781 - 2013-02-26

Note Title 2013-02-19

Goal: l1_b ; Show KB ∪ {~l1_b} is unsatisfiable

KB =

In computer:

{ *light1_broken* ← *sw_up* ∧ *power* ∧ *unlit_light1*.

sw_up.

power ← *lit_light2*.

unlit_light1.

lit_light2. }

In user's mind:

- *light1_broken*: light #1 is broken
- *sw_up*: switch is up
- *power*: there is power in the building
- *unlit_light1*: light #1 isn't lit
- *lit_light2*: light #2 is lit

Conclusion: *light1_broken* {l1_b}

- The computer doesn't know the meaning of the symbols
- The user can interpret the symbol using their meaning

{l1_b, ~sw_up, ~pow, ~un_l1}   {sw_up}

{l1_b, ~pow, ~un_l1}   {un_l1}

{l1_b, ~pow}   {pow, ~lit_l2}   {lit_l2}

{pow}

{~l1_b}   {l1_b}

[]

CNF and DNF

How to convert a prop. calculus formula to CNF & DNF.

Two main techniques.

One involves using the substitution theorem and a number of equivalences, including:

1. The "definition" of implication $P \to Q \equiv \sim P \vee Q$
2. The "definition" of equivalence $P \leftrightarrow Q \equiv (P \to Q) \wedge (Q \to P)$

3. Idempotency $(F \wedge F) \equiv F$, $(F \vee F) \equiv F$

4. Commutativity $(F \wedge G) \equiv (G \wedge F)$, $(F \vee G) \equiv (G \vee F)$

5. Associativity $((F \wedge G) \wedge H) \equiv (F \wedge (G \wedge H))$ $(\equiv (F \wedge G \wedge H))$

$((F \vee G) \vee H) \equiv (F \vee (G \vee H))$ $(\equiv (F \vee G \vee H))$

6. Absorption $(F \wedge (F \vee G)) \equiv F$, $(F \vee (F \wedge G)) \equiv F$

7. Distributivity $(F \wedge (G \vee H)) \equiv ((F \wedge G) \vee (F \wedge H))$

$(F \vee (G \wedge H)) \equiv ((F \vee G) \wedge (F \vee H))$

8. Double Negation $\neg\neg F \equiv F$

9. De Morgan's Laws $\neg(F \wedge G) \equiv (\neg F \vee \neg G)$

$\neg(F \vee G) \equiv (\neg F \wedge \neg G)$

10. Tautology Laws $(F \vee G) \equiv F$ if F is a tautology

$(F \wedge G) \equiv G$ if F is a tautology.

11. Unsatisfiability Laws $(F \vee G) \equiv G$ if F is unsatisfiable

$(F \wedge G) \equiv F$ if F is unsatisfiable

To convert F to CNF,

1. Push negation inwards, using

$$\neg\neg G \Rightarrow G$$

$$\neg(G \wedge H) \Rightarrow (\neg G \vee \neg H)$$

$$\neg(G \vee H) \Rightarrow (\neg G \wedge \neg H)$$

until no subformulas of the form on the LHS of these rules occur.

2. Substitute in F each occurrence of a subformula of the form

$$(F \vee (G \wedge H)) \Rightarrow ((F \vee G) \wedge (F \vee H))$$

$$((F \wedge G) \vee H) \Rightarrow ((F \vee H) \wedge (G \vee H))$$

until no such subformulas occur.

The resulting formula is in CNF (except for possible occurrences of tautologies).

(End of method)

Another technique is based on truth tables.
For formula F, to convert to DNF, take each row of its truth-table and build a conjunction with a positive literal of the corresponding variable is assigned 1 (T), and a negative literal of the corresponding variable is assigned 0.

To convert F to CNF, interchange the roles of 0 and 1, and construct a disjunction for each row of the truth table.

Example. Convert to DNF and CNF the following formula

$$F = \big((\neg A \to B) \wedge ((A \wedge \neg C) \leftrightarrow B)\big)$$

A	B	C	$\overset{A \vee B}{(\neg A \to B)}$	$(A \wedge \neg C)$	$((A \wedge \neg C) \leftrightarrow B)$	F
0	0	0	0	0	1	0
0	0	1	0	0	1	0
0	1	0	1	0	0	0
0	1	1	1	0	0	0
1	0	0	1	1	0	0
1	0	1	1	0	1	1
1	1	0	1	1	1	1
1	1	1	1	0	0	0

The equivalent DNF formula is

$$(A \wedge \neg B \wedge C) \vee (A \wedge B \wedge \neg C)$$

The equivalent CNF formula is

$$(A \vee B \vee C) \wedge (A \vee B \vee \neg C) \wedge (A \vee \neg B \vee C) \wedge (A \vee \neg B \vee \neg C) \wedge$$
$$(\neg A \vee B \vee C) \wedge (\neg A \vee \neg B \vee \neg C)$$, which is very opaque,

b/c this is also an equivalent CNF formula.

$$A \wedge (B \text{ XOR } C) \equiv A \wedge (B \vee C) \wedge (\neg B \vee \neg C)$$

Let's try the other method (for CNF):

$$F = ((\neg A \to B) \wedge ((A \wedge \neg C) \leftrightarrow B)) \equiv$$

$$\equiv ((\neg\neg A \vee B) \wedge (((A \wedge \neg C) \to B) \wedge (B \to (A \wedge \neg C)))) \equiv$$

$$\equiv ((A \vee B) \wedge ((\neg(A \wedge \neg C) \vee B) \wedge (\neg B \vee (A \wedge \neg C)))) \equiv$$

$$\equiv (A \vee B) \wedge (((\neg A \vee C) \vee B) \wedge (\neg B \vee (A \wedge \neg C))) \equiv$$

$$\equiv (A \vee B) \wedge ((\neg A \vee C \vee B) \wedge ((\neg B \vee A) \wedge (\neg B \vee \neg C))) \equiv$$

$$\equiv \underline{(A \vee B)} \wedge (\neg A \vee B \vee C) \wedge \underline{(A \vee \neg B)} \wedge (\neg B \vee \neg C) \equiv$$

$\equiv A \wedge (\neg A \vee B \vee C) \wedge (\neg B \vee \neg C) \equiv$

$\equiv A \wedge (B \vee C) \wedge (\neg B \vee \neg C)$

Resolution

The (propositional) resolution rule:

Assume: a formula in (conjunctive) clausal form;

start with F in CNF, so

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

(F is a conjunction of k clauses, where the ith clause contains n_i literals)

Write each clause as a set, so F is expressed as a set of clauses;

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

<u>Definition</u> <u>Resolvent</u>.

C_1 and C_2 are clauses. Then R is a <u>resolvent</u> of C_1 and C_2 if there is a literal $L \in C_1$ such that $\bar{L} \in C_2$ and $R = (C_1 - \{L\}) \cup (C_2 - \{\bar{L}\})$.

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

Graphical notation

$$\begin{array}{ccc} \{A, \neg C\} & & \{A, B, C\} \\ & \diagdown \quad \diagup & \\ & \{A, B\} & \end{array}$$

The resolvent of $\{A, \neg C\}$ and $\{A, B, C\}$ $\longrightarrow$ $\{A, B\}$

Theorem. 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 $F \cup \{R\}$ are equivalent.

<u>Definition</u> 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

$Res^0(F) = F$

$Res^{n+1}(F) = Res(Res^n(F))$ for $n \geq 0$, and

finally let

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

It can be proved that for every finite F, there is a k s.t. $Res^k(F) = Res^*(F)$.

<u>Defn.</u> The empty resolvent is the resolvent of $C_1 = \{L\}$ and $C_2 = \{\bar{L}\}$.

This is also called the <u>empty clause</u>, and is symbolized by $\square$.

$$\begin{tikzcd}[column sep=small] \{A_3\} \arrow[dr, dash] & & \{\neg A_3\} \arrow[dl, dash] \\ & \square & \end{tikzcd}$$

<u>The Resolution Theorem</u> (of propositional logic)

[J. A. Robinson proved this for FOL around 1960.]

A clause set F is unsatisfiable iff $\square \in Res^*(F)$.

<u>Example</u> (Ex 33 Schöning) Using resolution, show that $(A \land B \land C) = G$ is a consequence of

The clause set $F = \{\{\neg A, B\}, \{\neg B, C\}, \{A, \neg C\}, \{A, B, C\}\}$

Recall: $F \models G$ iff $\vdash F \to G$ iff $F \wedge \neg G$ is unsatisfiable

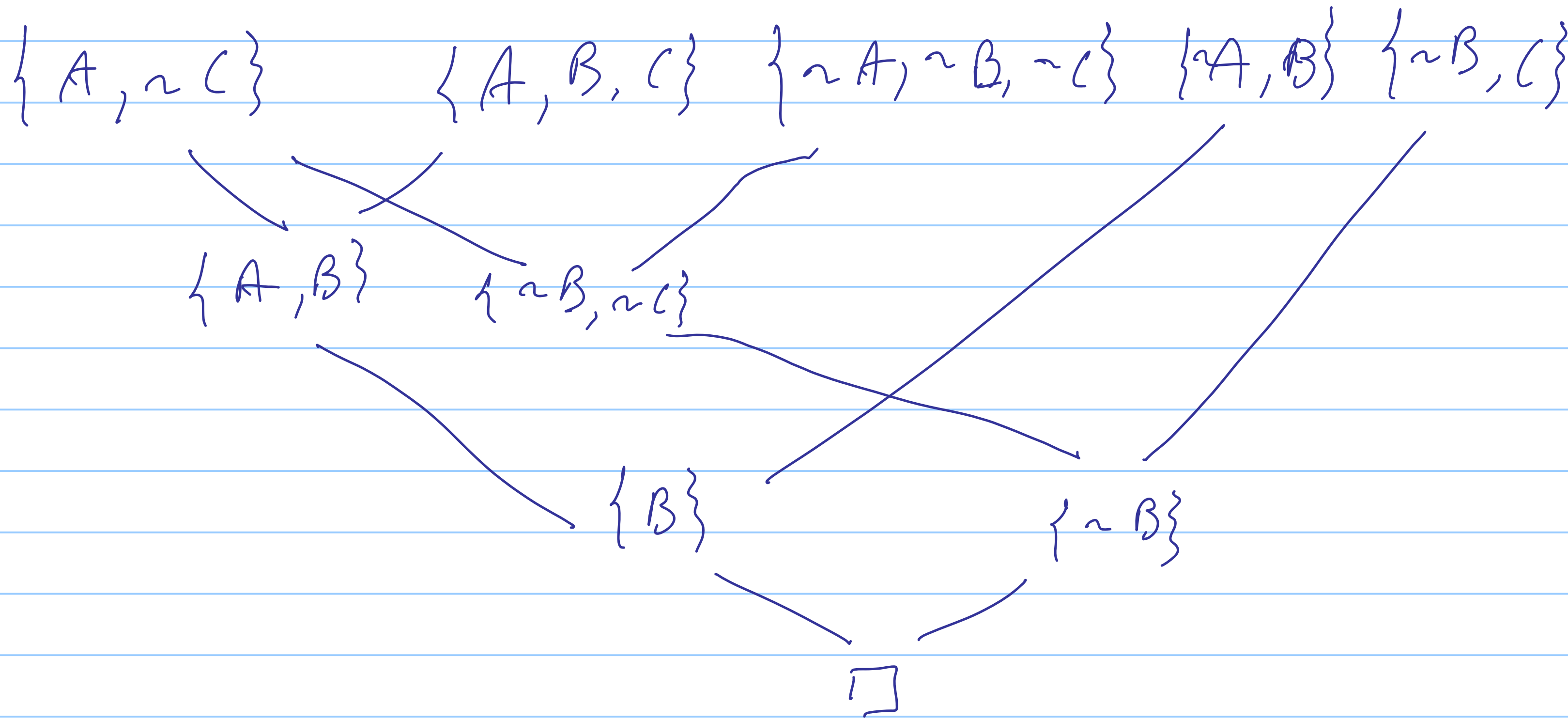
So. Negate G and show that $F \wedge \neg G$ is unsat.

$\neg G$: $\{\neg A, \neg B, \neg C\}$. Show that

$F \cup \neg G$: $\{\{\neg A, B\}, \{\neg B, C\}, \{A, \neg C\}, \{A, B, C\}, \{\neg A, \neg B, \neg C\}\}$

is unsat, i.e., by the res thm, show that

$Res^*(F \cup \neg G) = \square.$



Show by resolution that the following formula is unsatisfiable:

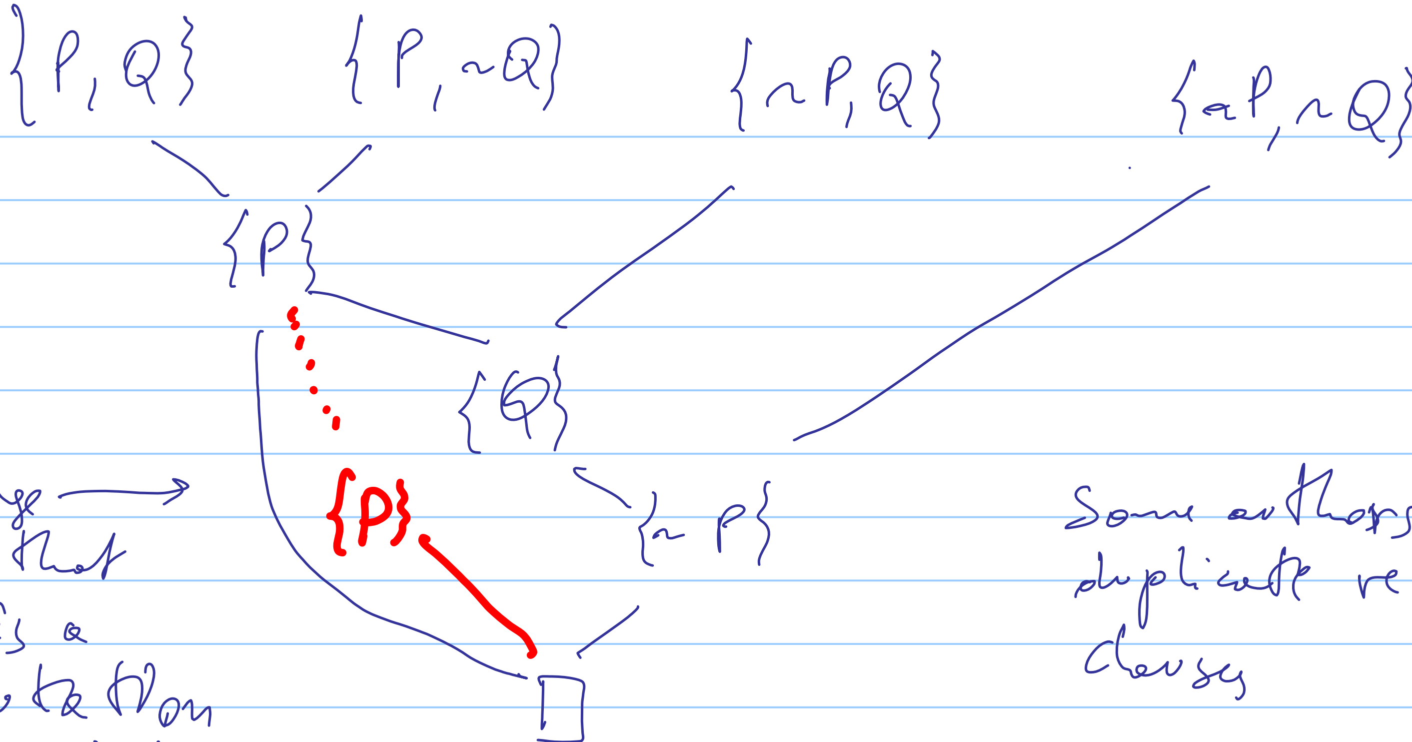
$$F = (P \vee Q) \wedge (P \vee \sim Q) \wedge (\sim P \vee Q) \wedge (\sim P \vee \sim Q)$$

$$\{P, Q\} \qquad \{P, \sim Q\} \qquad \{\sim P, Q\} \qquad \{\sim P, \sim Q\}$$

$$\{P\} \qquad\qquad\qquad \{\sim P\}$$

$$\square$$

Another proof of the same:

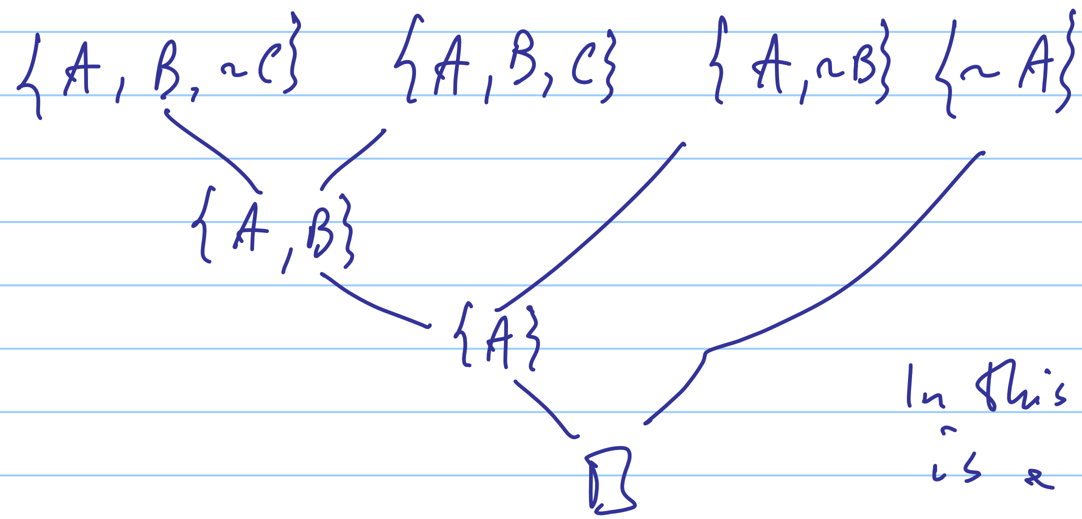


This edge → shows that this is a refutation graph (Schöning)

Some authors (e.g. Loveland) duplicate reused clauses

Another example - Show that

$F = \{\{A, B, \sim C\}, \{\sim A\}, \{A, B, C\}, \{A, \sim B\}\}$ is unsatisfiable



This is an example of a resolution graph.

In this case, the r. g. is a tree.

Exercise 34 [Schöning]

Using resolution, show that

$F = (\sim B \wedge \sim C \wedge D) \vee (\sim B \wedge \sim D) \vee (C \wedge D) \vee B$ is a tautology.

We show that $\sim F$ is unsatisfiable.

$\sim F = (B \vee C \vee \sim D) \wedge (B \vee D) \wedge (\sim C \vee \sim D) \wedge \sim B$, in clausal form:

$\{B, C, \sim D\} \quad \{B, D\} \quad \{\sim C, \sim D\} \quad \{\sim B\}$

$\{B, \sim D\}$

$\{B\}$

$\square$

A formula F in CNF is a Horn formula if every disjunction ("clause") in F contains at most one positive literal.

We distinguish three kinds of Horn clauses

① A — a fact (exactly and only one positive literal)

② $H \vee \sim B_1 \vee \dots \vee \sim B_k$ — a rule (one pos. literal & at least one negative literal)

③ $\sim B_1 \vee \dots \vee \sim B_k$ [i.e., $\sim (B_1 \wedge \dots \wedge B_k)$] — an integrity constraint (or indefinite clause) (only negative literals)

Horn clauses are often written in <u>implicative form</u>.

(Reminder/defn.: 1 stands for any tautology
0 " " " contradiction.)

1. $1 \to A$ (fact, usually written as A)
2. $B_1 \wedge \cdots \wedge B_k \to H$ (rule)
3. $B_1 \wedge \cdots \wedge B_k \to 0$ (integrity constraint)

There is an efficient (linear-time) algorithm to test satisfiability of Horn formulas. Here it is:

Input: a Horn formula F (i[illegible]tive form)

1. Mark every occurrence of an atomic formula (i.e., propositional variable) A in F if there is a clause $(1 \to A)$ in F.

2. <u>while</u> there is a clause G in F of form 2 or 3, where $B_1, \ldots, B_k$ are already marked (and H is not yet marked),
   <u>do</u>
   - if G is of form 2,
     then mark every occurrence of H
     else output "unsatisfiable" and halt.

3. Output "satisfiable" and halt
   (The satisfying assignment is given by the marking, in that variable $A_i = 1$ iff A_i has a mark.)

[The above algorithm is sound & complete. No proof given. It provides <u>the</u> minimal model for F.]

Exercise 22 [Schöning] Give an example of a formula that does not have an equivalent Horn formula.

$A \vee B$, because this formula does not have a single minimal model.

<u>Homework</u> (Ex. 21 [Schöning]): HW2 due Tuesday.

Apply the marking algorithm to

$F = (\sim A \vee \sim B \vee \sim D) \wedge \sim E \wedge (\sim C \vee A) \wedge C \wedge B \wedge (\sim G \vee D) \wedge G$

If the formula is unsatisfiable, please also show that by resolution.