Propositional Resolution[1]

Lecture 3

[1] The slides have been prepared using the textbook material available on the web, and the slides of the previous editions of the course by Prof. Luigia Carlucci Aiello

Summary

- resolution [RN, 7.5 + some additional material]

Resolution

- One inference rule
- Requires **normal** form

Clauses

♢ *Literals*: propositional symbols (*atoms*) or negated atoms.

♢ A *formula* is in *negation normal form (NNF)* if negations appear only in front of atoms.

♢ A *clause* is a disjuntion of literals $L_1 \vee L_2 \vee \cdots \vee L_n$.

♢ A formula is in *conjunctive normal form (CNF)* or in *clausal form* or it is called a *set of clauses* if:
it is in negation normal form and
it is $C_1 \wedge C_2 \wedge \cdots \wedge C_n$ where C_i are clauses.

Clauses (cont.)

♢ Since disjunction is commutative:

$$A_1 \vee A_2 \vee \cdots \vee A_n \vee \neg B_1 \vee \neg B_2 \vee \cdots \vee \neg B_m$$

where A_i and B_j are atoms.

♢ When $n = m = 0$ we have the *empty clause*, denoted $\{\}$, or $\Box$.

♢ Clause: $\{L_1, L_2, \ldots, L_p\}$.

♢ Let L be a literal and $C = \{L_1, L_2, \ldots, L_p\}$ be a clause $L \cup C$ denotes $C' = \{L\} \cup C$, i.e. $C' = \{L, L_1, L_2, \ldots, L_p\}$.

Horn Clauses

♢ A clause is **Horn** if $n \leq 1$:

example: $P \vee \neg Q \vee \neg R$

corresponding to $Q \wedge R \Rightarrow P$

♢ A CNF is Horn if all its clauses are Horn

♢ If $n = 1$:

$$A_1 \vee \neg B_1 \vee \ldots, \vee \neg B_m$$

we have a *definite (Horn) clause*.

Transformation in CNF

Apply the following steps:

1. biconditional elimination (using definition)
2. eliminate implication (with disjunction)
3. push negation inside
4. distribute conjunction over disjunction

Alternative way for steps 2-4

Conjunctive formulae α and disjunctive formulae β :

α	$\alpha_1 \quad \alpha_2$	β	$\beta_1 \quad \beta_2$
$A \wedge B$	$A \quad B$	$A \vee B$	$A \quad B$
$\neg(A \vee B)$	$\neg A \quad \neg B$	$\neg(A \wedge B)$	$\neg A \quad \neg B$
$\neg(A \Rightarrow B)$	$A \quad \neg B$	$A \Rightarrow B$	$\neg A \quad B$

Transformation in clausal form: algorithm

let F be a propositional formula:

Step 1 let $\{F\}$ be the initial set.

Step n+1 Let the result of step n be $\{D_1, \ldots, D_n\}$, where D_i is a disjunction $\{A_1^i, \ldots, A_k^i\}$; if A_j^i is not a literal, we do not have yet a CNF and we make one of the transformations shown next.

Transformation in clausal form: algorithm (cont.)

Choose a D_i which contains a non literal X:

a. if X is $\neg\top$ replace it with $\bot$;

b. if X is $\neg\bot$ replace it with $\top$;

c. if X is $\neg\neg A$ replace it with A;

d. if X is a β formula replace it with β_1, β_2;

e. if X is a α formula replace D_i with two clauses:
   D_i^1 that is D_i with α replaced by α_1
   D_i^2 that is D_i with α replaced by α_2.

Expansion rules

◇ Steps a – e and can be written as expansion rules:

$$1. \quad \frac{\neg\neg A}{A} \quad 2. \quad \frac{\neg\top}{\bot} \quad 3. \quad \frac{\neg\bot}{\top} \quad 4. \quad \frac{\beta}{\beta_1, \beta_2} \quad 5. \quad \frac{\alpha}{\alpha_1 \mid \alpha_2}$$

Properties

Lemma:

Let D be a disjunction $\{A_1, \ldots, A_k\}$. If D' is obtained from D by applying rules a – e of the algorithm:

$$D \equiv D'$$

.

Example

1. $F = (P \Rightarrow (Q \Rightarrow (S \vee T))) \Rightarrow (T \Rightarrow Q)$
2. $C = \{\neg(P \Rightarrow (Q \Rightarrow (S \vee T))), (T \Rightarrow Q)\}$
   (β-rule to F)
3. $C_1 = \{P, (T \Rightarrow Q)\}$, $C_2 = \{\neg(Q \Rightarrow (S \vee T)), (T \Rightarrow Q)\}$
   (α-rule to C)
4. $C_1 = \{P, \neg T, Q\}$, $C_2 = \{\neg(Q \Rightarrow (S \vee T)), (T \Rightarrow Q)\}$
   (β-rule to C_1)

Example (cont.)

5 $C_1 = \{P, \neg T, Q\}$, $C_2 = \{\neg(Q \Rightarrow (S \vee T)), \neg T, Q\}$
(β-rule to C_2)

6 $C_1 = \{P, \neg T, Q)\}$, $C_2 = \{Q, \neg T, Q\}$, $C_3 = \{\neg(S \vee T), \neg T, Q\}$
(α-rule to C_2)

7 $C_1 = \{P, \neg T, Q\}$, $C_2 = \{Q, \neg T, Q\}$, $C_3 = \{\neg S, \neg T, Q\}$, $C_4 = \{\neg T, \neg T, Q\}$
(α-rule to C_3)

Propositional resolution

♢ Let $\{L\} \cup D_1$ and $\{\neg L\} \cup D_2$ be two clauses. $D_1 \cup D_2$ is obtained from $\{L\} \cup D_1$ and $\{\neg L\} \cup D_2$ by a *resolution step* written:

$$\frac{\{L\} \cup D_1 \quad \{\neg L\} \cup D_2}{D_1 \cup D_2}$$

♢ A *resolution tree* is a binary tree whose nodes are labelled by clauses. Let C_1 and C_2 be 2 brother nodes, whose father is C, then $C_1 = D_1 \cup \{L\}$, $C_2 = D_2 \cup \{\neg L\}$ and $C = D_1 \cup D_2$. The label associated with the father is the result of a resolution step applied to the sons.

Propositional resolution (cont.)

$\diamondsuit$ Let KB be a finite set of clauses, and C a clause. C can be derived by *resolution* from KB iff there exists a resolution tree whose root is labelled by C and all the leaves are clauses in KB.

$KB \vdash_R C$ denotes that KB derives C by resolution.

Example

Let $KB = \{\{P\}, \{Q\}, \{\neg P, \neg Q\}\}$:

resolution tree for $KB \vdash_R \{\}$.

$$\dfrac{P \qquad \dfrac{Q \quad \neg P \vee \neg Q}{\neg P}}{\{\}}$$

Resolution and satisfiability

1. let Γ be a set of clauses, $\{L\} \cup D_1 \in \Gamma$ and $\{\neg L\} \cup D_2 \in \Gamma$. If Γ is satisfiable then $\Gamma \cup \{D_1 \cup D_2\}$ is satisfiable;
2. let Γ be a set of clauses, if $\Gamma \vdash_R \{\}$ then Γ is unsatisfiable.

We prove 1. Suppose Γ be satisfiable, i.e. there exists a model $\mathcal{M}$ such that Γ is true in $\mathcal{M}$ and suppose that L is true in $\mathcal{M}$; it follows that $\neg L$ is false in $\mathcal{M}$ and therefore D_2 must be true in $\mathcal{M}$. Consequently, $D_1 \cup D_2$ is true in $\mathcal{M}$. Since $\mathcal{M}$ is a model of Γ by hypothesis, $\Gamma \cup \{D_1 \cup D_2\}$ is true in $\mathcal{M}$.

$\mathcal{M} \models \neg L$ is analogous.

Resolution and satisfiability (cont.)

$\{\}$ denotes contradiction; the empty clause is unsatisfiable.

Since resolution preserves satisfiability, if the empty clause is generated then the initial set of clauses is unsatisfiable.

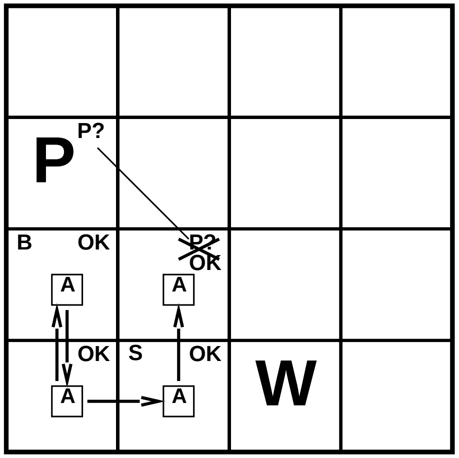
Resolution proofs work by **refutation**: $KB \vdash_R F$ is proven by checking whether $KB \cup \{\neg F\} \vdash_R \{\}$, where KB is a set of clauses and $\{\neg F\}$ is the negation of F in clausal form.

Example

We prove that $A \Rightarrow C$ follows from $A \Rightarrow B$ and $B \Rightarrow C$:

1. $(A \Rightarrow B) \wedge (B \Rightarrow C) \wedge \neg(A \Rightarrow C)$
2. $\{\neg A \vee B\}, \{\neg B \vee C\}, \{A\}, \{\neg C\}$ *(transf. in clausal form)*
3. $\{\neg A \vee C\}, \{A\}, \{\neg C\}$ *(res. B and $\neg B$ in 1^{st} and 2^{nd} claus. of 2)*
4. $\{C\}, \{\neg C\}$ *(res. $\neg A$ and A in 1^{st} and 2^{nd} claus. of 3)*
5. $\{\}$ *(res. C and $\neg C$ in 1^{st} and 2^{nd} claus. of 4)*.

Resolution for the wumpus world



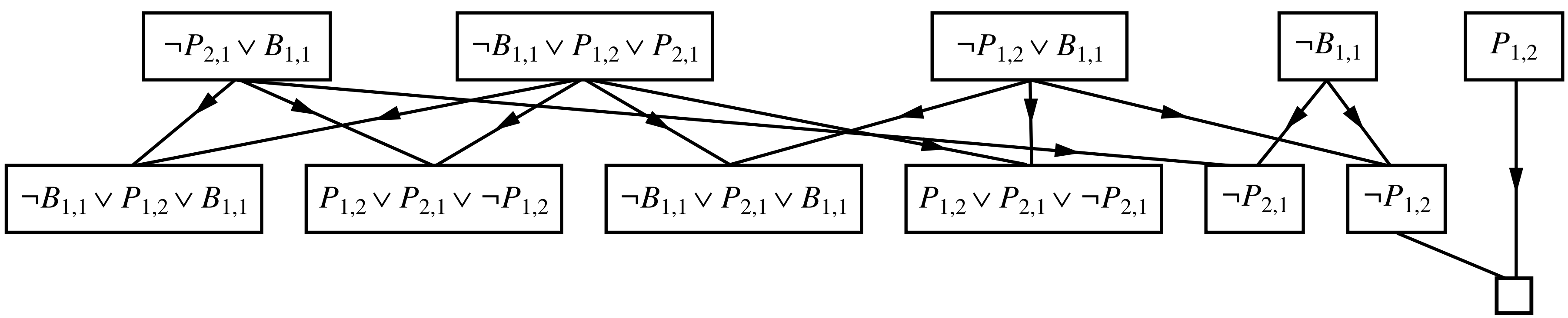
$$KB = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$$
$$\alpha = \neg P_{1,2}$$

$$KB \wedge \neg\alpha \Leftrightarrow$$
$$\{\{\neg B_{1,1}, P_{2,1}, P_{1,2}\}, \{B_{1,1}, \neg P_{2,1}\}, \{B_{1,1}, \neg P_{1,2}\}, \{\neg B_{1,1}\}, \{P_{1,2}\}\}$$

Resolution example

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

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



Resolution algorithm

Proof by contradiction, i.e., show $KB \wedge \neg\alpha$ unsatisfiable

function PL-Resolution(KB, α) **returns** *true* or *false*

  $clauses \leftarrow$ set of clauses in the CNF repres. of $KB \wedge \neg\alpha$
  $new \leftarrow \{\}$
  loop do
    for each C_i, C_j **in** $clauses$ **do**
      $resolvents \leftarrow$ PL-Resolve(C_i, C_j)
      if $\{\} \in resolvents$ **then return** *true*
      $new \leftarrow new \cup resolvents$
    if $new \subseteq clauses$ **then return** *false*
    $clauses \leftarrow clauses \cup new$

Resolution is satisfiability-complete

Let Γ be a set of clauses. Γ is unsatisfiable iff $\Gamma \vdash_R \{\}$.

Let $RC(KB)$ denote the **resolution closure** (i.e. the finite set of clauses that can be derived from KB using the resolution rule):

Γ is unsatisfiable iff $\{\} \in RC(\Gamma)$

Resolution is satisfiability-complete: proof

Proof by contrapposition: If $\{\} \notin RC(\Gamma)$ then Γ is satisfiable. Build a model from $RC(\Gamma)$ by assigning truth values to each $P_i \in \Gamma$:

- If a clause in $RC(\Gamma)$ contains the literal $\neg P_i$ and all its other literals are $false$ (given previous assignments), then assign $false$ to P_i.
- Otherwise assign $true$ to P_i

This assignment can falsify a clause only when in $RC(KB)$ there are two clauses: $false, false, \ldots, P_i$ and $false, false, \ldots, \neg P_i$; but this is impossible because $RC(\Gamma)$ is closed under resolution.

Resolution is sound and complete for propositional logic

... but not validity-complete

Let $KB = \{\{P\}\}$ and let $C = \{P, Q\}$;

KB logically entails C, and it can be derived by refutation,

but $KB \vdash_R C$ can not be direcly derived by applying a resolution step.

Summary

Resolution is a proof procedure that is sound and complete for satisfiability.

Resolution is the basis for reasoning systems, in particular for FOL.