

# 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 Prof 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

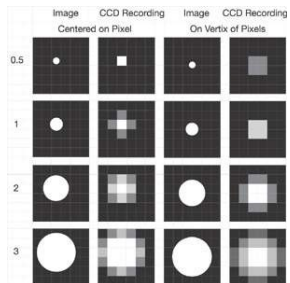
SAT is bad.

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

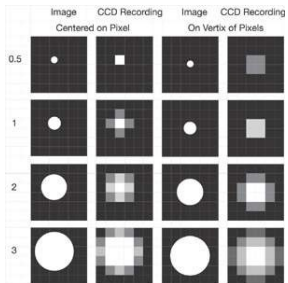
But, resolution has merits.

- 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.

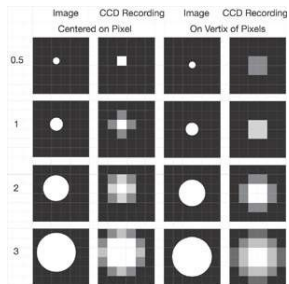
# Resolution



# Resolution



# 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.
- Lets us show a formula is unsatisfiable.
- For the proof by contradiction.
- Soundness: Anything proved to be unsatisfiable via resolution is unsatisfiable indeed.
- Completeness: Anything unsatisfiable can be proved to be unsatisfiable via resolution.



## 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

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:

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

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

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

If a CNF formula contains  $\square$ , 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  $\bar{L}$  is defined by

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

## Resolvents

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

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

### 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  $\bar{L} \in C_2$  such that

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

## Resolvents

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

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

### 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  $\bar{L} \in C_2$  such that

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

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  $\{\neg p_1\}$ .

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

## 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  $\{\neg p_1\}$ .

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

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  $C_1, C_2, \dots, C_m$  of clauses where

- $C_m = C$ ; and
- for each  $i = 1, 2, \dots, m$ , 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  $C_1, C_2, \dots, C_m$  of clauses where

- $C_m = C$ ; and
- for each  $i = 1, 2, \dots, m$ , 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  $\square$  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**:

$$\frac{\frac{\frac{\{x, \neg y\}}{\{x, z\}} \quad \{y, z\}}{\{z\}} \quad \frac{\{y, z\} \quad \{\neg x, \neg y, z\}}{\{\neg x, z\}} \quad \{\neg z\}}{\square}$$

## 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

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

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,  $\text{Res}(F)$  is defined as

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

Furthermore, define

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

and write

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

### Theorem

$C \in \text{Res}^*(F)$  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

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

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 to be unsatisfiable via resolution is unsatisfiable indeed.
- **Completeness:** Anything unsatisfiable can be proved to be unsatisfiable via resolution.
- For every  $F$ , there is a resolution refutation of  $F$  iff  $F$  is unsatisfiable.

## 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
- $R = (C_1 \setminus \{L\}) \cup (C_2 \setminus \{\bar{L}\})$  for some literal  $L \in C_1$  with  $\bar{L} \in C_2$ .

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
- $R = (C_1 \setminus \{L\}) \cup (C_2 \setminus \{\bar{L}\})$  for some literal  $L \in C_1$  with  $\bar{L} \in C_2$ .

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



## Soundness of resolution

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

## Soundness of resolution

**Soundness:** Can derive a contradiction only 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.

## Soundness of resolution

**Soundness:** Can derive a contradiction only 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  $C_1, C_2, \dots, C_m = \square$  is a resolution proof of  $\square$  from  $F$ .

## Soundness of resolution

**Soundness:** Can derive a contradiction only 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  $C_1, C_2, \dots, C_m = \square$  is a resolution proof of  $\square$  from  $F$ .

The repeated application of the Resolution Lemma shows

$$F \equiv F \cup \{C_1, C_2, \dots, C_m\}.$$

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  $\square$  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  $F \equiv \text{true}$  would be satisfiable. Thus,  $F = \{\square\}$ . 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  $F \equiv \text{true}$  would be satisfiable. Thus,  $F = \{\square\}$ . 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  $F \equiv \text{true}$  would be satisfiable. Thus,  $F = \{\square\}$ . We can give an one-line resolution refutation of  $F$ .
- Ex: Prove the inductive case.
- Suppose variables  $p_0, \dots, p_n$ . Since  $F$  is unsatisfiable, so is  $F_0 := F[\text{false}/p_n]$ . Induction hypothesis gives a resolution proof  $C_0, C_1, \dots, C_m = \square$  that derives  $\square$  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  $F \equiv \text{true}$  would be satisfiable. Thus,  $F = \{\square\}$ . We can give an one-line resolution refutation of  $F$ .
- Ex: Prove the inductive case.
- Suppose variables  $p_0, \dots, p_n$ . Since  $F$  is unsatisfiable, so is  $F_0 := F[\text{false}/p_n]$ . Induction hypothesis gives a resolution proof  $C_0, C_1, \dots, C_m = \square$  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  $F$ .

## 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  $F \equiv \text{true}$  would be satisfiable. Thus,  $F = \{\square\}$ . We can give an one-line resolution refutation of  $F$ .
- Ex: Prove the inductive case.
- Suppose variables  $p_0, \dots, p_n$ . Since  $F$  is unsatisfiable, so is  $F_0 := F[\text{false}/p_n]$ . Induction hypothesis gives a resolution proof  $C_0, C_1, \dots, C_m = \square$  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 it gives a res. proof  $C'_0, C'_1, \dots, C'_m$  from  $F$  where either  $C'_m = \square$  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  $F \equiv \text{true}$  would be satisfiable. Thus,  $F = \{\square\}$ . We can give an one-line resolution refutation of  $F$ .
- Ex: Prove the inductive case.
- Suppose variables  $p_0, \dots, p_n$ . Since  $F$  is unsatisfiable, so is  $F_0 := F[\text{false}/p_n]$ . Induction hypothesis gives a resolution proof  $C_0, C_1, \dots, C_m = \square$  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 it gives a res. proof  $C'_0, C'_1, \dots, C'_m$  from  $F$  where either  $C'_m = \square$  or  $C'_m = \{p_n\}$ .
- Apply similar reasoning to  $F_1 := F[\text{true}/p_n]$ . Get a proof of  $C''_1 = \square$  or  $C''_1 = \{\neg p_n\}$  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  $F \equiv \text{true}$  would be satisfiable. Thus,  $F = \{\square\}$ . We can give an one-line resolution refutation of  $F$ .
- Ex: Prove the inductive case.
- Suppose variables  $p_0, \dots, p_n$ . Since  $F$  is unsatisfiable, so is  $F_0 := F[\text{false}/p_n]$ . Induction hypothesis gives a resolution proof  $C_0, C_1, \dots, C_m = \square$  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 it gives a res. proof  $C'_0, C'_1, \dots, C'_m$  from  $F$  where either  $C'_m = \square$  or  $C'_m = \{p_n\}$ .
- Apply similar reasoning to  $F_1 := F[\text{true}/p_n]$ . Get a proof of  $C''_1 = \square$  or  $C''_1 = \{\neg p_n\}$  from  $F$ .
- If  $C'_m = \square$  or  $C''_1 = \square$ , done. Otherwise, glue together these two proofs and apply one more resolution step to  $\{p_n\}$  and  $\{\neg p_n\}$ .



## 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  $\square$  from  $F[false/r]$

$$\frac{\frac{\{p\} \quad \{\neg p, q\}}{\{q\}} \quad \{\neg q\}}{\square}$$

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

$$\frac{\frac{\{p, r\} \quad \{\neg p, q\}}{\{q, r\}} \quad \{\neg q, r\}}{\{r\}}$$



## 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  $\square$  from  $F[\text{false}/r]$

$$\frac{\frac{\{p\} \quad \{\neg p, q\}}{\{q\}} \quad \{\neg q\}}{\square}$$

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

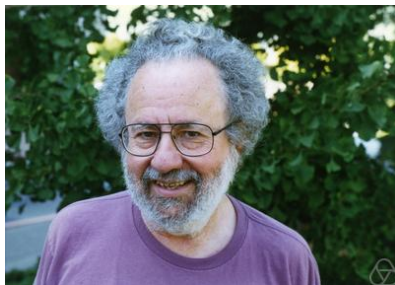
$$\frac{\frac{\{p, r\} \quad \{\neg p, q\}}{\{q, r\}} \quad \{\neg q, r\}}{\{r\}}$$

Ex: Handle the other case  $F[\text{true}/r]$ . Construct the derivation of  $\square$  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  $G := F[true/p]$ .
- 2 If  $p$  occurs only negatively in  $F$ ,  
delete all clauses containing  $\bar{p}$ , so  $G := F[false/p]$ .
- 3 Suppose  $p$  occurs both positively and negatively in  $F$ .  
For every pair of clauses  $C, D$  in  $F$  with  $p \in C$  and  $\bar{p} \in D$ ,  
add the resolvent of  $C$  and  $D$  to  $F$ .  
Delete all clauses containing  $p$  or  $\bar{p}$  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  $G := F[true/p]$ .
- 2 If  $p$  occurs only negatively in  $F$ ,  
delete all clauses containing  $\bar{p}$ , so  $G := F[false/p]$ .
- 3 Suppose  $p$  occurs both positively and negatively in  $F$ .  
For every pair of clauses  $C, D$  in  $F$  with  $p \in C$  and  $\bar{p} \in D$ ,  
add the resolvent of  $C$  and  $D$  to  $F$ .  
Delete all clauses containing  $p$  or  $\bar{p}$  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  $G := F[\text{true}/p]$ .
- 2 If  $p$  occurs only negatively in  $F$ ,  
delete all clauses containing  $\bar{p}$ , so  $G := F[\text{false}/p]$ .
- 3 Suppose  $p$  occurs both positively and negatively in  $F$ .  
For every pair of clauses  $C, D$  in  $F$  with  $p \in C$  and  $\bar{p} \in D$ ,  
add the resolvent of  $C$  and  $D$  to  $F$ .  
Delete all clauses containing  $p$  or  $\bar{p}$  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**  $F = \{\square\}$  **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  $(p, q, r, s)$ :

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

## 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:

$$t \mapsto 1$$

$$s \mapsto 1$$

$$r \mapsto 1$$

$$q \mapsto 0$$

$$p \mapsto 1$$

## 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.