Propositional Logic: Review

Big Ideas

- Logic is a great knowledge representation language for many AI problems
- **Propositional logic** is the simple foundation and fine for some AI problems
- **First order logic** (FOL) is much more expressive as a KR language and more commonly used in AI
- There are many variations: horn logic, higher order logic, three-valued logic, probabilistic logics, etc.

Propositional logic

- **Logical constants**: true, false
- **Propositional symbols**: P, Q,... (atomic sentences)
- Wrapping **parentheses**: ( ... )
- Sentences are combined by **connectives**:
  - $\land$ [conjunction]
  - $\lor$ [disjunction]
  - $\Rightarrow$ [implication / conditional]
  - $\Leftrightarrow$ [biconditional]
  - $\neg$ [negation]
- **Literal**: atomic sentence or negated atomic sentence $P, \neg P$
- **Clause**: a disjunction of literals $P \lor Q, P, P \lor Q \lor R$
- (binary) (unit)
- False is often regarded as empty clause

Examples of PL sentences

- $(P \land Q) \rightarrow R$
  - “If it is hot and humid, then it is raining”
- $Q \rightarrow P$
  - “If it is humid, then it is hot”
- Q
  - “It is humid.”
- We’re free to choose better symbols, btw:
  - Ho = “It is hot”
  - Hu = “It is humid”
  - R = “It is raining”

Propositional logic (PL)

- Simple language for showing key ideas and definitions
- User defines set of propositional symbols, like P and Q
- User defines **semantics** of each propositional symbol:
  - $P$ means “It is hot”, $Q$ means “It is humid”, etc.
- A sentence (well formed formula) is defined as follows:
  - A symbol is a sentence
  - If $S$ is a sentence, then $\neg S$ is a sentence
  - If $S$ is a sentence, then ($S$) is a sentence
  - If S and T are sentences, then ($S \lor T$), ($S \land T$), ($S \rightarrow T$), and ($S \leftrightarrow T$) are sentences
  - A sentence results from a finite number of applications of the rules

Some terms

- The meaning or **semantics** of a sentence determines its interpretation
- Given the truth values of all symbols in a sentence, it can be “evaluated” to determine its truth value (True or False)
- A model for a propositional sentence is a possible world – an assignment of truth values to propositional symbols that makes the sentence True.
- A KB (knowledge base) is a set of sentences; KB is equivalent to the conjunction of all the sentences.
Model for a KB

- Let the KB be \([P \land Q \rightarrow R, Q \rightarrow P]\)
- What are the possible models? Consider all possible assignments of T/F to P, Q and R and check truth tables
  - FFF: OK
  - FFT: OK
  - FTF: NO
  - FFT: NO
  - TFF: OK
  - TTT: OK
- If KB is \([P \land Q \rightarrow R, Q \rightarrow P, Q]\), then the only model is TTT

More terms

- A sentence (or KB) is valid or tautology, if every interpretation is a model.
- A sentence (or KB) is satisfiable, if it has a model.
- A sentence (or KB) is unsatisfiable, if it has no model model.
- Propositional Satisfiability is the problem of deciding if a sentence is satisfiable.
- Theorem Proving is the problem of deciding if a sentence is valid.

More terms

- \(P \text{ entails } Q\), written \(P \models Q\), means that whenever \(P\) is True, so is \(Q\). In other words, all models of \(P\) are also models of \(Q\), i.e., models(\(P\)) is a subset of models(\(Q\)).

Truth tables

- Truth tables are used to define logical connectives and to determine when a complex sentence is true given the values of the symbols in it.

<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 \equiv Q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</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>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

Example of a truth table used for a complex sentence

<table>
<thead>
<tr>
<th>(P)</th>
<th>(\neg P)</th>
<th>(P \rightarrow Q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

On the implies connective: \(P \rightarrow Q\)

- Note that \(\rightarrow\) is a logical connective
- So \(P \rightarrow Q\) is a logical sentence and has a truth value, i.e., is either true or false
- If we add this sentence to the KB, it can be used by an inference rule, *Modus Ponens*, to derive/infer/prove \(Q\) if \(P\) is also in the KB
- Given a KB where \(P=\text{True}\) and \(Q=\text{True}\), we can also derive/infer/prove that \(P \rightarrow Q\) is True

P \rightarrow Q

- When is \(P \rightarrow Q\) true? Check all that apply
  - \(P=Q=\text{true}\)
  - \(P=Q=\text{false}\)
  - \(P=\text{true}, Q=\text{false}\)
  - \(P=\text{false}, Q=\text{true}\)
P \rightarrow Q

• When is \( P \rightarrow Q \) true? Check all that apply
  \( \square \) \( P=Q=\text{true} \)
  \( \checkmark \) \( P=Q=\text{false} \)
  \( \square \) \( P=\text{true}, Q=\text{false} \)
  \( \checkmark \) \( P=\text{false}, Q=\text{true} \)
• We can get this from the truth table for \( \rightarrow \)
• Note: in FOL it’s much harder to prove that a conditional true.
  – Consider proving \( \text{prime}(x) \rightarrow \text{odd}(x) \)

Inference rules

• **Logical inference** creates new sentences that logically follow from a set of sentences (KB)
• An inference rule is **sound** if every sentence X it produces when operating on a KB logically follows from the KB
  – i.e., inference rule creates no contradictions
• An inference rule is **complete** if it can produce every expression that logically follows from (is entailed by) the KB.
  – Note analogy to complete search algorithms

Sound rules of inference

• Here are some examples of sound rules of inference
• Each can be shown to be sound using a truth table

<table>
<thead>
<tr>
<th>RULE</th>
<th>PREMISE</th>
<th>CONCLUSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modus Ponens</td>
<td>A, A \rightarrow B</td>
<td>B</td>
</tr>
<tr>
<td>And Introduction</td>
<td>A, B</td>
<td>A \land B</td>
</tr>
<tr>
<td>And Elimination</td>
<td>A \land B</td>
<td>A</td>
</tr>
<tr>
<td>Double Negation</td>
<td>\neg \neg A</td>
<td>A</td>
</tr>
<tr>
<td>Unit Resolution</td>
<td>A \lor B, \neg B</td>
<td>A</td>
</tr>
<tr>
<td><strong>Resolution</strong></td>
<td>A \lor B, \neg B \lor C</td>
<td>A \lor C</td>
</tr>
</tbody>
</table>

Soundness of modus ponens

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A \rightarrow B</th>
<th>OK?</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>√</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>√</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>√</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>√</td>
</tr>
</tbody>
</table>

Resolution

• **Resolution** is a valid inference rule producing a new clause implied by two clauses containing **complementary literals**
  – A literal is an atomic symbol or its negation, i.e., \( P, \neg P \)
• Amazingly, this is the only inference rule you need to build a sound and complete theorem prover
  – Based on proof by contradiction and usually called resolution refutation
• The resolution rule was discovered by Alan Robinson (CS, U. of Syracuse) in the mid 60s

Resolution

• A KB is actually a set of sentences all of which are true, i.e., a conjunction of sentences.
• To use resolution, put KB into **conjunctive normal form** (CNF), where each sentence written as a disjunction of (one or more) literals

<table>
<thead>
<tr>
<th>Tautologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A \rightarrow B) \leftrightarrow (\neg A \lor B)</td>
</tr>
<tr>
<td>(A \lor (B \land C)) \leftrightarrow (A \lor B) \land (A \lor C)</td>
</tr>
</tbody>
</table>

Example

• KB: \( [P \rightarrow Q, Q \rightarrow R \lor S] \)
  • KB in CNF: \( [\neg P \lor Q, \neg Q \lor R, \neg Q \lor S] \)
• Resolve KB(1) and KB(2) producing: \( \neg P \lor R \) (i.e., \( P \rightarrow R \))
• Resolve KB(1) and KB(3) producing: \( \neg P \lor S \) (i.e., \( P \rightarrow S \))
• New KB: \( [\neg P \lor Q, \neg Q \lor R \lor S, \neg P \lor R, \neg P \lor S] \)
Soundness of the resolution inference rule

<table>
<thead>
<tr>
<th>α</th>
<th>β</th>
<th>γ</th>
<th>α ∨ β</th>
<th>β ∨ γ</th>
<th>α ∨ γ</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>

From the rightmost three columns of this truth table, we can see that

\[(α ∨ β) ∧ (¬β ∨ γ) → (α ∨ γ)\]

is valid (i.e., always true regardless of the truth values assigned to α, β, and γ)

Logical equivalence

- Two sentences p and q are logically equivalent (α or β) if p ↔ q is a tautology
- (and therefore p and q have the same truth value for all truth assignments)

\[(α ∧ β) ≡ (β ∧ α) \text{ commutativity of } \land\]
\[(α ∨ β) ≡ (β ∨ α) \text{ commutativity of } \lor\]
\[(α ∧ (β ∨ γ)) ≡ (α ∧ β) \lor (α ∧ γ) \text{ distributivity of } \land \text{ over } \lor\]
\[(α ∨ (β ∧ γ)) ≡ (α ∨ β) \land (α ∨ γ) \text{ distributivity of } \lor \text{ over } \land\]

Theorem Proving

- A proof is a sequence of sentences, where each is a premise or is derived from earlier sentences in the proof by an inference rule
- The last sentence is the theorem (also called goal or query) that we want to prove
- Example for the “weather problem”

1. Hu premise “It’s humid”
2. Hu → Ho premise “If it’s humid, it’s hot”
3. Ho modus ponens(1,2) “It’s hot”
4. (Ho ∨ Hu) → R premise “If it’s hot & humid, it’s raining”
5. Ho ∨ Hu and introduction(1,3) “It’s hot and humid”
6. R modus ponens(4,5) “It’s raining”

Entailment and derivation

- Entailment: KB |= Q
  - Q is entailed by KB (set sentences) iff there is no logically possible world where Q is false while all the sentences in KB are true
  - Or, stated positively, Q is entailed by KB iff the conclusion is true in every logically possible world in which all the premises in KB are true
- Derivation: KB |- Q
  - We can derive Q from KB if there’s a proof consisting of a sequence of valid inference steps starting from the premises in KB and resulting in Q

Two important properties for inference

Soundness: If KB |- Q then KB |= Q
  - If Q is derived from KB using a given set of rules of inference, then Q is entailed by KB
  - Hence, inference produces only real entailments, or any sentence that follows deductively from the premises is valid

Completeness: If KB |= Q then KB |- Q
  - If Q is entailed by KB, then Q can be derived from KB using the rules of inference
  - Hence, inference produces all entailments, or all valid sentences can be proved from the premises

Propositional logic: pro and con

- Advantages
  - Simple KR language sufficient for some problems
  - Lays the foundation for higher logics (e.g., FOL)
  - Reasoning is decidable, though NP complete, and efficient techniques exist for many problems
- Disadvantages
  - Not expressive enough for most problems
  - Even when it is, it can very “un-concise”
PL is a weak KR language

• Hard to identify “individuals” (e.g., Mary, 3)
• Can’t directly talk about properties of individuals or relations between individuals (e.g., “Bill is tall”)
• Generalizations, patterns, regularities can’t easily be represented (e.g., “all triangles have 3 sides”)
• First-Order Logic (FOL) is expressive enough to represent this kind of information using relations, variables and quantifiers, e.g.,
  • Every elephant is gray: \( \forall x (\text{elephant}(x) \rightarrow \text{gray}(x)) \)
  • There is a white alligator: \( \exists x (\text{alligator}(X) \land \text{white}(X)) \)

PL Example

• Consider the problem of representing the following information:
  – Every person is mortal.
  – Confucius is a person.
  – Confucius is mortal.
• How can these sentences be represented so that we can infer the third sentence from the first two?

PL Example

• In PL we have to create propositional symbols to stand for all or part of each sentence, e.g.:
  \( P = \text{“person”}; Q = \text{“mortal”}; R = \text{“Confucius”} \)
• The above 3 sentences are represented as:
  \( P \rightarrow Q; R \rightarrow P; R \rightarrow Q \)
• The 3rd sentence is entailed by the first two, but we need an explicit symbol, R, to represent an individual, Confucius, who is a member of the classes person and mortal
• Representing other individuals requires introducing separate symbols for each, with some way to represent the fact that all individuals who are “people” are also “mortal”

Propositional logic summary

• Inference is the process of deriving new sentences from old
  – Sound inference derives true conclusions given true premises
  – Complete inference derives all true conclusions from a set of premises
• A valid sentence is true in all worlds under all interpretations
• If an implication sentence can be shown to be valid, then—given its premise—its consequent can be derived
• Different logics make different commitments about what the world is made of and what kind of beliefs we can have
• Propositional logic commits only to the existence of facts that may or may not be the case in the world being represented
  – Simple syntax and semantics suffices to illustrate the process of inference
  – Propositional logic can become impractical, even for very small worlds

A Survey of SAT

Why SAT?

• Fundamental problem from theoretical point of view
• Numerous applications:
  – Solving any NP problem...
  – Verification: Model Checking, theorem-proving, ...
  – AI: Planning, automated deduction, ...
  – Design and analysis: CAD, VLSI
  – Physics: statistical mechanics (models for spin-glass material)

Literals

• A literal is a variable or its negation.
• \( \text{var}(x) \) is the variable associated with a literal \( x \).
  e.g., \( \text{var}(P) = P \), \( \text{var}(\neg Q) = Q \)
• A literal is called negative if it is a negated variable, and positive otherwise.
**SAT basic definitions: literals**

- If \( \text{var}(x) \) is unassigned, then \( x \) is **unresolved**.
- Otherwise, \( x \) is **satisfied** by an assignment \( \alpha \) if \( \alpha(\text{var}(l)) = 1 \) and \( x \) is positive, or \( \alpha(\text{var}(l)) = 0 \) and \( x \) is negative, and **unsatisfied** otherwise.

**SAT basic definitions: clauses**

- The state of an \( n \)-literal clause \( C \) under a partial assignment \( \alpha \) is:
  - **Satisfied** if at least one of \( C \)'s literals is satisfied,
  - **Conflicting** if all of \( C \)'s literals are unsatisfied,
  - **Unit** if \( n-1 \) literals in \( C \) are unsatisfied and 1 literal is unresolved, and
  - **Unresolved** otherwise.

**Example**

Given the partial assignment

\[
(x_1 = 1, x_2 = 0, x_4 = 1)
\]

\((x_1 \lor x_3 \lor \neg x_4)\) is satisfied

\((\neg x_1 \lor x_2)\) is conflicting

\((\neg x_1 \lor \neg x_4 \lor x_3)\) is unit

\((\neg x_1 \lor x_3 \lor x_5)\) is unresolved

**SAT basic definitions: the unit clause rule**

- The **unit clause rule**: in a unit clause the unresolved literal must be satisfied.

**A Basic SAT algorithm**

Given \( \varphi \) in CNF: \((x,y,z),(-x,y),(-y,z),(-x,-y,-z)\)

**Basic Backtracking Search**

- Organize the search in the form of a **decision tree**
  - Each internal node corresponds to a **decision**
  - Depth of the node in the decision tree is called the **decision level**
  - Notation: \( x=\text{v} \) at decision level \( d \)
    - \( x \) is assigned \( \text{v} \) in \( \{0,1\} \) at decision level \( d \)
Backtracking Search in Action

\( \omega_1 = (x_2 \lor x_3) \)
\( \omega_2 = (\neg x_1 \lor \neg x_4) \)
\( \omega_3 = (\neg x_2 \lor x_4) \)
\( \omega_4 = (\neg x_1 \lor x_2 \lor \neg x_3) \)

\( x_1 = 0 \) \( \Rightarrow x_4 = 0 \) \( \Rightarrow x_2 = 0 \)
\( x_2 = 0 \) \( \Rightarrow x_3 = 1 \)
\( x_3 = 0 \) \( \Rightarrow x_4 = 0 \) \( \Rightarrow x_2 = 0 \)
\( x_4 = 0 \) \( \Rightarrow x_2 = 0 \)

No backtrack in this example, regardless of the decision!

Backtracking Search in Action

Add a clause
\( \omega_5 = (x_2 \lor x_3) \)
\( x_1 = 0 \) \( \Rightarrow x_4 = 0 \)
\( x_2 = 0 \) \( \Rightarrow x_3 = 1 \)
\( x_3 = 1 \) \( \Rightarrow x_4 = 0 \)
\( x_4 = 0 \) \( \Rightarrow x_2 = 0 \)

Decision heuristics

DLIS (Dynamic Largest Individual Sum)

- Maintain a counter for each literal: in how many unresolved clauses it appears?
- Decide on the literal with the largest counter.
- Requires \( O(#\text{literals}) \) queries for each decision.

Numerous progressing heuristics

- Hill-climbing
- Tabu-list
- Simulated-annealing
- Random-Walk
- ...
**Boolean Encoding for 4-Queen**

- **16 Boolean variables** $X_{ij}$, $i = 1$ to $4$, $j = 1$ to $4$
- **Constraints are of the form:**
  - $X_{ij} = 1$ iff “there is a
    queen on location $(i,j)$.”
- **Row and columns**
  - If $(X_{ij} = 1)$ then $(X_{ik} = 0)$ for all $k = 1$ to $4$, $k \neq j$ (logical constraint)
  - $X_{ij} = 1$ iff $X_{kj} = 0$ for all $k = 1$ to $4$, $k \neq i$
- **Diagonals**
  - $X_{ij} = 1$ iff $X_{i+j,j} = 0$ for all $l = 1$ to $3$, $i+l \leq 4$; $j+l \leq 4$ (right and up)
  - $X_{ij} = 1$ iff $X_{i-j,j} = 0$ for all $l = 1$ to $3$, $i-l \geq 1$; $j+l \leq 4$ (right and down)
  - $X_{ij} = 1$ iff $X_{i-j,j-l} = 0$ for all $l = 1$ to $3$, $i-l \geq 1$; $j-l \geq 1$ (left and down)
  - $X_{ij} = 1$ iff $X_{i+j,j-l} = 0$ for all $l = 1$ to $3$, $i+l \leq 4$; $j-l \geq 1$ (left and up)

Need $N$ (= 4) queens on board!
For each row $i$: $(X_{i1} \lor X_{i2} \lor X_{i3} \lor X_{i4})$