Lecture 5

Resolution

Resolution proof calculus, Davis-Putnam procedure

Dr Christoph Haase
University of Oxford
(with small changes by Javier Esparza)

Overview

SAT is bad:

- Truth tables: exponential time
- Horn-SAT, 2-SAT and X-SAT require special formulas
- **Resolution**: still worst case exponential time

Overview

SAT is bad:

- Truth tables: exponential time
- Horn-SAT, 2-SAT and X-SAT require special formulas
- **Resolution**: still worst case exponential time

But:

- very easy to automate
- very easy to analyse theoretically
- still sound and complete
- only takes polynomial time on Horn and 2-CNF formulas

Proof calculus

Resolution is a **proof calculus** for propositional logic

- rules of inference
- derive series of conclusions from series of hypothesis
- mechanical

Proof calculus

Resolution is a **proof calculus** for propositional logic

- rules of inference
- derive series of conclusions from series of hypothesis
- mechanical
- resolution has only one rule of inference
- is sound and complete:
  - soundness: anything that we prove is valid
  - completeness: anything that is valid can be proved

Set representation of CNF formulas

Resolution only works on CNF formulas.
Handy representation:

- clause $\rightarrow$ set of literals
- CNF formula $\rightarrow$ set of clauses

Set representation of CNF formulas

Resolution only works on CNF formulas.
Handy representation:

- clause $\rightarrow$ set of literals
- CNF formula $\rightarrow$ set of clauses

Example

$$(p_1 \vee \neg p_2) \wedge (p_3 \vee \neg p_4 \vee p_5) \wedge (\neg p_2)$$

is represented as

$$\{\{p_1, \neg p_2\}, \{p_3, \neg p_4, p_5\}, \{\neg p_2\}\}$$

Set representation of CNF formulas

Elements have no order or multiplicity, so set representation is only normal form modulo associativity, commutativity, and idempotence:

$$(p_3 \wedge (p_1 \vee p_1 \vee \neg p_2) \wedge p_3)$$
$$((\neg p_2 \vee p_1 \vee \neg p_2) \wedge (p_3 \vee p_3))$$
$$(p_3 \wedge (\neg p_2 \vee p_1))$$

all have representation $\{\{p_3\}, \{p_1, \neg p_2\}\}$

Set representation of CNF formulas

Elements have no order or multiplicity, so set representation is only normal form modulo associativity, commutativity, and idempotence:

$$(p_3 \wedge (p_1 \vee p_1 \vee \neg p_2) \wedge p_3)$$
$$((\neg p_2 \vee p_1 \vee \neg p_2) \wedge (p_3 \vee p_3))$$
$$(p_3 \wedge (\neg p_2 \vee p_1))$$

all have representation $\{\{p_3\}, \{p_1, \neg p_2\}\}$

- Empty clause, denoted $\square$, is equivalent to *false*
- If CNF formula contains $\square$, it is unsatisfiable
- If CNF formula is $\square$, it is equivalent to *true*

(Compare: sum of empty set of natural numbers is 0, but product of empty set of natural numbers is 1.)

Resolvents

Recall: for L, complementary one $\overline{L}$ is defined by

$$\overline{L} := \begin{cases} \neg p & \text{if } L = p \\ p & \text{if } L = \neg p \end{cases}$$

Resolvents

Recall: for L, complementary one $\overline{L}$ is defined by

$$\overline{L} := \begin{cases} \neg p & \text{if } L = p \\ p & \text{if } L = \neg p \end{cases}$$

Definition

Let C_1 and C_2 be clauses. A clause R is called a **resolvent** of C_1 and C_2 if there are complementary literals $L \in C_1$ and $\overline{L} \in C_2$ such that

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

Resolvents

Recall: for L, complementary one $\overline{L}$ is defined by

$$\overline{L} := \begin{cases} \neg p & \text{if } L = p \\ p & \text{if } L = \neg p \end{cases}$$

Definition

Let C_1 and C_2 be clauses. A clause R is called a **resolvent** of C_1 and C_2 if there are complementary literals $L \in C_1$ and $\overline{L} \in C_2$ such that

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

We say R **is derived from** C_1 **and** C_2 **by resolution**, and write

$$\frac{C_1 \quad C_2}{R}$$

Resolvents: example

Example

$\{p_1, p_3, \neg p_4\}$ resolves $\{p_1, p_2, \neg p_4\}$ and $\{\neg p_2, p_3\}$,
the empty clause is a resolvent of $\{p_1\}$ and $\{\neg p_1\}$:

$$\frac{\{p_1, p_2, \neg p_4\} \quad \{\neg p_2, p_3\}}{\{p_1, p_3, \neg p_4\}} \qquad \frac{\{p_1\} \quad \{\neg p_1\}}{\Box}$$

Derivations and refutations

Definition

A **derivation** (or **proof**) of a clause C from a set of clauses F is a sequence $C_1, C_2, \ldots, C_m$ of clauses where $C_m = C$ and for each $i = 1, 2, \ldots, m$ either $C_i \in F$ or C_i is a resolvent of C_j and C_k for some $j, k < i$.

Derivations and refutations

Definition

A **derivation** (or **proof**) of a clause C from a set of clauses F is a sequence $C_1, C_2, \ldots, C_m$ of clauses where $C_m = C$ and for each $i = 1, 2, \ldots, m$ either $C_i \in F$ or C_i is a resolvent of C_j and C_k for some $j, k < i$.

A derivation of the empty clause $\Box$ from a formula F is called a **refutation** of F.

Derivations: example

A resolution refutation of the CNF formula

$$\{\{x, \neg y\}, \{y, z\}, \{\neg x, \neg y, z\}, \{\neg z\}\}$$

is as follows:

1.	$\{x, \neg y\}$	(Assumption)	5.	$\{\neg x, z\}$	(2,4 Resolution)
2.	$\{y, z\}$	(Assumption)	6.	$\{\neg z\}$	(Assumption)
3.	$\{x, z\}$	(1,2 Resolution)	7.	$\{z\}$	(3,5 Resolution)
4.	$\{\neg x, \neg y, z\}$	(Assumption)	8.	$\square$	(6,7 Resolution)

Derivations: example

A resolution refutation of the CNF formula

$$\{\{x, \neg y\}, \{y, z\}, \{\neg x, \neg y, z\}, \{\neg z\}\}$$

is as follows:

1.	$\{x, \neg y\}$	(Assumption)	5.	$\{\neg x, z\}$	(2,4 Resolution)
2.	$\{y, z\}$	(Assumption)	6.	$\{\neg z\}$	(Assumption)
3.	$\{x, z\}$	(1,2 Resolution)	7.	$\{z\}$	(3,5 Resolution)
4.	$\{\neg x, \neg y, z\}$	(Assumption)	8.	$\square$	(6,7 Resolution)

Graphically represented by the following **proof tree**:

$$\dfrac{\dfrac{\dfrac{\{x, \neg y\} \qquad \{y, z\}}{\{x, z\}} \qquad \dfrac{\{y, z\} \qquad \{\neg x, \neg y, z\}}{\{\neg x, z\}}}{\{z\}} \qquad \{\neg z\}}{\square}$$

Refutations: comments

- A resolution refutation of a formula F can be seen as a proof that F is unsatisfiable
- Resolution can be used to prove entailments by transforming them to refutations
- For example, the refutation in previous example can be used to show that

$$(x \lor \neg y) \land (y \lor z) \land (\neg x \lor \neg y \lor z) \models z$$

Set of resolvents

Given set of clauses F, interested in set of all clauses derivable from F by resolution.

Definition

For set F of clauses, $Res(F)$ 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 write

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

Set of resolvents

Given set of clauses F, interested in set of all clauses derivable from F by resolution.

Definition

For set F of clauses, $Res(F)$ is defined as

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

Furthermore define

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

and write

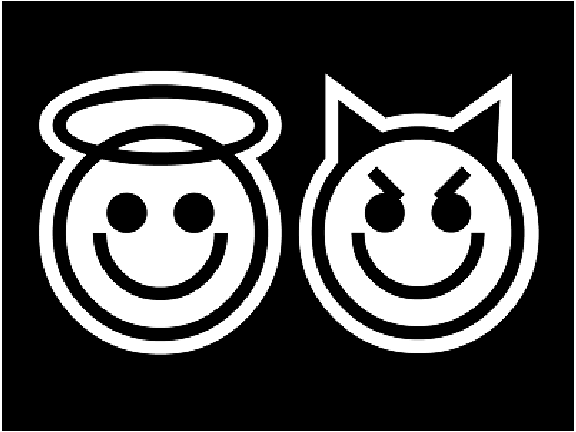
$$Res^*(F) = \bigcup_{n \geq 0} Res^n(F)$$

Theorem

$C \in Res^*(F)$ iff there is a derivation of C from F.

Soundness and completeness

Soundness: anything that we prove is valid
Completeness: anything that is valid can be proved



The resolution lemma

Lemma

Let F be CNF formula represented as set of clauses. If R is a resolvent of clauses C_1 and C_2 of F, then $F \equiv F \cup \{R\}$.

The resolution lemma

Lemma

Let F be CNF formula represented as set of clauses. If R is a resolvent of clauses C_1 and C_2 of F, then $F \equiv F \cup \{R\}$.

Proof.

For assignment $\mathcal{A}$, clearly, if $\mathcal{A} \models F \cup \{R\}$ then $\mathcal{A} \models F$. Conversely, suppose $\mathcal{A} \models F$ and $R = (C_1 \setminus \{L\}) \cup (C_2 \setminus \{\overline{L}\})$ for some literal L, where $L \in C_1$ and $\overline{L} \in C_2$.

- If $\mathcal{A} \models L$, then since $\mathcal{A} \models C_2$, it follows that $\mathcal{A} \models C_2 \setminus \{\overline{L}\}$, and thus $\mathcal{A} \models R$.
- If $\mathcal{A} \models \overline{L}$, then since $\mathcal{A} \models C_1$, it follows that $\mathcal{A} \models C_1 \setminus \{L\}$, and thus $\mathcal{A} \models R$.

$\square$

Soundness

Soundness: can only derive a contradiction from an unsatisfiable set of clauses.

Soundness

Soundness: can only derive a contradiction from an unsatisfiable set of clauses.

Theorem

If we can derive $\Box$ from F, then F is unsatisfiable.

Soundness

Soundness: can only derive a contradiction from an unsatisfiable set of clauses.

Theorem

If we can derive $\Box$ from F, then F is unsatisfiable.

Proof.

Suppose $C_1, C_2, \ldots, C_m = \Box$ is a proof of $\Box$ from F. Repeated application of the Resolution Lemma shows $F \equiv F \cup \{C_1, C_2, \ldots, C_m\}$. But the latter set of clauses includes the empty clause. $\Box$

Completeness

Completeness is converse of soundness: if a CNF formula is unsatisfiable then can derive the empty clause from it by resolution.

Completeness

Completeness is converse of soundness: if a CNF formula is unsatisfiable then can derive the empty clause from it by resolution.

Theorem

If F is unsatisfiable, then we can derive $\Box$ from F.

Completeness

Proof.

By induction on number n of variables in F.

Completeness

Proof.

By induction on number n of variables in F.

- If $n = 0$, then F has no variables, so either contains no clauses or only the empty clause. In the former case $F \equiv \mathit{true}$, which is satisfiable, so must have $F = \{\Box\}$, giving one-line resolution refutation of F.

Completeness

Proof.

By induction on number n of variables in F.

- If $n = 0$, then F has no variables, so either contains no clauses or only the empty clause. In the former case $F \equiv \mathit{true}$, which is satisfiable, so must have $F = \{\Box\}$, giving one-line resolution refutation of F.
- Suppose variables $p_0, \ldots, p_n$. Since F is unsatisfiable, so is $F_0 := F[\mathit{false}/p_n]$. Induction hypothesis gives resolution proof $C_0, C_1, \ldots, C_m = \Box$ that derives $\Box$ from F_0.

Completeness

Proof.

By induction on number n of variables in F.

- If $n = 0$, then F has no variables, so either contains no clauses or only the empty clause. In the former case $F \equiv \mathit{true}$, which is satisfiable, so must have $F = \{\Box\}$, giving one-line resolution refutation of F.
- Suppose variables $p_0, \ldots, p_n$. Since F is unsatisfiable, so is $F_0 := F[\mathit{false}/p_n]$. Induction hypothesis gives resolution proof $C_0, C_1, \ldots, C_m = \Box$ that derives $\Box$ from F_0. Each C_i from F_0 is either already in F or $C_i \cup \{p_n\}$ is in F.

Completeness

Proof.

By induction on number n of variables in F.

- If $n = 0$, then F has no variables, so either contains no clauses or only the empty clause. In the former case $F \equiv \mathit{true}$, which is satisfiable, so must have $F = \{\Box\}$, giving one-line resolution refutation of F.
- Suppose variables $p_0, \ldots, p_n$. Since F is unsatisfiable, so is $F_0 := F[\mathit{false}/p_n]$. Induction hypothesis gives resolution proof $C_0, C_1, \ldots, C_m = \Box$ that derives $\Box$ from F_0. Each C_i from F_0 is either already in F or $C_i \cup \{p_n\}$ is in F. Re-introducing p_n and propagating gives proof $C'_0, C'_1, \ldots, C'_m$ from F where either $C'_m = \Box$ or $C'_m = \{p_n\}$.

Completeness

Proof.

By induction on number n of variables in F.

- If $n = 0$, then F has no variables, so either contains no clauses or only the empty clause. In the former case $F \equiv \mathit{true}$, which is satisfiable, so must have $F = \{\square\}$, giving one-line resolution refutation of F.
- Suppose variables $p_0, \ldots, p_n$. Since F is unsatisfiable, so is $F_0 := F[\mathit{false}/p_n]$. Induction hypothesis gives resolution proof $C_0, C_1, \ldots, C_m = \square$ that derives $\square$ from F_0. Each C_i from F_0 is either already in F or $C_i \cup \{p_n\}$ is in F. Re-introducing p_n and propagating gives proof $C'_0, C'_1, \ldots, C'_m$ from F where either $C'_m = \square$ or $C'_m = \{p_n\}$.
- Apply similar reasoning to $F_1 := F[\mathit{true}/p_n]$, get proof of $\{\neg p_n\}$ from F.

Completeness

Proof.

By induction on number n of variables in F.

- If $n = 0$, then F has no variables, so either contains no clauses or only the empty clause. In the former case $F \equiv \mathit{true}$, which is satisfiable, so must have $F = \{\Box\}$, giving one-line resolution refutation of F.
- Suppose variables $p_0, \ldots, p_n$. Since F is unsatisfiable, so is $F_0 := F[\mathit{false}/p_n]$. Induction hypothesis gives resolution proof $C_0, C_1, \ldots, C_m = \Box$ that derives $\Box$ from F_0. Each C_i from F_0 is either already in F or $C_i \cup \{p_n\}$ is in F. Re-introducing p_n and propagating gives proof $C'_0, C'_1, \ldots, C'_m$ from F where either $C'_m = \Box$ or $C'_m = \{p_n\}$.
- Apply similar reasoning to $F_1 := F[\mathit{true}/p_n]$, get proof of $\{\neg p_n\}$ from F. Glue together these two proofs and apply one more resolution step to $\{p_n\}$ and $\{\neg p_n\}$.

$\Box$

Completeness: example

Example

Consider $F = \{\{p, r\}, \{\neg p, q\}, \{\neg q, r\}\}$.
Transform the following derivation of $\square$ from $F[\mathit{false}/r]$

$$\cfrac{\cfrac{\{p\} \qquad \{\neg p, q\}}{\{q\}} \qquad \{\neg q\}}{\square}$$

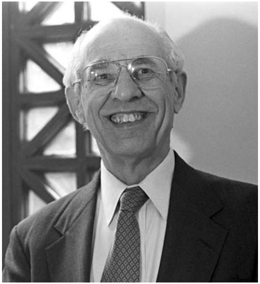
to the following derivation of $\{r\}$ from F:

$$\cfrac{\cfrac{\{p, r\} \qquad \{\neg p, q\}}{\{q, r\}} \qquad \{\neg q, r\}}{\{r\}}$$

The Davis–Putnam procedure

Can turn resolution into a **SAT solver**

Basic idea: **Davis–Putnam procedure**



Use resolution to perform **variable elimination**,
and compute satisfying valuation

Variable elimination

Eliminate p from CNF formula F to get new formula G:

1. If p occurs only positively in F,
   delete all clauses containing p, so $G := F[\mathit{true}/p]$
2. If p occurs only negatively in F,
   delete all clauses containing $\overline{p}$, so $G := F[\mathit{false}/p]$
3. Suppose p occurs both positively and negatively in F.
   For every pair of clauses C, D in F with $p \in C$ and $\overline{p} \in D$,
   add the resolvent of C and D (w.r.t. p) to F.
   Delete all clauses containing p or $\overline{p}$ from F to get G.

Variable elimination

Eliminate p from CNF formula F to get new formula G:

1. If p occurs only positively in F,
   delete all clauses containing p, so $G := F[\mathit{true}/p]$
2. If p occurs only negatively in F,
   delete all clauses containing $\overline{p}$, so $G := F[\mathit{false}/p]$
3. Suppose p occurs both positively and negatively in F.
   For every pair of clauses C, D in F with $p \in C$ and $\overline{p} \in D$,
   add the resolvent of C and D (w.r.t. p) to F.
   Delete all clauses containing p or $\overline{p}$ from F to get G.

Example

Eliminating p from $\{\{p\}, \{\neg p, q\}, \{\neg q, r\}, \{\neg r, s, t\}\}$ gives
$\{\{q\}, \{\neg q, r\}, \{\neg r, s, t\}\}$.

Variable elimination: correctness

Lemma (Elimination Lemma)

If eliminating variable p from F gives G then

- *F and G are equisatisfiable*
- *if $\mathcal{A} \models G$ then $\mathcal{A}_{[p \mapsto a]} \models F$ for some $a \in \{0, 1\}$ that can be determined from $\mathcal{A}$ and F.*

The Davis–Putnam algorithm

Davis–Putnam(F)
begin
remove all valid clauses from F
if $F = \{\square\}$ **then** return UNSAT
if $F = \emptyset$ **then** return the 0 assignment
let G arise by eliminating a variable p from F
if Davis–Putnam(G) = UNSAT **then** return UNSAT
if Davis–Putnam(G) = $\mathcal{A}$ **then** return $\mathcal{A}_{[p \mapsto a]}$,
  with a chosen as in the Elimination Lemma
end

Davis–Putnam: example

First eliminate variables (p, q, r, s):

$$\begin{aligned}
&\text{Davis–Putnam}(\{\{p\}, \{\neg p, q\}, \{\neg q, r\}, \{\neg r, s, t\}\}) \\
&= \text{Davis–Putnam}(\{\{q\}, \{\neg q, r\}, \{\neg r, s, t\}\}) \\
&= \text{Davis–Putnam}(\{\{r\}, \{\neg r, s, t\}\}) \\
&= \text{Davis–Putnam}(\{\{s, t\}\}) \\
&= \text{Davis–Putnam}(\emptyset)
\end{aligned}$$

Then recurse back up to get satisfying assignment:

$$\begin{aligned}
t &\mapsto 0 \\
s &\mapsto 1 \\
r &\mapsto 1 \\
q &\mapsto 1 \\
p &\mapsto 1
\end{aligned}$$

Complexity

Davis–Putnam is worst case exponential time
(unsurprising: intermediate clauses can become big)

Complexity

Davis–Putnam is worst case exponential time
(unsurprising: intermediate clauses can become big)

Questions:

- Can one efficiently precompute a (near)optimal variable ordering?

Complexity

Davis–Putnam is worst case exponential time
(unsurprising: intermediate clauses can become big)

Questions:

- Can one efficiently precompute a (near)optimal variable ordering?
- Given k, can one efficiently precompute a variable ordering such that Davis–Putnam only produces k-clauses?

Complexity

Davis–Putnam is worst case exponential time
(unsurprising: intermediate clauses can become big)

Questions:

- Can one efficiently precompute a (near)optimal variable ordering?
- Given k, can one efficiently precompute a variable ordering such that Davis–Putnam only produces k-clauses?
- More simply: suppose $F = F_1 \wedge F_2$, where F_1 and F_2 have only variable p in common. Should I eliminate p first, last or in some other position?

Answers:

next time ...

Summary

Resolution is:

- a proof calculus
- sound and complete
- very simple

Davis–Putnam:

- decision algorithm for SAT
- basis of SAT solvers
- polynomial time on nice formulas
- worst case exponential time
- depend on order of elimination