text{POP}(\text{agenda})\)
  unless inferred[\(p\)] do
    inferred[\(p\)] \(\leftarrow \text{true}\)
    for each Horn clause \(c\) in whose premise \(p\) appears do
      decrement count[\(c\)]
      if count[\(c\)] = 0 then do
        if HEAD[\(c\)] = \(q\) then return true
        Push(HEAD[\(c\)], agenda)
  return false
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example

Chapter 7
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 $m$
   \begin{itemize}
   \item \textbf{Proof:} Suppose a clause $a_1 \wedge \ldots \wedge a_k \Rightarrow b$ is false in $m$
   \item Then $a_1 \wedge \ldots \wedge a_k$ is true in $m$ and $b$ is false in $m$
   \item Therefore the algorithm has not reached a fixed point!
   \end{itemize}

4. Hence $m$ is a model of $KB$

5. If $KB \models q$, $q$ is true in \textbf{every} model of $KB$, including $m$

\textbf{General idea:} construct any model of $KB$ by sound inference, check $\alpha$
Backward chaining

Idea: work backwards from the query $q$:
- to prove $q$ by BC,
  - check if $q$ is known already, or
  - prove by BC all premises of some rule concluding $q$

Avoid loops: check if new subgoal is already on the goal stack

Avoid repeated work: check if new subgoal
  1) has already been proved true, or
  2) has already failed
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example

\[ Q \rightarrow P \rightarrow M \rightarrow L \rightarrow A \rightarrow B \]
Forward vs. backward chaining

FC is data-driven, cf. automatic, unconscious processing,
  e.g., object recognition, routine decisions

May do lots of work that is irrelevant to the goal

BC is goal-driven, appropriate for problem-solving,
  e.g., Where are my keys? How do I get into a PhD program?

Complexity of BC can be much less than linear in size of KB
Conjunctive Normal Form (CNF—universal)

conjunction of disjunctions of literals

clauses

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

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

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

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

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

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

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

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

4. Apply distributivity law (\( \lor \) over \( \land \)) and flatten:
   
   \[ (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1}) \]

Checking Satisfiability of a Clausal form sentence is also called the SAT problem (a CSP): the first NP-complete problem
Resolution algorithm

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

function **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

$clauses \leftarrow$ 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 $clauses$ do
    $resolvents \leftarrow$ PL-RESOLVE($C_i, C_j$)
    if $resolvents$ contains the empty clause then return *true*
    $new \leftarrow new \cup resolvents$
  if $new \subseteq clauses$ then return *false*
  $clauses \leftarrow clauses \cup new$
Resolution example

\[ KB = (B_{1,1} \iff (P_{1,2} \lor P_{2,1})) \land \lnot B_{1,1} \quad \alpha = \lnot P_{1,2} \]

I need to show DPLL model checking algorithm in clausal form at this stage
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.

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

Propositional logic lacks expressive power