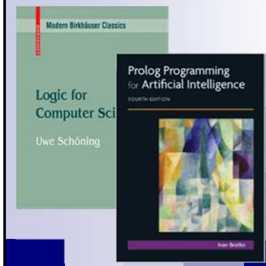


FIU | FLORIDA INTERNATIONAL UNIVERSITY

Propositional Logic (V)

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 Not to be published or publicly distributed without permission by the publisher.

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**";

  where **Res(F)**= F ∪ {R | R is a **resolvent** of two clauses in F}

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

Resolution Theorem

- In the following we want to distinguish between:
  - the clauses which are generated by the algorithm and
  - those clauses thereof which are <u>really relevant</u> to derive the empty clause.

<u>Definition</u>

- A **derivation** (or proof) of the <u>empty clause</u> from a clause set F is a sequence $C_1, C_2, \ldots, C_m$ of clauses such that C_m is the empty clause, and for every i (1, ... , m) C_i either is a clause in F or a **resolvent** of two clauses C_a, C_b with a, b < i.

3

Resolution Theorem

- From the **previous** definition we have a new formulation of the **Resolution Theorem**

<u>Theorem</u> (reformulation of Resolution Theorem):

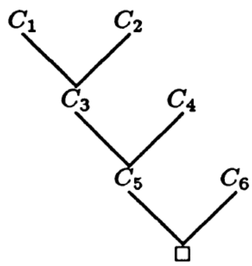
- A clause set F is **unsatisfiable** if and only if a **derivation** of the **empty clause** from F exists.

4

Resolution Theorem

Example: Let $F = \{\{A, B, \neg C\}, \{\neg A\}, \{A, B, C\}, \{A, \neg B\}\}$. F is unsatisfiable. This fact is proved by the following derivation $C_1, \ldots, C_7$ where

$$
\begin{array}{lcll}
C_1 & = & \{A, B, \neg C\} & \text{(clause in } F\text{)} \\
C_2 & = & \{A, B, C\} & \text{(clause in } F\text{)} \\
C_3 & = & \{A, B\} & \text{(resolvent of } C_1, C_2\text{)} \\
C_4 & = & \{A, \neg B\} & \text{(clause in } F\text{)} \\
C_5 & = & \{A\} & \text{(resolvent of } C_3, C_4\text{)} \\
C_6 & = & \{\neg A\} & \text{(clause in } F\text{)} \\
C_7 & = & \square & \text{(resolvent of } C_5, C_6\text{)}
\end{array}
$$



Resolution Theorem

Exercises (homework):

1. Use propositional resolution theorem show that the following sets of clauses are **unsatisfiable.**
   a) $\{p, q\}, \{\neg p, r\}, \{\neg p, \neg r\}, \{p, \neg q\}$
   b) $\{p, q, \neg r, s\}, \{\neg p, r, s\}, \{\neg q, \neg r\}, \{p, \neg s\}, \{\neg p, \neg r\}, \{r\}$

2. Use propositional **resolution refutation** to prove the following sentence.

$$((p \lor q) \land (p \to r)) \vDash (p \to r)$$

Resolution Strategies

- When doing **resolution** automatically, one has **to decide <u>in which order to resolve the clauses</u>**.
- This order can greatly affect the time needed to find a contradiction (empty clause).
- Strategies include:
  - **Unit resolution strategy**
  - **Set of support strategy**
  - **Davis-Putnam Procedure**

7

Unit Resolution Strategy

- **Unit resolution:** all resolutions involve **at least one unit clause.** Preference is given to clauses that have not been used yet.
- Prove P_4 from $P_1 \to P_2, \neg P_2, \neg P_1 \to P_3 \vee P_4$, $P_3 \to P_5$, $P_6 \to \neg P_5$ and P_6.

1.	$\neg P_1 \vee P_2$	Premise
2.	$\neg P_2$	Premise
3.	$P_1 \vee P_3 \vee P_4$	Premise
4.	$\neg P_3 \vee P_5$	Premise
5.	$\neg P_6 \vee \neg P_5$	Premise
6.	P_6	Premise
7.	$\neg P_4$	Negation of conclusion
8.	$\neg P_1$	Resolvent of 1, 2
9.	$\neg P_5$	Resolvent of 5, 6
10.	$P_1 \vee P_3$	Resolvent of 3, 7
11.	$\neg P_3$	Resolvent of 4, 9
12.	P_3	Resolvent of 8, 10
13.	$\square$	Resolvent of 11, 12

Unit Resolution Strategy

- **Unit resolution is not complete!**
- **Example:**
  - Check the following **logical consequence**:

    $(Q \vee R) \wedge (Q \vee \neg R) \wedge (\neg Q \vee R) \vDash (Q \wedge R)$
  - The set of clauses (including the **negation of the conclusion**):

    $\{\{Q,R\}, \{Q,\neg R\}, \{\neg Q,R\}, \{\neg Q,\neg R\}\}$
  - In this case there is no unit clause, which makes **unit resolution impossible**.

Set of Support Strategy

Basic ideas:

- One partitions all clauses into two sets:
  - the **set of support** and
  - the **auxiliary set**.
- The **auxiliary set** is formed in such a way that the formulas in it are not contradictory.
- For instance, the premises (facts/axioms) are usually not contradictory. The inconsistency only arises after one adds the negation of the conclusion.
- One often uses:
  - the **premises** as the **initial auxiliary set** and
  - the **negation of the conclusion** as the initial **set of support.**

Set of Support Strategy

- Then we have:
  - the **premises** as the **initial auxiliary set** and
  - the **negation of the conclusion** as the initial **set of support.**
- Since one cannot derive any contradiction by resolving clauses within the **auxiliary set**, one avoids such resolutions.
  - i.e. each resolution takes **at least one clause** from the **set of support**.
- The **resolvent** is then added to the **set of support**.
- **Resolution with the set of support strategy is complete!**

Set of Support Strategy

Example:

Prove P_4 from $P_1 \rightarrow P_2, \neg P_2, \neg P_1 \rightarrow P_3 \vee P_4, P_3 \rightarrow P_5, P_6 \rightarrow \neg P_5$ and P_6, by using the set of support strategy.

Initially the set of support is given by $\neg P_4$, the negation of the conclusion.

One then does all the possible resolutions involving $\neg P_4$, then all possible resolutions involving the resulting resolvents, and so on.

1.	$\neg P_1 \vee P_2$	Premise
2.	$\neg P_2$	Premise
3.	$P_1 \vee P_3 \vee P_4$	Premise
4.	$\neg P_3 \vee P_5$	Premise
5.	$\neg P_6 \vee \neg P_5$	Premise
6.	P_6	Premise
7.	$\neg P_4$	Negation of concl
8.	$P_1 \vee P_3$	Resolvent of 7, 3
9.	$P_2 \vee P_3$	Resolvent of 1, 8
10.	P_3	Resolvent of 2, 9
11.	P_5	Resolvent of 4, 10
12.	$\neg P_6$	Resolvent of 5, 11
13.	$\square$	Resolvent of 6, 12

Davis-Putnam Procedure – the algorithm

The main loop

- Given as input a nonempty **set of clauses** in the propositional literals P_1, P_2, …, P_n, the **Davis-Putnam Procedure (DPP)** repeats the following steps until there are no literals left:
  - Choose a literal P_i appearing in one of the clauses.
  - Add all possible **resolvents** using resolution on P_i to the set of clauses.
  - Discard all clauses with P_i or $\neg P_i$ in them.

13

Davis-Putnam Procedure – the algorithm

The stop criteria and the output

- If in some step of **DPP** we resolve $\{P_i\}$ and $\{\neg P_i\}$ then we obtain the **empty clause**, and it will be the only clause at the end of the procedure.
- If we never obtain a pair $\{P_i\}$ and $\{\neg P_i\}$ to resolve, then all the clauses will be thrown out and the output will be **no clauses (empty set of clauses)**.
- So the **output of DPP** either the **empty clause** or **no clauses (empty set of clauses)**.
- If the **output of DPP** is the **empty clause**, this indicates that both $\{P_i\}$ and $\{\neg P_i\}$ were produced, that is, the **formula is unsatisfiable**.
- If the **output of DPP** is **no clause (empty set of clauses)**, no contradiction can be found, and the **formula is satisfiable**.

14

Davis-Putnam Procedure - example

Example:

Theorem: $(\neg P \vee Q) \wedge (\neg Q \vee \neg R \vee S) \wedge P \wedge R \Rightarrow S$ **???**

$\Leftrightarrow$ F={{¬P,Q}, {¬Q,¬R,S}, {P}, {R}, {¬S}} is **unsatisfiable??**

Proof: (using Davis-Putnam Procedure)

Set of Literals of **F = {P, Q, R, S}**

By P: New clauses using resolution on P: {Q}
F={{¬P,Q}, {¬Q,¬R,S}, {P}, {R}, {¬S}, {Q}}
Discard all clauses with P or ¬P in them.
F={{¬Q,¬R,S}, {R}, {¬S}, {Q}}

By Q: New clauses using resolution on Q: {¬R,S}
F={{¬Q,¬R,S}, {R}, {¬S}, {Q}, {¬R,S}}
Discard all clauses with Q or ¬Q in them.
F={{R}, {¬S}, {¬R,S}}

By R: New clauses using resolution on R: {S}
F={{R}, {¬S}, {¬R,S}}, {S}}
Discard all clauses with R or ¬R in them.
F={{S}, {¬S}}

By S: New clauses using resolution on S: $\square$
F={{S}, {¬S}, $\square$ }
Discard all clauses with S or ¬S in them.
F={ $\square$ }

So the output is the **empty clause**, then F is **unsatisfiable** ➔ the original argument is valid (**theorem is proven!**)

15

Davis-Putnam Procedure

Exercise:

Theorem: $(\neg P \vee Q) \wedge (\neg Q \vee \neg R \vee \neg S) \wedge P \wedge R \Rightarrow S$ **???**

$\Updownarrow$ $\Updownarrow$

F={{¬P,Q}, {¬Q,¬R,¬S}, {P}, {R}, {¬S}} is **unsatisfiable ???**

Proof: (using Davis-Putnam Procedure)

16

Davis-Putnam Procedure

- **Theorem: (the DPP is correct and complete)**
  - Let S be a finite set of clauses. Then S is **unsatisfiable** if the **output** of the **Davis-Putnam Procedure** is the **empty clause**.

17

Resolution. Final remarks

- **Resolution** is a simple syntactic transformation applied to formulas.
- A collection of such transformation rules we call a **resolution calculus (or calculus)**.
- In the **resolution calculus**, the task is to prove **unsatisfiability** of a given formula.
- The definition of a **calculus** is sensible only if its **correctness** and its **completeness** can be established.
- **Correctness** (or **soundness**) means that every formula for which the **calculus** claims **unsatisfiability** indeed is **unsatisfiable**.
- **Completeness** means that for every **unsatisfiable** formula there is <u>a way to prove</u> this by means of the resolution **calculus**.

18

Resolution. Final remarks.

- We have seen that in some special case the resolution calculus leads to an efficient algorithm to determine (**un**)**satisfiablity**.
- In the case of arbitrary clause sets, it is possible to exhibit **unsatisfiable** clause sets such that every <u>derivation of the empty clause consists of exponentially many resolution steps</u>.
- **In general case, the expense of the resolution algorithm is comparable with the expense of the truth-table method.**
- Because of the "**NP-completeness**" of the **satisfiability problem**, there does not seem to exist any significantly faster algorithm.