UIT2504
Artificial Intelligence

Resolution in Propositional Logic

Logic in General

- Sentences written in logic must be well-formed formula and follow a grammar
- There are several possible interpretations for a set of sentences KB
- Interpretations in which KB evaluates to true are called as models of KB
- Given a new sentence α, KB logically entails α (written as $KB \models \alpha$) iff every model of KB is also a model of α
- We write $KB \vdash \alpha$ if α can be derived from KB using syntactic derivation rules
- Sentences in logic are usually written in a normal form
- There may be several strategies for effective application of the derivation rules

Derivations in Logic

- KB $\vdash \alpha$ if α can be **derived** from KB using syntactic **inference rules**

$$x + y = 4$$

$$x + y - y = 4 - y$$

$$x = 4 - y$$

- Inference procedure is **sound** if every α derivable is entailed by KB
- Inference procedure is **complete** if every α that is entailed by KB can be derived from KB
- When we have sound and complete inference procedure, KB $\overset{?}{\models} \alpha$ can be reduced to KB $\overset{?}{\vdash} \alpha$

Derivations

Given a set of sentences KB, new sentences can be derived using inference rules

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

Derivations

Given a set of sentences KB, new sentences can be derived using inference rules

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

Already available

Derivations

Given a set of sentences KB, new sentences can be derived using inference rules

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

Already available

Can be derived

Conjunctive Normal Form

$$
\begin{aligned}
CNFSentence &\rightarrow Clause_1 \wedge \cdots \wedge Clause_n \\
Clause &\rightarrow Literal_1 \vee \cdots \vee Literal_m \\
Fact &\rightarrow Symbol \\
Literal &\rightarrow Symbol \mid \neg Symbol \\
Symbol &\rightarrow P \mid Q \mid R \mid \ldots \\
HornClauseForm &\rightarrow DefiniteClauseForm \mid GoalClauseForm \\
DefiniteClauseForm &\rightarrow Fact \mid (Symbol_1 \wedge \cdots \wedge Symbol_l) \Rightarrow Symbol \\
GoalClauseForm &\rightarrow (Symbol_1 \wedge \cdots \wedge Symbol_l) \Rightarrow False
\end{aligned}
$$

Horn and Definite Clauses

- CNF(KB) is conjunction of clauses
- Clause is a disjunction of literals
- Literal is an atom or negation of an atom
- Definite clause is a clause with exactly one positive literal
- Horn clause is a clause with at most one positive literal
- Clauses with no positive literals are called as goal clauses

Resolution

$$\frac{l_1 \vee \ldots \vee l_k \qquad\qquad m_1 \vee \ldots \vee m_n}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

where l_i and m_j are complementary literals.

Resolution

$$\frac{l_1 \vee \ldots \vee l_k \qquad\qquad m_1 \vee \ldots \vee m_n}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

where l_i and m_j are complementary literals.

Example:

$$\frac{\neg B_{1,1} \vee P_{1,2} \vee P_{2,1} \qquad\qquad \neg P_{2,1} \vee B_{1,1}}{P_{1,2} \vee P_{2,1} \vee \neg P_{2,1}}$$

Resolution

$$\frac{\ell_1 \vee \ldots \vee \ell_k \qquad\qquad m_1 \vee \ldots \vee m_n}{\ell_1 \vee \ldots \vee \ell_{i-1} \vee \ell_{i+1} \vee \ldots \vee \ell_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

where l_i and m_j are complementary literals.

Example:

$$\frac{\neg B_{1,1} \vee P_{1,2} \vee P_{2,1} \qquad\qquad \neg P_{2,1} \vee B_{1,1}}{P_{1,2} \vee P_{2,1} \vee \neg P_{2,1}}$$

Two **input clauses** are **resolved** with each other to give the **resolvent**

Factoring

The resolvent should have only one copy of each literal

$$\frac{A \vee B \qquad\qquad \neg B \vee A}{A \vee A}$$

Factoring

The resolvent should have only one copy of each literal

$$\frac{A \vee B \qquad \neg B \vee A}{A \vee A}$$

$$\frac{A \vee B \qquad \neg B \vee A}{A}$$

Unit Resolution

$$\frac{\ell_1 \vee \ldots \vee \ell_k \qquad\qquad\qquad\qquad m}{\ell_1 \vee \ldots \vee \ell_{i-1} \vee \ell_{i+1} \vee \ldots \vee \ell_k}$$

where l_i and m are complementary literals.

Unit Resolution

$$\frac{\ell_1 \vee \ldots \vee \ell_k \qquad\qquad\qquad m}{\ell_1 \vee \ldots \vee \ell_{i-1} \vee \ell_{i+1} \vee \ldots \vee \ell_k}$$

where l_i and m are complementary literals.

Example:

$$\frac{P_{1,3} \vee P_{2,2} \qquad\qquad \neg P_{2,2}}{P_{1,3}}$$

Unit Resolution

$$\frac{\ell_1 \vee \ldots \vee \ell_k \qquad\qquad m}{\ell_1 \vee \ldots \vee \ell_{i-1} \vee \ell_{i+1} \vee \ldots \vee \ell_k}$$

where l_i and m are complementary literals.

Example:

$$\frac{P_{1,3} \vee P_{2,2} \qquad\qquad \neg P_{2,2}}{P_{1,3}}$$

Special Case:

$$\frac{P \qquad\qquad \neg P}{\square}$$

Unit Resolution

$$\frac{\ell_1 \vee \ldots \vee \ell_k \qquad\qquad m}{\ell_1 \vee \ldots \vee \ell_{i-1} \vee \ell_{i+1} \vee \ldots \vee \ell_k}$$

where l_i and m are complementary literals.

Example:

$$\frac{P_{1,3} \vee P_{2,2} \qquad\qquad \neg P_{2,2}}{P_{1,3}}$$

Special Case:

$$\frac{P \qquad\qquad \neg P}{\square}$$

Empty clause represents *Contradiction*

Resolution Procedure

- To show that KB entails α, show that $(KB \wedge \neg\alpha)$ is unsatisfiable (proof by contradiction; from deduction theorem)
- Use resolution to check if empty clause is derivable from CNF$(KB \wedge \neg\alpha)$
- Effective strategies may be required to design efficient procedure that is focused on the goal

Resolution Procedure

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

$clauses \leftarrow$ the set of clauses in the CNF representation of $KB \land \neg\alpha$
$new \leftarrow \{\}$
while *true* **do**
for each pair of clauses C_i, C_j **in** *clauses* **do**
$resolvents \leftarrow$ PL-RESOLVE(C_i, C_j)
if *resolvents* contains the empty clause **then return** *true*
$new \leftarrow new \cup resolvents$
if $new \subseteq clauses$ **then return** *false*
$clauses \leftarrow clauses \cup new$

Resolution Example

$KB = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$

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

Resolution Example

$KB = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$

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

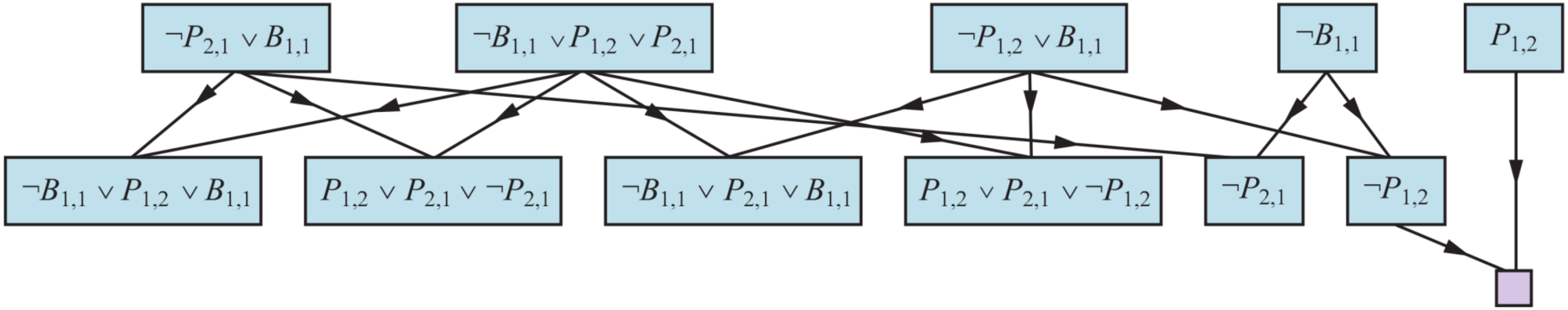
$CNF(KB \wedge \neg\alpha) = P_{1,2} \wedge \neg B_{1,1} \wedge (B_{1,1} \vee \neg P_{2,1})$
$(B_{1,1} \vee \neg P_{1,2}) \wedge (\neg B_{1,1} \vee P_{2,1} \vee P_{1,2})$

Resolution Example

$KB = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$

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

$CNF(KB \wedge \neg\alpha) = P_{1,2} \wedge \neg B_{1,1} \wedge (B_{1,1} \vee \neg P_{2,1})$
$(B_{1,1} \vee \neg P_{1,2}) \wedge (\neg B_{1,1} \vee P_{2,1} \vee P_{1,2})$



Resolution Example

$$KB = (B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1}$$

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

$$CNF(KB \land \neg\alpha) = P_{1,2} \land \neg B_{1,1} \land (B_{1,1} \lor \neg P_{2,1})$$

$$(B_{1,1} \lor \neg P_{1,2}) \land (\neg B_{1,1} \lor P_{2,1} \lor P_{1,2})$$

$$P_{1,2}$$

Resolution Example

$KB = (B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1}$

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

$CNF(KB \land \neg\alpha) = P_{1,2} \land \neg B_{1,1} \land (B_{1,1} \lor \neg P_{2,1})$
$(B_{1,1} \lor \neg P_{1,2}) \land (\neg B_{1,1} \lor P_{2,1} \lor P_{1,2})$

$P_{1,2}$

$B_{1,1} \lor \neg P_{1,2}$

$B_{1,1}$

Resolution Example

$$KB = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$$

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

$$CNF(KB \wedge \neg\alpha) = P_{1,2} \wedge \neg B_{1,1} \wedge (B_{1,1} \vee \neg P_{2,1}) \wedge (B_{1,1} \vee \neg P_{1,2}) \wedge (\neg B_{1,1} \vee P_{2,1} \vee P_{1,2})$$

$P_{1,2}$

$B_{1,1} \vee \neg P_{1,2}$

$B_{1,1}$

$\neg B_{1,1}$

$\square$

Forward Chaining (Linear Time for Horn Clauses)

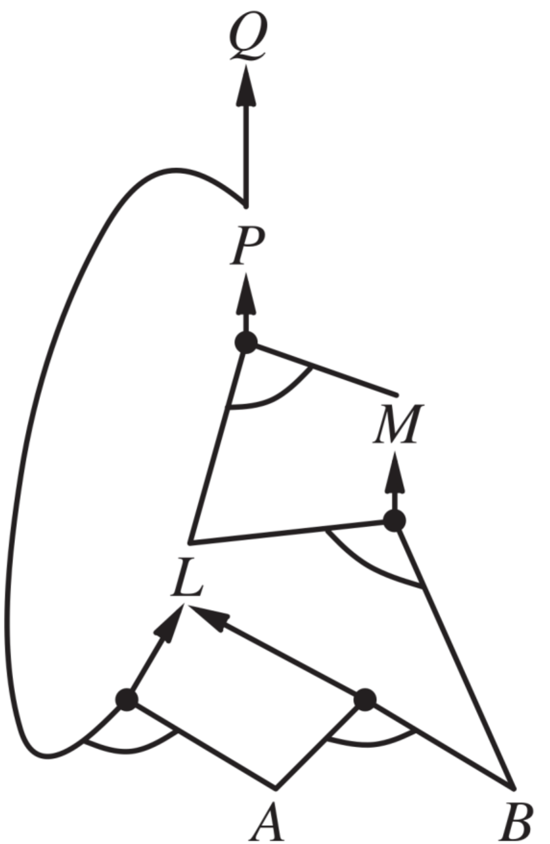
```
function PL-FC-ENTAILS?(KB, q) returns true or false
  inputs: KB, the knowledge base, a set of propositional definite clauses
          q, the query, a proposition symbol
  count ← a table, where count[c] is initially the number of symbols in clause c's premise
  inferred ← a table, where inferred[s] is initially false for all symbols
  queue ← a queue of symbols, initially symbols known to be true in KB

  while queue is not empty do
    p ← POP(queue)
    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 queue
  return false
```

Forward Chaining Example

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

(a)



(b)

Resolution Examples

- KB = {P V Q, ¬Q V R, ¬P V S, ¬S}

  α = R

- KB =

  a :- b, c.

  b :- d, e.

  b :- g, e.

  c :- e.

  f :- a, g.

  d.

  e.

  α = a

Effective Resolution

- Forward Chaining
- Backward Chaining
- Resolution Strategies
- Very effective for Horn clauses (runs in linear time!)
- Prolog is based on resolution

Soundness of Resolution

$$\frac{l_1 \vee \ldots \vee l_k \qquad\qquad m_1 \vee \ldots \vee m_n}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

where l_i and m_j are complementary literals.

Soundness of Resolution

$$\frac{l_1 \vee \ldots \vee l_k \qquad\qquad m_1 \vee \ldots \vee m_n}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

where l_i and m_j are complementary literals.

- l_i may be true or false
- If l_i is true, then m_j must be false. Then,

  $m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n$ must be true.
- If l_i is false, then

  $l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k$ must be true.
- Thus, we see that

$l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n$ must be true.

Completeness of Resolution

- The procedure PL-RESOLUTION terminates
- Returns the finite resolution closure RC(S) of a set of clauses S
- Let S uses propositions $P_1, \ldots, P_k$

Completeness of Resolution

- The procedure PL-RESOLUTION terminates
- Returns the finite resolution closure RC(S) of a set of clauses S
- Let S uses propositions $P_1, \ldots, P_k$

Ground Resolution Theorem:

If a set of clauses S is unsatisfiable, then the resolution closure RC(S) contains the empty clause.

Completeness of Resolution

- We will prove the contrapositive of this theorem: If the closure RC(S) does not contain the empty clause, then S is satisfiable.
- A model for S may be constructed as follows:

For *i* from 1 to *k*:

- If a clause in RC(S) contains the literal $\neg P_i$ and all its other literals are false under the assignment chosen for $P_1, \ldots, P_{i-1}$, then assign False to P_i
- Otherwise, assign True to P_i

Completeness of Resolution

- The assignment done thus for P_1, … , P_k must be a model of S
- Assume the contrary: at some stage *i*, assignment to P_i causes some clause C to become false

Completeness of Resolution

- The assignment done thus for $P_1, \ldots, P_k$ must be a model of S
- Assume the contrary: at some stage *i*, assignment to P_i causes some clause C to become false
- This is possible only if C is of the form (false $\vee$ false $\vee$ . . . false $\vee$ P_i) or (false $\vee$ false $\vee$ . . . false $\vee$ $\neg P_i$). If only one of them is present, then our assignment of truth value to P_i is correct. If both are present, then their resolvent must also be present which should have already been falsified!

Questions?