Inference Using Propositional Logic

CSE 415: Introduction to Artificial Intelligence
University of Washington
Spring 2017

© S. Tanimoto and University of Washington, 2017

Outline

• Review of propositional logic terminology
• Notation, syllogism, rules of inference
• Proof by perfect induction
• Clause form
• Proof by resolution

Readings for Logic

For basics of logic as a means of knowledge representation, consult the textbook material in:
Section 4.8. “Propositional and Predicate Logic” (in EAIP, Part 2).

Propositional logic is covered in 4.8.1-4.8.6.
(optional: Predicate logic is covered in 4.8.7-4.8.10.)

For methods of inference with logic, see Chapter 6 (in EAIP, Part 4).

Role of Logical Inference in AI

The single most important inference method.

But:

• Doesn’t handle uncertain information well.

• Needs algorithmic help
  – prone to the combinatorial explosion.

Propositional Calculus

A formal mathematical system of notation and evaluation rules for representing and processing true-false statements involving logical relationships.

An important foundation for knowledge representation in artificial intelligence.

Several modern techniques combine logic and probability (e.g., Markov Logic Networks).

A Logical Syllogism

If it is raining, then I am doing my homework.

It is raining.

Therefore, I am doing my homework.
Another Syllogism

It is not the case that steel cannot float.
Therefore, steel can float.

Terminology of the Propositional Calculus

Proposition symbols:
P, Q, R, P₁, P₂, ..., Q₁, Q₂, ..., R₁, R₂, ...

Atomic proposition: a statement that does not specifically contain substatements.
P: "It is raining."
Q: "Neither did Jack eat nor did he drink."

Compound proposition: A statement formed from one or more atomic propositions using logical connectives.
P ∨ Q: Either it is raining, or neither did Jack eat nor did he drink.

Logical Connectives

Negation: ¬P  not P
Conjunction: P ∧ Q  P and Q
Disjunction: P ∨ Q  P or Q
Exclusive OR: P ≡ Q  P exclusive-or Q

Logical Connectives (Cont)

NAND: ¬(P ∧ Q)  P nand Q
NOR: ¬(P ∨ Q)  P nor Q
Implies: P → Q  if P then Q

Logically Complete Sets of Connectives

{¬, ∨} form a logically complete set.
P ∧ Q  = ¬(¬P ∨ ¬Q)

{¬, →} form a logically complete set
P ∧ Q  = ¬(P → ¬Q)

{¬, ∧} form a logically complete set
P ∨ Q  = ¬(¬P ∧ ¬Q)

Syllogism: General Form

Premise 1
Premise 2
...
Premise n

Conclusion

P₁ ∧ P₂ ∧ ... ∧ Pₙ → C
Modus Ponens:
An important rule of inference

\[ P \rightarrow Q \quad \text{conditional} \]
\[ P \quad \text{antecedent} \]

\[ \downarrow \]
\[ Q \quad \text{consequent} \]

aka the “cut rule”

Modus Tollens:
(modus ponens in reverse)

\[ P \rightarrow Q \quad \text{conditional} \]
\[ \neg Q \quad \text{consequent denied} \]

\[ \downarrow \]
\[ \neg P \quad \text{antecedent denied} \]

Can be proved using “transposition” – taking the contrapositive of the conditional:

\[ P \rightarrow Q \iff \neg Q \rightarrow \neg P \]

\[ \therefore \text{by modus ponens}, \neg P \]

Algorithms for Logical Inference

Issues:

- Goal-directed or not?
- Always exponential in time? Space?
- Intelligible to users?
- Readily applicable to problem solving?

Proof by Perfect Induction

Prove that \( P \rightarrow \neg P \lor Q \implies Q \)

\[
\begin{array}{cccccc}
 P & Q & \neg P \lor Q & P \land \neg (P \lor Q) & (P \land \neg (P \lor Q)) \lor Q & Q \\
 T & T & T & T & T & T \\
 T & F & F & F & T & T \\
 F & T & T & F & T & T \\
 F & F & T & F & T & T \\
\end{array}
\]

Perfect Induction

- Given formulas: \( G_0, G_1, \ldots, G_{n-1} \), prove conclusion \( C \).
- Create a truth table with a column for each propositional variable, each premise, and the conclusion.
- Find all rows in which \( G_0 \) through \( G_{n-1} \) are all True.
- See if \( C \) is True in all those rows.
- If so, the syllogism is proven; otherwise, it’s not valid.

Perfect Induction: Characteristics

- Goal directed (compute only columns of interest)
- Always exponential in time AND space (as a function of the number of propositional variables)
- Somewhat understandable to non-technical users
- Straightforward algorithmically
- Not considered appropriate for general problem solving.
 Resolution is a general proof technique that supports logical reasoning.
It underlies the PROLOG language.
It requires that formulas be in a restricted form ("clause form").
PROLOG imposes a further constraint that the clauses be "Horn clauses."

Expressions such as \( P \), \( \neg P \), \( Q \), and \( \neg Q \) are called literals.
They are atomic formulas to which a negation may be prefixed.

A clause is an expression of the form \( L_1 \lor L_2 \lor \ldots \lor L_q \)
where each \( L_i \) is a literal. Here \( q \) is any non-negative integer.

Any propositional calculus formula can be represented as a set of clauses.

\[
\neg([P \land (Q \rightarrow R)]) \quad \text{starting formula}
\]
\[
\neg([P \land \neg Q] \lor [P \land R]) \quad \text{eliminate } \rightarrow
\]
\[
\neg([P \land \neg Q] \land [P \land R]) \quad \text{distribute } \land \text{over } \lor
\]
\[
\neg(\neg P \lor \neg Q) \land \neg(P \lor R) \quad \text{DeMorgan's law}
\]
\[
\neg(\neg P \lor Q) \land \neg(P \lor \neg R) \quad \text{" } \quad \text{"}
\]
\[
\neg P \lor Q, \quad \neg P \lor R \quad \text{ Double neg. and break into clauses}
\]

Two clauses having a pair of complementary literals can be resolved to produce a new clause that is logically implied by its parent clauses.

- For example,
  \[
  Q \lor \neg R \lor S, \quad R \lor \neg P \implies Q \lor S \lor \neg P
  \]

- For example,
  \[
  P \lor Q, \quad \neg Q \lor R \implies P \lor R
  \]

- For example,
  \[
  P \lor \neg P \lor R \implies R
  \]

- For example,
  \[
  P \lor \neg P \implies \bot \quad \text{(the null clause)}
  \]

Consider four examples:

- For example,
  \[
  P \implies \bot \quad \text{(the null clause)}
  \]

- For example,
  \[
  P \implies \neg P \lor R \implies R \quad \text{(modus ponens, since } P \implies R \iff \neg P \lor R \}
  \]

To prove: \((P \lor Q) \land (Q \rightarrow R) \implies (P \rightarrow R)\)

Negate the conclusion:

\[
(P \lor Q) \land (Q \rightarrow R) \implies (P \rightarrow R)
\]

Obtain clause form:

\[
\neg P \lor Q, \quad \neg Q \lor R, \quad P, \quad \neg R
\]

Derive the null clause \( [F] \) using resolution:

- \( Q \) by resolving \( P \) with \( \neg P \lor Q \).
- \( R \) by resolving \( Q \) with \( \neg Q \lor R \).
- \( F \) by resolving \( R \) with \( \neg R \).
Reductio ad Absurdum

A proof by resolution uses RAA (proof by contradiction).

Original syllogism:

Premise 1
Premise 2
...
Premise n
---------------
Conclusion

Syllogism for RAA:

Premise 1
Premise 2
...
Premise n
...Conclusion
---------------
[]

Resolution: Characteristics

- Not necessarily goal directed (can be used in either forward-chaining or backward-chaining systems).
- Time and space requirements depend on the algorithm in which resolution is embedded.
- Can be made understandable to non-technical users
- Needs to be combined with a search algorithm.
- Can be very appropriate for general problem solving (e.g., using PROLOG)