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 $\alpha \models \beta$ and $\beta \models \alpha$

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

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

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., \( A \lor B \)

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:
\( KB \models \alpha \) if and only if \( (KB \land \neg \alpha) \) is unsatisfiable
(known as proof by contradiction)
Inference rules

- Modus Ponens
  \[ A \implies B, \ A \quad \overline{\quad} \quad B \]

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**
  
  \[ A \implies B, \neg B \]
  
  \[ \neg A \]
  
  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. \( B_{1,1} \iff (P_{1,2} \lor P_{2,1}) \)

Let’s make some inferences:

1. \((B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})\) (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)
Inference using inference rules

- Inference using inference rules is sound.
- Is it complete? Depends if we have a rich enough set of rules.
- Inference is a search problem.
- Resolution: a sound inference rule that when coupled with a complete search method yields a complete inference algorithm
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. \( B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1}) \)

3. \( B_{2,1} \implies (P_{1,1} \lor P_{2,2} \lor P_{3,1}) \) (biconditional elimination)

4. \( P_{1,1} \lor P_{2,2} \lor P_{3,1} \) (modus ponens)

5. \( P_{1,1} \lor P_{3,1} \) (resolution rule)

6. \( P_{3,1} \) (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).

\[
P_{1,1} \lor P_{3,1}, \quad \neg P_{1,1} \quad \underbrace{\quad \therefore P_{3,1}}_{\text{resolution rule}}
\]
Resolution

Unit Resolution inference rule:

\[
\begin{array}{c}
l_1 \lor \ldots \lor l_k, \\
m \\
\hline
l_1 \lor \ldots \lor l_{i-1} \lor l_{i+1} \lor \ldots \lor l_k
\end{array}
\]

where \( l_i \) and \( m \) are complementary literals.

Full Resolution

\[
\begin{array}{c}
l_1 \lor \ldots \lor l_k, \\
m_1 \lor \ldots \lor m_n \\
\hline
l_1 \lor \ldots \lor l_{i-1} \lor l_{i+1} \lor \ldots \lor l_k \lor m_1 \lor \ldots \lor m_{j-1} \lor m_{j+1} \lor \ldots \lor m_n
\end{array}
\]

where \( l_i \) and \( m_j \) are complementary literals.
Resolution is sound

For simplicity let’s consider clauses of length two:

\[ l_1 \lor l_2, \quad \neg l_2 \lor l_3 \]

\[ l_1 \lor l_3 \]

To demonstrate 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 knowledgebase 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

B_{1,1} \iff (P_{1,2} \lor P_{2,1})

1. Eliminate \iff, replacing \alpha \iff \beta with (\alpha \implies \beta) \land (\beta \implies \alpha).
   \[(B_{1,1} \implies (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \implies B_{1,1})\]

2. Eliminate \implies, replacing \alpha \implies \beta with \neg \alpha \lor \beta.
   \[\neg B_{1,1} \lor P_{1,2} \lor P_{2,1} \land (\neg (P_{1,2} \lor P_{2,1}) \lor B_{1,1})\]

3. Move \neg inwards using de Morgan's rules and double-negation:
   \[\neg B_{1,1} \lor P_{1,2} \lor P_{2,1} \land (\neg P_{1,2} \lor \neg P_{2,1}) \lor B_{1,1}\]

4. Apply distributive law (\land over \lor) and flatten:
   \[\neg B_{1,1} \lor P_{1,2} \lor P_{2,1} \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1})\]
Converting to CNF

Every sentence can be converted to CNF

1. Replace $\alpha \iff \beta$ with $(\alpha \implies \beta) \land (\beta \implies \alpha)$
2. Replace $\alpha \implies \beta$ with $(\neg \alpha \lor \beta)$
3. Move $\neg$ “inward”
   1. Replace $\neg (\neg \alpha)$ with $\alpha$
   2. Replace $\neg (\alpha \land \beta)$ with $(\neg \alpha \lor \neg \beta)$
   3. Replace $\neg (\alpha \lor \beta)$ with $(\neg \alpha \land \neg \beta)$
4. Replace $(\alpha \lor (\beta \land \gamma))$ with $(\alpha \lor \beta) \land (\alpha \lor \gamma)$
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 \(\Leftrightarrow\) 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... $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1})$

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

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

- How do we automate the inference process?
  - Step 1: assume the negation of the consequent and add it to the knowledgebase
  - 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, α) returns true or false

inputs: KB, the knowledge base, a sentence in propositional logic
         α, the query, a sentence in propositional logic

clauses ← the set of clauses in the CNF representation of \( KB \land \neg \alpha \)

new ← {}

loop do

  for each pair of clauses \( C_i, C_j \) in clauses do
    resolvents ← PL-RESOLVE(\( C_i, C_j \))
    if resolvents contains the empty clause then return true
    new ← new U resolvents
    if new \( \subseteq \) clauses then return false

  clauses ← clauses U new
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**

- KB = conjunction of Horn clauses
- Horn clause = propositional symbol; or (conjunction of symbols) \( \Rightarrow \) symbol

Example of a Horn clause: \( (C \land D) \Rightarrow B \)

Example of a Horn KB: \( C \land (B \Rightarrow A) \land ((C \land D) \Rightarrow B) \)

Horn form is a special case of CNF where a clause can have at most one positive literal
Inference for Horn clauses

Horn Form

KB = conjunction of Horn clauses
Horn clause =
  propositional symbol; or
  (conjunction of symbols) ⇒ symbol

Example: C ∧ (B ⇒ A) ∧ ((C ∧ D) ⇒ B)

Modus Ponens is a natural way to make inference in Horn KBs

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

\[ \quad \beta \]

- 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 example
Forward chaining algorithm

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

Forward chaining is sound and complete for Horn KB
Backward chaining

Idea: work backwards from the query $q$:
check if $q$ is known already, or
prove by backward chaining all premises of some rule concluding $q$
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining

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
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?

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

Two families of efficient algorithms for satisfiability:

- Backtracking search algorithms:
  - DPLL 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:

\[ \neg P_{1,1} \]
\[ \neg W_{1,1} \]
\[ B_{x,y} \iff (P_{x,y+1} \lor P_{x,y-1} \lor P_{x+1,y} \lor P_{x-1,y}) \]
\[ S_{x,y} \iff (W_{x,y+1} \lor W_{x,y-1} \lor W_{x+1,y} \lor W_{x-1,y}) \]
\[ W_{1,1} \lor W_{1,2} \lor \ldots \lor W_{4,4} \]
\[ \neg W_{1,1} \lor \neg W_{1,2} \]
\[ \neg W_{1,1} \lor \neg W_{1,3} \]
\[ \ldots \]

\[ \Rightarrow 64 \text{ distinct proposition symbols, 155 sentences} \]
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 a knowledge base
  - **Inference:** deriving sentences from other sentences
  - **Soundness:** derivations produce only entailed sentences
  - **Completeness:** derivations can produce all entailed sentences
Summary

Methods for inference
- **Resolution** is complete for propositional logic
- **Forward, backward chaining** are linear-time, complete for Horn clauses

General observation:
- Propositional logic lacks expressive power