Logical Agents

Chapter 7
Why Do We Need Logic?

• Problem-solving agents were very inflexible: hard code every possible state.

• Search is almost always exponential in the number of states.

• Problem solving agents cannot infer unobserved information.

• We want an agent that can reason similarly to humans.
Knowledge & Reasoning

To address these issues we will introduce

• A knowledge base (KB): a list of facts that are known to the agent.

• Rules to infer new facts from old facts using rules of inference.

• Logic provides the natural language for this.
Knowledge Bases

• Knowledge base:
  – set of *sentences* in a *formal* language.

• Declarative approach to building an agent:
  – *Tell* it what it needs to know.
  – *Ask* it what to do → answers should follow from the KB.
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:** 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 – things we do have an impact.
- **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

If the Wumpus were here, stench should be here. Therefore it is here. Since, there is no breeze here, the pit must be there.

We need rather sophisticated reasoning here!
Exploring a wumpsus world
Exploring a wumpus world
Exploring a wumpus world
Logic

• We used logical reasoning to find the gold.
• 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 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:

  \[ \text{KB} \models \alpha \]

- Knowledge base \( \text{KB} \) entails sentence \( \alpha \) if and only if \( \alpha \) is true in **all worlds** where \( \text{KB} \) is true
  
  - E.g., the KB containing “the Giants won and the Reds won” entails “The Giants won”.
  - E.g., \( x+y = 4 \) entails \( 4 = x+y \)
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 *α* if *α* is true in *m*.

- *M(α)* is the set of all models of *α*.

- Then KB ⊨ α iff *M(KB) ⊆ M(α)*
  - E.g. KB = Giants won and Reds won, α = Giants won

- Think of KB and α as collections of constraints and of models m as possible states. M(KB) are the solutions KB and M(α) the solutions to α. Then, KB ⊨ α when all solutions to KB are also solutions to α.
Entailment in the wumpus world

- Consider possible models for $KB$ assuming only pits and a reduced Wumpus world

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

All possible ways to fill in the ?’s.
Wumpus models

- $KB = \text{all possible wumpus-worlds consistent with the observations and the "physics" of the Wumpus world.}$
Wumpus models

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

\[ \alpha_2 = \text{"[2,2] is safe"}, \ KB \not\vdash \alpha_2 \]
Inference Procedures

- $KB \models_i \alpha =$ sentence $\alpha$ can be derived from $KB$ by procedure $i$

- **Soundness**: $i$ is sound if whenever $KB \models_i \alpha$, it is also true that $KB \models \alpha$ (*no wrong inferences, but maybe not all inferences*)

- **Completeness**: $i$ is complete if whenever $KB \not\models \alpha$, it is also true that $KB \models_i \alpha$ (*all inferences can be made, but maybe some wrong extra ones as well*)
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/world specifies true or false for each proposition symbol

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

\[
\begin{array}{ccc}
\text{false} & \text{true} & \text{false} \\
\end{array}
\]

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., \( S_1 \not\Rightarrow S_2 \) is false iff \( S_1 \) is true and \( S_2 \) is false

\( S_1 \Leftrightarrow S_2 \) is true iff \( S_1 \not\Rightarrow S_2 \) is true and \( S_2 \not\Rightarrow S_1 \) 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>$P \Rightarrow Q$</th>
<th>$P \iff Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</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>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>

**OR:** $P$ or $Q$ is true or both are true.

**XOR:** $P$ or $Q$ is true but not both.

**Implication is always true when the premises are False!**
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 $[i, j]$.

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

• "Pits cause breezes in adjacent squares"

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

- Enumeration of all models is sound and complete.
- For $n$ symbols, time complexity is $O(2^n)$...
- We need a smarter way to do inference!
- In particular, we are going to infer new logical sentences from the data-base and see if they match a query.
Logical equivalence

• To manipulate logical sentences we need some rewrite rules.

• Two sentences are logically equivalent iff they are true in same models: \( \alpha \equiv \beta \) iff \( \alpha \models \beta \) 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 ∨ ¬A,  A ⇒ A,  (A ∧ (A ⇒ B)) ⇒ B

Validity is connected to inference via the **Deduction Theorem**:

KB |= α if and only if (KB ⇒ α) is valid

A sentence is **satisfiable** if it is true in **some** model

e.g., A ∨ B,  C

A sentence is **unsatisfiable** if it is false in **all** models

e.g., A ∧ ¬A

Satisfiability is connected to inference via the following:

KB |= α if and only if (KB ∧ ¬α) is unsatisfiable

 THERE IS NO MODEL FOR WHICH KB=TRUE AND μ IS FALSE

Proof methods

• Proof methods divide into (roughly) two kinds:

  Application of inference rules:
  Legitimate (sound) generation of new sentences from old.
  – Resolution
  – Forward & Backward chaining

Model checking
Searching through truth assignments.
• Improved backtracking: Davis--Putnam-Logemann-Loveland (DPLL)
• Heuristic search in model space: Walksat.
Normal Form

We like to prove: $\text{KB} \models \alpha$

equivalent to: $\text{KB} \land \neg \alpha$ unsatisfiable

We first rewrite $\text{KB} \land \neg \alpha$ into conjunctive normal form (CNF).

A “conjunction of disjunctions”

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

Clause

Clause

• Any KB can be converted into CNF.
• In fact, any KB can be converted into CNF-3 using clauses with at most 3 literals.
Example: 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 \( \neg \alpha \lor \beta \).
   \[ (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg (P_{1,2} \lor P_{2,1}) \lor B_{1,1}) \]

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

4. Apply distributive 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

- Resolution: inference rule for CNF: sound and complete!

\[(A \lor B \lor C)\]
\[\neg A\]
\[\therefore (B \lor C)\]

“If A or B or C is true, but not A, then B or C must be true.”

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

“If A is false then B or C must be true, or if A is true then D or E must be true, hence since A is either true or false, B or C or D or E must be true.”

\[(A \lor B)\]
\[\neg A \lor B\]
\[\therefore (B \lor B) \equiv B\]

Simplification
Resolution Algorithm

- The resolution algorithm tries to prove: $KB \models \alpha$ equivalent to $KB \land \neg \alpha$ unsatisfiable

- Generate all new sentences from KB and the query.
- One of two things can happen:

1. We find $P \land \neg P$ which is unsatisfiable. I.e. we can entail the query.

2. We find no contradiction: there is a model that satisfies the sentence $KB \land \neg \alpha$ (non-trivial) and hence we cannot entail the query.
Resolution example

• $KB = (B_{1,1} \iff (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1}$
• $\alpha = \neg P_{1,2}$
Horn Clauses

• Resolution can be exponential in space and time.

• If we can reduce all clauses to “Horn clauses” resolution is linear in space and time.

A clause with at most 1 positive literal.

\[ A \lor \neg B \lor \neg C \]

• Every Horn clause can be rewritten as an implication with
  a conjunction of positive literals in the premises and a single
  positive literal as a conclusion.

\[ B \land C \Rightarrow A \]

• 1 positive literal: definite clause

• 0 positive literals: Fact or integrity constraint:

\[\neg (A \lor \neg B) \equiv (A \land B \Rightarrow False)\]

• Forward Chaining and Backward chaining are sound and complete
  with Horn clauses and run linear in space and time.
Try it Yourselves

• 7.9 page 238: (Adapted from Barwise and Etchemendy (1993).) If the unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical if it is horned.

• Derive the KB in normal form.
• Prove: Horned, Prove: Magical.
Forward chaining

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

Forward chaining is sound and complete for Horn KB
Forward chaining example

“OR” Gate

“AND” gate
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 $q$
  • check if $q$ is known already, or
  • prove by BC all premises of some rule concluding $q$
  • Hence BC maintains a stack of sub-goals that need to be proved to get to $q$.

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

we need P to prove L and L to prove P.
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
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
Model Checking

Two families of efficient algorithms:

• Complete backtracking search algorithms: DPLL algorithm

• Incomplete local search algorithms
  – WalkSAT algorithm
The DPLL algorithm

Determine if an input propositional logic sentence (in CNF) is satisfiable. **This is just backtracking search for a CSP.**

**Improvements:**

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. (if there is a model for \(S\), then making a pure symbol true is also a model).

3. **Unit clause heuristic**
   Unit clause: only one literal in the clause
   The only literal in a unit clause must be true.

   Note: literals can become a pure symbol or a unit clause when other literals obtain truth values. e.g. \((A \lor \text{True}) \land (\neg A \lor B)\)  \(A = \text{pure}\)
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 (5)
  \(n = \) number of symbols (5)

  – Hard problems seem to cluster near \(m/n = 4.3\) (critical point)
Hard satisfiability problems
Hard satisfiability problems

• Median runtime for 100 satisfiable random 3-CNF sentences, $n = 50$
Inference-based agents in the wumpus world

A wumpus-world agent using propositional logic:

\[-P_{1,1}\] (no pit in square [1,1])
\[-W_{1,1}\] (no Wumpus in square [1,1])

\[B_{x,y} \iff (P_{x,y+1} \lor P_{x,y-1} \lor P_{x+1,y} \lor P_{x-1,y})\] (Breeze next to Pit)

\[S_{x,y} \iff (W_{x,y+1} \lor W_{x,y-1} \lor W_{x+1,y} \lor W_{x-1,y})\] (stench next to Wumpus)

\[W_{1,1} \lor W_{1,2} \lor \ldots \lor W_{4,4}\] (at least 1 Wumpus)
\[-W_{1,1} \lor -W_{1,2}\] (at most 1 Wumpus)
\[-W_{1,1} \lor -W_{8,9}\]

\[\Rightarrow \text{64 distinct proposition symbols, 155 sentences}\]
Expressiveness limitation of propositional logic

- KB contains "physics" sentences for every single square

- For every time $t$ and every location $[x,y]$, $L_{x,y}^t \land FacingRight^t \land Forward^t \implies L_{x+1,y}^{t+1}$

  position $(x,y)$ at time $t$ of the agent.

- Rapid proliferation of clauses. First order logic is designed to deal with this through the introduction of variables.
2. (12 pts) **Inference**

Consider the following logical sentence in CNF form:

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

a. (2 pts) Is this sentence in Horn form?
Are all clauses definite clauses?

b. (4 pts) Use either resolution or forward chaining to show that this sentence is unsatisfiable. In other words, there is no assignment of False/True values for the variables A,B,C that will render the sentence True.

c. (2 pts) Next consider the following sentence:

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

Is this sentence in Horn form?

d. (4 pts) We will solve this problem using backtracking search. We use the following heuristics: 1) order the variables according to the number of clauses they are involved in (the variable involved in the largest number is first), 2) choose a value for a variable that satisfies the largest number of clauses.

Provide a solution (i.e. an assignment of values True/False to the variables such that all clauses are satisfied).
2. (12 pts) **Inference**

Consider the following logical sentence in CNF form:

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

a. (2 pts) Is this sentence in Horn form?
   Are all clauses definite clauses?
   a) answer: Yes, No (last clause).

b. (4 pts) Use either resolution or forward chaining to show that this sentence is unsatisfiable. In other words, there is no assignment of False/True values for the variables A,B,C that will render the sentence True.

c. (2 pts) Next consider the following sentence:

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

Is this sentence in Horn form?

d. (4 pts) We will solve this problem using backtracking search. We use the following heuristics: 1) order the variables according to the number of clauses they are involved in (the variable involved in the largest number is first), 2) choose a value for a variable that satisfies the largest number of clauses.

Provide a solution (i.e., an assignment of values True/False to the variables such that all clauses are satisfied).
2. (12 pts) **Inference**

Consider the following logical sentence in CNF form:

\[ (-A \lor \neg B \lor C) \land (-A \lor B) \land (A) \land (-A \lor \neg B \lor \neg C). \]

a. (2 pts) Is this sentence in Horn form?
Are all clauses definite clauses?

a) answer: Yes, No (last clause).

b. (4 pts) Use either resolution or forward chaining to show that this sentence is unsatisfiable. In other words, there is no assignment of False/True values for the variables A,B,C that will render the sentence True.

b) answer: FC: First rewrite all clauses into implications, and then derive A,B,C, False in that order. With resolution you use derive A,B,C in the same order and then show that you get an empty clause.

c. (2 pts) Next consider the following sentence:

\[ (-A \lor \neg B \lor C) \land (-A \lor B) \land (A \lor \neg B \lor \neg C) \land (A \lor B) \land (A \lor \neg C). \]
Is this sentence in Horn form?

d. (4 pts) We will solve this problem using backtracking search. We use the following heuristics: 1) order the variables according to the number of clauses they are involved in (the variable involved in the largest number is first), 2) choose a value for a variable that satisfies the largest number of clauses.
Provide a solution (i.e. an assignment of values True/False to the variables such that all clauses are satisfied).
2. (12 pts) **Inference**

Consider the following logical sentence in CNF form:

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

a. (2 pts) Is this sentence in Horn form?
   Are all clauses definite clauses?
   a) answer: Yes, No (last clause).

b. (4 pts) Use either resolution or forward chaining to show that this sentence is unsatisfiable. In other words, there is no assignment of False/True values for the variables A, B, C that will render the sentence True.

b) answer: FC: First rewrite all clauses into implications, and then derive A, B, C, False in that order. With resolution you use derive A, B, C in the same order and then show that you get an empty clause.

c. (2 pts) Next consider the following sentence:

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

Is this sentence in Horn form?

e. answer: No (fourth clause).

d. (4 pts) We will solve this problem using backtracking search. We use the following heuristics: 1) order the variables according to the number of clauses they are involved in (the variable involved in the largest number is first), 2) choose a value for a variable that satisfies the largest number of clauses.

Provide a solution (i.e. an assignment of values True/False to the variables such that all clauses are satisfied).
2. (12 pts) **Inference**

Consider the following logical sentence in CNF form:

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

a. (2 pts) Is this sentence in Horn form?
Are all clauses definite clauses?

a) answer: Yes, No (last clause).

b. (4 pts) Use either resolution or forward chaining to show that this sentence is unsatisfiable. In other words, there is no assignment of False/True values for the variables A,B,C that will render the sentence True.

b) answer: FC: First rewrite all clauses into implications, and then derive A,B,C,False in that order. With resolution you use derive A,B,C in the same order and then show that you get an empty clause.

c. (2 pts) Next consider the following sentence:

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

Is this sentence in Horn form?

c) answer: No (fourth clause).

d. (4 pts) We will solve this problem using backtracking search. We use the following heuristics: 1) order the variables according to the number of clauses they are involved in (the variable involved in the largest number is first), 2) choose a value for a variable that satisfies the largest number of clauses.

Provide a solution (i.e. an assignment of values True/False to the variables such that all clauses are satisfied).

d) answer: A=F, B=T, C=F (no backtracking needed).
2. (10 pts) **Propositional Logic**
Consider the following knowledge base: \( KB_0 = \neg A \lor \neg B \lor C \) and the sentence \( \alpha = \neg A \lor \neg B \)

a. (2 pts) Use De Morgan’s law to rewrite \( \neg \alpha = \neg(\neg A \lor \neg B) \)

b. (2 pts) Consider the updated KB: \( KB_1 = KB_0 \land \neg \alpha \). Is the \( KB_1 \) in Horn form? Are all clauses in \( KB_1 \) definite clauses?

c. (4 pts) Use resolution to prove \( C = true \).

d. (2 pts) Is \( \alpha \) entailed by \( KB_0 \), i.e. \( KB_0 \models \alpha \)?
2. (10 pts) **Propositional Logic**

Consider the following knowledge base: $KB_0 = \neg A \lor \neg B \lor C$ and the sentence $\alpha = \neg A \lor \neg B$

a. (2 pts) Use De Morgan’s law to rewrite $\neg \alpha = \neg(\neg A \lor \neg B)$

   a) answer: $\neg(\neg A \lor \neg B) = A \land B$.

b. (2 pts) Consider the updated KB: $KB_1 = KB_0 \land \neg \alpha$. Is the $KB_1$ in Horn form? Are all clauses in $KB_1$ definite clauses?

c. (4 pts) Use resolution to prove $C = true$.

d. (2 pts) Is $\alpha$ entailed by $KB_0$, i.e. $KB_0 \models \alpha$?
2. (10 pts) **Propositional Logic**
Consider the following knowledge base: \( KB_0 = \neg A \lor \neg B \lor C \) and the sentence \( \alpha = \neg A \lor \neg B \)

a. (2 pts) Use De Morgan’s law to rewrite \( \neg \alpha = \neg (\neg A \lor \neg B) \)

a) answer: \( \neg (\neg A \lor \neg B) = A \land B \).

b. (2 pts) Consider the updated KB: \( KB_1 = KB_0 \land \neg \alpha \). Is the \( KB_1 \) in Horn form? Are all clauses in \( KB_1 \) definite clauses?

b) answer: Yes and yes. There are 3 clauses, each clause contains exactly 1 positive literal.

c. (4 pts) Use resolution to prove \( C = \text{true} \).

d. (2 pts) Is \( \alpha \) entailed by \( KB_0 \), i.e. \( KB_0 \models \alpha \)?
2. (10 pts) Propositional Logic
Consider the following knowledge base: $KB_0 = \neg A \lor \neg B \lor C$ and the sentence $\alpha = \neg A \lor \neg B$

a. (2 pts) Use De Morgan’s law to rewrite $\neg \alpha = \neg (\neg A \lor \neg B)$
   a) answer: $\neg (\neg A \lor \neg B) = A \land B$.

b. (2 pts) Consider the updated KB: $KB_1 = KB_0 \land \neg \alpha$. Is the $KB_1$ in Horn form? Are all clauses in $KB_1$ definite clauses?
   b) answer: Yes and yes. There are 3 clauses, each clause contains exactly 1 positive literal.

c. (4 pts) Use resolution to prove $C = true$.
   c) answer: First Use $A$ and $\neg A \lor \neg B \lor C$ to conclude $\neg B \lor C$. Then use that and $B$ to conclude $C$.

d. (2 pts) Is $\alpha$ entailed by $KB_0$, i.e. $KB_0 \models \alpha$?
2. (10 pts) **Propositional Logic**
Consider the following knowledge base: \( KB_0 = \neg A \lor \neg B \lor C \) and the sentence \( \alpha = \neg A \lor \neg B \)

a. (2 pts) Use De Morgan’s law to rewrite \( \neg \alpha = \neg (\neg A \lor \neg B) \)

a) answer: \( \neg (\neg A \lor \neg B) = A \land B \).

b. (2 pts) Consider the updated KB: \( KB_1 = KB_0 \land \neg \alpha \). Is the \( KB_1 \) in Horn form? Are all clauses in \( KB_1 \) definite clauses?

b) answer: Yes and yes. There are 3 clauses, each clause contains exactly 1 positive literal.

c. (4 pts) Use resolution to prove \( C = \text{true} \).

c) answer: First use \( A \) and \( \neg A \lor \neg B \lor C \) to conclude \( \neg B \lor C \). Then use that and \( B \) to conclude \( C \).

d. (2 pts) Is \( \alpha \) entailed by \( KB_0 \), i.e. \( KB_0 \models \alpha \)?

d) answer: No: we have just shown that \( KB_1 = KB_0 \land \neg \alpha \) has a solution and is therefore not unsatisfiable.
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