Propositional and First-Order Logic

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**:

  $\wedge$ and [conjunction]

  $\vee$ or [disjunction]

  $\Rightarrow$ implies [implication / conditional]

  $\Leftrightarrow$ is equivalent [biconditional]

  $\neg$ not [negation]
- **Literal**: atomic sentence or negated atomic sentence

  P, $\neg$ P

Examples of PL sentences

- $(P \wedge Q) \rightarrow R$

  "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 $(S \vee T)$, $(S \wedge T)$, $(S \rightarrow T)$, and $(S \leftrightarrow T)$ are sentences
  – A sentence results from a finite number of applications of the rules

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

Predicate Logic

INT404

A Predicate Logic Example

1. Marcus was a man.
2. Marcus was a Pompeian.
3. All Pompeian were Romans.
4. Caesar was a ruler.
5. All Romans were either loyal to Caesar or hated him.
6. Everyone is loyal to someone.
7. People only try to assassinate rulers they are not loyal to.
8. Marcus tried to assassinate Caesar.

A Predicate Logic Example

1. Marcus was a man.
   $man(Marcus)$
2. Marcus was a Pompeian.
   $Pompeian(Marcus)$
3. All Pompeians were Romans.
   $\forall x : Pompeian(x) \rightarrow Roman(x)$
4. Caesar was a ruler.
   $ruler(Caesar)$
5. All Romans were either loyal to Caesar or hated him.
   $\forall x : Roman(x) \rightarrow loyalto(x, Caesar) \vee hate(x, Caesar)$
6. Everyone is loyal to someone.
   $\forall x : \exists y : loyalto(x, y)$
7. People only try to assassinate rulers they aren't loyal to.
   $\forall x : \forall y : person(x) \wedge ruler(y) \wedge tryassassinate(x, y) \rightarrow \neg loyalto(x, y)$
8. Marcus tried to assassinate Caesar.
   $tryassassinate(Marcus, Caesar)$
9. All men are people.
   $\forall x : man(x) \rightarrow person(x)$

An Attempt to Prove $\neg\, loyalto(Marcus,\ Caesar)$

$$
\begin{array}{l}
\neg\, loyalto(Marcus,\ Caesar) \\
\qquad\qquad \uparrow \qquad\qquad (7,\ \text{substitution}) \\
person(Marcus) \wedge \\
ruler(Caesar) \wedge \\
tryassassinate(Marcus, Caesar) \\
\qquad\qquad \uparrow \qquad\qquad (4) \\
person(Marcus) \\
tryassassinate\{Marcus,\ Caesar) \\
\qquad\qquad \uparrow \qquad\qquad (8) \\
person(Marcus) \\
\qquad\qquad \uparrow \quad (9) \\
\quad man(Marcus) \\
\qquad\qquad \uparrow \quad (1) \\
\qquad\quad nil
\end{array}
$$

Three Ways of Representing Class Membership

1. $man(Marcus)$
2. $Pompeian(Marcus)$
3. $\forall x : Pompeian(x) \rightarrow Roman(x)$
4. $ruler(Caesar)$
5. $\forall x : Roman(x) \rightarrow loyalto(x, Caesar) \vee hate(x, Caesar)$

1. $instance(Marcus, man)$
2. $instance(Marcus, Pompeian)$
3. $\forall x : instance(x, Pompeian) \rightarrow instance(x, Roman)$
4. $instance(Caesar, ruler)$
5. $\forall x : instance(x, Roman) \rightarrow loyalto(x, Caesar) \vee hate(x, Caesar)$

1. $instance(Marcus, man)$
2. $\text{instance}(Marcus, Pompeian)$
3. $isa(Pompeian, Roman)$
4. $instance(Caesar, ruler)$
5. $\forall x : instance(x, Roman) \rightarrow loyalto(x, Caesar) \vee hate(x, Caesar)$
6. $\forall x : \forall y : \forall z : instance(x, y) \wedge \text{isa}(y, z) \rightarrow instance(x, z)$

Overriding Defaults

Suppose we add: Make Paulus an exception to the general rule about the Romans and their feeling towards Caesar.

$$Pompeian(Paulus)$$
$$\neg\ [loyalto(Paulus,\ Caesar) \vee hate(Paulus, Caesar)]$$

But now we have a problem with 5:

$$\forall x : Roman(x) \rightarrow loyalto(x, Caesar) \vee hate(x, Caesar)$$

So we need to change it to :

$$\forall x : Roman(x) \wedge \neg eq(x, Paulus) \rightarrow$$
$$loyalto(x, Caesar) \vee hate(x, Caesar)$$

Every exception to a general rule must be stated twice, once in a particular statement
And once in exception list that forms part of general rule

Another Predicate Logic Example

1. Marcus was a man.
2. Marcus was a Pompeian.
3. Marcus was born in 40 A.D.
4. All men are mortal.
5. All Pompeians died when the volcano erupted in 79 A.D.
6. No mortal lives longer than 150 years.
7. It is now 1991.
8. Alive means not dead.
9. If someone dies, then he is dead at all later times.

Is Marcus alive?

A Set of Facts about Marcus

1. $man(Marcus)$
2. $Pompeian(Marcus)$
3. $born(Marcus, 40)$
4. $\forall x : man(x) \longrightarrow mortal(x)$
5. $\forall : Pompeian(x) \longrightarrow died(x, 79)$
6. $erupted(yolcano, 79)$
7. $\forall_x : \forall t_1 : \forall t_2 : mortal(x) \wedge born(x, t_1) \wedge gt(t_2 - t_1, 150) \longrightarrow dead(x, t_2)$
8. $now = 1991$
9. $\forall x : \forall t$: $[\text{alive}(x, t) \longrightarrow \neg dead(x, t)] \wedge [\neg dead(x, t) \longrightarrow alive(x, t)]$
10. $\forall x : \forall t_1 : \forall t_2 : died(x, t_1) \wedge gt(t_2, t_1) \longrightarrow dead(x, t_2)$

Computable predicates

One Way of Proving That Marcus Is Dead

$$
\begin{array}{ll}
\neg alive(Marcus, now) & \\
\uparrow & (9, \text{substitution}) \\
dead(Marcus, now) & \\
\uparrow & (10, \text{substitution}) \\
died(Marcus, t_1) \wedge gt(now, t_1) & \\
\uparrow & (5, \text{substitution}) \\
Pompeian(Marcus) \wedge gt(now, 79) & \\
\uparrow & (2) \\
gt(now, 79) & \\
\uparrow & (8, \text{substitute equals}) \\
gt(1991, 79) & \\
\uparrow & (\text{compute gt}) \\
nil &
\end{array}
$$

The term nil at end of each proof indicate that the list of conditions remaining is empty
So the proof has succeeded.

Another Way of Proving That Marcus Is Dead

$$
\begin{array}{ll}
\neg alive(Marcus, now) & \\
\uparrow & (9,\ \text{substitution}) \\
dead(Marcus, now) & \\
\uparrow & (7,\ \text{substitution}) \\
mortal(Marcus)\ \wedge & \\
born(Marcus, t_1)\ \wedge & \\
gt(now - t_1, 150) & \\
\uparrow & (4,\ \text{substitution}) \\
man(Marcus)\ \wedge & \\
born(Marcus, t_1)\ \wedge & \\
gt(now - t_1, 150) & \\
\uparrow & (1) \\
born(Marcus, t_1)\ \wedge & \\
gt(now - t_1, 150) & \\
\uparrow & (3) \\
gt(now - 40, 150) & \\
\uparrow & (8) \\
gt(1991 - 40, 150) & \\
\uparrow & (\text{compute minus}) \\
gt(1951, 150) & \\
\uparrow & (\text{compute gt}) \\
nil &
\end{array}
$$

- **Very simple conclusion can require many steps to prove.**
- **A variety of processes such as Matching, substitution and application of modus ponens are involved in production of proof.**

Resolution

- Resolution produces proof by refutation. i.e. ***to prove a statement, resolution attempts to show that the negation of statement produces a contradiction with known statements.***
- This approach contrast with technique that we have been using to generate proofs by chaining backward from theorem to be proved axioms.
- It operates on statements that have been converted to a very convenient standard form.
- The formula would be easier to work with if
  - *It were flatter i.e. there was less embedding of components.*
  - *The quantifiers were separated from the rest of formula so they did not need to be consider.*
- Conjunctive Normal Form (CNF)has both these properties.

$$[p_{21} \vee p_{22} \vee \text{..............} p_{2n}] \wedge$$
$$[p_{11} \vee p_{12} \vee \text{..............} p_{1n}] \wedge$$
$$\text{- -}$$

Algorithm : Convert to Clause Form

1. Eliminate $\rightarrow$, using: $a \rightarrow b = \neg a \vee b$.
2. Reduce the scope of each $\neg$ to a single term, using:
   - $\neg(\neg p) = p$
   - deMorgan's laws: $\neg(a \wedge b) = \neg a \vee \neg b$
     $\neg(a \vee b) = \neg a \wedge \neg b$
   - $\neg \forall x : P(x) = \exists x : \neg P(x)$
   - $\neg \exists x : P(x) = \forall x : \neg P(x)$
3. Standardize variables.
4. Move all quantifiers to the left of the formula without changing their relative order.
5. Eliminate existential quantifiers by inserting Skolem functions.
6. Drop the prefix.
7. Convert the matrix into a conjunction of disjuncts, using associativity and distributivity.
8. Create a separate clause for each conjunct.
9. Standardize apart the variables in the set of clauses generated in step 8, using the fact that

   $$(\forall x : P(x) \wedge Q(x)) = \forall x : P(x) \wedge \forall x : Q(x)$$

Examples of Conversion to Clause Form

Given Axioms

P

$(P \wedge Q) \rightarrow R$

$(S \vee T) \rightarrow Q$

T

Examples of Conversion to Clause Form

Example : Suppose we know that all Romans who know Marcus either hate Caesar or think That anyone who hates anyone is crazy.

$$\forall x : [Roman(x) \wedge know(x, Marcus)] \rightarrow [hate(x, Caesar) \vee (\forall y : \exists z : hate(y, z) \rightarrow thinkcrazy(x, y))]$$

1 Eliminate $\rightarrow$

$$\vee x : \neg[Roman(x) \wedge know< x, Marcus)] \vee [hate(x, Caesar) \vee (\forall y : \neg(\exists z : hate(y, z)) \vee thinkcrazy(x,y))]$$

2 Reduce scope of $\neg$

$$\forall x : [\neg Roman(x) \vee \neg know(x, Marcus)] \vee [hate(x, Caesar) \vee (\forall y : \forall z : \neg hate(y, z) \vee thinkcrazy(x, y))]$$

3 Standardize Variables.

$\forall x : P(x) \vee \forall x : Q(x)$

would be converted to

$\forall x : P(x) \vee \forall y : Q(y)$

Examples of Conversion to Clause Form

4 Move quantifiers. Prenex normal form

$$\forall x : \forall y : \forall z : [\neg Roman(x) \vee \neg know(x\ Marcus)] \vee$$
$$[hate(x,\ Caesar) \vee (\neg hate(y,\ z) \vee thinkcrazy(x,y))]$$

5 Eliminate existential quantifiers.

$$\exists y : President(y)$$

will be converted to

$$President(S1)$$

while

$$\forall x : \exists y : father\text{-}of(y,x)$$

will be converted to

$$\forall x : father\text{-}of(S2(x),x))$$

6 Drop the prefix.

$$[\neg Roman(x) \vee \neg know(x,\ Marcus)] \vee$$
$$[hate(x,\ Caesar) \vee (\neg hate(y,\ z) \vee thinkcrazy\{x,\ y))]$$

7 Convert to a conjunction of disjuncts.

$$\neg Roman(x) \vee \neg know(x,\ Marcus) \vee$$
$$hate(x,\ Caesar) \vee \neg hate(y,\ z) \vee thinkcrazy(x,\ y)$$

Convert a matrix to conjunction of disjuncts.

The Formula

$$(winter \wedge wearingboots) \vee (summer \wedge wearingsandals)$$

becomes, after one application of the rule

$$[winter \vee (summer \wedge wearingsandals)]$$
$$\wedge [wearingboots \vee (summer \wedge wearingsandals)]$$

becomes

$$[winter \vee (summer \wedge wearingsandals)]$$
$$\wedge [wearingboots \vee (summer \wedge wearingsandals)]$$

and then becomes

$$(winter \vee summer) \wedge$$
$$(\text{winter} \vee wearingsandals) \wedge$$
$$(wearingboots \vee summer) \wedge$$
$$(wearingboots \vee wearingsandals)$$

Thank You!!!