

Drawbacks of Gilmore's Algorithm:

- How can one find a suitable instantiation of variables by ground terms? Sect 3.4
- How can one check satisfiability of a propositional formula efficiently? Sect 3.3

Resolution: main proof technique in logic prog.

First: ground resolution (for formulas without variables)

To check a formula $\forall X_1, \dots, X_n \varphi$ in Skolem NF for unsatisfiability by resolution, one first has to transform φ into conjunctive normal form (CNF).

Such formulas can then be represented as clause sets.

Def 331 (CNF, clause, literal)

A formula φ is in conjunctive normal form iff it is quantifier-free and has the form

$$(L_{1,1} \vee \dots \vee L_{1,n_1}) \wedge \dots \wedge (L_{m,1} \vee \dots \vee L_{m,n_m}).$$

Here, $L_{i,j}$ are literals, i.e., atomic formulas or negated atomic formulas of the form $p(t_1, \dots, t_n)$ or $\neg p(t_1, \dots, t_n)$.

For every literal L , its negation \bar{L} is defined as

$$\bar{L} = \begin{cases} \neg A, & \text{if } L = A \in \text{At}(\Sigma, \Delta, \mathcal{V}) \\ A & \text{if } L = \neg A, A \in \text{At}(\Sigma, \Delta, \mathcal{V}) \end{cases}$$

A clause is a set of literals and it represents the

universally quantified disjunction of the literals.

A clause set represents the conjunction of its clauses.

So every formula φ in CNF can be represented as a clause set

$$\mathcal{K}(\varphi) = \{ \{L_{1,1}, \dots, L_{1,n_1}\}, \dots, \{L_{m,1}, \dots, L_{m,n_m}\} \}$$

Therefore, we also speak of satisfiability and entailment of clause sets.

The empty clause is denoted \square . It is unsatisfiable by definition (empty disjunction).

Thm 332 (Transformation to CNF)

Every quantifier-free formula φ can be transformed into an equivalent formula φ' in CNF automatically.

Proof: First replace all sub-formulas $\varphi_1 \leftrightarrow \varphi_2$ by $(\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1)$.

Then replace all sub-formulas $\varphi_1 \rightarrow \varphi_2$ by $\neg \varphi_1 \vee \varphi_2$.

Afterwards, use the following algorithm CNF:

- Input: φ (quantifier-free, without \leftrightarrow or \rightarrow)
- Output: equivalent formula in CNF

- If φ is an atomic formula, then return φ .

- If $\varphi = \varphi_1 \wedge \varphi_2$, then return $\text{CNF}(\varphi_1) \wedge \text{CNF}(\varphi_2)$.

- If $\varphi = \varphi_1 \vee \varphi_2$, then compute

$$\begin{aligned} \text{CNF}(\varphi_1) &= \varphi_1' \wedge \dots \wedge \varphi_{m_1}' \\ \text{CNF}(\varphi_2) &= \varphi_1'' \wedge \dots \wedge \varphi_{m_2}'' \end{aligned} \quad \begin{aligned} &= (\varphi_1' \wedge \dots \wedge \varphi_{m_1}') \vee \\ &= (\varphi_1'' \wedge \dots \wedge \varphi_{m_2}'') \end{aligned}$$

$$\text{Return } \bigwedge_{\substack{i \in \{1, \dots, m_1\} \\ j \in \{1, \dots, m_2\}}} \varphi_i' \vee \varphi_j''$$

← use the distributivity law

- If $\psi = \neg \psi_1$, then compute

$$\text{CNF}(\psi_1) = \bigwedge_{i \in \{1, \dots, m\}} \left(\bigvee_{j \in \{1, \dots, n_i\}} L_{i,j} \right)$$

De Morgan's law states that the negation of this formula is

$$\bigvee_{i \in \{1, \dots, m\}} \left(\bigwedge_{j \in \{1, \dots, n_i\}} \overline{L_{i,j}} \right)$$

Due to the distributivity law, we return

$$\bigwedge_{\substack{j_1 \in \{1, \dots, n_1\}, \\ \vdots \\ j_m \in \{1, \dots, n_m\}}} \left(\overline{L_{1,j_1}} \vee \dots \vee \overline{L_{m,j_m}} \right)$$

□

Ex 333 Let $p, q, r \in \Delta_0$.

Transform the following formula into CNF:

$$\neg(\neg p \wedge (\neg q \vee r))$$

↓ De Morgan

$$p \vee (q \wedge \neg r)$$

↓ Distributivity

$$(p \vee q) \wedge (p \vee \neg r)$$

Alg. of Gilmore: To check unsatisf. of ψ , consider $E(\psi)$ and prove unsatisf. of these propositional formulas.

Therefore: Now introduce a technique to prove unsatisfiability of prop. formulas in CNF.

In other words: Prove unsatisf. of clause sets

without variables.

Resolution: $(L_1 \vee L) \wedge (L_2 \vee \bar{L})$

implies $L_1 \vee L_2$

Def 334 (Propositional Resolution)

Slide 13

Let K_1, K_2 be two clauses without variables.

Then the clause R is a resolvent of K_1 and K_2 iff

there exists a $L \in K_1$ with $\bar{L} \in K_2$

and $R = (K_1 \setminus \{L\}) \cup (K_2 \setminus \{\bar{L}\})$.

For a clause set \mathcal{K} we define

$\text{Res}(\mathcal{K}) = \mathcal{K} \cup \{R \mid R \text{ is resolvent of two clauses from } \mathcal{K}\}$

We define

$\text{Res}^0(\mathcal{K}) = \mathcal{K}$

$\text{Res}^{u+1}(\mathcal{K}) = \text{Res}(\text{Res}^u(\mathcal{K}))$ for all $u \geq 0$

Moreover: $\text{Res}^*(\mathcal{K}) = \bigcup_{u \geq 0} \text{Res}^u(\mathcal{K})$

Idea: Construct $\text{Res}^*(\mathcal{K})$ until one obtains

\square . Since adding resolvents is equivalence-preserving, this means that \mathcal{K} is unsatisfiable.

Clearly: $\square \in \text{Res}^*(\mathcal{K})$ iff

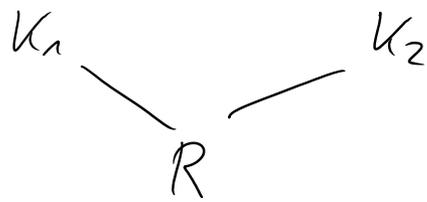
there is a sequence of clauses K_1, \dots, K_m with $K_m = \square$

where for all $1 \leq i \leq m$, we have

• $K_i \in \mathcal{K}$ or

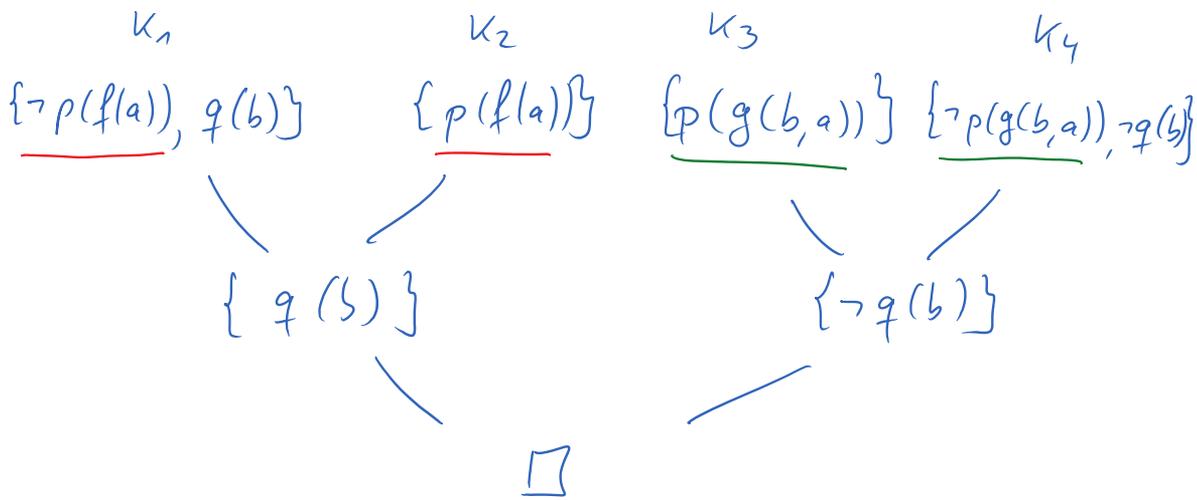
- K_i is a resolvent of K_j, K_k for $j, k < i$.

To denote resolution proofs:



means that
 R is resolvent of
 K_1 and K_2 .

Ex. 335 Let $\Delta_1 = \{p, q\}, \Sigma_0 = \{a, b\}, \Sigma_1 = \{f\}, \Sigma_2 = \{g\}$.
 We regard the clause set $\mathcal{K} = \{K_1, K_2, K_3, K_4\}$ with



Slide
13

The resolution calculus is sound and complete:

Completeness: If \mathcal{K} is unsat., then $\square \in \text{Res}^*(\mathcal{K})$.

Soundness: If $\square \in \text{Res}^*(\mathcal{K})$, then \mathcal{K} is unsat..

To prove soundness, we need the following lemma.

Lemma 336 (Propositional Resolution Lemma)

Let \mathcal{K} be a set of clauses without variables.

If $K_1, K_2 \in \mathcal{K}$ and R is a resolvent of K_1 and K_2 ,
 then \mathcal{K} and $\mathcal{K} \cup \{R\}$ are equivalent.

Proof: " \Leftarrow ": Every structure that satisfies $\mathcal{K} \cup \{R\}$ also
 satisfies \mathcal{K} . Thus: $\mathcal{K} \cup \{R\} \models \mathcal{K}$

" \Rightarrow ": Let S be a structure with $S \models \mathcal{K}$.

Let $L \in \mathcal{K}_1$, $\bar{L} \in \mathcal{K}_2$, $R = (\mathcal{K}_1 \setminus \{L\}) \cup (\mathcal{K}_2 \setminus \{\bar{L}\})$.

Assume $S \not\models \mathcal{K} \cup \{R\}$. Thus: $S \not\models R$

If $S \models L$, then $S \models \mathcal{K}_2$ implies $S \models \mathcal{K}_2 \setminus \{\bar{L}\}$

Therefore: $S \models R$ $\quad \text{!}$

If $S \not\models L$, then $S \models \bar{L}$ and $S \models \mathcal{K}_1$ implies $S \models \mathcal{K}_1 \setminus \{L\}$

Therefore: $S \models R$. $\quad \text{!}$

\square

Thm 3.37 (Soundness and Completeness of Prop. Resolution)

Let \mathcal{K} be a set of clauses without variables.

Then \mathcal{K} is unsatisfiable iff $\square \in \text{Res}^*(\mathcal{K})$.

Proof: Soundness " \Leftarrow ":

By the resolution lemma 3.3.6., \mathcal{K} and $\text{Res}(\mathcal{K})$ are equivalent. By induction on n , one can show that \mathcal{K} and $\text{Res}^n(\mathcal{K})$ are equivalent for all $n \in \mathbb{N}$.

$\square \in \text{Res}^*(\mathcal{K})$

\leadsto there exists an $n \in \mathbb{N}$ with $\square \in \text{Res}^n(\mathcal{K})$

$\leadsto \text{Res}^n(\mathcal{K})$ is unsat.

$\leadsto \mathcal{K}$ is unsat.

Completeness " \Rightarrow ":

If \mathcal{K} is unsatisfiable, then there is a finite subset $\mathcal{K}' \subseteq \mathcal{K}$ which is also unsat. (by the compactness thm. of prop. logic).

Let n be the number of different atomic formulas in \mathcal{K}' . We use induction on n .

Ind. Base: $n=0$

There are only two clause sets without atomic formulas:

$\mathcal{K}' = \emptyset$ ← empty conjunction: valid, i.e., true in every interpretation \checkmark

$\mathcal{K}' = \{ \square \}$. Then $\square \in \text{Res}^0(\mathcal{K}') \subseteq \text{Res}^0(\mathcal{K})$.

Ind. Step: $n > 0$

Let A be an atomic formula that occurs in the unsat. clause set \mathcal{K}' .

Let \mathcal{K}^+ result from \mathcal{K}' by

- removing all clauses that contain literal A
- drop $\neg A$ from all remaining clauses

Thus: $\mathcal{K}^+ = \{ K \setminus \{ \neg A \} \mid K \in \mathcal{K}', A \notin K \}$

Similarly $\mathcal{K}^- = \{ K \setminus \{ A \} \mid K \in \mathcal{K}', \neg A \notin K \}$

\mathcal{K}^+ is unsat.: If $S \models \mathcal{K}^+$, then extend S to a structure S' with $S' \models A$.

Then $S' \models \mathcal{K}'$ \checkmark to the unsat. of \mathcal{K}'

Similarly, \mathcal{K}^- is unsat.

Since \mathcal{K}^+ and \mathcal{K}^- do not contain A , we can apply the ind. hypothesis, which yields:

$\square \in \text{Res}^*(\mathcal{K}^+)$, $\square \in \text{Res}^*(\mathcal{K}^-)$.

$\square \in \text{Res}^*(\mathcal{K}^+)$ means that there is a sequence

K_1, \dots, K_m with $\square = K_m$ where

for all $1 \leq i \leq m$:

• $K_i \in \mathcal{K}^+$ or

• K_i is resolvent of K_j and K_k for $j, k < i$

$\left\{ \begin{array}{l} \cdot K_i \text{ is resolvent of } K_j \text{ and } K_k \text{ for } j, k < i \end{array} \right.$

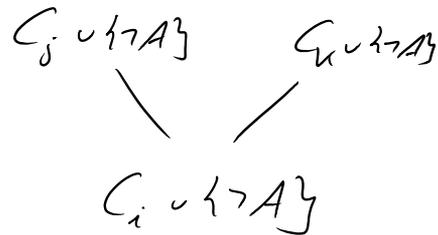
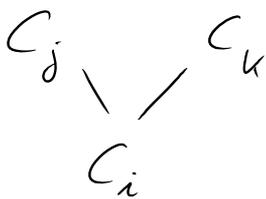
If all these K_i are also contained in \mathcal{K}' , then we have proved $\square \in \text{Res}^*(\mathcal{K}') \subseteq \text{Res}^*(\mathcal{K})$.

Otherwise: add $\neg A$ again to all clauses where it had been removed.

Then, obtain a resolution proof for

$$\{\neg A\} \in \text{Res}^*(\mathcal{K}')$$

Reason:



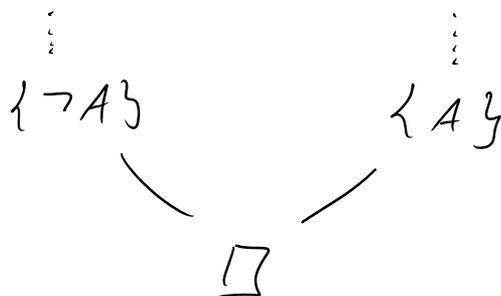
Similarly: $\square \in \text{Res}^*(\mathcal{K}^-)$ implies

$$\square \in \text{Res}^*(\mathcal{K}') \text{ or}$$

$$\{A\} \in \text{Res}^*(\mathcal{K}').$$

Thus: $\{A\}, \{\neg A\} \in \text{Res}^*(\mathcal{K}')$

implies $\square \in \text{Res}^*(\mathcal{K}')$



□

Now the alg. of Gilmore can be improved to the ground resolution algorithm.

sounds il ab ... "L" " " " " ...

Slide 14

sound: if alg. returns "true", then $\{\varphi_1, \dots, \varphi_k\} \models \varphi$

complete: if $\{\varphi_1, \dots, \varphi_k\} \models \varphi$, then alg. terminates and returns "true"

if $\{\varphi_1, \dots, \varphi_k\} \not\models \varphi$, then the alg. might not terminate

⇒ semi-decision procedure

Step 5: Advantage over Gilmore's alg.

⇒ better check for unsat. of propositional clause sets

Step 4: Still inefficient, since we don't know how to instantiate variables by ground terms in a "clever" way.

Ex 338: Show unsatisfiability of

$$\forall X, Y \underbrace{(\neg p(X) \vee \neg p(f(a)) \vee q(Y)) \wedge p(Y) \wedge (\neg p(g(b, X)) \vee \neg q(b))}_{\gamma}$$

Corresponding clause set $\mathcal{K}(\gamma)$:

$$\{\neg p(X), \neg p(f(a)), q(Y)\}, \{p(Y)\}, \{\neg p(g(b, X)), \neg q(b)\}$$

$$u_1: [X/f(a), Y/b] \quad u_2: [Y/f(a)] \quad u_3: [Y/g(b, a)] \quad u_4: [X/a]$$

$$\{\neg p(f(a)), q(b)\} \quad \{p(f(a))\} \quad \{p(g(b, a))\} \quad \{p(g(b, a)), \neg q(b)\}$$

$$\begin{array}{ccc} \diagdown & & \diagup \\ \{q(b)\} & & \{\neg q(b)\} \\ \diagup & & \diagdown \end{array}$$

□

• Instantiations can unify several literals of the same clause.

• We can use several different instantiations of the same

clause.

How can one find such suitable instantiations automatically?