Logic
Why logic?

- Search: can compactly write down, solve problems like Sudoku
- 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**: \( \neg, \land, \lor, \Rightarrow \)
- **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”
  - $happy(John) \implies \neg happy(Jack)$ means the same as $x \implies \neg y$
But what does it mean?

- A formula defines a mapping
  \[(assignment \text{ to variables}) \mapsto \{T, F\}\]
- Assignment to variables = model
- For example, formula $\neg x$ yields mapping:

<table>
<thead>
<tr>
<th></th>
<th>$x$</th>
<th>$\neg x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td></td>
<td>$F$</td>
</tr>
<tr>
<td>$F$</td>
<td></td>
<td>$T$</td>
</tr>
</tbody>
</table>
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>
Truth table for implication

- \((a \Rightarrow b)\) is logically equivalent to \((\neg a \lor b)\)
- 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

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>a \Rightarrow b</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>-----------------</td>
</tr>
<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>
Complex formulas

- 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
Another example

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

\[x = F, y = T, z = F\]
Questions about models and sentences

- How many models make a sentence true?
  - *Sentence is satisfiable* if true in some model (famous NP-complete problem)
  - *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\} satisfying models?
- This is the most frequent question an agent would ask: given my assumptions, can I conclude $X$? Can I rule $X$ out?
- $SAT$ answers all the above questions
Bigger Examples
3-coloring

\[
\text{Vars: } aR, aG, ab, bR, bG, bB \ldots
\]

\[
(aR \lor aG \lor ab) \land (bR \lor bG \lor bB) \land \ldots
\]

\[
(a\overline{R} \lor b\overline{R}) \land (a\overline{G} \lor b\overline{G}) \land \ldots
\]
Constraint satisfaction problems

- Like SAT, but:
  - variable domains are arbitrary (vs. TF)
  - complex constraints (vs. \( a \lor b \lor \neg c \))
- Sudoku: “at most one 3 in row 5”
- Can translate SAT \iff CSP
  - often CSP more compact
Minesweeper

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) \},... \}

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

image courtesy Andrew Moore
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
Handling uncertainty

- **Minesweeper: what if no safe move?**
- **Say each mine initially present w/ prob p**
- **Common situation: independent “Nature” choices, deterministic rules thereafter**
- **Logic represents deterministic rules ⇒ use logical reasoning as subroutine**

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Handling uncertainty

- **Minesweeper**: what if no safe move?
- **Say each mine initially present w/ prob p**
- **Common situation**: independent “Nature” choices, deterministic rules thereafter
- **Logic represents deterministic rules ⇒ use logical reasoning as subroutine**

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>✗</td>
<td>1</td>
<td>✗</td>
<td>1</td>
<td>✗</td>
</tr>
</tbody>
</table>
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1
Handling uncertainty

- **Minesweeper**: what if no safe move?
- Say each mine initially present with prob $p$
- Common situation: independent “Nature” choices, deterministic rules thereafter
- Logic represents deterministic rules $\implies$ use logical reasoning as subroutine
Handling uncertainty

- **Minesweeper**: what if no safe move?
- **Say each mine initially present w/ prob p**
- **Common situation**: independent “Nature” choices, deterministic rules thereafter
- **Logic represents deterministic rules ⇒ use logical reasoning as subroutine**

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Working with formulas
Truth tables get big fast

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

- Two sentences are equivalent, $A \equiv B$, if they have the same truth value in every model.
  - $(\text{rains} \implies \text{pours}) \equiv (\neg \text{rains} \lor \text{pours})$

- Reflexive, transitive, symmetric

- Simplifying = transforming a formula into a simpler, equivalent formula
Transformation rules

\[(\alpha \land \beta) \equiv (\beta \land \alpha)\] commutativity of \(\land\)

\[(\alpha \lor \beta) \equiv (\beta \lor \alpha)\] commutativity of \(\lor\)

\[((\alpha \land \beta) \land \gamma) \equiv (\alpha \land (\beta \land \gamma))\] associativity of \(\land\)

\[((\alpha \lor \beta) \lor \gamma) \equiv (\alpha \lor (\beta \lor \gamma))\] associativity of \(\lor\)

\[\neg(\neg \alpha) \equiv \alpha\] double-negation elimination

\[(\alpha \land (\beta \lor \gamma)) \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma))\] distributivity of \(\land\) over \(\lor\)

\[(\alpha \lor (\beta \land \gamma)) \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma))\] distributivity of \(\lor\) over \(\land\)

\(\alpha, \beta, \gamma\) 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

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

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

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

... can be derived from truth tables

For example:

- \((a \lor \neg a) \equiv True\)
- \((True \lor a) \equiv True \ (T\ elim)\)
- \((False \land a) \equiv False \ (F\ elim)\)
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
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)

\[
\begin{align*}
happy(John) \land \\
(\neg happy(Bill) \lor happy(Sue)) \land \\
man(Socrates) \land \\
(\neg man(Socrates) \lor mortal(Socrates))
\end{align*}
\]
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

\[(\text{rains } \lor \text{ pours}) \land (\neg \text{pours } \Rightarrow \text{fishing})\]
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 in CNF:*

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

\[(\neg a \lor \neg b \lor \neg d \lor \neg e)\]
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*
Put the following formula in CNF:

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

Parse tree:
Tseitin transformation

- *Introduce temporary variables*
  - \( x = (a \land b) \)
  - \( y = (c \lor d) \)
  - \( z = (y \land e) \)
Tseitin transformation

- To ensure $x = (a \land b)$, want
  - $x \implies (a \land b)$
  - $(a \land b) \implies x$
Tseitin transformation

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

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

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

- $y \Rightarrow (c \lor d)$
- $\neg y \lor c \lor d$
- $(c \lor d) \Rightarrow y$
- $((\neg c \land \neg d) \lor y)$
- $(\neg c \lor y) \land (\neg d \lor y)$
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

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

\[(-x \lor a) \land (-x \lor b) \land (-a \lor -b \lor x) \land\]

\[(-y \lor c \lor d) \land (-c \lor y) \land (-d \lor y) \land\]

\[(-z \lor y) \land (-z \lor e) \land (-y \lor -e \lor z) \land\]

\[(x \lor z)\]
Compositional Semantics
Recall: meaning of a formula is a function models $\rightarrow \{T, F\}$

Why this choice? So that meanings are compositional

Write $[\alpha]$ for meaning of formula $\alpha$

$[\alpha \land \beta](M) = [\alpha](M) \land [\beta](M)$

Similarly for $\lor$, $\neg$, etc.
Proofs
Entailment

- Sentence $A$ entails sentence $B$, $A \models B$, if $B$ is true in every model where $A$ is
  - same as saying that $(A \Rightarrow B)$ is valid
Proof tree

- A tree with a formula at each node
- At each internal node, children $\models$ parent
- Leaves: assumptions or premises
- Root: consequence
- If we believe assumptions, we should also believe consequence
Proof tree example

\[ \text{rains} \Rightarrow \text{pours} \]
\[ \text{pours} \land \text{outside} \Rightarrow \text{rusty} \]
\[ \text{rains} \]
\[ \text{outside} \]
Proof by contradiction

○ Assume opposite of what we want to prove, show it leads to a contradiction

○ Suppose we want to show $KB \vdash S$

○ Write $KB'$ for $(KB \land \neg S)$

○ Build a proof tree with
  ○ assumptions drawn from clauses of $KB'$
  ○ conclusion = $F$
  ○ so, $(KB \land \neg S) \vdash F$ (contradiction)
Proof by contradiction

\[
\text{KB} \\
\text{rains} \rightarrow \text{pours} \\
\text{pours \& outside} \rightarrow \text{rusty} \\
\text{rains} \\
\text{outside} \\
\neg \text{rusty} \\
\uparrow \text{negation of desired conclusion}
\]
Proof by contradiction

KB

\[
\frac{\text{rains} \Rightarrow \text{pours} \quad \therefore \text{pours}}{\text{pours} \land \text{outside} \Rightarrow \text{rusty}}
\]

\[
\text{rains} \quad \text{outside}
\]

\[
\neg \text{rusty}
\]

\[
\neg \text{rusty} 
\]

\[
\neg \text{negative of desired conclusion}
\]

\[
\}
\]