Lecture 5

Resolution

Resolution proof calculus, Davis-Putnam procedure

Introduction to Logic for Computer Science

Prof Hongseok Yang
KAIST

These slides are minor variants of those made by Prof Worrell and Dr Haase for their logic course at Oxford.

Overview

SAT is bad.

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

Overview

SAT is bad.

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

But, resolution has merits.

- Can exploit the structure of a given formula.
- Only takes polynomial time on Horn and 2-CNF formulas.
- Very easy to automate.
- Very easy to analyse theoretically.
- Still sound and complete.

Resolution

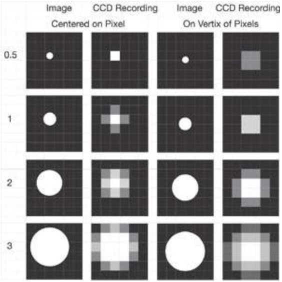


Image
CCD Recording
Image
CCD Recording
Centered on Pixel
On Vertix of Pixels
0.5
1
2
3

Resolution

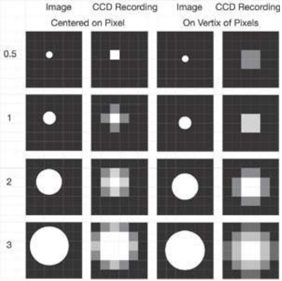
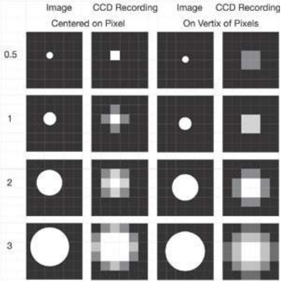


Image
CCD Recording
Image
CCD Recording
Centered on Pixel
On Vertix of Pixels
0.5
1
2
3

Resolution



The latter is more related to resolution in logic.

Proof calculus

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

- A proof calculus consists of rules of inference.
- Enables to derive series of conclusions from series of hypothesis.
- Mechanical.

Proof calculus

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

- A proof calculus consists of rules of inference.
- Enables to derive series of conclusions from series of hypothesis.
- Mechanical.
- Resolution has only one rule of inference.
- Sound and complete.
- Soundness: Anything proved is valid.
- Completeness: Anything 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 $\to$ set of literals.
- CNF formula $\to$ 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

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

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

Ex1: Write $(p_1 \vee \neg p_2 \vee p_1) \wedge (\neg p_2 \vee p_1) \wedge (p_3 \vee \neg p_4) \wedge (\neg p_2)$ as a set.

Ex2: What is good about this set representation?

Set representation of CNF formulas

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

$$
\begin{aligned}
&(p_3 \wedge (p_1 \vee p_1 \vee \neg p_2) \wedge p_3), \\
&((\neg p_2 \vee p_1 \vee \neg p_2) \wedge (p_3 \vee p_3)), \text{ and} \\
&(p_3 \wedge (\neg p_2 \vee p_1))
\end{aligned}
$$

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:

$$(p_3 \wedge (p_1 \vee p_1 \vee \neg p_2) \wedge p_3),$$
$$((\neg p_2 \vee p_1 \vee \neg p_2) \wedge (p_3 \vee p_3)), \text{ and}$$
$$(p_3 \wedge (\neg p_2 \vee p_1))$$

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

Empty clause (empty set of literals) is equivalent to *false*. Denoted $\Box$.

If a CNF formula contains $\Box$, it is unsatisfiable.

If a CNF formula is the empty set, it is equivalent to *true*.

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

Resolvents

Recall: for a literal L, its **complementary** literal $\overline{L}$ is defined by

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

Resolvents

Recall: for a literal L, its **complementary** literal $\overline{L}$ 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 a literal L, its **complementary** literal $\overline{L}$ 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 $\{\neg p_1\}$.

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

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 $\{\neg p_1\}$.

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

Ex: Compute a resolvent of each of the following pairs of clauses:

1. $\{p_1, \neg p_2, p_4\}$ and $\{p_1, \neg p_4, \neg p_5\}$.
2. $\{p_1, \neg p_2, p_4\}$ and $\{\neg p_1, \neg p_4, \neg p_5\}$.
3. $\{p_1, \neg p_2, p_4\}$ and $\{\neg p_1\}$.

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

$$(x \vee \neg y) \wedge (y \vee z) \wedge (\neg x \vee \neg y \vee z) \models z.$$

- Intuitively, proof by contradiction.

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

$$(x \vee \neg y) \wedge (y \vee z) \wedge (\neg x \vee \neg y \vee z) \models z.$$

- Intuitively, proof by contradiction.
- Ex: Suppose that we would like to prove $F \models G$ for CNF formulas F and G. How to do this using resolution?

Set of resolvents

Given a set F of clauses, we are interested in the set of all clauses derivable from F by resolution.

Definition

For a 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

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

and write

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

Set of resolvents

Given a set F of clauses, we are interested in the set of all clauses derivable from F by resolution.

Definition

For a 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

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

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.

Set of resolvents

Given a set F of clauses, we are interested in the set of all clauses derivable from F by resolution.

Definition

For a 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

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

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.

Ex: Prove the theorem.

Soundness and completeness

- **Soundness**: Anything proved is valid.
- **Completeness**: Anything valid can be proved.
- For every F, there is a resolution refutation of F iff $\neg F$ is valid.

The resolution lemma

Lemma

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

Ex: Show $F \cup \{R\} \models F$.

The resolution lemma

Lemma

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

Ex: Show $F \cup \{R\} \models F$. Now show that $F \models F \cup \{R\}$.

The resolution lemma

Lemma

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

Ex: Show $F \cup \{R\} \models F$. Now show that $F \models F \cup \{R\}$.

Proof.

We focus on proving $F \models F \cup \{R\}$. Suppose

- $\mathcal{A} \models F$, and
- $R = (C_1 \setminus \{L\}) \cup (C_2 \setminus \{\overline{L}\})$ for some literal $L \in C_1$ with $\overline{L} \in C_2$.

We will have to show $\mathcal{A} \models F \cup \{R\}$.

The resolution lemma

Lemma

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

Ex: Show $F \cup \{R\} \models F$. Now show that $F \models F \cup \{R\}$.

Proof.

We focus on proving $F \models F \cup \{R\}$. Suppose

- $\mathcal{A} \models F$, and
- $R = (C_1 \setminus \{L\}) \cup (C_2 \setminus \{\overline{L}\})$ for some literal $L \in C_1$ with $\overline{L} \in C_2$.

We will have to show $\mathcal{A} \models F \cup \{R\}$.

- 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 of resolution

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

Soundness of resolution

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

Theorem

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

Ex: Prove this using the Resolution Lemma.

Soundness of resolution

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

Theorem

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

Ex: Prove this using the Resolution Lemma.

Proof.

Suppose $C_1, C_2, \ldots, C_m = \square$ is a resolution proof of $\square$ from F.

Soundness of resolution

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

Theorem

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

Ex: Prove this using the Resolution Lemma.

Proof.

Suppose $C_1, C_2, \ldots, C_m = \square$ is a resolution proof of $\square$ from F.

The 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. Thus, F is unsatisfiable. $\square$

Completeness

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

Completeness

Completeness is the converse of soundness: if a CNF formula is unsat., then we 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 the number n of variables in F.

Completeness

Proof.

By induction on the number n of variables in F.

- If $n = 0$, then F has no variables. So either it contains no clauses or only the empty clause. The former is impossible because then $F \equiv \mathit{true}$ would be satisfiable. Thus, $F = \{\Box\}$. We can give an one-line resolution refutation of F.

Completeness

Proof.

By induction on the number n of variables in F.

- If $n = 0$, then F has no variables. So either it contains no clauses or only the empty clause. The former is impossible because then $F \equiv$ *true* would be satisfiable. Thus, $F = \{\Box\}$. We can give an one-line resolution refutation of F.
- Ex: Prove the inductive case.

Completeness

Proof.

By induction on the number n of variables in F.

- If $n = 0$, then F has no variables. So either it contains no clauses or only the empty clause. The former is impossible because then $F \equiv \mathit{true}$ would be satisfiable. Thus, $F = \{\Box\}$. We can give an one-line resolution refutation of F.
- Ex: Prove the inductive case.
- Suppose variables $p_0, \ldots, p_n$. Since F is unsatisfiable, so is $F_0 := F[\mathit{false}/p_n]$. Induction hypothesis gives a resolution proof $C_0, C_1, \ldots, C_m = \Box$ that derives $\Box$ from F_0.

Completeness

Proof.

By induction on the number n of variables in F.

- If $n = 0$, then F has no variables. So either it contains no clauses or only the empty clause. The former is impossible because then $F \equiv \mathit{true}$ would be satisfiable. Thus, $F = \{\Box\}$. We can give an one-line resolution refutation of F.
- Ex: Prove the inductive case.
- Suppose variables $p_0, \ldots, p_n$. Since F is unsatisfiable, so is $F_0 := F[\mathit{false}/p_n]$. Induction hypothesis gives a 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 the number n of variables in F.

- If $n = 0$, then F has no variables. So either it contains no clauses or only the empty clause. The former is impossible because then $F \equiv \mathit{true}$ would be satisfiable. Thus, $F = \{\Box\}$. We can give an one-line resolution refutation of F.
- Ex: Prove the inductive case.
- Suppose variables $p_0, \ldots, p_n$. Since F is unsatisfiable, so is $F_0 := F[\mathit{false}/p_n]$. Induction hypothesis gives a 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 it gives a res. 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 the number n of variables in F.

- If $n = 0$, then F has no variables. So either it contains no clauses or only the empty clause. The former is impossible because then $F \equiv \mathit{true}$ would be satisfiable. Thus, $F = \{\Box\}$. We can give an one-line resolution refutation of F.
- Ex: Prove the inductive case.
- Suppose variables $p_0, \ldots, p_n$. Since F is unsatisfiable, so is $F_0 := F[\mathit{false}/p_n]$. Induction hypothesis gives a 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 it gives a res. 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 a proof of $C_l'' = \Box$ or $C_l'' = \{\neg p_n\}$ from F.

Completeness

Proof.

By induction on the number n of variables in F.

- If $n = 0$, then F has no variables. So either it contains no clauses or only the empty clause. The former is impossible because then $F \equiv \mathit{true}$ would be satisfiable. Thus, $F = \{\Box\}$. We can give an one-line resolution refutation of F.
- Ex: Prove the inductive case.
- Suppose variables $p_0, \ldots, p_n$. Since F is unsatisfiable, so is $F_0 := F[\mathit{false}/p_n]$. Induction hypothesis gives a 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 it gives a res. 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 a proof of $C_l'' = \Box$ or $C_l'' = \{\neg p_n\}$ from F.
- If $C_m' = \Box$ or $C_l'' = \Box$, done. Otherwise, glue together these two proofs and apply one more resolution step to $\{p_n\}$ and $\{\neg p_n\}$.

$\Box$

Completeness: example

Example

Consider $F = \{\{p, r\}, \{\neg p, q\}, \{\neg q, r\}, \{\neg q, \neg r\}, \{p, \neg r\}\}$.

Transform the following derivation of $\Box$ from $F[\mathit{false}/r]$

$$\dfrac{\dfrac{\{p\} \qquad \{\neg p, q\}}{\{q\}} \qquad \{\neg q\}}{\Box}$$

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

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

Completeness: example

Example

Consider $F = \{\{p, r\}, \{\neg p, q\}, \{\neg q, r\}, \{\neg q, \neg r\}, \{p, \neg r\}\}$.

Transform the following derivation of $\Box$ from $F[\mathit{false}/r]$

$$\dfrac{\dfrac{\{p\} \qquad \{\neg p, q\}}{\{q\}} \qquad \{\neg q\}}{\Box}$$

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

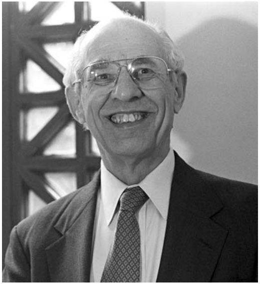
$$\dfrac{\dfrac{\{p, r\} \qquad \{\neg p, q\}}{\{q, r\}} \qquad \{\neg q, r\}}{\{r\}}$$

Ex: Handle the other case $F[\mathit{true}/r]$. Construct the derivation of $\Box$ from F.

The Davis–Putnam procedure

Can turn resolution into a **SAT solver**.

Davis–Putnam procedure.



Basic idea: Use resolution to perform **variable elimination** when searching for a satisfying assignment.

Variable elimination

Eliminate p from a CNF formula F to get a 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 resolvant of C and D to F.
   Delete all clauses containing p or $\overline{p}$ from F to get G.

Variable elimination

Eliminate p from a CNF formula F to get a 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 resolvant of C and D to F.
   Delete all clauses containing p or $\overline{p}$ from F to get G.

Example

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

Variable elimination

Eliminate p from a CNF formula F to get a 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 resolvant of C and D to F.
   Delete all clauses containing p or $\overline{p}$ from F to get G.

Example

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

Ex: Eliminate r from the above set of clauses.

Variable elimination: correctness

Lemma (Elimination Lemma)

If eliminating a variable p from F gives G then

- *F and G are equisatisfiable; and*
- *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.*

Variable elimination: correctness

Lemma (Elimination Lemma)

If eliminating a variable p from F gives G then

- *F and G are equisatisfiable; and*
- *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.*

Ex: Prove the lemma.

The Davis–Putnam algorithm

Davis–Putnam(F)
begin
remove all valid clauses from F
if $F = \{\Box\}$ **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):

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

Davis–Putnam: example

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

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

Then recurse back up to get satisfying assignment:

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

Complexity

Davis–Putnam takes exponential time in the worst case.
(unsurprising: intermediate clauses can become big).

Complexity

Davis–Putnam takes exponential time in the worst case.
(unsurprising: intermediate clauses can become big).

Questions:

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

Complexity

Davis–Putnam takes exponential time in the worst case.
(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 at most k-clauses?

Complexity

Davis–Putnam takes exponential time in the worst case.
(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 at most 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?

Complexity

Davis–Putnam takes exponential time in the worst case.
(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 at most 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?

Actually, I don't know the answers. But I recommend you to think about this type of questions.

Summary

Resolution:

- 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 the order of variable elimination.