Adversarial Search
&
Logic and Reasoning

CSEP 573
Recall from Last Time: Adversarial Games as Search

Convention: first player is called MAX,
2nd player is called MIN
MAX moves first and they take turns until game is over
Winner gets reward, loser gets penalty
Utility values stated from MAX’s perspective
Initial state and legal moves define the game tree
MAX uses game tree to determine next move
Tic-Tac-Toe Example

MAX (X)

MIN (O)

MAX (X)

MIN (O)

TERMINAL

Utility

-1  0  +1
Optimal Strategy: Minimax Search

Find the contingent strategy for MAX assuming an infallible MIN opponent

Assumption: Both players play optimally!

Given a game tree, the optimal strategy can be determined by using the minimax value of each node (defined recursively):

\[
\text{MINIMAX-VALUE}(n) = \begin{cases} 
\text{UTILITY}(n) & \text{If } n \text{ is a terminal} \\
\max_{s \in \text{succ}(n)} \text{MINIMAX-VALUE}(s) & \text{If } n \text{ is a MAX node} \\
\min_{s \in \text{succ}(n)} \text{MINIMAX-VALUE}(s) & \text{If } n \text{ is a MIN node}
\end{cases}
\]
Two-Ply Game Tree
Two-Ply Game Tree
Two-Ply Game Tree

Minimax decision = $A_1$

Minimax maximizes the worst-case outcome for max
Is there anyway I could speed up this search?
Pruning trees

Minimax algorithm explores depth-first
Pruning trees
Pruning trees

No need to look at or expand these nodes!!
Pruning trees

MAX

MIN

3 12 8 2 14

X X
Pruning trees

MAX

MIN

The diagram shows a pruning tree with nodes labeled with numbers and conditions. The tree is rooted at the top with conditions for pruning:
- The node at the top is labeled with $\geq 3$.
- The node below $\geq 3$ is labeled with $\leq 2$.
- The nodes below $\leq 2$ are labeled with $X$.
- The node at the bottom right is labeled with $\leq 5$.
Pruning trees
Prune this tree!
Do we need to check this node?
No, because \( \max(-29, -37) = -29 \) and other children of min can only lower min's value of -37 (because of the min operation)
Another pruning opportunity!
Pruning can eliminate entire subtrees!
This form of tree pruning is known as alpha-beta pruning

alpha = the highest (best) value for MAX along path
beta = the lowest (best) value for MIN along path
Why is it called $\alpha$-\(\beta\$?

$\alpha$ is the value of the best (i.e., highest-value) choice found so far at any choice point along the path for $\text{max}$.

If $v$ is worse than $\alpha$, $\text{max}$ will avoid it

$\Rightarrow$ prune that branch

Define $\beta$ similarly for $\text{min}$.
The α-β algorithm
(minimax with four lines of added code)

```plaintext
function ALPHA-BETA-SEARCH(state) returns an action
  inputs: state, current state in game
  \( v \leftarrow \text{MAX-VALUE}(state, -\infty, +\infty) \)
  return the action in SUCCESSORS(state) with value \( v \)

function MAX-VALUE(state, \( \alpha, \beta \)) returns a utility value
  inputs: state, current state in game
  \( \alpha \), the value of the best alternative for MAX along the path to state
  \( \beta \), the value of the best alternative for MIN along the path to state
  if TERMINAL-TEST(state) then return UTILITY(state)
  \( v \leftarrow -\infty \)
  for \( a, s \) in SUCCESSORS(state) do
    \( v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(s, \alpha, \beta)) \)
    if \( v \geq \beta \) then return \( v \)
    \( \alpha \leftarrow \text{MAX}(\alpha, v) \)
  return \( v \)
```

New

Pruning
The α-β algorithm (cont.)

```
function Min-Value(state, α, β) returns a utility value
    inputs: state, current state in game
            α, the value of the best alternative for MAX along the path to state
            β, the value of the best alternative for MIN along the path to state

    if Terminal-Test(state) then return Utility(state)
    v ← +∞
    for a, s in Successors(state) do
        v ← Min(v, Max-Value(s, α, β))
        if v ≤ α then return v
        β ← Min(β, v)
    return v
```
Does alpha-beta pruning change the final result? Is it an approximation?
Properties of $\alpha - \beta$

Pruning does not affect final result

Effectiveness of pruning can be improved through good move ordering
(e.g., in chess, captures $> \text{threats} > \text{forward moves}$ $> \text{backward moves}$)

With "perfect ordering," time complexity $= O(b^{m/2})$
   $\Rightarrow$ allows us to search deeper - doubles depth of search

A simple example of the value of reasoning about which computations are relevant (a form of metareasoning)
Good enough?

Chess:
- branching factor $b \approx 35$
- game length $m \approx 100$
- $\alpha$-$\beta$ search space $b^{m/2} \approx 35^{50} \approx 10^{77}$

The Universe:
- number of atoms $\approx 10^{78}$
- age $\approx 10^{21}$ milliseconds
Can we do better?

Strategies:

• search to a fixed depth (cut off search)
• iterative deepening search
Evaluation Function

- When search space is too large, create game tree up to a certain depth only.
- Art is to estimate utilities of positions that are not terminal states.
- Example of simple evaluation criteria in chess:
  - Material worth: pawn=1, knight =3, rook=5, queen=9.
  - Other: king safety, good pawn structure
  - Rule of thumb: 3-point advantage = certain victory

\[
eval(s) = \sum_{i} w_i \cdot \text{feature}_i(s)
\]

- \(w_1 \cdot \text{material}(s)\)
- \(w_2 \cdot \text{mobility}(s)\)
- \(w_3 \cdot \text{king safety}(s)\)
- \(w_4 \cdot \text{center control}(s)\)
Does it work in practice?

If \( b^m = 10^6 \) and \( b = 35 \) \( \Rightarrow \) \( m = 4 \)

4-ply lookahead is a hopeless chess player!

- 4-ply \( \approx \) human novice
- 8-ply \( \approx \) typical PC, human master
- 14-ply \( \approx \) Deep Blue, Kasparov
- 18-ply \( \approx \) Hydra (64-node cluster with FPGAs)
What about Games that Include an Element of Chance?

White has just rolled 6-5 and has 4 legal moves.
Game Tree for Games with an Element of Chance

- In addition to MIN- and MAX nodes, we include chance nodes (e.g., for rolling dice).

Search costs increase: Instead of $O(b^d)$, we get $O((bn)^d)$, where $n$ is the number of chance outcomes.

Expectiminimax Algorithm:
For chance nodes, compute expected value over successors.
Imperfect Information

E.g. card games, where opponents’ initial cards are unknown or Scrabble where letters are unknown

Idea: For all deals consistent with what you can see

- compute the minimax value of available actions for each of possible deals
- compute the expected value over all deals
Game Playing in Practice

- **Chess**: Deep Blue defeated human world champion Gary Kasparov in a 6 game match in 1997. Deep Blue searched 200 million positions per second, used very sophisticated evaluation functions, and undisclosed methods for extending some lines of search up to 40 ply.

- **Checkers**: Chinook ended 40 year reign of human world champion Marion Tinsley in 1994; used an endgame database defining perfect play for all positions involving 8 or fewer pieces on the board, a total of 443,748,401,247 positions (!).

- **Othello**: human champions refuse to play against computers because software is too good.

- **Go**: human champions refuse to play against computers because software is too bad.
Summary of Game Playing using Search

Basic idea: Minimax search (but can be slow)
Alpha-Beta pruning can increase max depth by factor up to 2
Limited depth search may be necessary
Static evaluation functions necessary for limited depth search
Opening and End game databases can help
Computers can beat humans in some games (checkers, chess, othello) but not in others (Go)
Next: Logic and Reasoning
"Thinking Rationally"

- Computational models of human “thought” processes
- Computational models of human behavior
- Computational systems that “think” rationally
- Computational systems that behave rationally
Logical Agents

Chess program calculates legal moves, but doesn't know that no piece can be on 2 different squares at the same time.

Logic (Knowledge-Based) agents combine general knowledge about the world with current percepts to infer hidden aspects of current state prior to selecting actions.

- Crucial in partially observable environments.
Outline

Knowledge-based agents
Wumpus world
Logic in general
Propositional logic
  • Inference, validity, equivalence and satisfiability
  • Reasoning
    – Resolution
    – Forward/backward chaining
**Knowledge Base**

Knowledge Base: set of sentences represented in a knowledge representation language

- stores assertions about the world

Inference rule: when one ASKS questions of the KB, the answer should follow from what has been TELLed to the KB previously.

![Diagram of Inference Engine and Knowledge Base with TELL and ASK arrows]
Generic KB-Based Agent

function KB-AGENT(\textit{percept}) \textbf{returns} an action
\begin{itemize}
  \item static: \textit{KB}, a knowledge base
  \item $t$, a counter, initially 0, indicating time
\end{itemize}

\textbf{TELL}(\textit{KB}, \textbf{MAKE-PERCEPT-SENTENCE}(\textit{percept}, t))
\textbf{action} $\leftarrow$ \textbf{ASK}(\textit{KB}, \textbf{MAKE-ACTION-QUERY}(t))
\textbf{TELL}(\textit{KB}, \textbf{MAKE-ACTION-SENTENCE}(\textbf{action}, t))
\textbf{t} $\leftarrow$ \textbf{t} + 1
\textbf{return} \textbf{action}
Abilities of a KB agent

Agent must be able to:

- Represent states and actions
- Incorporate new percepts
- Update internal representation of the world
- Deduce hidden properties of the world
- Deduce appropriate actions
Description level

Agents can be described at different levels

• Knowledge level
  - What they know, regardless of the actual implementation (Declarative description)

• Implementation level
  - Data structures in KB and algorithms that manipulate them, e.g., propositional logic and resolution
A Typical Wumpus World

- **Wumpus**
  - Row 3, Column 1

- **You (Agent)**
  - Row 1, Column 1

- **PIT**
  - Row 4, Column 3

- **Gold**
  - Row 3, Column 1

- **Stench**
  - Row 3, Column 1, Row 2, Column 1

- **Breeze**
  - Row 3, Column 3, Row 2, Column 3
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
Climbing in [1,1] gets agent out of the cave

Sensors  Stench, Breeze, Glitter, Bump, Scream

Actuators  TurnLeft, TurnRight, Forward, Grab, Shoot, Climb
Wumpus World Characterization

Observable?
Deterministic?
Episodic?
Static?
Discrete?
Single-agent?
Wumpus World Characterization

Observable? No, only local perception
Deterministic?
Episodic?
Static?
Discrete?
Single-agent?
Observable? No, only local perception
Deterministic? Yes, outcome exactly specified
Episodic?
Static?
Discrete?
Single-agent?
Wumpus World Characterization

Observable?  No, only local perception
Deterministic? Yes, outcome exactly specified
Episodic?  No, sequential at the level of actions
Static?
Discrete?
Single-agent?
Wumpus World Characterization

Observable? No, only local perception
Deterministic? Yes, outcome exactly specified
Episodic? No, sequential at the level of actions
Static? Yes, Wumpus and pits do not move
Discrete?
Single-agent?
Wumpus World Characterization

Observable? No, only local perception
Deterministic? Yes, outcome exactly specified
Episodic? No, sequential at the level of actions
Static? Yes, Wumpus and pits do not move
Discrete? Yes
Single-agent?
Wumpus World Characterization

Observable? No, only local perception
Deterministic? Yes, outcome 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 of the environment
Exploring the Wumpus World

[1,1] KB initially contains the rules of the environment. First percept is [none, none, none, none, none, none], move to safe cell e.g. 2,1
[2,1] Breeze which indicates that there is a pit in [2,2] or [3,1], return to [1,1] to try next safe cell
Exploring the Wumpus World

[1,2] Stench in cell which means that wumpus is in [1,3] or [2,2] but not in [1,1]
YET ... wumpus not in [2,2] or stench would have been detected in [2,1]
THUS ... wumpus must be in [1,3]
THUS [2,2] is safe because of lack of breeze in [1,2]
THUS pit in [3,1]
move to next safe cell [2,2]
Exploring the Wumpus World

[2,2] Move to [2,3]

[2,3] Detect glitter, smell, breeze
    Grab gold
    THUS pit in [3,3] or [2,4]
How do we represent rules of the world and percepts encountered so far?

Why not use logic?
What is a logic?

A formal language

- Syntax – what expressions are legal (well-formed)
- Semantics – what legal expressions mean
  - In logic the truth of each sentence evaluated with respect to each possible world

E.g. the language of arithmetic

- \( x+2 \geq y \) is a sentence, \( x2y+ = \) is not a sentence
- \( x+2 \geq y \) is true in a world where \( x=7 \) and \( y=1 \)
- \( x+2 \geq y \) is false in a world where \( x=0 \) and \( y=6 \)
How do we draw conclusions and deduce new facts about the world using logic?
Entailment

Knowledge Base = KB
Sentence \( \alpha \)

\[ KB \models \alpha \text{ (KB "entails" sentence } \alpha \text{) if and only if } \alpha \text{ is true in all worlds (models) where KB is true.} \]

E.g. \( x+y=4 \) entails \( 4=x+y \)
(because \( 4=x+y \) is true for all values of \( x, y \) for which \( x+y=4 \) is true)
Models and Entailment

$m$ is a model of a sentence $\alpha$ if $\alpha$ is true in $m$

e.g. $\alpha$ is “$4=x+y$” and $m = \{x=2, y=2\}$

$M(\alpha)$ is the set of all models of $\alpha$

Then $KB \models \alpha$ iff $M(KB) \subseteq M(\alpha)$

E.g. $KB = CSEP \ 573$ students are bored and
     CSEP 573 students are sleepy;
     $\alpha = CSEP \ 573$ students are bored
Wumpus world model

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 possible world models

- Diagrams of different possible world models for the Wumpus world.
Wumpus world models consistent with observations

\[ KB = \text{wumpus-world rules + observations} \]
Example of Entailment

Is [1,2] safe?
Example of Entailment

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

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

$M(KB) \subseteq M(\alpha_1)$
Another Example

Is [2,2] safe?

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

\[ K B = \text{wumpus-world rules} + \text{observations} \]

\[ \alpha_2 = \text{"[2,2] is safe"}, \quad K B \nvdash \alpha_2 \]

\[ M(KB) \nvdash M(\alpha_2) \]
Soundness and Completeness

If an inference algorithm only derives entailed sentences, it is called *sound* (or *truth preserving*). Otherwise it just makes things up.

- Algorithm \( i \) is sound if whenever \( KB \models i \alpha \) (i.e. \( \alpha \) is derived by \( i \) from \( KB \)) it is also true that \( KB \models \alpha \)

**Completeness:** An algorithm is complete if it can derive any sentence that is entailed.

\[ i \text{ is complete if whenever } KB \models \alpha \text{ it is also true that } KB \models i \alpha \]
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.
Propositional Logic: Syntax

Propositional logic is the simplest logic – illustrates basic ideas.

Atomic sentences = proposition symbols = A, B, P_{1,2}, P_{2,2} etc. used to denote properties of the world

- *Can be either True or False*

E.g. \( P_{1,2} = \) “There’s a pit in location [1,2]” is either true or false in the wumpus world
Propositional Logic: Syntax

Complex sentences constructed from simpler ones recursively

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

A model specifies true/false for each proposition symbol.

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

Rules for evaluating truth w.r.t. a model \( m \):

\[
\begin{align*}
\neg S & \quad \text{is true iff } S \text{ is false} \\
S_1 \land S_2 & \quad \text{is true iff } S_1 \text{ is true and } S_2 \text{ is true} \\
S_1 \lor S_2 & \quad \text{is true iff } S_1 \text{ is true or } S_2 \text{ is true} \\
S_1 \Rightarrow S_2 & \quad \text{is true iff } S_1 \text{ is false or } S_2 \text{ is true} \\
S_1 \Leftrightarrow S_2 & \quad \text{is true iff both } S_1 \Rightarrow S_2 \text{ and } S_2 \Rightarrow S_1 \text{ are true}
\end{align*}
\]
## 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>
Propositional Logic: Semantics

Simple recursive process can be used to evaluate an arbitrary sentence

E.g., Model: \( P_{1,2} \quad P_{2,2} \quad P_{3,1} \)
\[
\begin{array}{ccc}
\text{false} & \text{true} & \text{false} \\
\end{array}
\]

\[\neg P_{1,2} \land (P_{2,2} \lor P_{3,1}) = true \land (true \lor false) = true \land true = true\]
Example: Wumpus World

Proposition Symbols and Semantics:
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 $[i, j]$. 

![Diagram of the Wumpus World environment]
Wumpus KB

Knowledge Base (KB) includes the following sentences:

Statements currently known to be true:

\[ \lnot P_{1,1} \]
\[ \lnot B_{1,1} \]
\[ B_{2,1} \]

Properties of the world: E.g., "Pits cause breezes in adjacent squares"

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

(and so on for all squares)
Can a Wumpus-Agent use this logical representation and KB to avoid pits and the wumpus, and find the gold?

Is there no pit in [1,2]?

Does KB ⊨ ¬P_{1,2} ?
Inference by Truth Table Enumeration

\[
\text{¬}P_{1,2} \text{ true in all models in which KB is true.}
\]

Therefore, KB \models \text{¬}P_{1,2}
Another Example

Is there a pit in [2,2]?
Inference by Truth Table Enumeration

<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>$KB$</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>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>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>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>
</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>
</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>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</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>false</td>
</tr>
</tbody>
</table>

$P_{2,2}$ is false in a model in which $KB$ is true.

Therefore, $KB \not\models P_{2,2}$
Inference by TT Enumeration

Algorithm: Depth-first enumeration of all models (see Fig. 7.10 in text for pseudocode)

- Algorithm is sound & complete

For $n$ symbols:

time complexity = $O(2^n)$, space = $O(n)$
Two sentences are logically equivalent iff they are true in the same models: $\alpha \equiv \beta$ iff $\alpha \vdash \beta$ and $\beta \vdash \alpha$

\[
\begin{align*}
(\alpha \land \beta) & \equiv (\beta \land \alpha) \quad \text{commutativity of } \land \\
(\alpha \lor \beta) & \equiv (\beta \lor \alpha) \quad \text{commutativity of } \lor \\
((\alpha \land \beta) \land \gamma) & \equiv (\alpha \land (\beta \land \gamma)) \quad \text{associativity of } \land \\
((\alpha \lor \beta) \lor \gamma) & \equiv (\alpha \lor (\beta \lor \gamma)) \quad \text{associativity of } \lor \\
\neg(\neg\alpha) & \equiv \alpha \quad \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) & \equiv (\neg\beta \Rightarrow \neg\alpha) \quad \text{contraposition} \\
(\alpha \Rightarrow \beta) & \equiv (\neg\alpha \lor \beta) \quad \text{implication elimination} \\
(\alpha \iff \beta) & \equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination} \\
\neg(\alpha \land \beta) & \equiv (\neg\alpha \lor \neg\beta) \quad \text{de Morgan} \\
\neg(\alpha \lor \beta) & \equiv (\neg\alpha \land \neg\beta) \quad \text{de Morgan} \\
(\alpha \land (\beta \lor \gamma)) & \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)) \quad \text{distributivity of } \land \text{ over } \lor \\
(\alpha \lor (\beta \land \gamma)) & \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) \quad \text{distributivity of } \lor \text{ over } \land
\end{align*}
\]


Concepts for Other Techniques: Validity and Satisfiability

A sentence is **valid** if it is true in all models (a tautology)

\[ \text{e.g., True, } A \lor \neg A, \ A \Rightarrow A, \ (A \land (A \Rightarrow B)) \Rightarrow B \]

Validity is connected to inference via the Deduction Theorem:

\[ KB \models a \text{ if and only if } (KB \Rightarrow a) \text{ is valid} \]

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

\[ \text{e.g., } A \lor B, \ C \]

A sentence is **unsatisfiable** if it is true in no models

\[ \text{e.g., } A \land \neg A \]

Satisfiability is connected to inference via the following: \[ KB \models a \text{ if and only if } (KB \land \neg a) \text{ is unsatisfiable (proof by contradiction)} \]
Inference Techniques for Logical Reasoning
Inference/Proof Techniques

Two kinds (roughly):

Model checking
- Truth table enumeration (always exponential in $n$)
- Efficient backtracking algorithms
e.g., Davis-Putnam-Logemann-Loveland (DPLL)
- Local search algorithms (sound but incomplete)
e.g., randomized hill-climbing (WalkSAT)

Successive application of inference rules
- Generate new sentences from old in a sound way
- Proof = a sequence of inference rule applications
- Use inference rules as successor function in a standard search algorithm
Inference Technique I: Resolution

Terminology:

Literal = proposition symbol or its negation
  E.g., \( A, \neg A, B, \neg B \), etc.

Clause = disjunction of literals
  E.g., \( (B \lor \neg C \lor \neg D) \)

Resolution assumes sentences are in Conjunctive Normal Form (CNF): 

sentence = conjunction of clauses
  E.g., \( (A \lor \neg B) \land (B \lor \neg C \lor \neg D) \)
Conversion to CNF

E.g., $B_{1,1} \iff (P_{1,2} \lor P_{2,1})$

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

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

   This is in CNF – Done!
Resolution motivation

There is a pit in [1,3] or
There is a pit in [2,2]  There is no pit in [2,2]

There is a pit in [1,3]

More generally,

\[
\begin{align*}
\ell_1 \lor \cdots \lor \ell_k & \quad \neg \ell_i \\
\hline
\ell_1 \lor \cdots \lor \ell_{i-1} \lor \ell_{i+1} \lor \cdots \lor \ell_k
\end{align*}
\]
Inference Technique: Resolution

General Resolution inference rule (for CNF):

\[
\begin{align*}
\ell_1 \lor \ldots \lor \ell_k & \quad \quad \quad \quad \quad m_1 \lor \ldots \lor m_n \\
\ell_1 \lor \ldots \lor \ell_{i-1} \lor \ell_{i+1} \lor \ldots \lor \ell_k \lor m_1 \lor \ldots \lor m_{j-1} \lor m_{j+1} \ldots \lor m_n
\end{align*}
\]

where \( \ell_i \) and \( m_j \) are complementary literals (\( \ell_i = \neg m_j \))

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

\[
\begin{align*}
P_{1,3}
\end{align*}
\]

Resolution is sound and complete for propositional logic
Soundness

Proof of soundness of resolution inference rule:

\[ \neg (l_1 \lor \ldots \lor l_{i-1} \lor l_{i+1} \lor \ldots \lor l_k) \Rightarrow l_i \]
\[ \neg m_j \Rightarrow (m_1 \lor \ldots \lor m_{j-1} \lor m_{j+1} \lor \ldots \lor m_n) \]

\[ \neg (l_i \lor \ldots \lor l_{i-1} \lor l_{i+1} \lor \ldots \lor l_k) \Rightarrow (m_1 \lor \ldots \lor m_{j-1} \lor m_{j+1} \lor \ldots \lor m_n) \]

(since \( l_i = \neg m_j \))
Resolution algorithm

To show $KB \models \alpha$, use proof by contradiction, i.e., show $KB \land \neg \alpha$ unsatisfiable

```
function PL-RESOLUTION(KB, \alpha) returns true or false
    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 ∪ resolvents
        if new ⊆ clauses then return false
        clauses ← clauses ∪ new
    end do
```

PL-RESOLUTION can be shown to be complete (see text)
Resolution example

Given no breeze in $[1,1]$, prove there's no pit in $[1,2]$

$$KB = (B_{1,1} \iff (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1} \text{ and } \alpha = \neg P_{1,2}$$

Resolution: Convert to CNF and show $KB \land \neg \alpha$ is unsatisfiable
Resolution example
Resolution example

Empty clause
(i.e., KB ∧ ¬α unsatisfiable)
Inference Technique II: Forward/Backward Chaining

Require sentences to be in Horn Form:

- $KB = \text{conjunction of Horn clauses}$
  - Horn clause =
    - proposition symbol or
    - “(conjunction of symbols) ⇒ symbol”
      (i.e. clause with at most 1 positive literal)
  - E.g., $KB = C \land (B \Rightarrow A) \land ((C \land D) \Rightarrow B)$

F/B chaining is based on “Modus Ponens” rule:

- $\alpha_1, \ldots, \alpha_n \quad \alpha_1 \land \ldots \land \alpha_n \Rightarrow \beta$
- $\beta$
- Complete for Horn clauses

Very natural and linear time complexity in size of KB
Forward chaining

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

KB:

\[
\begin{align*}
P & \Rightarrow Q \\
L \land M & \Rightarrow P \\
B \land L & \Rightarrow M \\
A \land P & \Rightarrow L \\
A \land B & \Rightarrow L \\
A & \\
B & 
\end{align*}
\]

Query: “Is Q true?”

AND-OR Graph for KB
Forward chaining algorithm

```
function PL-FC-ENTAILS?(KB, q) returns true or false
    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 to be true

    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]    // Decrement # premises
                if count[c] = 0 then do
                    if HEAD[c] = q then return true
                    PUSH(HEAD[c], agenda)
            end
        end
    end
    return false
```

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

Query = Q
(i.e. “Is Q true?”)

# premises
Forward chaining example

A is known to be true
Forward chaining example

B is also known to be true

\[
\text{count} = 0; \quad \text{therefore, L is true}
\]

B is also known to be true
Forward chaining example

count = 0; therefore, M is true
Forward chaining example

count = 0; therefore, P is true
Forward chaining example

Query = Q
(i.e. “Is Q true?”)

\[
\text{count} = 0; \\
\text{therefore, Q is true}
\]
Backward chaining

Idea: work backwards from the query \( q \)
to prove \( q \):
check if \( q \) is known already, OR
prove by backward chaining all premises of
some rule concluding \( q \)

Avoid loops: check if new subgoal is already on 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
Forward vs. backward chaining

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

FC may do lots of work that is irrelevant to the goal

BC is goal-driven, appropriate for problem-solving  
  e.g., How do I get an A in this class?  
  e.g., What is my best exit strategy out of the classroom?  
  e.g., How can I impress my date tonight?

Complexity of BC can be much less than linear in size of KB
Next Class: More logic
&
Uncertainty

Note: No homework this week, HW #2 will be assigned next week