Admin
Class Lectures: Tuesdays and Thursdays 10:30-11:50am in 4623 Wean Hall

Recitations: TBA

This course is targeted at graduate students who need to learn about and perform current-day research in artificial intelligence—the discipline of designing intelligent decision-making machines. Techniques from probability, statistics, game theory, algorithms, operations research, and optimal control are increasingly important tools for improving the intelligence and autonomy of machines, whether those machines are robots surveying Antarctica, schedulers moving billions of dollars of inventory, spacecraft deciding which experiments to perform, or vehicles negotiating for lanes on the freeway. This AI course is a review of a selected set of these tools. The course will cover the ideas underlying these tools, their implementation, and how to use them or extend them in your research. Students entering the class should have a pre-existing working knowledge of

http://www.cs.cmu.edu/~ggordon/780/
http://www.cs.cmu.edu/~sandholm/cs15-780S09/
Website highlights

- Grading: 4–5 HWs, “mid”term, project
- Project: proposal, 2 interim reports, final report, poster
- Office hours
Website highlights

- Authoritative source for readings, HWs
- Please check the website regularly for readings (for Lec. 1–3, Russell & Norvig Chapters 7–9)
Background

- No prerequisites
- But, suggest familiarity with at least some of the following:
  - Linear algebra
  - Calculus
  - Algorithms & data structures
  - Complexity theory
Waitlist, Audits

- If you need us to approve something, send us email
Course email list

15780students@…

domain cs.cmu.edu

To subscribe/unsubscribe:

email 15780students-request@…

word “help” in subject or body
Matlab

- Should all have access to Matlab via school computers
  - Those with access to CS license servers, please use if possible
  - Limited number of Andrew licenses
- Tutorial TBA soon
- HWs: please use C, C++, Java, or Matlab
Intro
What is AI?

- Easy part: A
- Hard part: I
  - Anything we don’t know how to make a computer do yet
  - Corollary: once we do it, it isn’t AI anymore :-)

Definition by examples

- Card games
  - Poker
  - Bridge

- Board games
  - Deep Blue
  - TD-Gammon
  - Samuels's checkers player
Web search
Web search, cont’d

Senior Marketing Representative
Crawley Warren Insurance Services, Inc. (San Francisco, California)

Salary: $20 to $30
Salary Details: depending on experience
Position Type: Parttime
Ref Code: 60576596
Minimum Education Level:
Some College Coursework Completed
Minimum Career Level: Experienced (Non-Manager)

Save Job to my monster
APPLY NOW
Recommender systems
Computer algebra systems

from http://www.math.wpi.edu/IQP/BVCalcHist/calctoc.html
Grand Challenge road race
Getting from A to B

- **ITA software** ([http://beta.itasoftware.com](http://beta.itasoftware.com))
Robocup
Kidney exchange

- In US, ≥ 50,000/yr get lethal kidney disease
- Cure = transplant, but donor must be compatible (blood type, tissue type, etc.)
- Wait list for cadaver kidneys: 2–5 years
- Live donors: have 2 kidneys, can survive w/ 1
- Illegal to buy/sell, but altruists/friends/family donate
Kidney Exchange

Pair 1

Patient
Donor

Pair 2

Patient
Donor
Optimization: cycle cover

Cycle length constraint => extremely hard (NP-complete) combinatorial optimization problem
National market predicted to have 10,000 patients at any one time
Optimization performance

![Graph comparing clearing times for CPLEX and Sandholm et al.'s algorithm. The x-axis represents the number of patients, and the y-axis represents the clearing time in seconds. The graph shows that CPLEX has a lower clearing time compared to Sandholm et al.'s algorithm, especially as the number of patients increases.]
More examples

- Motor skills: riding a bicycle, learning to walk, playing pool, ...
- Vision
More examples

- Valerie and Tank, the Roboceptionists
- Social skills: attending a party, giving directions, …
More examples

- Natural language
- Speech recognition
Common threads

- Finding the needle in the haystack
  - Search
  - Optimization
  - Summation / integration
- Set the problem up well (so that we can apply a standard algorithm)
Common threads

- Managing uncertainty
  - chance outcomes (e.g., dice)
  - sensor uncertainty ("hidden state")
  - opponents

- The more different types of uncertainty, the harder the problem (and the slower the solution)
Classic AI

- No uncertainty, pure search
  - Mathematica
  - deterministic planning
  - Sudoku
- This is the topic of Part I of the course

Outcome uncertainty

- In backgammon, don’t know ahead of time what the dice will show
- When driving down a corridor, wheel slippage causes unexpected deviations
- Open a door, find out what’s behind it
- MDPs (later)
Sensor uncertainty

- For given set of immediate measurements, multiple world states may be possible
- Image of a handwritten digit $\rightarrow 0, 1, \ldots, 9$
- Image of room $\rightarrow$ person locations, identities
- Laser rangefinder scan of a corridor $\rightarrow$ map, robot location
Sensor uncertainty example
Opponents cause uncertainty

- In chess, must guess what opponent will do; cannot directly control him/her
- Alternating moves (game trees)
  - not really uncertain (Part I)
- Simultaneous or hidden moves: game theory (later)
Other agents cause uncertainty

- In many AI problems, there are other agents who aren’t (necessarily) opponents
  - Ignore them & pretend part of Nature
  - Assume they’re opponents (pessimistic)
  - Learn to cope with what they do
  - Try to convince them to cooperate (paradoxically, this is the hardest case)
- More later
Logic
Why logic?

- **Search**: for problems like Sudoku, can write compact description of rules
- **Reasoning**: figure out consequences of the knowledge we’ve given our agent
- …and, **logical inference is a special case of probabilistic inference**
Propositional logic

- **Constants**: T or F
- **Variables**: x, y (values T or F)
- **Connectives**: ∧, ∨, ¬
  - *Can get by w/ just NAND*
  - *Sometimes also add others:*
    - ⊕, →, ↔, …
Propositional logic

- Build up expressions like $\neg x \Rightarrow y$
- Precedence: $\neg$, $\wedge$, $\vee$, $\Rightarrow$
- Terminology: variable or constant with or w/o negation = literal
- Whole thing = formula or sentence
Expressive variable names

- Rather than variable names like $x, y$, may use names like “rains” or “happy(John)”
- For now, “happy(John)” is just a string with no internal structure
  - there is no “John”
  - happy(John) $\Rightarrow \neg$happy(Jack) means the same as $x \Rightarrow \neg y$
But what does it mean?

- A formula defines a mapping
  
  \[(\text{assignment to variables}) \mapsto \{T, F\}\]
  
- Assignment to variables = \textit{model}

- For example, formula \(\neg x\) yields mapping:

<table>
<thead>
<tr>
<th></th>
<th>(\neg x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x)</td>
<td>(\neg x)</td>
</tr>
<tr>
<td>(T)</td>
<td>(F)</td>
</tr>
<tr>
<td>(F)</td>
<td>(T)</td>
</tr>
</tbody>
</table>
Questions about models and sentences

- *How many models make a sentence true?*
  - A sentence is *satisfiable* if it is True in some model
  - *If not satisfiable, it is a contradiction* (False in every model)
  - A sentence is *valid* if it is True in every model (called a *tautology*)
Questions about models and sentences

- How is the variable $X$ set in \{some, all\} true models?
- This is the most frequent question an agent would ask: given my assumptions, can I conclude $X$? Can I rule $X$ out?
### More truth tables

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$x \land y$</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>$F$</td>
<td>$T$</td>
<td>$F$</td>
</tr>
<tr>
<td>$F$</td>
<td>$F$</td>
<td>$F$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$x \lor y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>$T$</td>
<td>$T$</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 ⇒ b) is logically equivalent to (¬a ∨ b)

- If a is True, b must be True too
- If a False, no requirement on b
- E.g., “if I go to the movie I will have popcorn”: if no movie, may or may not have popcorn

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>a ⇒ b</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>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>
To evaluate a bigger formula

\((x \lor y) \land (x \lor \neg y)\) when \(x = F, y = F\)

Build a parse tree

Fill in variables at leaves using model

Work upwards using truth tables for connectives
Example

○ $(x \lor y) \land (x \lor \neg y)$ when $x = F, y = F$
Another example

\[(x \lor y) \Rightarrow z\]

\[x = F, y = T, z = F\]
Example
3-coloring

vars: aR ab aG
     bR bB ... 

consts: (aR u aB u aG) \n     (bR u bB u bG) ... 
     (aR u \overline{bB}) \n     (aB u \overline{bB}) \n     (aG u \overline{bG}) \n     (aR u \overline{zR}) ...
Sudoku

http://www.cs.qub.ac.uk/~I.Spence/SuDoku/SuDoku.html
Minesweeper

\[
\begin{array}{ccc}
0 & 0 & 1 \\
0 & 0 & 1 \\
0 & 0 & 1 \\
1 & 1 & 2 \\
v8 & v7 & v6 \\
v5 & & \\
\end{array}
\]

\[ V = \{ v1, v2, v3, v4, v5, v6, v7, v8 \}, D = \{ B (bomb), S (space) \} \]

\[ C = \{ (v1,v2) : \{ (B, S), (S,B) \}, (v1,v2,v3) : \{ (B,S,S), (S,B,S), (S,S,B) \} \}, ... \]
Propositional planning

init: have(cake)
goal: have(cake), eaten(cake)
eat(cake):
  pre: have(cake)
  eff: -have(cake), eaten(cake)
bake(cake):
  pre: -have(cake)
  eff: have(cake)
Other important logic problems

- *Scheduling* (e.g., of factory production)
- *Facility location*
- *Circuit layout*
- *Multi-robot planning*

- *Important issue: handling uncertainty*
Working with formulas
Truth tables get big fast

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$z$</th>
<th>$(x \lor y) \implies z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>$T$</td>
<td>$T$</td>
<td></td>
</tr>
<tr>
<td>$T$</td>
<td>$T$</td>
<td>$F$</td>
<td></td>
</tr>
<tr>
<td>$T$</td>
<td>$F$</td>
<td>$T$</td>
<td></td>
</tr>
<tr>
<td>$T$</td>
<td>$F$</td>
<td>$F$</td>
<td></td>
</tr>
<tr>
<td>$F$</td>
<td>$T$</td>
<td>$T$</td>
<td></td>
</tr>
<tr>
<td>$F$</td>
<td>$T$</td>
<td>$F$</td>
<td></td>
</tr>
<tr>
<td>$F$</td>
<td>$F$</td>
<td>$T$</td>
<td></td>
</tr>
<tr>
<td>$F$</td>
<td>$F$</td>
<td>$F$</td>
<td></td>
</tr>
</tbody>
</table>
Truth tables get big fast

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$z$</th>
<th>$a$</th>
<th>$(x \lor y \lor a) \Rightarrow z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>$T$</td>
<td>$T$</td>
<td>$T$</td>
<td>$T$</td>
</tr>
<tr>
<td>$T$</td>
<td>$T$</td>
<td>$F$</td>
<td>$T$</td>
<td></td>
</tr>
<tr>
<td>$T$</td>
<td>$F$</td>
<td>$T$</td>
<td>$T$</td>
<td></td>
</tr>
<tr>
<td>$T$</td>
<td>$F$</td>
<td>$F$</td>
<td>$T$</td>
<td></td>
</tr>
<tr>
<td>$F$</td>
<td>$T$</td>
<td>$T$</td>
<td>$T$</td>
<td></td>
</tr>
<tr>
<td>$F$</td>
<td>$T$</td>
<td>$F$</td>
<td>$T$</td>
<td></td>
</tr>
<tr>
<td>$F$</td>
<td>$F$</td>
<td>$T$</td>
<td>$T$</td>
<td></td>
</tr>
<tr>
<td>$F$</td>
<td>$F$</td>
<td>$F$</td>
<td>$T$</td>
<td></td>
</tr>
<tr>
<td>$T$</td>
<td>$T$</td>
<td>$T$</td>
<td>$F$</td>
<td></td>
</tr>
<tr>
<td>$T$</td>
<td>$T$</td>
<td>$F$</td>
<td>$F$</td>
<td></td>
</tr>
<tr>
<td>$T$</td>
<td>$F$</td>
<td>$T$</td>
<td>$F$</td>
<td></td>
</tr>
<tr>
<td>$T$</td>
<td>$F$</td>
<td>$F$</td>
<td>$F$</td>
<td></td>
</tr>
<tr>
<td>$F$</td>
<td>$T$</td>
<td>$T$</td>
<td>$F$</td>
<td></td>
</tr>
<tr>
<td>$F$</td>
<td>$T$</td>
<td>$F$</td>
<td>$F$</td>
<td></td>
</tr>
<tr>
<td>$F$</td>
<td>$F$</td>
<td>$T$</td>
<td>$F$</td>
<td></td>
</tr>
<tr>
<td>$F$</td>
<td>$F$</td>
<td>$F$</td>
<td>$F$</td>
<td></td>
</tr>
</tbody>
</table>
Definitions

- Two sentences are *equivalent*, $A \equiv B$, if they have same truth value in every model
  - $(\text{rains } \Rightarrow \text{ pours}) \equiv (\neg \text{rains } \lor \text{ pours})$
  - reflexive, transitive, commutative
- *Simplifying* = transforming a formula into a shorter*, equivalent formula
Transformation rules

\[(\alpha \land \beta) \equiv (\beta \land \alpha) \quad \text{commutativity of } \land\]

\[(\alpha \lor \beta) \equiv (\beta \lor \alpha) \quad \text{commutativity of } \lor\]

\[((\alpha \land \beta) \land \gamma) \equiv (\alpha \land (\beta \land \gamma)) \quad \text{associativity of } \land\]

\[((\alpha \lor \beta) \lor \gamma) \equiv (\alpha \lor (\beta \lor \gamma)) \quad \text{associativity of } \lor\]

\[\neg(\neg \alpha) \equiv \alpha \quad \text{double-negation elimination}\]

\[(\alpha \land (\beta \lor \gamma)) \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)) \quad \text{distributivity of } \land \text{ over } \lor\]

\[(\alpha \lor (\beta \land \gamma)) \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) \quad \text{distributivity of } \lor \text{ over } \land\]

\[\alpha, \beta, \gamma \text{ are arbitrary formulas}\]
More rules

\[(\alpha \Rightarrow \beta) \equiv (\neg \beta \Rightarrow \neg \alpha)\] contraposition

\[(\alpha \Rightarrow \beta) \equiv (\neg \alpha \lor \beta)\] implication elimination

\[(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha))\] biconditional elimination

\[\neg (\alpha \land \beta) \equiv (\neg \alpha \lor \neg \beta)\] de Morgan

\[\neg (\alpha \lor \beta) \equiv (\neg \alpha \land \neg \beta)\] de Morgan

\[\alpha, \beta \text{ are arbitrary formulas}\]
Still more rules…

…can be derived from truth tables

For example:

- \((a \lor \neg a) \equiv True\)
- \((True \lor a) \equiv True\)
- \((False \land a) \equiv False\)
Example

\[(a \lor \neg b) \land (a \lor \neg c) \land (\neg(b \lor c) \lor \neg a)\]
Normal Forms
Normal forms

- A normal form is a standard way of writing a formula

  E.g., conjunctive normal form (CNF)
  - conjunction of disjunctions of literals
    $(x \lor y \lor \neg z) \land (x \lor \neg y) \land (z)$
  - Each disjunct called a clause

- Any formula can be transformed into CNF w/o changing meaning
CNF cont’d

```latex
\begin{align*}
\text{happy}(John) & \land \\
(\neg \text{happy}(Bill) & \lor \text{happy}(Sue)) \land \\
\text{man}(Socrates) & \land \\
(\neg \text{man}(Socrates) & \lor \text{mortal}(Socrates))
\end{align*}
```

- Often used for storage of knowledge database
- called **knowledge base** or **KB**
- Can add new clauses as we find them out
- Each clause in KB is separately true (if KB is)
Another normal form: DNF

- DNF = disjunctive normal form = disjunction of conjunctions of literals
- Doesn’t compose the way CNF does: can’t just add new conjuncts w/o changing meaning of KB
- Example:

\[(\text{rains} \lor \neg \text{pours}) \land \text{fishing} \equiv (\text{rains} \land \text{fishing}) \lor (\neg \text{pours} \land \text{fishing})\]
To CNF

\[(a \lor b \lor \neg c) \land \neg (d \lor (e \land f)) \land (c \lor d \lor e)\]
To DNF

\[(a \lor b \lor \neg c) \land \neg (d \lor (e \land f)) \land (c \lor d \lor e)\]

\[\neg (d \lor (e \land f)) \lor (\neg (d \lor (e \land f)) \land d)\]

\[\lor (\neg (d \lor (e \land f)) \land e)\]

\[\alpha \land (\beta \lor \gamma) \rightarrow \frac{\text{big}}{\text{small}} \quad \frac{\alpha \land \beta \lor \alpha \land \gamma}{\text{almost big}}\]
Transforming to CNF or DNF

- **Naive algorithm:**
  - replace all connectives with $\land \lor \neg$
  - move negations inward using De Morgan’s laws and double-negation
  - repeatedly distribute over $\land$ over $\lor$ for DNF ($\lor$ over $\land$ for CNF)
Example

Put the following formula in CNF

\[(a \lor b \lor \neg c) \land \neg(d \lor (e \land f)) \land (c \lor d \lor e)\]
Example

- Now try DNF

\[(a \lor b \lor \neg c) \land \neg (d \lor (e \land f)) \land (c \lor d \lor e)\]
Discussion

- Problem with naive algorithm: it’s exponential! (Space, time, size of result.)
- Each use of distributivity can almost double the size of a subformula
A smarter transformation

- Can we avoid exponential blowup in CNF?
- Yes, if we’re willing to introduce new variables
Tseitin example

- Put the following formula in CNF:
  
  \[(a \land b) \lor ((c \lor d) \land e)\]

- Parse tree:
Tseitin transformation

- Introduce temporary variables
  - \( x = (a \land b) \)
  - \( y = (c \lor d) \)
  - \( z = (y \land e) \)
To ensure $x = (a \land b)$, want

- $x \Rightarrow (a \land b)$
- $(a \land b) \Rightarrow x$
Tseitin transformation

\[ x \Rightarrow (a \land b) \]
\[ (\neg x \lor (a \land b)) \]
\[ (\neg x \lor a) \land (\neg x \lor b) \]
Tseitin transformation

- \((a \land b) \Rightarrow x\)
- \((- (a \land b) \lor x)\)
- \((- a \lor \neg b \lor x)\)
To ensure $y = (c \lor d)$, want

- $y \Rightarrow (c \lor d)$
- $(c \lor d) \Rightarrow y$
Tseitin transformation

- $y \implies (c \lor d)$
- $(\neg y \lor c \lor d)$
- $(c \lor d) \implies y$
- $((\neg c \land \neg d) \lor y)$
- $(\neg c \lor y) \land (\neg d \lor y)$
Tseitin transformation

- Finally, \( z = (y \land e) \)
- \( z \Rightarrow (y \land e) \equiv (\neg z \lor y) \land (\neg z \lor e) \)
- \( (y \land e) \Rightarrow z \equiv (\neg y \lor \neg e \lor z) \)
Tseitin end result

\[(a \land b) \lor ((c \lor d) \land e) \equiv\]

\[\neg x \lor a \land \neg x \lor b \land \neg a \lor \neg b \lor x \land\]
\[\neg y \lor c \lor d \land \neg c \lor y \land \neg d \lor y \land\]
\[\neg z \lor y \land \neg z \lor e \land \neg y \lor \neg e \lor z \land\]
\[x \lor z\]