CS 771
Artificial Intelligence
Propositional Logic
Why Do We Need Logic?

- Problem-solving agents were very inflexible
  - hard code every possible state
  - E.g., in the transition of 8-puzzle problem, knowledge of what each action can do is hidden inside the RESULT function

- Logic is used as a general class of representations to support knowledge based agents or logical agents

- Logical agents to some extent resembles reasoning in humans
Knowledge-Based Agents

- **KB = knowledge base**
  - A set of sentences or facts
  - e.g., a set of statements in a logic language
  - There must be a way to add new sentence to the knowledge base (TELL) and a way to query what is known (ASK)

- **Inference**
  - Deriving new sentences from old
  - e.g., using a set of logical statements to infer new ones

- **Each time a knowledge based agent program is called, it does three things**
  - It TELLS the knowledge base what it perceives
  - It ASKS the knowledge base what action it should perform
    - In the process of answering the query, extensive reasoning may be done about the current state of the world, about the outcome of possible action sequence etc
  - Agent program TELLS the knowledge base which action was chosen, and the agent executes the action
Types of Logics

- **Propositional logic** deals with specific objects and concrete statements that are either true or false
  - E.g., John is married to Sue.

- **Predicate logic (also called first order logic, first order predicate calculus)** allows statements to contain variables, functions, and quantifiers
  - For all X, Y: If X is married to Y then Y is married to X

- Next we will show an environment where logic based agents can show their worth
Wumpus World PEAS description

- **Performance measure**
  - gold: +1000, death: -1000
  - -1 per step, -10 for using the arrow
  - Game ends when
    - either agent dies
    - or climbs out of tree
Wumpus World PEAS description

• 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
  – Agent always starts in square labeled [1, 1] facing right
  – Location of wumpus and gold are chosen randomly, with a uniform distribution, from the squares other than the start square
  – Each square other than the start square can be a pit, with probability 0.2
Wumpus World PEAS description

- **Actuators:** Agent can move
  - Forward
  - Turnleft by 90 degrees
  - Turnright by 90 degrees
  - Grab
    - can be used to pick up the gold if it is in the same square as the agent
  - Shoot
    - can be used to fire an arrow in a straight line in the direction the agent is facing
    - The arrow continues until either it hits (and kills) the Wumpus or hits a wall
    - The agent has only one arrow, so only the first shoot action has any effect
  - Climb
    - Can be used to climb out of the cave but only from square [1,1]

- **Agent dies if it enters a square containing a pit or a live Wumpus**
Wumpus World PEAS description

- Agent has five sensors
  - In the squares containing the wumpus and in directly adjacent squares, agent will perceive a Stench
  - In the square directly adjacent to a pit, the agent will perceive a Breeze
  - In the square where the gold is, the agent will perceive a Glitter
  - When an agent walks into a wall, it will perceive a Bump
  - When a wumpus is killed, it emits a woeful Scream that can be perceived anywhere in the case
  - A typical sensor percept is
    - [Stench, Breeze, None, None, None]
Exploring wumpus world

- Initially the agent is ignorant regarding the configuration of the environment
- Overcoming this ignorance requires logical reasoning
- In most cases it is possible for the agent to retrieve gold safely
- About 21% of the environments are utterly unfair because gold is in a pit or is surrounded by pits
Exploring wumpus world

- Agent’s initial knowledge base contains rule of the environment and in addition it knows that it is in [1,1] and that [1,1] is a safe square
  - We denote this with an “A” and “OK” in cell [1,1]
- The first sensor percept is
  - [None, None, None, None, None]
  - From this the agent knows that squares [1,2] and [2,1] are free of dangers
Exploring wumpus world

• A cautious agent will only move to a square that knows to be OK
• Suppose agent moves to square [2,1]
• Agent perceives breeze (denoted by “B”) so there must be a pit in a neighboring square
  • Pit can not be in [1,1] by rules of the game
  • So pit must be in [2,2] or [3,1] or both
  • We denote it by notation “P?”

<table>
<thead>
<tr>
<th></th>
<th>1,1</th>
<th>2,1</th>
<th>3,1</th>
<th>4,1</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td></td>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OK</td>
<td></td>
<td>OK</td>
<td>B</td>
<td>P?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>1,4</th>
<th>2,4</th>
<th>3,4</th>
<th>4,4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,1</td>
<td>2,2</td>
<td></td>
<td>3,2</td>
<td>4,2</td>
</tr>
<tr>
<td>1,2</td>
<td></td>
<td></td>
<td>3,3</td>
<td>4,3</td>
</tr>
<tr>
<td>1,3</td>
<td>2,3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
At this point agent knows square [1,2] is okay so it moves there.

Agent perceives stench in [1,2] (denoted by “S”) so there must be a wumpus in a neighboring square.

- Wumpus can not be in [1,1] by rules of the game.
- It can not be in [2,2] (or agent would have detected a stench while it was in [2,1]).
- Therefore agent can infer that wumpus is in [1,3].
- We use “W!” to denote this inference.
- Moreover, lack of breeze in [1,2] implies that there is no pit in [2,2].
- This means there must be a pit in [3,1].
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" or interpretation of sentences;
  – connects symbols to real events in the world,
  – 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 \( x = 0, y = 6 \)
Model

• Semantics defines the truth of each sentence with respect to each possible world
• We will use the term model in place of “possible world”
• Possible world might be thought of as real environments that the agent might or might not be in, models are mathematical abstractions each of which simply fixes the truth of falsehood of every relevant sentence
• Example: x men and y women sitting at a table and playing bridge and the sentence \(x+y=4\) is true when there are 4 people in total
  – Possible models are all possible assignments of real numbers to variable \(x\) and \(y\)
  – The above sentence is true in the model \{(0,4), (1,3),(2,2), (3,1),(4,0)\}
• If a sentence \(\alpha\) is true in a model \(m\), we say \(m\) satisfies \(\alpha\) or sometimes \(m\) is a model of \(\alpha\)
  – We use the notation \(M(\alpha)\) to denote set of all models of \(\alpha\)
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 \( 4 = x+y \)
  - E.g., “Mary is Sue’s sister and Amy is Sue’s daughter” entails “Mary is Amy’s aunt.”
If KB is true in the real world, then any sentence $\alpha$ entailed by KB is also true in the real world.
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 \( KB \models \alpha \) iff \( M(KB) \subseteq M(\alpha) \)
  - Every model in which KB is true, \( \alpha \) is also true.
  - E.g. \( KB = \) Giants won and Reds won \( \alpha = \) Giants won.

- Think of KB and \( \alpha \) as collections of constraints and of models \( m \) as possible states. \( M(KB) \) are the solutions to KB and \( M(\alpha) \) the solutions to \( \alpha \).
  Then, \( KB \models \alpha \) when all solutions to KB are also solutions to \( \alpha \).
Wumpus models

- Consider the situation when agent has detected nothing in $[1,1]$ and a breeze in $[2,1]$
  - These percepts combined with agents' knowledge of rules of the wumpus world constitute the KB

- The agent is interested in whether the adjacent squares $[1,2], [2,2]$ and $[3,1]$ contain pits

- Each of three squares might or might not contain a pit, so there are $2^3=8$ possible models
• $KB = \text{all possible wumpus-worlds consistent with the observations and the }$
  \text{“physics” of the Wumpus world}

• $KB$ is false in any model in which [1,2] contains a pit, because there is no breeze in
  [1,1]

• There are only 3 models in which $KB$ is true
Wumpus models

- Consider a possible conclusion $\alpha_1 = \text{"there is no pit in[1,2]"}$
- In every model where $KB$ is true $\alpha_1$ is also true
- $KB \models \alpha_1$, proved by model checking
Wumpus models

- Consider another possible conclusion $\alpha_2 = \text{“There is no pit in [2,2]”}
- In some models where KB is true $\alpha_2$ is false
- $KB \nvdash \alpha_2$
- Agent can not conclude that there is no pit in [2,2]
Inference Procedures
(next lecture)

• \( KB \models_i \alpha \) = 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 \( KB \models \alpha \), it is also true that \( KB \models_i \alpha \) (*all inferences can be made, but maybe some wrong extra ones as well*)
Recap 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**)
Recap propositional logic: Semantics

Each model/world specifies true or false for each proposition symbol

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

false, true, false

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

Rules for evaluating truth with respect to a model \( m \):

\[ \neg S \text{ is true iff } S \text{ is false} \]

\[ S_1 \land S_2 \text{ is true iff } S_1 \text{ is true and } S_2 \text{ is true} \]

\[ S_1 \lor S_2 \text{ is true iff } S_1 \text{ is true or } S_2 \text{ is true} \]

\[ S_1 \Rightarrow S_2 \text{ is true iff } S_1 \text{ is false or } S_2 \text{ is true} \]

i.e., \( \neg S \) is false iff \( S_1 \) is true and \( S_2 \) is false

\[ S_1 \iff S_2 \text{ is true iff } S_1 \Rightarrow S_2 \text{ is true and } S_2 \Rightarrow S_1 \text{ is true} \]

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

\[ \neg P_{1,2} \land (P_{2,2} \lor P_{3,1}) = true \land (true \lor false) = true \land true = true \]
Recap 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>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!
A Simple Knowledgebase

- 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 $[i,j]$.

The knowledge base includes the following sentences, each one labeled for convenience:

- There is no pit in $[1,1]$:
  \[
  R_1 : \quad \neg P_{1,1} .
  \]

- A square is breezy if and only if there is a pit in a neighboring square. This has to be stated for each square; for now, we include just the relevant squares:
  \[
  R_2 : \quad B_{1,1} \iff (P_{1,2} \lor P_{2,1}) .
  \]
  \[
  R_3 : \quad B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1}) .
  \]

- The preceding sentences are true in all wumpus worlds. Now we include the breeze percepts for the first two squares visited in the specific world the agent is in, leading up to the situation in Figure 7.3(b).
  \[
  R_4 : \quad \neg B_{1,1} .
  \]
  \[
  R_5 : \quad B_{2,1} .
  \]

The knowledge base, then, consists of sentences $R_1$ through $R_5$. It can also be considered as a single sentence—the conjunction $R_1 \land R_2 \land R_3 \land R_4 \land R_5$—because it asserts that all the individual sentences are true.
A Simple knowledge base

<table>
<thead>
<tr>
<th></th>
<th>1,4</th>
<th>2,4</th>
<th>3,4</th>
<th>4,4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,3</td>
<td>2,3</td>
<td>3,3</td>
<td>4,3</td>
<td></td>
</tr>
<tr>
<td>1,2</td>
<td>P?</td>
<td>3,2</td>
<td>4,2</td>
<td></td>
</tr>
<tr>
<td>OK</td>
<td>V</td>
<td>2,2</td>
<td>3,1</td>
<td>4,1</td>
</tr>
<tr>
<td>OK</td>
<td>OK</td>
<td>A</td>
<td>B</td>
<td>P?</td>
</tr>
</tbody>
</table>
A Simple Inference Procedure

• Our goal is to decide whether $\text{KB} \models \alpha$ for some $\alpha$
  – We are interested in $\alpha = \neg P_{1,2}$
• With 7 variables there are $2^7 = 128$ models, in three of those, $\text{KB}$ is true, in those three models $\neg P_{1,2}$ is true
Inference by enumeration
(generate the truth table)

• 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 \( \beta \models \alpha \)

\[
\begin{align*}
(\alpha \land \beta) & \equiv (\beta \land \alpha) \quad \text{commutativity of } \land \\
(\alpha \lor \beta) & \equiv (\beta \lor \alpha) \quad \text{commutativity of } \lor \\
((\alpha \land \beta) \land \gamma) & \equiv (\alpha \land (\beta \land \gamma)) \quad \text{associativity of } \land \\
((\alpha \lor \beta) \lor \gamma) & \equiv (\alpha \lor (\beta \lor \gamma)) \quad \text{associativity of } \lor \\
\neg(\neg \alpha) & \equiv \alpha \quad \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) & \equiv (\neg \beta \Rightarrow \neg \alpha) \quad \text{contraposition} \\
(\alpha \Rightarrow \beta) & \equiv (\neg \alpha \lor \beta) \quad \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) & \equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination} \\
\neg(\alpha \land \beta) & \equiv (\neg \alpha \lor \neg \beta) \quad \text{de Morgan} \\
\neg(\alpha \lor \beta) & \equiv (\neg \alpha \land \neg \beta) \quad \text{de Morgan} \\
(\alpha \land (\beta \lor \gamma)) & \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)) \quad \text{distributivity of } \land \text{ over } \lor \\
(\alpha \lor (\beta \land \gamma)) & \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) \quad \text{distributivity of } \lor \text{ over } \land
\end{align*}
\]
Ontology:
What kind of things exist in the world?
What do we need to describe and reason about?

Reasoning

Representation
A Formal Symbol System

Syntax
What is said

Semantics
What it means

Inference
Formal Pattern Matching

Schema
Rules of Inference

Execution
Search Strategy

Preceding lecture
This lecture
Review

• Definitions:
  – Syntax, Semantics, Sentences, Propositions, Entails, Follows, Derives, Inference, Sound, Complete, Model

• Syntactic Transformations:
  – E.g., \((A \implies B) \iff (\neg A \lor B)\)

• Truth Tables
  – Negation, Conjunction, Disjunction, Implication, Equivalence (Biconditional)
  – Inference by Model Enumeration
If KB is true in the real world, then any sentence $\alpha$ entailed by KB is also true in the real world.
So --- how do we keep it from “Just making things up.”?

Is this inference correct?
How do you know?
How can you tell?

How can we make correct inferences?
How can we avoid incorrect inferences?
If KB is true in the real world, then any sentence \( \alpha \) derived from KB by a sound inference procedure is also true in the real world.
Logical inference

• The notion of entailment can be used for logic inference.
  – Model checking (see wumpus example):
    enumerate all possible models and check whether \( \alpha \) is true.

• **Sound** (or truth preserving):
  The algorithm **only** derives entailed sentences.
  – Otherwise it just makes things up.
    
    i is sound iff whenever \( KB \vdash \alpha \) it is also true that \( KB \models \alpha \)
  – E.g., model-checking is sound

• **Complete**:
  The algorithm can derive **every** entailed sentence.
  
  i is complete iff whenever \( KB \models \alpha \) it is also true that \( KB \not\vdash \alpha \)
Validity and satisfiability

• A sentence is **valid** if it is true in all models
  – e.g., $P \lor \neg P$
  – Valid sentences are **tautologies** : they are necessarily true
  – Because the sentence True is true in all models, every valid sentence is logically equivalent to True

• Deduction Theorem ( or why validity is important)
  – For any sentence $\alpha$ and $\beta$, $\alpha \vdash \beta$ if and only if $(\alpha \implies \beta)$ is valid

• A sentence is **satisfiable**, if it is true in, or satisfied by, some model
  – For the knowledge base given earlier, $(R_1 \land R_2 \land R_3 \land R_4 \land R_5)$ is satisfiable because there are three models in which it is true

• Validity and satisfiability are related
  – $\alpha$ is valid iff $\neg \alpha$ is unsatisfiable
  – $\alpha$ is satisfiable iff $\neg \alpha$ is not valid
  – $\alpha \vdash \beta$ if and only if the sentence $(\alpha \land \neg \beta)$ is unsatisfiable i.e., (there is no model in which $\alpha$ is True and $\beta$ is false
Proof by contradiction

- We have seen that, $\alpha \not\models \beta$ if and only if the sentence $(\alpha \land \neg \beta)$ is unsatisfiable, i.e., (there is no model in which $\alpha$ is True and $\beta$ is false)

- Proving $\beta$ from $\alpha$ by checking unsatisfiability of $(\alpha \land \neg \beta)$ corresponds exactly to the standard mathematical proof technique “proof by contradiction”
  - One assumes a sentence $\beta$ to be false
  - Then show that this leads to a contradiction with known axioms $\alpha$
    - $\neg \beta$ is true
    - If $(\alpha \land \neg \beta)$ unsatisfiable then $\alpha$ must be false
    - But this is a contradiction since we know axiom $\alpha$ is true
  - This contradiction is exactly what is meant by saying that $(\alpha \land \neg \beta)$ is unsatisfiable
Monotonic property of logical systems

• Set of entailed sentence can only increase as information is added to the knowledge base
  – For any $\alpha$ and $\beta$, if $\text{KB} \vDash \alpha$ then $\text{KB} \land \beta \vDash \alpha$
  – How do you prove it?
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.
Inference Rules

• Reasoning or Inference uses KB statements to reach new conclusions
• Inference rules can be written as:

\[
\text{antecedent} \quad \Rightarrow \quad \text{consequent}
\]

- If the KB contains the antecedent, you can add the consequent (the KB entails the consequent)
- Whenever antecedent is given as true, consequent can be inferred (consequent is true)
Commonly Used Inference Rules

- **Modus Ponens**

  \[ \alpha_1 \Rightarrow \alpha_2, \alpha_1 \]

  \[ \alpha_2 \]

- **Modus Tollens**

  \[ \alpha_1 \Rightarrow \alpha_2, \neg \alpha_2 \]

  \[ \neg \alpha_1 \]

- **Unit Resolution**

  \[ \alpha_1 \lor \alpha_2, \neg \alpha_2 \]

  \[ \alpha_1 \]

- **And Elimination**

  \[ \alpha_1 \land \alpha_2 \]

  \[ \alpha_2, \alpha_1 \]

- **Or Introduction**

  \[ \alpha_1 \]

  \[ \alpha_2 \Rightarrow \alpha_1 \land \alpha_2 \]

- **And Introduction**

  \[ \alpha_1, \alpha_2 \]

  \[ \alpha_2 \land \alpha_1 \]
Using Inference: Logical Proofs

• Starting with a KB ...
  – ASK(P1): is P1 true given what is in KB

• Derive P1 from the KB
  1. Use inference rules to add new statements
  2. Use logical equivalence to rewrite existing statements (see Fig 7.11)
Example

• 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.
  – How do you write the knowledge base?
  – Is the unicorn Horned?
  – Is the unicorn magical?
Proof by resolution

• All inference rules are sound but necessarily complete
• Resolution is a single inference rule that yields a complete inference algorithm when coupled with any complete search algorithm
• However to apply resolution technique its requires to represent KB as well as any sentence $\alpha$ that we wish to derive in a special format known as Conjunctive Normal Form (CNF)
Conjunctive Normal Form

We’d like to prove: $KB \models \alpha$

equivalent to: $KB \land \neg \alpha$ unsatisfiable

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

A “conjunction of disjunctions”

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

• Any KB can be converted into CNF.
• In fact, any KB can be converted into CNF-3 using clauses with at most 3 literals.
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 is “refutation complete” in that it can prove the truth of any entailed sentence by refutation.
Example: 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 \( \neg \alpha \lor \beta \).
   \[ (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land ((\neg P_{1,2} \lor \neg P_{2,1}) \lor B_{1,1}) \]

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:
   \[ (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1}) \]
Example: Conversion to CNF

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

5. KB is the conjunction of all of its sentences (all are true), so write each clause (disjunct) as a sentence in KB:

\[ \cdots \]
\[ (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \]
\[ (\neg P_{1,2} \lor B_{1,1}) \]
\[ (\neg P_{2,1} \lor B_{1,1}) \]
\[ \cdots \]
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 (negated) query in CNF form
- Resolution rule is applied to the resulting clauses and added to the set of CNF clauses
- The process continues until one of two things can happen:

1. We find $\mathcal{P} \land \neg\mathcal{P}$ which is unsatisfiable (empty clause). I.e. we can entail the query (we are done)

2. There are no new clauses to be added, in which case we cannot entail the query.
Resolution example

- When the agent is in square [1,1], there is no breeze. Show that there is no Pit in square [1,2]
- $KB = (B_{1,1} \iff (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1}$
- $\alpha = \neg P_{1,2}$
Horn Clauses

- Resolution is complete but 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.

e.g. \( A \lor \neg B \lor \neg C \)

- 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.

  e.g. \( B \land C \Rightarrow A \)

- Horn clauses are closed under resolution
  - If you resolve two Horn clauses you get back a Horn clause

- **Definite clause**
  - Disjunction of literals of which exactly one is positive
  - All definite clauses are Horn clauses
Horn clauses are important for three reasons

- Every definite clause (and so Horn clause) can be written as implication whose premise is a conjunction of positive literals and whose conclusion is a single positive literal
  - In the Horn form, the premise is called the body and the conclusion is called the head
  - A sentence consisting of a single positive literal is called a fact
- Inference with Horn clauses can be done through the forward chaining and backward chaining algorithms
  - Both these algorithms are natural in that the inference steps are obvious and easy for humans to follow
- Deciding entailment with Horn clauses can be done in time that is linear in the size of the knowledge base (KB)
Forward chaining (FC)

- **Idea:** fire any rule whose premises are satisfied in the $KB$, add its conclusion to the $KB$, until query is found. Start with the known facts, i.e., single positive literals

$$KB \Rightarrow Q$$

- This proves that $Q$ is true in all possible worlds (i.e. trivial), and hence it proves entailment.

$$P \Rightarrow Q$$
$$L \land M \Rightarrow P$$
$$B \land L \Rightarrow M$$
$$A \land P \Rightarrow L$$
$$A \land B \Rightarrow L$$

$A$

$B$

- 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 (BC)

Idea: work backwards from the query $q$

- check if $q$ is known already (no work needed), or
- prove by BC all premises of some rule whose conclusion is $q$
- 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

As soon as you can move forward, do so.
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? What shall I do now?

• Complexity of BC can be **much less** than linear in size of KB