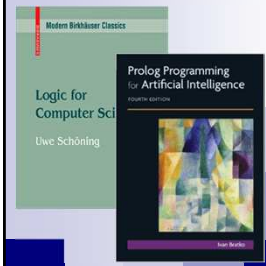


FLORIDA INTERNATIONAL UNIVERSITY

Propositional Logic (IV)

Dr. Antonio L. Bajuelos

FIU School of Computing & Information Sciences

Note: The most of the information of these slides was extracted and adapted from Schoning's book, "Logic for Computer Scientists". They are provided for COT-3541 students only. Not to be published or publicly distributed without permission by the publisher.

SAT – problem

Remember that …

- The **CNF Satisfiability Problem** (**SAT**) is:
  - Given a propositional **CNF** formula **F**;
  - Question: Is **F satisfiable**?
- **Example:**
  - Given $F = (A \vee B) \wedge (\neg B \vee C \vee \neg D) \wedge (\neg A \vee D)$. F is satisfiable?
  - Answer: $\mathcal{A}(F) = 1$ for this assignment: $\mathcal{A}(A) = 1$, $\mathcal{A}(B) = 0$, $\mathcal{A}(C) = 0$, and $\mathcal{A}(D) = 1$.
  - Then **F is satisfiable**.

2

Resolution

- **Resolution** is a simple syntactic transformation applied to formulas.
- **Process:**
  - From **two** given **formulas** in a **resolution step** (if provided resolution is applicable to the formulas), a **third formula is generated**.
  - This new formula can then be used in further resolution steps, and so on.
- A collection of such "mechanical" transformation rules we call a **calculus**.

In the case of the **resolution calculus**, the task is to prove **unsatisfiability** of a given formula.

Calculus. Correctness and Completeness

- The definition of a **calculus** is sensible only if its **correctness** and its **completeness** can be established.
- **Correctness** (**soundness**) means that every formula for which the resolution **calculus** claims **unsatisfiability** indeed is **unsatisfiable**.
- **Completeness** means that for every **unsatisfiable** formula there is a way to prove this by means of the resolution **calculus**.

Resolution

- A general precondition for the application of **resolution** to a formula is that the formula (or set of formulas) is in **CNF**. That is, if necessary, the formula has to be transformed into an equivalent **CNF formula**

$$F = (L_{1,1} \vee \cdots \vee L_{1,n_1}) \wedge \cdots \wedge (L_{k,1} \vee \cdots \vee L_{k,n_k})$$

where the $L_{i,j}$ are literals, i.e.

$$L_{i,j} \in \{A_1, A_2, \cdots\} \cup \{\neg A_1, \neg A_2, \cdots\}$$

- For the presentation of resolution it is advantageous to represent formulas in **CNF** as sets of so-called **clauses** where a clause is a set of literals:

$$F = \{\{L_{1,1}, \ldots, L_{1,n_1}\}, \ldots, \{L_{k,1}, \ldots, L_{k,n_k}\}\}$$

- **Example:**
  - Given F = (A ∨ B) ∧ (¬B ∨ C ∨ ¬D) ∧ (¬A ∨ D).

    F = {{A,B}, {¬B,C,¬D}, {¬A,D}}

Resolvent

<u>Definition</u> (resolvent)

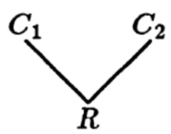
- Let $\mathbf{C_1}$, $\mathbf{C_2}$ and **R** be **clauses.** Then **R** is called a **resolvent of $\mathbf{C_1}$ and $\mathbf{C_2}$** if there is a literal $L \in C_1$ such that $\overline{L} \in C_2$ and R has the form:

$$R = (C_1 - \{L\}) \cup (C_2 - \{\overline{L}\}).$$

here $\overline{L}$ is defined as:

$$\overline{L} = \begin{cases} \neg A_i & \text{if } L = A_i\ , \\ A_i & \text{if } L = \neg A_i \end{cases}$$

- Graphically we denote this situation by the following diagram:



- **Example:**
  - Given F = (A ∨ B) ∧ (¬B ∨ C ∨ ¬D) ∧ (¬A ∨ D).

    F = {{A,B}, {¬B,C,¬D}, {¬A,D}}

  C_1 = {A,B}, C_2= {¬B,C,¬D} then R = {A,C,¬D}

Resolvent

- The definition of **resolvent** also includes the case that R is the **empty set** (if C_1,=$\{L\}$ and C_1,=$\{\bar{L}\}$ for some literal L) This **empty clause** is denoted by the special symbol $\square$.
- Please note that, the **empty clause,** $\square$, is **unsatisfiable**. Therefore, a clause set which contains $\square$ as an element is **unsatisfiable**.
- **Why?**
  - Let's see how the resolution rule is applied to $C_1 = \{S\}$ and $C_2 = \{\neg S\}$.
  - It is easy to see that $\mathbf{S \equiv False \vee S}$ and $\mathbf{\neg S \equiv False \vee \neg S}$
  - So $C_1 = \{False, S\}$ and $C_2 = \{False, \neg S\}$ then resolvent of C_1 and C_2 is $C_{1,2} = \{False\}$.
  - F= $\{C_1, C_2, \ldots, C_n,$ **False**$\}$ is **False** i.e is **unsatisfiable**.

Resolvent

Examples:

- The following are some examples for **resolvents**:

$$\{A_3, \neg A_4, A_1\} \quad \{A_4, \neg A_1\}$$
$$\{A_3, A_1, \neg A_1\}$$

$$\{A_3, \neg A_4, A_1\} \quad \{A_4, \neg A_1\}$$
$$\{A_3, \neg A_4, A_4\}$$

$$\{A_{17}\} \quad \{\neg A_{17}\}$$
$$\square$$

- **Exercise:** Give the entire list of **resolvents** which can be generated from the set of clauses:

$$\{\{A, E, \neg B\}, \{\neg A, B, C\}, \{\neg A, \neg D, \neg E\}, \{A, D\}\}$$

Resolvent

Very Important!

- Note that in resolving two clauses, **only one pair of literals may be resolved at a time**, even though there are multiple resolvable pairs.
- For example, the following is **not a legal application** of resolution. $C_1 = \{p,q\}$ and $C_2 = \{\neg p, \neg q\}$ and the **resolvent** is the empty clause, □ **Wrong!**
- For $C_1 = \{p,q\}$ and $C_2 = \{\neg p, \neg q\}$ the legal **resolvents** are $C_3 = \{p, \neg p\}$ and $C_4 = \{q, \neg q\}$

Resolution

<u>Lemma</u> (resolution lemma)
Let F be a **CNF** formula, represented as a set of clauses. Let R be a **resolvent** of two clauses C_1 and C_2 in F. Then **F and F ∪ {R} are equivalent**.

Proof: Let $\mathcal{A}$ be an assignment that is suitable for F (and also for $F \cup \{R\}$). If $\mathcal{A} \models F \cup \{R\}$ then immediately, $\mathcal{A} \models F$. Conversely, suppose $\mathcal{A} \models F$, that is, for all clauses $C \in F$, $\mathcal{A} \models C$. Assume the resolvent R has the form $R = (C_1 - \{L\}) \cup (C_2 - \{\overline{L}\})$ where $C_1, C_2 \in F$ and $L \in C_1, \overline{L} \in C_2$.

Case 1: $\mathcal{A} \models L$.

Then, by $\mathcal{A} \models C_2$ and $\mathcal{A} \not\models \overline{L}$, it follows $\mathcal{A} \models (C_2 - \{\overline{L}\})$, and therefore $\mathcal{A} \models R$.

Case 2: $\mathcal{A} \not\models L$.

Then, by $\mathcal{A} \models C_1$, it follows $\mathcal{A} \models (C_1 - \{L\})$, and therefore $\mathcal{A} \models R$. ■

Resolution

Definition (*Res**(*F*))

Let F be a set of clauses. Then $Res(F)$ is defined as

$$Res(F) = F \cup \{R \mid R \text{ is a resolvent of two clauses in } F\}.$$

Furthermore, define

$$\begin{aligned} Res^0(F) &= F \\ Res^{n+1}(F) &= Res(Res^n(F)) \quad \text{for } n \geq 0 . \end{aligned}$$

and finally, let

$$Res^*(F) = \bigcup_{n \geq 0} Res^n(F) .$$

- **Exercise:** For the following set of clauses,

  $$F = \{\{A, \neg B, C\}, \{B, C\}, \{\neg A, C\}, \{B, \neg C\}, \{\neg C\}\}$$

  determine $Res^n(F)$ for n =0, 1, and 2.

Resolution Theorem

Theorem (Resolution Theorem)

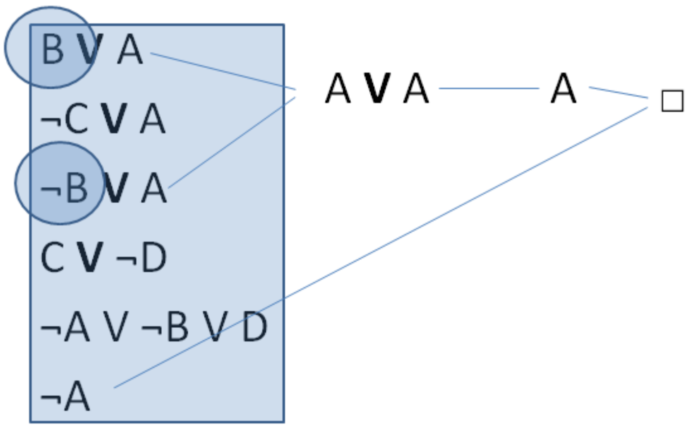
- A clause set F is **unsatisfiable** if and only if $\square \in Res^*(F)$.

(see Prof. in the Schoning's book, Chapter 1)

- Note that from the **Resolution Theorem** we can infer the **correctness** and **completeness** of the **resolution calculus** (with respect to **unsatisfiability**).

Resolution Theorem

Example:



Resolution Theorem

Example:

Facts (1-6)	**Resolvents**	
1) B **V** A	7) ¬C	From 2,6
2) ¬C **V** A	8) ¬D	From 4,7
3) ¬B **V** A	9) ¬A **V** ¬B	From 5,8
4) C **V** ¬D	10) B	From 1,6
5) ¬A **V** ¬B **V** D	11) ¬A	From 9,10
6) ¬A	12) ¬B	From 3,11

□

Algorithm (from Resolution Theorem)

- Algorithm that decides **satisfiability** for a given input formula in CNF form

  Instance: a formula F in CNF
  1. Form a clause set from F (and continue to call it F);
  2. repeat
     - G := F;
     - F := Res(F);

     until (□ ∈ F) or (F = G);
  3. if □ ∈ F then "F is **unsatisfiable**"

     else "F is **satisfiable**";

- Note that in some cases this algorithm can come up with a decision quite fast, but there do exist examples for **unsatisfiable** formulas where exponentially many resolvents have to be generated before the until condition is satisfied.

15

Resolution Refutation in Propositional Logic

- Suppose you need to prove this **logical consequence**
  - **F ⊨ G (G is True when F is True) – Theorem!**

    (set of axioms ⇒ conclusion)

    Note: F is True ⇒ the set of axioms is consistent

- Using the definition of **F → G**, it's easy to prove that:
  - if **F ⊨ G** then **F ∧ ¬ G** is a **contradiction**

16

Resolution Refutation in Propositional Logic

- if **F** $\models$ **G** then **F** $\wedge$ $\neg$ **G** is a **contradiction**

- <u>**Algorithm:**</u>
  1. Transform the set of axioms in a **clausal form,** S
  2. Negate the theorem (**negate the conclusion**). Transform the negated conclusion in a clausal form and add it to the set of axioms in a clausal form, S.
  3. **repeat**
     - 3.1. Select two appropriate clauses C_1 and C_2 from S
     - 3.2. Compute R = Res(C_1,C_2)
     - 3.3. **if** R $\neq$ ☐ **then** add R to S

     until R = ☐ **or** there are no two other clauses that resolve
  4. **if** R = ☐ **then**

     the theorem is proven
  5. **else**

     it is not a theorem

     (the conclusion cannot be proved from the set of axioms)

17

Resolution Refutation in Propositional Logic

<u>**Example:**</u> **(Resolution Refutation)**

- Prove: **(remember that "$\models$" $\equiv$ "$\Rightarrow$")**

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

  - Negate the conclusion:

    $F = (P \rightarrow Q) \wedge (Q \rightarrow R) \Rightarrow \neg(P \rightarrow R)$

  - Using $P \rightarrow Q \equiv \neg P \vee Q$, we obtain clause form set:

    $\{\{\neg P,Q\}, \{\neg Q,R\}, \{P\}, \{\neg R\}\}$

  - Derive the **empty clause** using resolution:

    {Q} resolving {P} with {¬P,Q}

    {R} resolving {Q} with {¬Q,R}

    ☐ resolving {R} with {¬R}

- **Conclusion:** F is **unsatisfiable** then

  $(P \rightarrow Q) \wedge (Q \rightarrow R) \Rightarrow (P \rightarrow R)$ is a **theorem**

18

Resolution Refutation in Propositional Logic

- From the previous example we prove that:

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

  - In particular we prove that:
  - "**Toby will die**" from the statements that:
    - "Toby is a dog" and
    - "all dogs are animals" and
    - "all animals will die"

19

Resolution Refutation in Propositional Logic

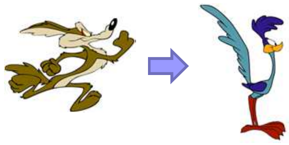
- Suppose you need to prove this **logical consequence**
  - **F $\models$ G (G is True when F is True) – Theorem!**

    (set of axioms $\Rightarrow$ conclusion)
- **Remember that!**
  - By the **Resolution Refutation**:
    - If we derive a **contradiction (empty clause)**, then the conclusion follows from the axioms.
    - If we **can't obtain a contradiction**, then the conclusion cannot be proved from the axioms.
    - **Example:**
      - Prove by resolution refutation: **P $\vee$ Q $\models$ P $\wedge$ Q**

20

Resolution Refutation in Propositional Logic

Example (**Roadrunner & Coyote**):



- **Facts/Axioms:**
  - Coyote chases Roadrunner
  - If Roadrunner is smart, Coyote does not catch it
  - If Coyote chases Roadrunner and does not catch it, then Coyote is annoyed.
  - Roadrunner is smart
- **Theorem:** Coyote is annoyed **????**

21

Resolution Refutation in Propositional Logic

Example (**Roadrunner and Coyote**):

- **Facts/Axioms:**
  - Coyote chases Roadrunner
  - If Roadrunner is smart, Coyote does not catch it
  - If Coyote chases Roadrunner and does not catch it, then Coyote is annoyed.
  - Roadrunner is smart

Theorem: Coyote is annoyed **????**

- **Proof strategy:**
  - We try to prove that **"Coyote is NOT annoyed"** is a **contradiction**
  - We add "Coyote is NOT annoyed" to the clause form set, and prove False
  - We will use the following set of literals:
    - C = Coyote chases Roadrunner
    - S = Roadrunner is smart
    - B = Coyote catches Roadrunner
    - A = Coyote is annoyed
- **Exercise:** Prove this theorem by resolution refutation

22