Outline

- Knowledge-based agents
- Logic in general—models and entailment
- Propositional (Boolean) logic
- Equivalence, validity, satisfiability
- Inference rules and theorem proving
  - forward chaining
  - backward chaining
  - resolution
knowledge-based agents
Knowledge-based Agent

- **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

```plaintext
function KB-A G E N T ( percept ) returns an action
    static: KB, a knowledge base
              t, a counter, initially 0, indicating time
    T E L L ( KB , MAKE-PERCEPT-SENTENCE ( percept , t ) )
    action ← ASK ( KB , MAKE-ACTION-QUERY ( t ) )
    T E L L ( 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
example
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
Wumpus World Characterization

- Observable? Yes—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
Exploring a Wumpus World
Exploring a Wumpus World
Exploring a Wumpus World
Exploring a Wumpus World

![Wumpus World Diagram](image-url)
Exploring a Wumpus World
Exploring a Wumpus World
Tight Spot

- Breeze in (1,2) and (2,1)  
  \[\rightarrow\text{ no safe actions}\]

- Assuming pits uniformly distributed,  
  (2,2) has pit w/ prob 0.86, vs. 0.31
Tight Spot

- 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
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

- Note: brains process syntax (of some sort)
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 $\alpha$.

$\Rightarrow KB \models \alpha$ if and only if $M(KB) \subseteq M(\alpha)$.

- E.g. $KB = \text{Giants won and Reds won}$
  $\alpha = \text{Giants won}$
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 $\Rightarrow$ 8 possible models
Wumpus Models
Wumpus Models

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

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

\( \alpha_2 = "[2,2] \text{ is safe}, \ KB \not\models \alpha_2 " \)
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
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. $P_{1,2}$ $P_{2,2}$ $P_{3,1}$
  
  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

  $S_1 \implies S_2$ 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>$P \Leftrightarrow Q$</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>

Philipp Koehn Artificial Intelligence: Logical Agents 6 October 2015
Wumpus World Sentences

- Let $P_{i,j}$ be true if there is a pit in $[i,j]$.
  - Observation $R_1: \neg P_{1,1}$

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

- “Pits cause breezes in adjacent squares”
  - Rule $R_2: B_{1,1} \iff (P_{1,2} \lor P_{2,1})$
  - Rule $R_3: B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1})$
  - Observation $R_4: \neg B_{1,1}$
  - Observation $R_5: B_{2,1}$

- What can we infer about $P_{1,2}$, $P_{2,1}$, $P_{2,2}$, etc.?
### 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>false</td>
<td>true</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>false</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>false</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>true</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>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</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>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</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>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</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>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</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>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>
</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

```
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
equivalence, validity, satisfiability
Logical Equivalence

- Two sentences are logically equivalent iff true in same models:
  \[ \alpha \equiv \beta \text{ if and only if } \alpha \models \beta \text{ and } \beta \models \alpha \]

\[
\begin{align*}
(\alpha \land \beta) & \equiv (\beta \land \alpha) \quad \text{commutativity of } \land \\
(\alpha \lor \beta) & \equiv (\beta \lor \alpha) \quad \text{commutativity of } \lor \\
((\alpha \land \beta) \land \gamma) & \equiv (\alpha \land (\beta \land \gamma)) \quad \text{associativity of } \land \\
((\alpha \lor \beta) \lor \gamma) & \equiv (\alpha \lor (\beta \lor \gamma)) \quad \text{associativity of } \lor \\
\neg(\neg \alpha) & \equiv \alpha \quad \text{double-negation elimination} \\
(\alpha \implies \beta) & \equiv (\neg \beta \implies \neg \alpha) \quad \text{contraposition} \\
(\alpha \implies \beta) & \equiv (\neg \alpha \lor \beta) \quad \text{implication elimination} \\
(\alpha \iff \beta) & \equiv ((\alpha \implies \beta) \land (\beta \implies \alpha)) \quad \text{biconditional elimination} \\
\neg(\alpha \land \beta) & \equiv (\neg \alpha \lor \neg \beta) \quad \text{De Morgan} \\
\neg(\alpha \lor \beta) & \equiv (\neg \alpha \land \neg \beta) \quad \text{De Morgan} \\
(\alpha \land (\beta \lor \gamma)) & \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)) \quad \text{distributivity of } \land \text{ over } \lor \\
(\alpha \lor (\beta \land \gamma)) & \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) \quad \text{distributivity of } \lor \text{ over } \land
\end{align*}
\]
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**
inference
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)
  \[ KB = \text{conjunction of Horn clauses} \]

- Horn clause =
  - proposition symbol; or
  - (conjunction of symbols) \( \implies \) symbol

  e.g., \( C \land (B \implies A) \land (C \land D \implies B) \)

- **Modus Ponens** (for Horn Form): complete for Horn KBs
  \[
  \frac{\alpha_1, \ldots, \alpha_n}{\beta, \alpha_1 \land \ldots \land \alpha_n \implies \beta}
  \]

- Can be used with **forward chaining** or **backward chaining**

- These algorithms are very natural and run in **linear** time
Example

- 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
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, init. 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 ← 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)
        end for
    end unless
end while
return false
Forward Chaining Example

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

- Agenda: \( A, B \)

- Annotate horn clauses with number of premises
Forward Chaining Example

- Process agenda item $A$
- Decrease count for horn clauses in which $A$ is premise
Forward Chaining Example

- Process agenda item $B$
- Decrease count for horn clauses in which $B$ is premise
- $A \land B \implies L$ has now fulfilled premise
- Add $L$ to agenda
Forward Chaining Example

- Process agenda item $L$
- Decrease count for horn clauses in which $L$ is premise
- $B \land L \implies M$ has now fulfilled premise
- Add $M$ to agenda
Forward Chaining Example

- Process agenda item $M$
- Decrease count for horn clauses in which $M$ is premise
- $L \land M \implies P$ has now fulfilled premise
- Add $P$ to agenda
Forward Chaining Example

- Process agenda item $P$
- Decrease count for horn clauses in which $P$ is premise
- $P \implies Q$ has now fulfilled premise
- Add $Q$ to agenda
- $A \land P \implies L$ has now fulfilled premise
Forward Chaining Example

- Process agenda item $P$
- Decrease count for horn clauses in which $P$ is premise
  - $P \implies Q$ has now fulfilled premise
- Add $Q$ to agenda
  - $A \land P \implies L$ has now fulfilled premise
- But $L$ is already inferred
Forward Chaining Example

- Process agenda item $Q$
- $Q$ is inferred
- Done
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 $\alpha$
backward chaining
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

- $A$ and $B$ are known to be true
- $Q$ needs to be proven
Backward Chaining Example

- Current goal: $Q$
- $Q$ can be inferred by $P \implies Q$
- $P$ needs to be proven
Backward Chaining Example

- Current goal: $P$
- $P$ can be inferred by $L \land M \implies P$
- $L$ and $M$ need to be proven
Backward Chaining Example

- Current goal: $L$
- $L$ can be inferred by $A \land P \implies L$
- $P$ is already a goal
- $A$ is already true
Backward Chaining Example

- Current goal: $L$
Backward Chaining Example

- Current goal: $L$
- $L$ can be inferred by $A \land B \implies L$
- Both are true
Backward Chaining Example

- Current goal: $L$
- $L$ can be inferred by $A \land B \implies L$
- Both are true

$\Rightarrow L$ is true
Backward Chaining Example

- Current goal: $M$
- $M$ can be inferred by $B \land L \implies M$
Backward Chaining Example

- Current goal: $M$
- $M$ can be inferred by $B \land L \implies M$
- Both are true

$\Rightarrow M$ is true
Backward Chaining Example

- Current goal: $P$
- $P$ can be inferred by $L \land M \implies P$
- Both are true

$\Rightarrow P$ is true
Backward Chaining Example

- Current goal: $Q$
- $Q$ can be inferred by $P \implies Q$
- $P$ is true

$\Rightarrow Q$ is true
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
resolution
Resolution

- Conjunction of disjunctions of literals

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.,

\[
\frac{P_{1,3} \lor P_{2,2}, \quad \neg P_{2,2}}{P_{1,3}}
\]

- Resolution is sound and complete for propositional logic
• Rules such as: “If breeze, then a pit adjacent.”

\[ B_{1,1} \iff (P_{1,2} \lor P_{2,1}) \]
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 \( \lnot \alpha \lor \beta \).
   
   \[ (\lnot B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\lnot (P_{1,2} \lor P_{2,1}) \lor B_{1,1}) \]

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

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

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

```plaintext
function PL-RESOLUTION(KB, \alpha) returns true or false
    inputs: KB, the knowledge base, a sentence in propositional logic
             \alpha, the query, a sentence in propositional logic
    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 
```

Philipp Koehn
Artificial Intelligence: Logical Agents
6 October 2015
Resolution Example

- To disprove: $\alpha = \neg P_{1,2}$

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

  reformulated as:
  $$(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1})$$

- Observation: $\neg B_{1,1}$

- Resolution

  $\frac{\neg P_{1,2} \lor B_{1,1}}{\neg B_{1,1}} \quad \frac{\neg B_{1,1}}{\neg P_{1,2}}$

- Resolution

  $\frac{\neg P_{1,2} \quad P_{1,2}}{false}$
Resolution Example

- In practice: all resolvable pairs of clauses are combined
Logical Agent

- Logical agent for Wumpus world explores actions
  - observe glitter → done
  - unexplored safe spot → plan route to it
  - if Wampus in possible spot → shoot arrow
  - take a risk to go possibly risky spot

- Propositional logic to infer state of the world

- Heuristic search to decide which action to take
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.

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

- Propositional logic lacks expressive power