Proof Methods for Propositional Logic

Russell and Norvig Chapter 7

Logical equivalence

- Two sentences are logically equivalent iff they are true in the same models: \( \alpha \equiv \beta \) iff \( \alpha \vdash \beta \) and \( \beta \vdash \alpha \)

\[
\begin{align*}
(\alpha \land \beta) & \equiv (\beta \land \alpha) \quad \text{commutativity of } \land \\
(\alpha \lor \beta) & \equiv (\beta \lor \alpha) \quad \text{commutativity of } \lor \\
((\alpha \land \beta) \land \gamma) & \equiv (\alpha \land (\beta \land \gamma)) \quad \text{associativity of } \land \\
((\alpha \lor \beta) \lor \gamma) & \equiv (\alpha \lor (\beta \lor \gamma)) \quad \text{associativity of } \lor \\
\neg \neg \alpha & \equiv \alpha \quad \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) & \equiv (\neg \alpha \lor \beta) \quad \text{implication elimination} \\
(\alpha \iff \beta) & \equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination} \\
\neg (\alpha \land \beta) & \equiv (\neg \alpha \lor \neg \beta) \quad \text{de Morgan} \\
\neg (\alpha \lor \beta) & \equiv (\neg \alpha \land \neg \beta) \quad \text{de Morgan} \\
(\alpha \lor (\beta \lor \gamma)) & \equiv ((\alpha \lor \beta) \lor (\alpha \lor \gamma)) \quad \text{distributivity of } \lor \text{ over } \land \\
(\alpha \land (\beta \land \gamma)) & \equiv ((\alpha \land \beta) \land (\alpha \land \gamma)) \quad \text{distributivity of } \land \text{ over } \lor
\end{align*}
\]

Validity and satisfiability

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

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

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

- A sentence is unsatisfiable if it is false in all models
  - e.g., \( A \land \neg A \)

Satisfiability is connected to inference via the following:
\( KB \vdash \alpha \) if and only if \( (KB \land \neg \alpha) \) is unsatisfiable
(known as proof by contradiction)

Inference rules

- Modus Ponens

\[
\begin{align*}
A \Rightarrow B, \quad A \\
\hline
B
\end{align*}
\]

Example:
“raining implies soggy courts”, “raining”
Infer: “soggy courts”
Example

- KB: \{A \Rightarrow B, B \Rightarrow C, A\}
- Is C entailed?
- Yes.

<table>
<thead>
<tr>
<th>Given</th>
<th>Rule</th>
<th>Inferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>A \Rightarrow B, A</td>
<td>Modus Ponens</td>
<td>B</td>
</tr>
<tr>
<td>B \Rightarrow C, B</td>
<td>Modus Ponens</td>
<td>C</td>
</tr>
</tbody>
</table>

Inference rules (cont.)

- Modus Tollens
  \[ A \Rightarrow B, \neg B \]
  \[ \neg A \]
  Example:
  “raining implies soggy courts”, “courts not soggy”
  Infer: “not raining”
- And-elimination
  \[ A \land B \]
  \[ A \]

Reminder: The Wumpus World

- Performance measure
  - gold: +1000, death: -1000
  - -1 per step, -10 for using the arrow
- Environment
  - Squares adjacent to wumpus: smelly
  - Squares adjacent to pit: breezy
  - Glitter iff gold is in the same square
  - Shooting kills wumpus if you are facing it
  - Shooting uses up the only arrow
  - Grabbing picks up gold if in same square
- Sensors: Stench, Breeze, Glitter, Bump
- Actuators: Left turn, Right turn, Forward, Grab, Release, Shoot

Inference in the wumpus world

Given:
1. \neg B_{1,1}
2. \neg B_{1,1} \iff (P_{1,2} \lor P_{2,1})
Let’s make some inferences:
1. \neg B_{1,1} \iff (P_{1,2} \lor P_{2,1}) \iff B_{1,1} (By definition of the biconditional)
2. (P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1} (And-elimination)
3. \neg B_{1,1} \Rightarrow \neg (P_{1,2} \lor P_{2,1}) (equivalence with contrapositive)
4. \neg (P_{1,2} \lor P_{2,1}) (modus ponens)
5. \neg P_{1,2} \land \neg P_{2,1} (DeMorgan’s rule)
6. \neg P_{1,2} (And-elimination)
7. \neg P_{2,1} (And-elimination)
More inference

- Recall that when we were at (2,1) we could not decide on a safe move, so we backtracked, and explored (1,2), which yielded $\neg B_{1,2}$. This yields $\neg P_{2,2} \land \neg P_{1,3}$.

- Now we can consider the implications of $B_{2,1}$.

Resolution

1. $\neg P_{2,2}, \neg P_{1,1}$
2. $B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1})$ (biconditional elimination)
3. $B_{2,1} \Rightarrow (P_{1,1} \lor P_{2,2} \lor P_{3,1})$ (modus ponens)
4. $P_{1,1} \lor P_{2,2} \lor P_{3,1}$ (resolution rule)
5. $P_{1,1} \lor P_{3,1}$ (resolution rule)

The resolution rule: if there is a pit in (1,1) or (3,1), and it’s not in (1,1), then it’s in (3,1).

$P_{1,1} \lor P_{3,1} \quad \neg P_{1,1}$

$P_{3,1}$

Resolution is sound

For simplicity let’s consider clauses of length two:

$$l_1 \lor \ldots \lor l_k \quad \neg m_1 \lor \ldots \lor m_n$$

To derive the soundness of resolution consider the values $l_2$ can take:
- If $l_2$ is True, then since we know that $\neg l_2 \lor l_3$ holds, it must be the case that $l_3$ is True.
- If $l_2$ is False, then since we know that $l_1 \lor l_2$ holds, it must be the case that $l_1$ is True.
Resolution

- Properties of the resolution rule:
  - Sound
  - Complete
- Resolution can be applied only to disjunctions of literals. How can it be complete?
- Turns out any knowledge base can be expressed as a conjunction of disjunctions (conjunctive normal form, CNF).
- Example: \((A \lor \neg B) \land (B \lor \neg C \lor \neg D)\)

Conversion to CNF

1. Eliminate \(\iff\), replacing \(\alpha \iff \beta\) with \((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)\).
2. Eliminate \(\Rightarrow\), replacing \(\alpha \Rightarrow \beta\) with \(\neg \alpha \lor \beta\).
3. Move \(\neg\) inwards using de Morgan’s rules and double-negation:
   \((-B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (-P_{1,2} \lor -P_{2,1}) \lor B_{1,1}\)
4. Apply distributive law (\(\land\) over \(\lor\)) and flatten:
   \((-B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (-P_{1,2} \lor -P_{2,1}) \land (-P_{2,1} \lor B_{1,1})\)

Converting to CNF (II)

- While converting expressions, note that
  - \(((\alpha \lor \beta) \lor \gamma)\) is equivalent to \((\alpha \lor \beta \lor \gamma)\)
  - \(((\alpha \land \beta) \land \gamma)\) is equivalent to \((\alpha \land \beta \land \gamma)\)
- Why does this algorithm work?
  - Because \(\Rightarrow\) and \(\iff\) are eliminated
  - Because \(\neg\) is always directly attached to literals
  - Because what is left must be \(\land\)’s and \(\lor\)’s, and they can be distributed over to make CNF clauses
Using resolution

- Even if our KB entails a sentence $\alpha$, resolution is not guaranteed to produce $\alpha$.
- To get around this we use proof by contradiction, i.e., show that $KB \land \neg \alpha$ is unsatisfiable.
- Resolution is complete with respect to proof by contradiction.

Example of proof by resolution

$KB = (B_{1,1} \iff (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1}$

in CNF… $\neg B_{1,1} \lor (\neg P_{1,2} \lor P_{2,1}) \land (\neg P_{2,1} \lor B_{1,1})$

$\alpha = \neg P_{1,2}$

Resolution yielded the empty clause.
The empty clause is False (a disjunction is True only if at least one of its disjuncts is true).

Automated Theorem Proving

- How do we automate the inference process?
  - Step 1: assume the negation of the consequent and add it to the knowledge base
  - Step 2: convert KB to CNF
    - i.e. a collection of disjunctive clauses
  - Step 3: Repeatedly apply resolution until:
    - It produces an empty clause (contradiction), in which case the consequent is proven, or
    - No more terms can be resolved, in which case the consequent cannot be proven

Resolution Algorithm

function PL-RESOLUTION(KB, $\alpha$) returns true or false

inputs: KB, the knowledge base, a sentence in propositional logic $\alpha$, the query, a sentence in propositional logic
 clauses $\leftarrow$ the set of clauses in the CNF representation of $KB \land \neg \alpha$
 new $\leftarrow \{\}$
 loop do
   for each pair of clauses $C_i,C_j$ in clauses do
      resolvents $\leftarrow$ PL-RESOLVE($C_i,C_j$)
      if resolvents contains the empty clause then return true
      new $\leftarrow$ new U resolvents
   if new $\subseteq$ clauses then return false
   clauses $\leftarrow$ clauses U new

CS440 Fall 2015 17
CS440 Fall 2015 18
CS440 Fall 2015 19
CS440 Fall 2015 20
Another example

- If it rains, I get wet.
- If I’m wet, I get mad.
- Given that I’m not mad, prove that it’s not raining.

Inference for Horn clauses

Horn Form (special form of CNF)

- KB = conjunction of Horn clauses
- Horn clause = disjunction of literals of which at most one is positive
  - Example: C ∨ B ∨ ¬A

Modus Ponens is a natural way to make inference in Horn KBs (recall a⇒b is equivalent to ¬a ∨ b)

\[ \alpha_1, \ldots, \alpha_n, \quad \alpha_1 \land \ldots \land \alpha_n \Rightarrow \beta \]

- Successive application of modus ponens leads to algorithms that are sound and complete, and run in linear time

Forward chaining

- Idea: fire any rule whose premises are satisfied in the KB
  - add its conclusion to the KB, until query is found
  - \[ P \Rightarrow Q \]
  - \[ L \land M \Rightarrow P \]
  - \[ B \land L \Rightarrow M \]
  - \[ A \land P \Rightarrow L \]
  - \[ A \land B \Rightarrow L \]
  - \[ A \]
  - \[ B \]

Forward chaining example
Forward chaining example

Forward chaining example

Forward chaining example

Forward chaining example

Forward chaining algorithm

Forward chaining is sound and complete for Horn KB

function PL-FC-ENTAILS?(KB, q) returns true or false
inputs: KB, knowledge base, a set of propositional definite clauses
        q, the query, a propositional symbol
count ← a table, where count[c] is the number of symbols in c’s premise
inferred ← a table, where inferred[s] is initially false for all symbols
agenda ← a queue of symbols, initially symbols known to be true in KB
while agenda is not empty do
    p ← POP(agenda)
    if p=q then return true
    if inferred[p]=false then
        inferred[p] ← true
        for each clause c in KB where p is in c.PREMISE do
            decrement count[c]
        if count[c]=0 then add c.CONCLUSION to agenda
    return false
Backward chaining
aka Goal Directed reasoning
Idea: work backwards from the query $q$:

- check if $q$ is known already, or
- prove by backward chaining all premises of some rule concluding $q$

Avoid loops:
- check if new subgoal is already on the goal stack

Avoid repeated work: check if new subgoal
- has already been proved true, or
- has already failed

Backward chaining example
Backward chaining example
Forward vs. backward chaining

- FC is data-driven
- May do lots of work that is irrelevant to the goal

- BC is goal-driven, appropriate for problem-solving,
  - e.g., What courses do I need to take to graduate? How do I get into a PhD program?

- Complexity of BC can be much less than linear in size of KB
Efficient propositional inference

Two families of efficient algorithms for satisfiability:

- Complete backtracking search algorithms:
  - DPSSL algorithm (Davis, Putnam, Logemann, Loveland)

- Local search algorithms
  - WalkSAT algorithm:
    - Start with a random assignment
    - At each iteration pick an unsatisfied clause and pick a symbol in the clause to flip; alternate between:
      - Pick the symbol that minimizes the number of unsatisfied clauses
      - Pick a random symbol

Is WalkSAT sound? Complete?

In the wumpus world

A wumpus-world agent using propositional logic:

$$
\neg P_{1,1} \\
\neg W_{1,1} \\
B_{x,y} \equiv (P_{x+1,y} \lor P_{x-1,y} \lor P_{x,y+1} \lor P_{x,y-1}) \\
S_{x,y} \equiv (W_{x+1,y} \lor W_{x-1,y} \lor W_{x,y+1} \lor W_{x,y-1}) \\
W_{1,1} \lor W_{1,2} \lor \ldots \lor W_{4,4} \\
\neg W_{1,1} \lor \neg W_{1,2} \\
\neg W_{1,1} \lor \neg W_{1,3} \\
\ldots \\
\Rightarrow 64 \text{ distinct proposition symbols, 155 sentences}
$$

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: 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

- Resolution is complete for propositional logic

- Forward, backward chaining are linear-time, complete for Horn clauses

- Propositional logic lacks expressive power