Warm-up:

What is the relationship between number of constraints and number of possible solutions?

In other words, as the number of the constraints increases, does the number of possible solutions:

A) Increase
B) Decrease
C) Stay the same
Announcements

Assignments:
- P2: Optimization
  - Due Thu 2/21, 10 pm

Midterm 1 Exam
- Mon 2/18, in class
- Recitation Fri is a review session
- See Piazza post for details

Alita Class Field Trip!
- Moved to Saturday, 2/23, afternoon

White card feedback
Warm-up:

What is the relationship between number of constraints and number of possible solutions?

In other words, as the number of the constraints increases, does the number of possible solutions:

A) Increase
B) Decrease
C) Stay the same

Where is the knowledge in our CSPs?
AI: Representation and Problem Solving

Propositional Logic

Instructors: Pat Virtue & Stephanie Rosenthal

Slide credits: CMU AI, http://ai.berkeley.edu
Logic Representation and Problem Solving

To honk or not to honk
Logical Agents

Logical agents and environments

Agent

Sensors

Knowledge Base

Inference

Actuators

Environment

Percepts

Actions
Wumpus World

Logical Reasoning as a CSP

- $B_{ij} =$ breeze felt
- $S_{ij} =$ stench smelt
- $P_{ij} =$ pit here
- $W_{ij} =$ wumpus here
- $G =$ gold

http://thiagodnf.github.io/wumpus-world-simulator/
A Knowledge-based Agent

function KB-AGENT(percept) returns an action

persistent: KB, a knowledge base
    t, an integer, initially 0

TELL(KB, PROCESS-PERCEPT(percept, t))

action ← ASK(KB, PROCESS-QUERY(t))

TELL(KB, PROCESS-RESULT(action, t))

t ← t + 1

return action
Logical Agents

So what do we TELL our knowledge base (KB)?

- **Facts** (sentences)
  - The grass is green
  - The sky is blue
- **Rules** (sentences)
  - Eating too much candy makes you sick
  - When you’re sick you don’t go to school
- **Percepts and Actions** (sentences)
  - Pat ate too much candy today

What happens when we ASK the agent?

- Inference – new sentences created from old
  - Pat is not going to school today
Logical Agents

Sherlock Agent

- Really good knowledge base
  - Evidence
  - Understanding of how the world works (physics, chemistry, sociology)

- Really good inference
  - Skills of deduction
  - “It’s elementary my dear Watson”

Dr. Strange?
Alan Turing?
Kahn?
Worlds

What are we trying to figure out?

- Who, what, when, where, why
- Time: past, present, future
- Actions, strategy
- Partially observable? Ghosts, Walls

Which world are we living in?
Models

How do we represent possible worlds with models and knowledge bases? How do we then do inference with these representations?
Wumpus World

Possible Models

- $P_{1,2}$  $P_{2,2}$  $P_{3,1}$
Wumpus World

Possible Models

- $P_{1,2} \ P_{2,2} \ P_{3,1}$

- Knowledge base
  - Nothing in [1,1]
  - Breeze in [2,1]
Wumpus World

Possible Models

- $P_{1,2} P_{2,2} P_{3,1}$

- Knowledge base
  - Nothing in [1,1]
  - Breeze in [2,1]

- Query $\alpha_1$:
  - No pit in [1,2]
Wumpus World

Possible Models

- $P_{1,2} P_{2,2} P_{3,1}$

- Knowledge base
  - Nothing in [1,1]
  - Breeze in [2,1]

- Query $\alpha_2$:
  - No pit in [2,2]
Logic Language

Natural language?

Propositional logic

- Syntax: \( P \lor (\neg Q \land R) \); \( X_1 \Leftrightarrow (\text{Raining} \Rightarrow \text{Sunny}) \)
- Possible world: \{P=true, Q=true, R=false, S=true\} or 1101
- Semantics: \( \alpha \land \beta \) is true in a world iff \( \alpha \) is true and \( \beta \) is true (etc.)

First-order logic

- Syntax: \( \forall x \exists y P(x,y) \land \neg Q(Joe,f(x)) \Rightarrow f(x)=f(y) \)
- Possible world: Objects \( o_1, o_2, o_3 \); \( P \) holds for \(<o_1,o_2>\); \( Q \) holds for \(<o_3>\); \( f(o_1)=o_1; Joe=o_3 \); etc.
- Semantics: \( \phi(\sigma) \) is true in a world if \( \sigma=o_j \) and \( \phi \) holds for \( o_j \); etc.
Propositional Logic
Propositional Logic

Symbol:
- Variable that can be true or false
- We’ll try to use capital letters, e.g. A, B, P_{1,2}
- Often include True and False

Operators:
- \( \neg A \): not A
- \( A \land B \): A and B (conjunction)
- \( A \lor B \): A or B (disjunction) Note: this is not an “exclusive or”
- \( A \Rightarrow B \): A implies B (implication). If A then B
- \( A \iff B \): A if and only if B (biconditional)

Sentences
Propositional Logic Syntax

Given: a set of proposition symbols \{X_1, X_2, \ldots, X_n\}

- (we often add True and False for convenience)

\(X_i\) is a sentence

If \(\alpha\) is a sentence then \(\neg\alpha\) is a sentence

If \(\alpha\) and \(\beta\) are sentences then \(\alpha \land \beta\) is a sentence

If \(\alpha\) and \(\beta\) are sentences then \(\alpha \lor \beta\) is a sentence

If \(\alpha\) and \(\beta\) are sentences then \(\alpha \Rightarrow \beta\) is a sentence

If \(\alpha\) and \(\beta\) are sentences then \(\alpha \Leftrightarrow \beta\) is a sentence

And p.s. there are no other sentences!
Notes on Operators

$\alpha \lor \beta$ is inclusive or, not exclusive
Truth Tables

\( \alpha \lor \beta \) is inclusive or, not exclusive

<table>
<thead>
<tr>
<th>( \alpha )</th>
<th>( \beta )</th>
<th>( \alpha \land \beta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \alpha )</th>
<th>( \beta )</th>
<th>( \alpha \lor \beta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
Notes on Operators

\( \alpha \lor \beta \) is **inclusive** or, not exclusive

\( \alpha \Rightarrow \beta \) is equivalent to \( \neg \alpha \lor \beta \)
- Says who?
Truth Tables

$\alpha \Rightarrow \beta$ is equivalent to $\neg \alpha \lor \beta$

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$\alpha \Rightarrow \beta$</th>
<th>$\neg \alpha$</th>
<th>$\neg \alpha \lor \beta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>
Notes on Operators

\( \alpha \lor \beta \) is inclusive or, not exclusive

\( \alpha \Rightarrow \beta \) is equivalent to \( \neg \alpha \lor \beta \)

- Says who?

\( \alpha \iff \beta \) is equivalent to \( (\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha) \)

- Prove it!
Truth Tables

$\alpha \iff \beta$ is equivalent to $(\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)$

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$\alpha \iff \beta$</th>
<th>$\alpha \Rightarrow \beta$</th>
<th>$\beta \Rightarrow \alpha$</th>
<th>$(\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

Equivalence: it’s true in all models. Expressed as a logical sentence:

$\left( \alpha \iff \beta \right) \iff \left( \left( \alpha \Rightarrow \beta \right) \land \left( \beta \Rightarrow \alpha \right) \right)$
A literal is an atomic sentence:

- True
- False
- Symbol
- $\neg$ Symbol
Monty Python Inference

There are ways of telling whether she is a witch
Sentences as Constraints

Adding a sentence to our knowledge base constrains the number of possible models:

KB: Nothing

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>Q</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>2</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>3</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>4</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>5</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>6</td>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>7</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>8</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>
Sentences as Constraints

Adding a sentence to our knowledge base constrains the number of possible models:

KB: Nothing

KB: [(P ∧ ¬Q) ∨ (Q ∧ ¬P)] \implies R

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>
Sentences as Constraints

Adding a sentence to our knowledge base constrains the number of possible models:

KB: Nothing

KB: \[(P \land \neg Q) \lor (Q \land \neg P)\] \Rightarrow R

KB: R, \[(P \land \neg Q) \lor (Q \land \neg P)\] \Rightarrow R

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>Q</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>1</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>2</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>3</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>4</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>5</td>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>6</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>7</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>
Sherlock Entailment

“When you have eliminated the impossible, whatever remains, however improbable, must be the truth” – Sherlock Holmes via Sir Arthur Conan Doyle

- Knowledge base and inference allow us to remove impossible models, helping us to see what is true in all of the remaining models
Entailment

Entailment: $\alpha \models \beta$ (“$\alpha$ entails $\beta$” or “$\beta$ follows from $\alpha$”) iff in every world where $\alpha$ is true, $\beta$ is also true

- I.e., the $\alpha$-worlds are a subset of the $\beta$-worlds [$\text{models}(\alpha) \subseteq \text{models}(\beta)$]

Usually we want to know if $\text{KB} \models \text{query}$

- $\text{models}(\text{KB}) \subseteq \text{models}(\text{query})$
- In other words
  - $\text{KB}$ removes all impossible models (any model where $\text{KB}$ is false)
  - If $\beta$ is true in all of these remaining models, we conclude that $\beta$ must be true

Entailment and implication are very much related

- However, entailment relates two sentences, while an implication is itself a sentence (usually derived via inference to show entailment)
Wumpus World

Possible Models

- $P_{1,2}$ $P_{2,2}$ $P_{3,1}$

- Knowledge base
  - Nothing in [1,1]
  - Breeze in [2,1]
Wumpus World

Possible Models

- $P_{1,2} P_{2,2} P_{3,1}$

- Knowledge base
  - Nothing in [1,1]
  - Breeze in [2,1]

- Query $\alpha_1$:
  - No pit in [1,2]
Wumpus World

Possible Models

- $P_{1,2} \ P_{2,2} \ P_{3,1}$

- Knowledge base
  - Nothing in [1,1]
  - Breeze in [2,1]

- Query $\alpha_2$:
  - No pit in [2,2]
# Propositional Logic Models

<table>
<thead>
<tr>
<th>Model Symbols</th>
<th>All Possible Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0 0 0 0 1 1 1 1 1</td>
</tr>
<tr>
<td>B</td>
<td>0 0 1 1 0 0 1 1</td>
</tr>
<tr>
<td>C</td>
<td>0 1 0 1 0 1 0 1</td>
</tr>
</tbody>
</table>
Piazza Poll 1
Does the KB entail query C?

Entailment: \( \alpha \models \beta \)
“\( \alpha \) entails \( \beta \)” iff in every world where \( \alpha \) is true, \( \beta \) is also true

<table>
<thead>
<tr>
<th>Model Symbols</th>
<th>A</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

| Knowledge Base | A       | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
|                | B⇒C    | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
|                | A⇒B∨C | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |

| Query | C       | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

All Possible Models
Piazza Poll 1

Does the KB entail query C?

Yes 🇮

Entailment: $\alpha \models \beta$

“$\alpha$ entails $\beta$” iff in every world where $\alpha$ is true, $\beta$ is also true

<table>
<thead>
<tr>
<th>Model Symbols</th>
<th>Knowledge Base</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1$</td>
<td>$0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1$</td>
</tr>
<tr>
<td>$B$</td>
<td>$0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 1 \ 1$</td>
<td>$1 \ 1 \ 0 \ 1 \ 1 \ 1 \ 0 \ 1$</td>
</tr>
<tr>
<td>$C$</td>
<td>$0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1$</td>
<td>$1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 1 \ 1$</td>
</tr>
</tbody>
</table>

$\text{KB} \models \neg B$
Entailment

How do we implement a logical agent that proves entailment?

- Logic language
  - Propositional logic
  - First order logic

- Inference algorithms
  - Theorem proving
  - Model checking
Propositional Logic

Check if sentence is true in given model
In other words, does the model \textit{satisfy} the sentence?

function \textsc{PL-TRUE?}(\alpha, \text{model}) returns true or false

\begin{itemize}
  \item if \(\alpha\) is a symbol then return \text{Lookup}(\alpha, \text{model})
  \item if \(\text{Op}(\alpha) = \neg\) then return \text{not}(\text{PL-TRUE?}(\text{Arg1}(\alpha), \text{model}))
  \item if \(\text{Op}(\alpha) = \land\) then return \text{and}(\text{PL-TRUE?}(\text{Arg1}(\alpha), \text{model}), \text{PL-TRUE?}(\text{Arg2}(\alpha), \text{model}))
\end{itemize}

etc.

(Sometimes called “recursion over syntax”)
Simple Model Checking

function TT-ENTAILS?(KB, α) returns true or false
   return TT-CHECK-ALL(KB, α, symbols(KB) U symbols(α),{})

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
      P ← first(symbols)
      rest ← rest(symbols)
      return and (TT-CHECK-ALL(KB, α, rest, model U {P = true})
                   TT-CHECK-ALL(KB, α, rest, model U {P = false }))
Simple Model Checking, contd.

Same recursion as backtracking
$O(2^n)$ time, linear space
We can do much better!