Inference Using Propositional Logic

CSE 415: Introduction to Artificial Intelligence
University of Washington
Spring 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, P_1, P_2, … , Q_1, Q_2, …, R_1, R_2, …

***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 v 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 v Q	***P or Q***
Exclusive OR: P <> Q	***P exclusive-or Q***

Logical Connectives (Cont)

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

Logically Complete Sets of Connectives

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

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

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

Syllogism: General Form

Premise 1
Premise 2
…
Premise n

Conclusion

$P_1 \wedge P_2 \wedge \ldots \wedge P_n \rightarrow C$

Modus Ponens: An important rule of inference

$P \rightarrow Q$	conditional
P	antecedent

Q	consequent

aka the "cut rule"

Modus Tollens: (*modus ponens* in reverse)

$P \rightarrow Q$	conditional
$\neg Q$	consequent denied

$\neg P$	antecedent denied

Can be proved using "transposition" – taking the *contrapositive* of the conditional:

$P \rightarrow Q \Leftrightarrow \neg Q \rightarrow \neg P$

$\neg Q$

therefore, 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, $\neg P \vee Q \Rightarrow Q$

P	Q	$\neg P \vee Q$	$P \wedge (\neg P \vee Q)$	$(P \wedge (\neg P \vee Q)) \rightarrow Q$
T	T	T	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	T

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.

Getting Ready for Resolution

- 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."

Clause Form

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

A clause is an expression of the form $L_1 \vee L_2 \vee \ldots \vee L_q$
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.

$\neg(P \wedge (Q \rightarrow R))$	starting formula
$\neg(P \wedge (\neg Q \vee R))$	eliminate →
$\neg((P \wedge \neg Q) \vee (P \wedge R))$	distribute ∧ over v.
$\neg(P \wedge \neg Q) \wedge \neg(P \wedge R)$	DeMorgan's law
$(\neg P \vee \neg\neg Q) \wedge (\neg P \vee \neg R)$	" "
$\neg P \vee Q$, $\neg P \vee \neg R$	Double neg. and break into clauses

Propositional Resolution

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

e.g.,

$Q \vee \neg R \vee S$, $R \vee \neg P$ ⇒ $Q \vee S \vee \neg P$

$P \vee Q$, $\neg Q \vee R$ ⇒ $P \vee R$

P, $\neg P \vee R$ ⇒ R

P, ¬P ⇒ [] (the null clause)

Why Does Resolution Work?

Consider four examples:

P, ¬P ⇒ [] (the null clause)
(a contradiction)

P, $\neg P \vee R$ ⇒ R
(modus ponens, since $P \rightarrow R \equiv \neg P \vee R$)

Why Does Resolution Work?

$P \vee Q$, $\neg Q \vee R$ ⇒ $P \vee R$
(Suppose Q is false, then P must be true;
If Q is not false, then ¬Q is false and R must be true.
One way or the other, either P or R must be true.

$P_1 \vee \ldots \vee P_n \vee Q$, $\neg Q \vee R_1 \vee \ldots \vee R_m$ ⇒ $P_1 \vee \ldots \vee P_n \vee R_1 \vee \ldots \vee R_m$
(We get this from the previous version by letting
$P = P_1 \vee \ldots \vee P_n$ and $R = R_1 \vee \ldots \vee R_m$.)

This last example is the general case.

Proof Using Resolution

To Prove: $(P \rightarrow Q) \wedge (Q \rightarrow R) \Rightarrow (P \rightarrow R)$

Negate the conclusion:
$(P \rightarrow Q) \wedge (Q \rightarrow R) \Rightarrow \neg(P \rightarrow R)$

Obtain clause form:
$\neg P \vee Q$, $\neg Q \vee R$, P, ¬R

Derive the null clause (F) using resolution:
Q by resolving P with $\neg P \vee Q$.
R by resolving Q with $\neg Q \vee R$.
F by resolving R with ¬R.

Reductio ad Absurdum

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

Original syllogism:	*Syllogism for RAA:*
Premise 1	**Premise 1**
Premise 2	**Premise 2**
…	**…**
Premise n	**Premise n**
---------------	**¬Conclusion**
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)