Artificial Intelligence

CSE 4205/3201

First Order Predicate Logic(FOL) and Resolution

Inference Rules

- Inference rules allow the construction of new sentences from existing sentences
- An inference procedure generates new sentences on the basis of inference rules.
- An inference rule is **sound** if every sentence X it produces when operating on a KB logically follows from the KB
  - i.e., inference rule creates no contradictions
- An inference rule is **complete** if it can produce every expression that logically follows from (is entailed by) the KB.
  - Note analogy to complete search algorithms

Sound Rules of Inference

- Here are some examples of sound rules of inference
- Each can be shown to be sound using a truth table

RULE	**PREMISE**	**CONCLUSION**
Modus Ponens	$A, A \rightarrow B$	B
And Introduction	A, B	$A \wedge B$
And Elimination	$A \wedge B$	A
Double Negation	$\neg\neg A$	A
Unit Resolution	$A \vee B, \quad B$	A
Resolution	$\mathbf{A \vee B, \neg B \vee C}$	$\mathbf{A \vee C}$

Inference Rules

– **modus ponens**
  - from an implication and its premise one can infer the conclusion

$$\frac{\alpha \Rightarrow \beta,\ \ \alpha}{\beta}$$

– **and-elimination**
  - from a conjunct, one can infer any of the conjuncts

$$\frac{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}{\alpha_i}$$

– **and-introduction**
  - from a list of sentences, one can infer their conjunction

$$\frac{\alpha_1, \alpha_2, \ldots, \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}$$

– **or-introduction**
  - from a sentence, one can infer its disjunction with anything else

$$\frac{\alpha_1, \alpha_2, \ldots, \alpha_n}{\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n}$$

Inference Rules

- **double-negation elimination**
  - a double negations infers the positive sentence

$$\frac{\neg\neg\alpha}{\alpha}$$

  - if one of the disjuncts in a disjunction is false, then the other one must be true

$$\frac{\alpha \vee \beta, \qquad \neg\beta}{\alpha}$$

- **resolution**
  - β cannot be true and false, so one of the other disjuncts must be true

$$\frac{\alpha \vee \beta, \qquad \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

  - can also be restated as "implication is transitive

$$\frac{\neg\alpha \Rightarrow \beta, \beta \Rightarrow \gamma}{\neg\alpha \Rightarrow \gamma}$$

Modus Ponens

IF A Then B
A

Therefore B

IF it's raining out
Then Ann puts the top up on her convertible
It's raining out

Therefore Ann puts the top up on her convertible

IF it's raining out
Then Ann puts the top up on her convertible
Ann puts the top up on her convertible

Therefore ???

Modus Tollens

IF A Then B
Not(B)

Therefore Not (A)

IF it's raining out
Then Ann puts the top up on her convertible
Ann did not put the top up on her convertible

Therefore it's not raining out

IF it's raining out
Then Ann puts the top up on her convertible
It is not raining out

Therefore ???

Resolution

- **Resolution** is a valid inference rule producing a new clause implied by two clauses containing *complementary literals.*
- Amazingly, this is the only inference rule you need to build a sound and complete theorem prover.

  – Based on *proof by contradiction* and usually called *resolution refutation*.
- To use resolution, put knowledge base into *conjunctive normal form* (CNF), where each sentence written as a disjunction of (one or more) literals.

The Resolution Principle

- Here it is:
  - From $X \vee$ someLiterals
    and $\neg X \vee$ someOtherLiterals
    ----
    conclude: someLiterals $\vee$ someOtherLiterals
- That's all there is to it!
- Example:
  - broke(Bob) $\vee$ well-fed(Bob)
    ¬broke(Bob) $\vee$ ¬hungry(Bob)
    ----
    well-fed(Bob) $\vee$ ¬hungry(Bob)

A Common Error

- You can only do *one* resolution at a time
- Example:

  $broke(Bob) \vee well\text{-}fed(Bob) \vee happy(Bob)$
  $\neg broke(Bob) \vee \neg hungry(Bob) \vee \neg happy(Bob)$

- You can resolve on broke to get:

  $well\text{-}fed(Bob) \vee happy(Bob) \vee \neg hungry(Bob) \vee \neg happy(Bob) \equiv T$

- Or you can resolve on happy to get:

  $broke(Bob) \vee well\text{-}fed(Bob) \vee \neg broke(Bob) \vee \neg hungry(Bob) \equiv T$

- But you *cannot* resolve on both at once to get:

  $well\text{-}fed(Bob) \vee \neg hungry(Bob)$

Refutation Resolution

- The previous example was easy because it had very few clauses
- When we have a lot of clauses, we want to *focus* our search on the thing we would like to prove
- We can do this as follows:
  - Add the *negation* of the thing we want to prove to the fact base
  - Show that the fact base is now inconsistent
  - Conclude the thing we want to prove

Clause Form

- A **clause** is a **disjunction** (“or”) of literals
  - A literal is a an atomic symbol or its negation, i.e **P or ¬P**

- Example:
  sinks(X) ∨ dissolves(X, water) ∨ ¬denser(X, water)

- Notice that clauses use only “or” and “not”
  – they do not use “and,” “implies,” or either of the quantifiers “for all” or “there exists”

A First Example

- "Everywhere that John goes, Rover goes.
- John is at school."
  - $at(John, X) \Rightarrow at(Rover, X)$ (not yet in clause form)
  - $at(John, school)$ (already in clause form)

- We use implication elimination to change the first of these into clause form:
  - $\neg at(John, X) \vee at(Rover, X)$
  - $at(John, school)$

Example of Refutation Resolution

- **Prove that "Rover is at school."**
  1. ¬at(John, X) ∨ at(Rover, X)
  2. at(John, school)
  3. ¬at(Rover, school) (this is the added clause)
- Resolve #1 and #3:
  4. ¬at(John, X)
- Resolve #2 and #4:
  5. NIL
- Conclude the negation of the added clause: at(Rover, school)
- This seems a roundabout approach for such a simple example, but it works well for larger problems

The resolution is a simple iterative process:
– At each step,
• Two clauses (parent clauses) are compared (resolved), yielding a new clause that has been produced from them.

FOL Resolution Proof – Another Example

Prove: Mortal(Marcus) given:

$\forall x\ (Man(x) \rightarrow Mortal(x))$	$\neg Man(x) \vee Mortal(x)$
$Man(Marcus)$	$Man(Marcus)$

Add: $\neg Mortal(Marcus)$

$\neg Man(x) \vee Mortal(x)$ $\quad$ $\neg Mortal(Marcus)$

Marcus/x

$Man(Marcus)$ $\quad$ $\neg\ Man(Marcus)$

nil

Conversion To Clause Form

A nine-step process

Example

- All Romans who know Marcus either hate Caesar or think that anyone who hates anyone is crazy.

$$\forall x, [\ Roman(x) \wedge know(x, Marcus)\ ] \Rightarrow$$
$$[hate(x, Caesar) \vee$$
$$(\forall y, \exists z, hate(y, z) \Rightarrow thinkCrazy(x, y))]$$

Step 1: Eliminate Implication

- Eliminate $\rightarrow$.

  $$P \rightarrow Q \equiv \neg P \vee Q$$

- $\forall x$, [Roman(x) $\wedge$ know(x, Marcus)] $\Rightarrow$
  [hate(x, Caesar) $\vee$
  ($\forall y$, $\exists z$, hate(y, z) $\Rightarrow$ thinkCrazy(x, y))]

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

Step 2: Reduce The Scope of $\neg$

- Reduce the scope of negation ($\neg$) to a single term, using:
  - $\neg(\neg p) \equiv p$
  - $\neg(a \wedge b) \equiv (\neg a \vee \neg b)$
  - $\neg(a \vee b) \equiv (\neg a \wedge \neg b)$
  - $\neg\forall x,\ p(x) \quad \equiv \quad \exists x,\ \neg p(x)$
  - $\neg\exists x,\ p(x) \quad \equiv \quad \forall x,\ \neg p(x)$

- $\forall x,\ \neg[\ \text{Roman}(x) \wedge \text{know}(x, \text{Marcus})\ ] \vee$
  $[\text{hate}(x, \text{Caesar}) \ \vee$
  $(\forall y,\ \neg(\exists z,\ \text{hate}(y, z) \vee \text{thinkCrazy}(x, y))]$

- $\forall x,\ [\ \neg\text{Roman}(x) \vee \neg\text{know}(x, \text{Marcus})\ ] \vee$
  $[\text{hate}(x, \text{Caesar}) \ \vee$
  $(\forall y,\ \forall z,\ \neg\text{hate}(y, z) \vee \text{thinkCrazy}(x, y))]$

Step 3: Standardize Variables Apart

- Standardize variables so that each quantifier binds a unique variable.
- $\forall x, P(x) \vee \forall x, Q(x)$
  becomes
  $\forall x, P(x) \vee \forall y, Q(y)$
- This is just to keep the scopes of variables from getting confused
- Not necessary in our running example

Step 4: Move Quantifiers

- Move all quantifiers to the left without changing their relative order.

  $$(\forall x\colon P(x)) \vee (\exists y\colon Q(y)) \equiv \forall x\colon \exists y\colon (P(x) \vee (Q(y))$$

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

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

Step 5: Eliminate Existential Quantifiers

- Eliminate $\exists$ (Skolemization).

  $\exists x: P(x) \equiv P(c)$ Skolem constant

  $\forall x: \exists y\ P(x, y) \equiv \forall x: P(x, f(x))$ Skolem function

- We do this by introducing Skolemization:
  - If $\exists x, p(x)$ then just pick one; call it x'
  - If the existential quantifier is under control of a universal quantifier, then the picked value has to be a function of the universally quantified variable:
    - If $\forall x, \exists y, p(x, y)$ then $\forall x, p(x, y(x))$

- Not necessary in our running example

Step 6: Drop The Prefix (Quantifiers)

- Drop $\forall$.

  $\forall x: P(x) \equiv P(x)$

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

- At this point, all the quantifiers are universal quantifiers

- We can just take it for granted that all variables are universally quantified

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

Step 7: Convert to CNF/Create A Conjunction Of Disjuncts

- Convert the formula into a conjunction of disjuncts.

  $(P \wedge Q) \vee R \equiv (P \vee R) \wedge (Q \vee R)$

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

  becomes

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

Step 8: Create Separate Clauses

- Create a separate clause corresponding to each conjunct.
- Every place we have an $\wedge$, we break our expression up into separate pieces
- Not necessary in our running example

Step 9: Standardize Apart

- Rename variables so that no two clauses have the same variable
- Not necessary in our running example

  Final result:

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

- That's it! It's a long process, but easy enough to do mechanically

Conjunctive Normal Form(CNF)

- A literal is a variable or a negated variable.
- A clause is either a single literal or the disjunction of two or more literals.

  P, $P \vee \neg P$, and $P \vee \neg Q \vee R \vee S$ are clauses.

  $\neg(R \vee S)$ and $P \rightarrow \neg Q$ are not clauses.

- A wff(**Well Formed Formulas**) is in conjunctive normal form iff it is either a single clause or the conjunction of two or more clauses.

  $(P \vee \neg Q \vee R \vee S) \wedge (\neg P \vee \neg R)$ is in cnf

  $(P \wedge \neg Q \wedge R \wedge S) \vee (\neg P \wedge \neg R)$ is not in cnf

Conversion to Conjunctive Normal Form

Let w be the wff :

$$P \rightarrow \neg(R \vee \neg Q).$$

Then w can be converted to conjunctive normal form as follows:

Step 1 produces: $\neg P \vee \neg(R \vee \neg Q)$.

Step 2 produces: $\neg P \vee (\neg R \wedge Q)$.

Step 3 produces: $(\neg P \vee \neg R) \wedge (\neg P \vee Q)$.

Horn clause

- A **Horn sentence** or **Horn clause:**

  - A horn clause is a disjunction with at most one positive literal.

$$\sim X1 \lor \sim X2 \lor \ldots \lor \sim Xn \lor Y$$

e.g. $A \lor \neg B \lor \neg C$

Many wffs can be translated into Horn clauses:

$$(A \land B) \rightarrow C$$

$$= \sim(A \land B) \lor C$$

$$= \sim A \lor \sim B \lor C$$

Horn clause

- Every Horn clause can be rewritten as an implication with a conjunction of positive literals in the premises and at most a single positive literal as a conclusion.

e.g. $B \wedge C \Rightarrow A$

- 1 positive literal: definite clause.