Have some fun

- Checker: http://www.cs.caltech.edu/~vhuang/cs20/c/applet/more.html

Local Beam Search

- Run multiple searches to find the solution
  - The best K states are selected.
- Like parallel hill climbing from different start points but:
  - Advantage: They communicate to localize the search
  - Drawback: may miss the global optima
Stochastic Beam Search

- Instead of selecting the best $K$ successors, use probability to select.

Online Search

- Explore the environment like MarsRover
- Agent knows:
  - ACTIONS(s): list of possible actions
  - Step cost $c(s,a,s')$: cost of going from $s$ to $s'$ taking action $a$
  - GOAL-TEST(s)
- Competitive ratio:
  - path cost using exploration/optimal path cost
  - Optimal path cost = off-line planning knowing the map
Online Search (II)

- Competitive ratio:
  - Can be infinite if there are dead ends

- Assumption:
  - Safely explorable: there are paths from each reachable state to goal.

Online Search (III)

- DFS is acceptable method

- Hill climbing is doable too

- LRTA*: Learning Real-Time A*
Knowledge and reasoning

- Knowledge representation
- Logic and representation
- Propositional (Boolean) logic
- Normal forms
- Inference in propositional logic
- Wumpus world example

Knowledge-Based Agent

Domain independent algorithms

- ASK
- TELL

Inference engine

Knowledge Base

Domain specific content
Generic knowledge-based agent

function KB-AGENT( percept ) returns an action
static: KB, a knowledge base
     t, a counter, initially 0, indicating time
TELL(KB, MAKE-PERCEPT-SENTENCE( percept, t ))
action ← ASK(KB, MAKE-ACTION-QUERY(t ))
TELL(KB, MAKE-ACTION-SENTENCE( action, t ))
t ← t + 1
return action

1. TELL KB what was perceived

2. ASK KB what to do.

Wumpus world example

Percepts Breeze, Glitter, Smell

Actions Left turn, Right turn,
     Forward, Grab, Release, Shoot

Goals Get gold back to start
     without entering pit or wumpus square

Environment
     Squares adjacent to wumpus are smelly
     Squares adjacent to pit are breezy
     Glitter if and only if gold is in the same square
     Shooting kills the wumpus if you are facing it
     Shooting uses up the only arrow
     Grabbing picks up the gold if in the same square
     Releasing drops the gold in the same square
Wumpus world characterization

- Deterministic?
- Accessible?
- Static?
- Discrete?
- Episodic?

Let's play a game ☺

- http://www.cogsci.rpi.edu/Otter/Wumpus/
Exploring a Wumpus world

A = Agent
B = Breeze
S = Smell
P = Pit
W = Wumpus
OK = Safe
V = Visited
G = Glitter

Breeze in (1,2) and (2,1)
⇒ no safe actions

Assuming pits uniformly distributed,
(2,2) is most likely to have a pit

Smell in (1,1)
⇒ cannot move

Can use a strategy of coercion:
- shoot straight ahead
- wumpus was there ⇒ dead ⇒ safe
- wumpus wasn’t there ⇒ safe
Another example solution

<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>2.2</td>
<td>3.2</td>
<td>4.2</td>
<td></td>
</tr>
<tr>
<td>1.1</td>
<td>2.1</td>
<td>3.1</td>
<td>4.1</td>
<td></td>
</tr>
</tbody>
</table>

No perception ⇒ 1,2 and 2,1 OK
Move to 2,1

<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>2.2</td>
<td>3.2</td>
<td>4.2</td>
<td></td>
</tr>
<tr>
<td>1.1</td>
<td>2.1</td>
<td>3.1</td>
<td>4.1</td>
<td></td>
</tr>
</tbody>
</table>

B in 2,1 ⇒ 2,2 or 3,1 P?
1,1 V ⇒ no P in 1,1
Move to 1,2 (only option)

Let’s try blinded


- Geeky games: Office environment
  - Dark environment
  - Black holes suck you in (you feel slight breeze)
  - Trudy with big mallet to smash you (you will feel her perfume)
  - You have to find the check and your way back to the starting point
  - Pay check can be in Trudy or black hole cubical.
  - Trudy can be in the black hole cubical
  - Computer wants to find your check too
    - Nick is conservative
    - Ki is taking chance once or twice
    - Fooker is an aggressive one.
Logic in general

Logics are formal languages for representing information such that conclusions can be drawn.

Syntax defines the sentences in the language.

Semantics define the “meaning” of sentences; i.e., define truth of a sentence in a world.

E.g., the language of arithmetic:

\( x + 2 \geq y \) is a sentence; \( x2 + y > \) is not a sentence.

\( x + 2 \geq y \) is true iff the number \( x + 2 \) is no less than the number \( y \).

\( x + 2 \geq y \) is true in a world where \( x = 7, y = 1 \).

\( x + 2 \geq y \) is false in a world where \( x = 0, y = 6 \).

Types of logic

Logics are characterized by what they commit to as “primitives.”


Epistemological commitment: what states of knowledge?

<table>
<thead>
<tr>
<th>Language</th>
<th>Ontological Commitment</th>
<th>Epistemological Commitment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propositional logic</td>
<td>facts</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>First-order logic</td>
<td>facts, objects, relations</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>Temporal logic</td>
<td>facts, objects, relations, times</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>Probability theory</td>
<td>facts</td>
<td>degree of belief 0...1</td>
</tr>
<tr>
<td>Fuzzy logic</td>
<td>degree of truth</td>
<td>degree of belief 0...1</td>
</tr>
</tbody>
</table>
The Semantic Wall

Physical Symbol System

+BLOCKA+
+BLOCKB+
+BLOCKC+

P₁:(IS_ON +BLOCKA+ +BLOCKB+)
P₂:(IS_RED +BLOCKA+)

Truth depends on Interpretation

Representation 1

A
B
ON(A,B) T

World

A
B
ON(A,B) F
Entailment

\[ KB \models \alpha \]

Knowledge base \( KB \) entails sentence \( \alpha \)
if and only if
\( \alpha \) is true in all worlds where \( KB \) is true

E.g., the KB containing "the Giants won" and "the Reds won" entails "Either the Giants won or the Reds won"
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 $m$.

Inference

$KB \vdash \alpha = \text{sentence } \alpha \text{ can be derived from } KB \text{ by procedure } i$.

Soundness: $i$ is sound if whenever $KB \vdash \alpha$, it is also true that $KB \models \alpha$. 
Basic symbols

Expressions only evaluate to either “true” or “false.”

- P
- ¬P
- P V Q
- P ^ Q
- P => Q
- P ⇔ Q

Propositional logic: syntax

Propositional logic is the simplest logic.

The proposition symbols $P_1, P_2$ etc are sentences

If $S$ is a sentence, $¬S$ is a sentence

If $S_1$ and $S_2$ is a sentence, ...

If $S_1$ and $S_2$ is a sentence, ...

If $S_1$ and $S_2$ is a sentence, ...
Propositional logic: semantics

Each model specifies true/false for each proposition symbol.

E.g. $A \quad B \quad C$

$\text{True True False}$

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 $S_1 \Leftarrow S_2$ is true.

Truth tables

- Truth value: whether a statement is true or false.
- Truth table: complete list of truth values for a statement given all possible values of the individual atomic expressions.

Example:

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>Q</th>
<th>P V Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td></td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td></td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td></td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td></td>
<td>F</td>
</tr>
</tbody>
</table>
### Truth tables for basic connectives

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>¬P</th>
<th>¬Q</th>
<th>P V Q</th>
<th>P ^ Q</th>
<th>P =&gt; Q</th>
<th>P ⇔ Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**P Implies Q: (X>Y ^ Y>Z)**

\[ \Rightarrow X>Z \]

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>P =&gt; Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>
Propositional logic: basic manipulation rules

- ¬(¬A) = A  
  Double negation

- ¬(A ^ B) = (¬A) V (¬B)  
  Negated “and”

- ¬(A V B) = (¬A) ^ (¬B)  
  Negated “or”

- A ^ (B V C) = (A ^ B) V (A ^ C)  
  Distributivity of ^ on V

- A => B = (¬A) V B  
  by definition

- ¬(A => B) = A ^ (¬B)  
  using negated or

- A ⇔ B = (A => B) ^ (B => A)  
  by definition

- ¬(A ⇔ B) = (A ^ (¬B)) V (B ^ (¬A))  
  using negated and & or

Propositional inference: enumeration method

- Let α = A V B and KB = (A V C) ^ (B V ¬C)

Is it the case that KB ⊨ α?
Check all possible models—α must be true wherever KB is t:

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>A V C</td>
<td>B V ¬C</td>
<td>KB</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td></td>
<td></td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td></td>
<td></td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td></td>
<td></td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td></td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td></td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td></td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td></td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

31
Propositional inference: normal forms

Other approaches to inference use syntactic operations on sentences, often expressed in standardized forms

**Conjunctive Normal Form (CNF—universal)**

\[
\text{conjunction of } \text{disjunctions of literals}
\]

E.g., \((A \lor \neg B) \land (B \lor \neg C \lor \neg D)\)

**Disjunctive Normal Form (DNF—universal)**

\[
\text{disjunction of } \text{conjunctions of literals}
\]

E.g., \((A \land B) \lor (A \land \neg C) \lor (A \land \neg D) \lor (\neg B \land \neg C) \lor (\neg B \land \neg D)\)

**Horn Form** (restricted)

\[
\text{conjunction of Horn clauses } (\text{clauses with } \leq 1 \text{ positive literal})
\]

E.g., \((A \lor \neg B) \land (B \lor \neg C \lor \neg D)\)

Often written as set of implications:

\[B \Rightarrow A \text{ and } (C \land D) \Rightarrow B\]

---

Deriving expressions from functions

- Given a boolean function in truth table form, find a propositional logic expression for it that uses only \(\lor\), \(\land\), and \(\neg\).

- **Idea:** We can easily do it by disjoining the “T” rows of the truth table.

Example: XOR function

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>RESULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
A more formal approach

- To construct a logical expression in disjunctive normal form from a truth table:
  - Build a "minterm" for each row of the table, where:
    - For each variable whose value is T in that row, include the variable in the minterm
    - For each variable whose value is F in that row, include the negation of the variable in the minterm
  - Link variables in minterm by conjunctions
  - The expression consists of the disjunction of all minterms.

Example: adder with carry

Takes 3 variables in: x, y and ci (carry-in);

results: sum (s) and carry-out (co).

To get you used to other notations, here we assume T = 1, F = 0, V = OR, ^ = AND, ¬ = NOT.
Example: adder with carry

c0 is:

s is:

---

Tautologies

- Simplify the logical expressions that are always true.

Examples:

T

T V A

A V (¬A)

¬(A ^ (¬A))

A ⇔ A

((P V Q) ⇔ P) V (¬P ^ Q)

(P ⇔ Q) => (P => Q)
Validity and satisfiability

A sentence is valid if it is true in all models
e.g., \( A \lor \neg A \), \( A \Rightarrow A \), \( (A \land (A \Rightarrow B)) \Rightarrow B \)

Validity is connected to inference via the Deduction Theorem
\( KB \models \alpha \) if and only if \( (KB \Rightarrow \alpha) \) is valid

A sentence is **satisfiable** if it is true in some model
e.g., \( A \lor B \), \( C \)

A sentence is **unsatisfiable** if it is true in no models
e.g., \( A \land \neg A \)

Satisfiability is connected to inference via the following:
\( KB \models \alpha \) if and only if \( (KB \land \neg \alpha) \) is unsatisfiable
i.e., prove \( \alpha \) by *reductio ad absurdum*

---

Model of a Formula

<table>
<thead>
<tr>
<th>assignment</th>
<th>a</th>
<th>b</th>
<th>a ( \Rightarrow ) b</th>
<th>( a \land b )</th>
<th>( \neg (a \lnot\leftrightarrow b) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

a) The assignments A, B and D are models of the formula \( a \Rightarrow b \).
b) The assignment D is model of the formula \( a \land b \).
c) The assignments A and D are models of the formula \( \neg (a \lnot\leftrightarrow b) \).
Satisfiability Example

<table>
<thead>
<tr>
<th>assignment</th>
<th>a</th>
<th>b</th>
<th>a=&gt;b</th>
<th>a^b</th>
<th>(a↔ b)</th>
<th>~(a V b)</th>
<th>a V (~a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Proof methods

Proof methods divide into (roughly) two kinds:

Model checking
- truth table enumeration (sound and complete for propositional)
- heuristic search in model space (sound but incomplete)
  e.g., the GSAT algorithm (Ex. 6.15)

Application of inference rules
- Legitimate (sound) generation of new sentences from old
  Proof = a sequence of inference rule applications
  Can use inference rules as operators in a standard search alg.
Inference Rules

◊ **Modus Ponens** or **Implication-Elimination**: (From an implication and the premise of the implication, you can infer the conclusion.)

\[
\frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta}
\]

◊ **And-Elimination**: (From a conjunction, you can infer any of the conjuncts.)

\[
\frac{\alpha_1 \land \alpha_2 \land \ldots \land \alpha_n}{\alpha_i}
\]

◊ **And-Introduction**: (From a list of sentences, you can infer their conjunction.)

\[
\frac{\alpha_1, \quad \alpha_2, \quad \ldots, \quad \alpha_n}{\alpha_1 \land \alpha_2 \land \ldots \land \alpha_n}
\]

◊ **Or-Introduction**: (From a sentence, you can infer its disjunction with anything else at all.)

\[
\frac{\alpha_i}{\alpha_1 \lor \alpha_2 \lor \ldots \lor \alpha_n}
\]

Inference Rules

◊ **Double-Negation Elimination**: (From a doubly negated sentence, you can infer a positive sentence.)

\[
\frac{\neg \neg \alpha}{\alpha}
\]

◊ **Unit Resolution**: (From a disjunction, if one of the disjuncts is false, then you can infer the other one is true.)

\[
\frac{\alpha \lor \beta, \quad \neg \beta}{\alpha}
\]

◊ **Resolution**: (This is the most difficult. Because \( \beta \) cannot be both true and false, one of the other disjuncts must be true in one of the premises. Or equivalently, implication is transitive.)

\[
\frac{\alpha \lor \beta, \quad \neg \beta \lor \gamma}{\alpha \lor \gamma} \quad \text{or equivalently} \quad \frac{\neg \alpha \Rightarrow \beta, \quad \beta \Rightarrow \gamma}{\neg \alpha \Rightarrow \gamma}
\]
Wumpus world: example

- **Facts:** Percepts inject (TELL) facts into the KB
  - [stench at 1,1 and 2,1] \(\rightarrow\) \(S_{1,1}\); \(S_{2,1}\)

- **Rules:** if square has no stench then neither the square or adjacent square contain the wumpus
  - R1: \(!S_{1,1} \Rightarrow !W_{1,1} \land !W_{1,2} \land !W_{2,1}\)
  - R2: \(!S_{2,1} \Rightarrow !W_{1,1} \land !W_{2,2} \land !W_{2,2} \land !W_{3,1}\)
  - ...

- **Inference:**
  - KB contains \(!S_{1,1}\) then using Modus Ponens we infer
    - \(!W_{1,1} \land !W_{1,2} \land !W_{2,1}\)
  - Using And-Elimination we get: \(!W_{1,1} \land !W_{1,2} \land !W_{2,1}\)
  - ...

Limitations of Propositional Logic

1. It is too weak (has very limited expressiveness):

2. It cannot keep track of changes:
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.

Propositional logic suffices for some of these tasks

Truth table method is sound and complete for propositional logic.

Next time

- First-order logic: [AIMA] Chapter 7