Propositional Logic
Resolution and DPLL

Mario Alviano

University of Calabria, Italy

A.Y. 2018/2019

Outline

CNF transformation

1. Eliminate $\top$, $\bot$, $\rightarrow$, $\leftrightarrow$
2. Apply double negation equivalences:
   - $\neg\neg\phi \equiv \phi$
3. Apply De Morgan equivalences:
   - $\neg(\phi \vee \psi) \equiv \neg\psi \wedge \neg\phi$
   - $\neg(\phi \wedge \psi) \equiv \neg\psi \vee \neg\phi$
4. Apply distributivity equivalence:
   - $\phi \vee (\psi \wedge \gamma) \equiv (\phi \vee \psi) \wedge (\phi \vee \gamma)$

Order of 2-4 does not matter!

Example of CNF transformation (1)

$\neg((A \to B) \wedge (B \leftrightarrow C))$

1 $\neg((A \to B) \wedge (B \leftrightarrow C)) \equiv \neg((\neg A \vee B) \wedge (B \leftrightarrow C))$

1 $\neg((\neg A \vee B) \wedge (B \leftrightarrow C)) \equiv \neg((\neg A \vee B) \wedge ((\neg B \vee C) \wedge (\neg C \vee B)))$

3 $\neg((\neg A \vee B) \wedge ((\neg B \vee C) \wedge (\neg C \vee B))) \equiv$
$\neg(\neg A \vee B) \vee \neg((\neg B \vee C) \wedge (\neg C \vee B))$

3 $\neg(\neg A \vee B) \vee \neg((\neg B \vee C) \wedge (\neg C \vee B)) \equiv$
$(\neg\neg A \wedge \neg B) \vee \neg((\neg B \vee C) \wedge (\neg C \vee B))$

2 $(\neg\neg A \wedge \neg B) \vee \neg((\neg B \vee C) \wedge (\neg C \vee B)) \equiv$
$(A \wedge \neg B) \vee \neg((\neg B \vee C) \wedge (\neg C \vee B))$

3 $(A \wedge \neg B) \vee \neg((\neg B \vee C) \wedge (\neg C \vee B)) \equiv$
$(A \wedge \neg B) \vee (\neg(\neg B \vee C) \vee \neg(\neg C \vee B))$

Example of CNF transformation (2)

3 $(A \wedge \neg B) \vee (\neg(\neg B \vee C) \vee \neg(\neg C \vee B)) \equiv$
$(A \wedge \neg B) \vee ((\neg\neg B \wedge \neg C) \vee \neg(\neg C \vee B))$

2 $(A \wedge \neg B) \vee ((\neg\neg B \wedge \neg C) \vee \neg(\neg C \vee B)) \equiv$
$(A \wedge \neg B) \vee ((B \wedge \neg C) \vee \neg(\neg C \vee B))$

3 $(A \wedge \neg B) \vee ((B \wedge \neg C) \vee \neg(\neg C \vee B)) \equiv$
$(A \wedge \neg B) \vee ((B \wedge \neg C) \vee (\neg\neg C \wedge \neg B))$

2 $(A \wedge \neg B) \vee ((B \wedge \neg C) \vee (\neg\neg C \wedge \neg B)) \equiv$
$(A \wedge \neg B) \vee ((B \wedge \neg C) \vee (C \wedge \neg B))$

So far so good!

With the exception of 1, we are reducing the size of the formula

Example of CNF transformation (3)

4 $(A \wedge \neg B) \vee ((B \wedge \neg C) \vee (C \wedge \neg B)) \equiv$
$(A \vee ((B \wedge \neg C) \vee (C \wedge \neg B))) \wedge (\neg B \vee ((B \wedge \neg C) \vee (C \wedge \neg B)))$

4 $(A \vee ((B \wedge \neg C) \vee (C \wedge \neg B))) \wedge (\neg B \vee ((B \wedge \neg C) \vee (C \wedge \neg B))) \equiv$
$(A \vee (((B \wedge \neg C) \vee C) \wedge ((B \wedge \neg C) \vee \neg B))) \wedge (\neg B \vee ((B \wedge \neg C) \vee (C \wedge \neg B)))$

4 $(A \vee (((B \wedge \neg C) \vee C) \wedge ((B \wedge \neg C) \vee \neg B))) \wedge (\neg B \vee ((B \wedge \neg C) \vee (C \wedge \neg B))) \equiv (A \vee (((B \vee C) \wedge (\neg C \vee C)) \wedge ((B \wedge \neg C) \vee \neg B))) \wedge (\neg B \vee ((B \wedge \neg C) \vee (C \wedge \neg B)))$

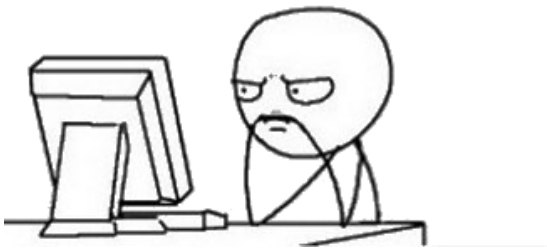
4 $(A \vee (((B \vee C) \wedge (\neg C \vee C)) \wedge ((B \wedge \neg C) \vee \neg B))) \wedge (\neg B \vee ((B \wedge \neg C) \vee (C \wedge \neg B))) \equiv (A \vee (((B \vee C) \wedge (\neg C \vee C)) \wedge ((B \vee \neg B) \wedge (\neg C \vee \neg B)))) \wedge (\neg B \vee ((B \wedge \neg C) \vee (C \wedge \neg B)))$

Example of CNF transformation (4)

4 $(A \vee (((B \vee C) \wedge (\neg C \vee C)) \wedge ((B \vee \neg B) \wedge (\neg C \vee \neg B)))) \wedge (\neg B \vee ((B \wedge \neg C) \vee (C \wedge \neg B))) \equiv ((A \vee ((B \vee C) \wedge (\neg C \vee C))) \wedge (A \vee ((B \vee \neg B) \wedge (\neg C \vee \neg B)))) \wedge (\neg B \vee ((B \wedge \neg C) \vee (C \wedge \neg B)))$

4 $((A \vee ((B \vee C) \wedge (\neg C \vee C))) \wedge (A \vee ((B \vee \neg B) \wedge (\neg C \vee \neg B)))) \wedge (\neg B \vee ((B \wedge \neg C) \vee (C \wedge \neg B))) \equiv (((A \vee (B \vee C)) \wedge (A \vee (\neg C \vee C))) \wedge (A \vee ((B \vee \neg B) \wedge (\neg C \vee \neg B)))) \wedge (\neg B \vee ((B \wedge \neg C) \vee (C \wedge \neg B)))$

4 $(((A \vee (B \vee C)) \wedge (A \vee (\neg C \vee C))) \wedge (A \vee ((B \vee \neg B) \wedge (\neg C \vee \neg B)))) \wedge (\neg B \vee ((B \wedge \neg C) \vee (C \wedge \neg B))) \equiv (((A \vee (B \vee C)) \wedge (A \vee (\neg C \vee C))) \wedge ((A \vee (B \vee \neg B)) \wedge (A \vee (\neg C \vee \neg B)))) \wedge (\neg B \vee ((B \wedge \neg C) \vee (C \wedge \neg B)))$

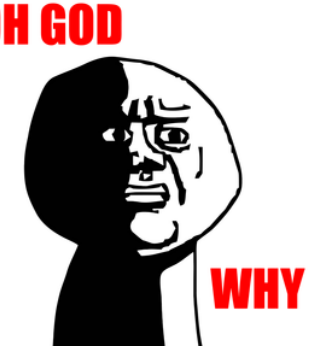


Example of CNF transformation (5)

4 $(((A \vee (B \vee C)) \wedge (A \vee (\neg C \vee C))) \wedge ((A \vee (B \vee \neg B)) \wedge (A \vee (\neg C \vee \neg B)))) \wedge (\neg B \vee ((B \wedge \neg C) \vee (C \wedge \neg B))) \equiv$
$(((A \vee (B \vee C)) \wedge (A \vee (\neg C \vee C))) \wedge ((A \vee (B \vee \neg B)) \wedge (A \vee (\neg C \vee \neg B)))) \wedge (\neg B \vee (((B \wedge \neg C) \vee C) \wedge ((B \wedge \neg C) \vee \neg B)))$

4 $(((A \vee (B \vee C)) \wedge (A \vee (\neg C \vee C))) \wedge ((A \vee (B \vee \neg B)) \wedge (A \vee (\neg C \vee \neg B)))) \wedge (\neg B \vee (((B \wedge \neg C) \vee C) \wedge ((B \wedge \neg C) \vee \neg B))) \equiv$
$(((A \vee (B \vee C)) \wedge (A \vee (\neg C \vee C))) \wedge ((A \vee (B \vee \neg B)) \wedge (A \vee (\neg C \vee \neg B)))) \wedge (\neg B \vee (((B \vee C) \wedge (\neg C \vee C)) \wedge ((B \wedge \neg C) \vee \neg B)))$

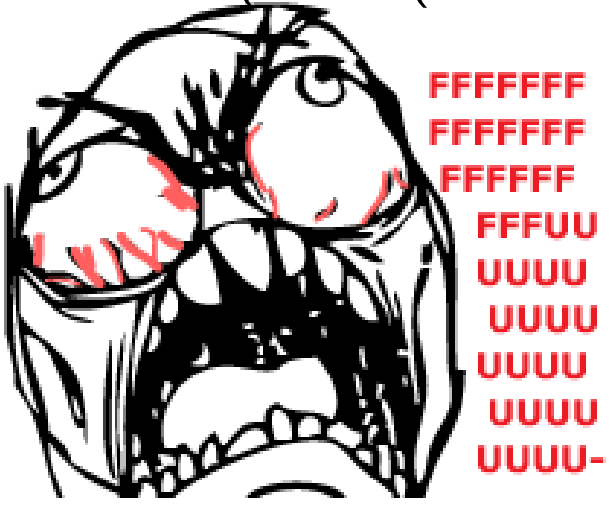
4 $(((A \vee (B \vee C)) \wedge (A \vee (\neg C \vee C))) \wedge ((A \vee (B \vee \neg B)) \wedge (A \vee (\neg C \vee \neg B)))) \wedge (\neg B \vee (((B \vee C) \wedge (\neg C \vee C)) \wedge ((B \wedge \neg C) \vee \neg B))) \equiv$
$(((A \vee (B \vee C)) \wedge (A \vee (\neg C \vee C))) \wedge ((A \vee (B \vee \neg B)) \wedge (A \vee (\neg C \vee \neg B)))) \wedge (\neg B \vee (((B \vee C) \wedge (\neg C \vee C)) \wedge ((B \vee \neg B) \wedge (\neg C \vee \neg B))))$



Example of CNF transformation (6)

4 $(((A \vee (B \vee C)) \wedge (A \vee (\neg C \vee C))) \wedge ((A \vee (B \vee \neg B)) \wedge (A \vee (\neg C \vee \neg B)))) \wedge (\neg B \vee (((B \vee C) \wedge (\neg C \vee C)) \wedge ((B \vee \neg B) \wedge (\neg C \vee \neg B)))) \equiv (((A \vee (B \vee C)) \wedge (A \vee (\neg C \vee C))) \wedge ((A \vee (B \vee \neg B)) \wedge (A \vee (\neg C \vee \neg B)))) \wedge ((\neg B \vee ((B \vee C) \wedge (\neg C \vee C))) \wedge (\neg B \vee ((B \vee \neg B) \wedge (\neg C \vee \neg B))))$

4 $(((A \vee (B \vee C)) \wedge (A \vee (\neg C \vee C))) \wedge ((A \vee (B \vee \neg B)) \wedge (A \vee (\neg C \vee \neg B)))) \wedge ((\neg B \vee ((B \vee C) \wedge (\neg C \vee C))) \wedge (\neg B \vee ((B \vee \neg B) \wedge (\neg C \vee \neg B)))) \equiv (((A \vee (B \vee C)) \wedge (A \vee (\neg C \vee C))) \wedge ((A \vee (B \vee \neg B)) \wedge (A \vee (\neg C \vee \neg B)))) \wedge (((\neg B \vee (B \vee C)) \wedge (\neg B \vee (\neg C \vee C))) \wedge (\neg B \vee ((B \vee \neg B) \wedge (\neg C \vee \neg B))))$



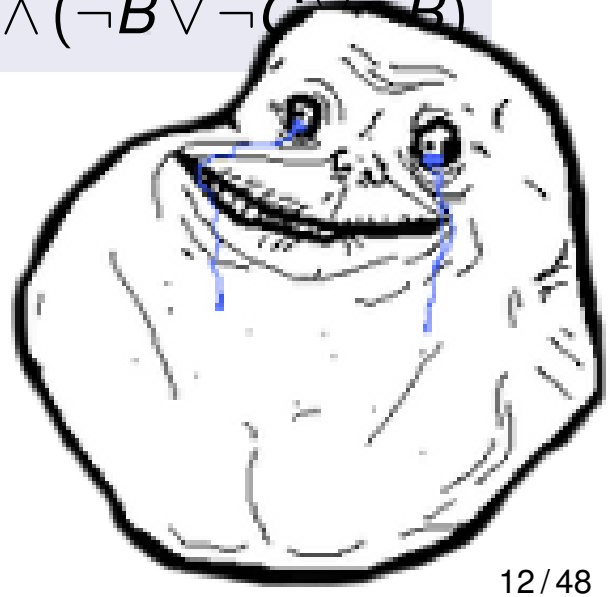
Example of CNF transformation (7)

4 $(((A \vee (B \vee C)) \wedge (A \vee (\neg C \vee C))) \wedge ((A \vee (B \vee \neg B)) \wedge (A \vee (\neg C \vee \neg B)))) \wedge (((\neg B \vee (B \vee C)) \wedge (\neg B \vee (\neg C \vee C))) \wedge (\neg B \vee ((B \vee \neg B) \wedge (\neg C \vee \neg B)))) \equiv (((A \vee (B \vee C)) \wedge (A \vee (\neg C \vee C))) \wedge ((A \vee (B \vee \neg B)) \wedge (A \vee (\neg C \vee \neg B)))) \wedge (((\neg B \vee (B \vee C)) \wedge (\neg B \vee (\neg C \vee C))) \wedge ((\neg B \vee (B \vee \neg B)) \wedge (\neg B \vee (\neg C \vee \neg B))))$

Flattening

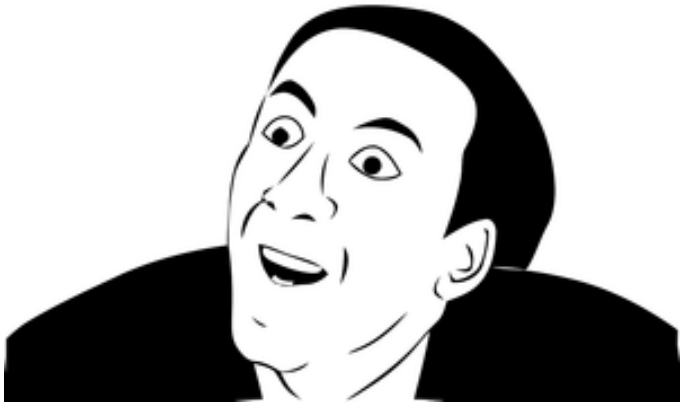
$(A \vee B \vee C) \wedge (A \vee \neg C \vee C) \wedge (A \vee B \vee \neg B) \wedge (A \vee \neg C \vee \neg B) \wedge (\neg B \vee B \vee C) \wedge (\neg B \vee \neg C \vee C) \wedge (\neg B \vee B \vee \neg B) \wedge (\neg B \vee \neg C \vee \neg B)$

Eliminate clauses containing A and $\neg A$ (those are equivalent to $\top$ and are hence neutral for $\wedge$)

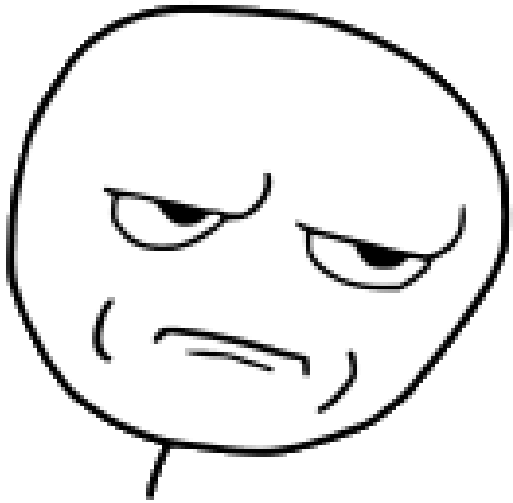


Be careful! This algorithm outputs formulas of exponential size in general

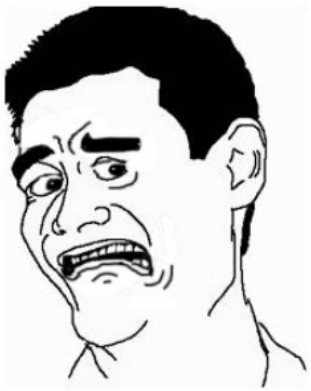
CNF transformation in practice (1)

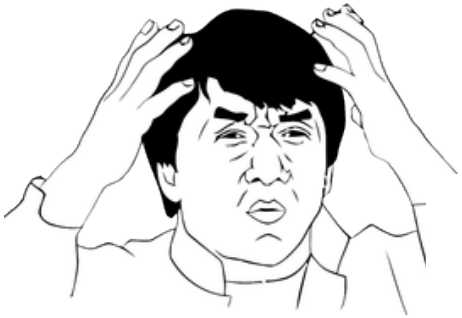


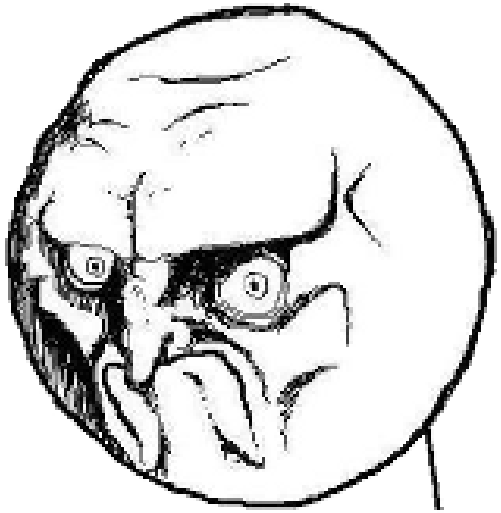
Yes, I say!



Seriously

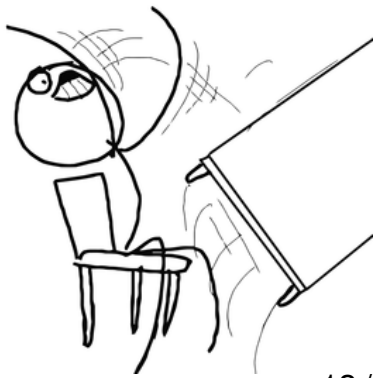






NO.

Trust me...



CNF transformation in practice (2)

Work bottom-up on the structure of the formula and build a set of clauses preserving only (un)satisfiability (not equivalence)

1. Replace each $L_1 \odot L_2$ (L_1, L_2 literals, $\odot \in \{\vee, \wedge, \rightarrow, \leftrightarrow\}$) by an auxiliary (fresh) variable X and introduce formula $X \leftrightarrow L_1 \odot L_2$, i.e.,

$(\odot = \vee)$	$\neg X \vee L_1 \vee L_2$	$[X \rightarrow L_1 \vee L_2]$
	$X \vee \neg L_1, X \vee \neg L_2$	$[L_1 \vee L_2 \rightarrow X]$
$(\odot = \wedge)$	$\neg X \vee L_1, \neg X \vee L_2$	$[X \rightarrow L_1 \wedge L_2]$
	$X \vee \neg L_1 \vee \neg L_2$	$[L_1 \wedge L_2 \rightarrow X]$
$(\odot = \rightarrow)$	$\neg X \vee \neg L_1 \vee L_2$	$[X \rightarrow (L_1 \rightarrow L_2)]$
	$X \vee L_1, X \vee \neg L_2$	$[(L_1 \rightarrow L_2) \rightarrow X]$
$(\odot = \leftrightarrow)$	$\neg X \vee \neg L_1 \vee L_2, \neg X \vee \neg L_2 \vee L_1$	$[X \rightarrow (L_1 \leftrightarrow L_2)]$
	$X \vee \neg L_1 \vee \neg L_2, X \vee L_1 \vee L_2$	$[(L_1 \leftrightarrow L_2) \rightarrow X]$

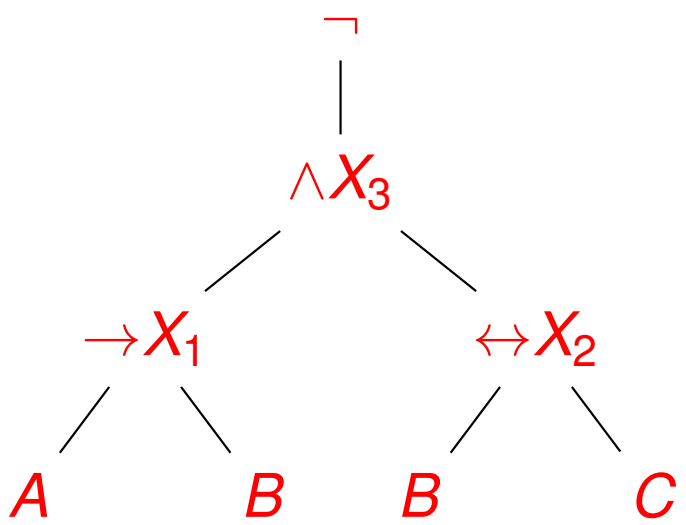
2. Apply double negation equivalences:
   - $\neg\neg\phi \equiv \phi$
3. Simplify $\top$, $\bot$

You may also apply De Morgan equivalences, but they are not necessary

And do not apply distributivity equivalence!

Example of CNF transformation (reloaded)

- We started with $\neg((A \to B) \wedge (B \leftrightarrow C))$



- $\neg X_1 \vee \neg A \vee B$
  $X_1 \vee A$, $X_1 \vee \neg B$
- $\neg X_2 \vee \neg B \vee C$,
  $\neg X_2 \vee \neg C \vee B$
  $X_2 \vee \neg B \vee \neg C$, $X_2 \vee B \vee C$
- $\neg X_3 \vee X_1$, $\neg X_3 \vee X_2$
  $X_3 \vee \neg X_1 \vee \neg X_2$
- $\neg X_3$

CNF representation

$$(L_{1_1} \vee \cdots \vee L_{m_1}) \wedge \ldots \wedge (L_{1_n} \vee \cdots \vee L_{m_n})$$

- It is clear where which connectives are
- Let us write the CNF as a set of clauses
  - Write clauses as sets of literals
  - Write CNFs as a set of sets of literals

$$\{\{L_{1_1}, \ldots, L_{m_1}\}, \ldots, \{L_{1_n}, \ldots, L_{m_n}\}\}$$

DNF transformation

1. Eliminate $\top$, $\bot$, $\rightarrow$, $\leftrightarrow$
2. Apply double negation equivalence:
   - $\neg\neg\phi \equiv \phi$
3. Apply De Morgan equivalences:
   - $\neg(\phi \vee \psi) \equiv \neg\phi \wedge \neg\psi$
   - $\neg(\phi \wedge \psi) \equiv \neg\phi \vee \neg\psi$
4. Apply distributivity equivalence:
   - $\phi \wedge (\psi \vee \gamma) \equiv (\phi \wedge \psi) \vee (\phi \wedge \gamma)$

Order of 2-4 does not matter!

Propositional resolution (1)

Main observation

$$(a \vee b) \wedge (\neg a \vee c) \equiv (a \vee b) \wedge (\neg a \vee c) \wedge (b \vee c)$$

$$(a \vee b) \wedge (\neg a \vee c) \models (b \vee c)$$

More general

$$C_1 \wedge \cdots \wedge C_k \wedge (a \vee L_1^1 \vee \ldots \vee L_n^1) \wedge (\neg a \vee L_1^2 \vee \ldots \vee L_m^2) \models (L_1^1 \vee \ldots \vee L_n^1 \vee L_1^2 \vee \ldots \vee L_m^2)$$

This is known as **resolution**!

Additional observation

$$(a \vee a \vee B) \equiv (a \vee B)$$

- Remove duplicated literals in clauses
- It is known as factorization

Resolution (set notation)

$$C_1, \ldots, C_k, \{a, L_1^1, \ldots, L_n^1\}, \{\neg a, L_1^2, \ldots, L_m^2\} \models \{L_1^1, \ldots, L_n^1, L_1^2, \ldots, L_m^2\}$$

Factorization comes “for free”!

Derivation by resolution (1)

Resolvent

Given two clauses C_1 and C_2 such that $a \in C_1$ and $\neg a \in C_2$, $(C_1 \setminus \{a\}) \cup (C_2 \setminus \{\neg a\})$ is the resolvent of C_1 and C_2.

Derivation

Given a set Γ of clauses, a derivation by resolution of a clause C from Γ, denoted $\Gamma \vdash_R C$, is a sequence $C_1, \ldots, C_n$ such that $C_n = C$ and for each C_i $(1 \leq i \leq n)$ we have

1. $C_i \in \Gamma$, or
2. C_i is a resolvent of C_j and C_k, where $j < i$ and $k < i$.

Lemma

If $\Gamma \vdash_R C$ then $\Gamma \models C$.

Proof. By induction on the sequence $C_1, \ldots, C_n$.

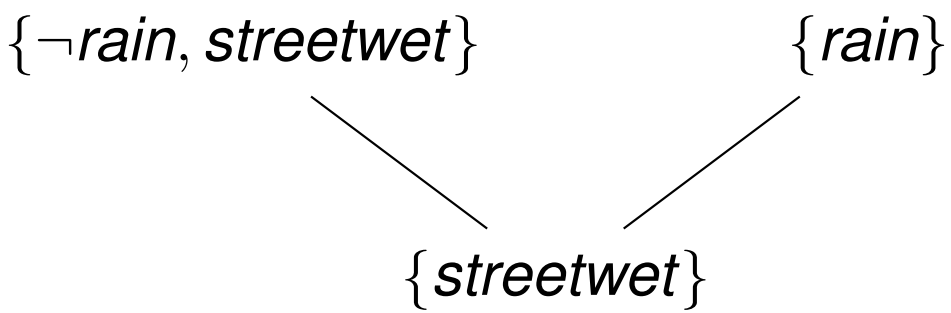
Derivation by resolution (2)

Consider

- $\Gamma = \{rain \rightarrow streetwet,\ rain\}$, or equivalently
- $\Gamma = \{\neg rain \vee streetwet,\ rain\}$, or equivalently
- $\Gamma = \{\{\neg rain, streetwet\},\ \{rain\}\}$

The following is a derivation by resolution:

- $C_1 = \{\neg rain, streetwet\}$
- $C_2 = \{rain\}$
- $C_3 = \{streetwet\}$



$$rain \rightarrow streetwet,\ rain \vdash_R streetwet$$

Refutation (1)

- Our goal is to model $\Gamma \models \bot$
- Let $\square$ be the empty clause
  - $\square$ is like $\bot$
  - $\square$ is different from an empty set of formulas!

Refutation

A derivation by resolution of $\square$ from Γ is called a refutation of Γ.

Resolution Theorem

$\Gamma \vdash_R \square$ if and only if Γ is unsatisfiable.

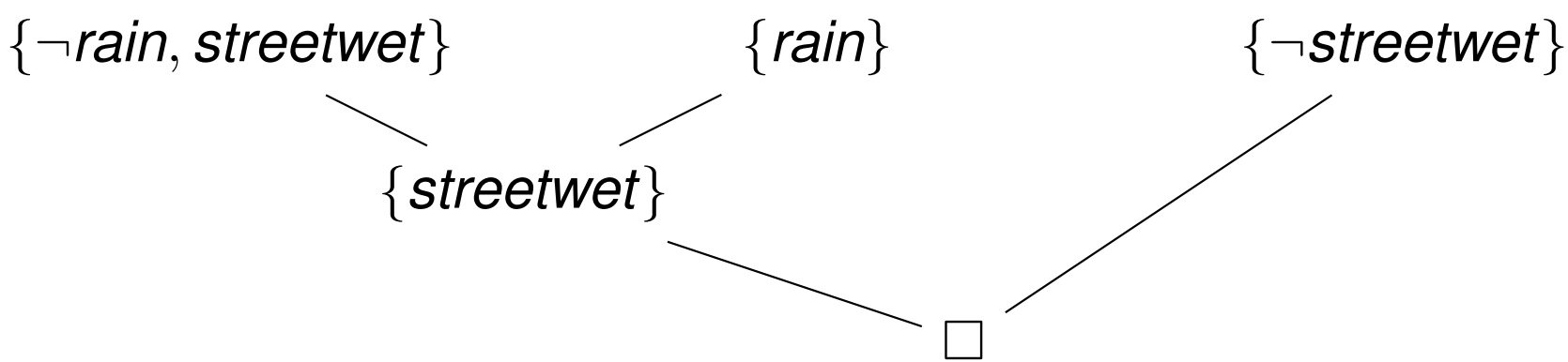
Proof. Soundness: $\Gamma \vdash_R \square$ implies $\Gamma \models \square$ (by the previous Lemma).
Completeness: by induction over the number of variables in Γ.

Refutation (2)

Goal: $\{rain \rightarrow streetwet,\ rain\} \models streetwet$
IFF: $\{rain \rightarrow streetwet,\ rain\} \cup \{\neg streetwet\} \models \bot$

The following is a refutation of
$\{rain \rightarrow streetwet,\ rain,\ \neg streetwet\}$:

- $C_1 = \{\neg rain, streetwet\}$
- $C_2 = \{rain\}$
- $C_3 = \{\neg streetwet\}$
- $C_4 = \{streetwet\}$
- $C_5 = \{\} = \square$



$$rain \rightarrow streetwet,\ rain,\ \neg streetwet \vdash_R \square$$

Validity, equivalence, entailment, ...

Validity

$\models \phi$ iff $\neg\phi \models \bot$

- Test whether $\neg\phi \vdash_R \square$

Equivalence

$\phi \equiv \psi$ iff $\neg(\phi \leftrightarrow \psi) \models \bot$

- Test whether $\neg(\phi \leftrightarrow \psi) \vdash_R \square$

Entailment

$\Gamma \models \phi$ iff $\Gamma,\ \neg\phi \models \bot$

- Test whether $\Gamma,\ \neg\phi \vdash_R \square$

Satisfiability

ϕ is satisfiable iff $\neg\phi$ is not valid

- Test whether $\phi \nvdash_R \square$

Implementing resolution (1)

Algorithm: SAT by resolution

Input : a set Γ of wffs
Output: true if Γ is SAT; false otherwise

```
1  begin
2      Γ^CNF := trasformToCNF(Γ);
3      repeat
4          if □ ∈ Γ^CNF then
5              return false;
6          Γ_old := Γ^CNF;
7          Γ^CNF := Γ^CNF ∪ resolveAll(Γ^CNF);
8      until Γ_old = Γ^CNF;
9      return true;
```

Function resolveAll(Γ^{CNF}: set of clauses)

```
1  begin
2      Γ_res := ∅;
3      foreach C1 ∈ Γ^CNF and foreach atom a ∈ C1 do
4          foreach C2 ∈ Γ^CNF such that ¬a ∈ C2 do
5              Γ_res := Γ_res ∪ {C1 \ {a} ∪ C2 \ {¬a}};
6      return Γ_res
```

Implementing resolution (2)

Complexity

Deciding $\Gamma \vdash_R \square$ requires up to an exponential number of steps (with respect to the size of the formula)

Since unsatisfiability of a formula is coNP-complete, this is "reasonable"

Example

Is the following formula satisfiable?

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

Refinements (1)

Drop tautological clauses

A clause C is a tautology if there is $a \in V$ such that $a \in C$ and $\neg a \in C$

Drop subsumed clauses

A clause C_1 subsumes a clause C_2 if $C_1 \subseteq C_2$

Linear Resolution

Any intermediate derivation uses the clause obtained in the previous step.

Theorem

Linear resolution is refutation complete: If a set of wffs is unsatisfiable then a refutation by linear resolution exists.

$$\{\{A, B\}, \{A, \neg B\}, \{\neg A, B\}, \{\neg A, \neg B\}\}$$

Refinements (2)

Linear Input Resolution

Any intermediate derivation uses the clause obtained in the previous step and a clause of the original formula.

Theorem

Linear input resolution is refutation complete for (sets of) Horn clauses, where a Horn clause is a clause containing at most one positive atom.

Example

1. $\{\{A, B\}, \{A, \neg B\}, \{\neg A, B\}, \{\neg A, \neg B\}\}$
2. $\{\{A\}, \{B\}, \{A, \neg B\}, \{\neg A, B\}, \{\neg A, \neg B\}\}$

Examples

1. Is $(A_1 \vee A_2) \wedge (\neg A_2 \vee \neg A_3) \wedge (A_3 \vee A_4) \wedge (\neg A_4 \vee \neg A_1)$ satisfiable?
2. Does A follow from $(A \vee B \vee C) \wedge (\neg C \vee B) \wedge (A \vee \neg B)$?
3. Does $\neg A$ follow from $(A \vee B \vee C) \wedge (\neg C \vee B) \wedge (A \vee \neg B)$?
4. Does $A \wedge B$ follow from $(\neg A \rightarrow B) \wedge (A \rightarrow B) \wedge (\neg A \rightarrow \neg B)$?

DPLL algorithm

- By $\neg\ell$ we denote the opposite literal of ℓ
  - if $\ell = \neg a$ then $\neg\ell = a$
- Simplify is often called **Unit Propagation**
  - Call-by-name parameters (references)

Algorithm: DPLL

Input : a set Γ of clauses
Output: true if Γ is SAT; false otherwise

```
1 begin
2     Simplify(Γ);
3     if Γ = ∅ then return true ;
4     if □ ∈ Γ then return false ;
5     ℓ := ChooseLiteral(Γ);
6     return DPLL(Γ ∪ {{ℓ}}) or DPLL(Γ ∪ {{¬ℓ}});
```

Procedure Simplify(Γ)

```
1 begin
2     while {ℓ} ∈ Γ do
3         foreach c ∈ Γ do
4             if ℓ ∈ c then Γ := Γ \ {c} ;
5             else if ¬ℓ ∈ c then Γ := (Γ \ {c}) ∪ {(c \ {¬ℓ})} ;
```

Properties

- DPLL(Γ) returns true if Γ is satisfiable, and false otherwise
- DPLL(Γ) can be (easily) modified in order to compute one (or all) models of Γ
  - DPLL is sound and complete
- DPLL(Γ) works in polynomial-space

Features

Simplification

The input set of clauses is simplified at each branch using (at least) unit clause propagation

Branching

When no further simplification is possible, a literal is selected using some heuristic criterion (ChooseLiteral) and assumed as a unit clause in the current set of clauses

Backtracking

When a contradiction (empty clause) arises, the search resumes from some previous assumption ℓ by assuming $\neg\ell$ instead

Examples (1)

Example

$\Gamma = \{\{x_1, x_2, \neg x_3\}, \{\neg x_2\}, \{x_4, \neg x_3\}\}$

- Simplify using $\{\neg x_2\}$: $\Gamma = \{\{x_1, \neg x_3\}, \{x_4, \neg x_3\}\}$
- ChooseLiteral returns $\neg x_3$: $\Gamma = \emptyset$, i.e., the formula is SAT!

Example

$\Gamma = \{\{x_1, x_2, \neg x_3, \neg x_4\}, \{\neg x_2\}, \{x_4, \neg x_3\}\}$

- Simplify using $\{\neg x_2\}$: $\Gamma = \{\{x_1, \neg x_3, \neg x_4\}, \{x_4, \neg x_3\}\}$
- If ChooseLiteral returns $\neg x_3$, the process is the same as before; otherwise, if x_1 is returned, another choice has to be made

Branching order (ChooseLiteral) can make big differences!

Examples (2)

1. $\{\{x_1, x_2, x_3\}, \{x_1, x_2, \neg x_3\}, \{x_1, \neg x_2, x_3\}, \{x_1, \neg x_2, \neg x_3\}, \{\neg x_1, x_4\}, \{x_1, \neg x_4, \neg x_5, x_6\}, \{\neg x_1, x_7\}\}$
2. $\{\{x_1, x_2\}, \{x_1, \neg x_2\}, \{\neg x_1, x_2\}, \{\neg x_1, \neg x_2\}\}$

Refinements

- How to efficiently detect unit clauses?
  - 2-watched literals
- How to implement ChooseLiteral?
  - Look-ahead heuristics
  - Look-back heuristics
- How to take advantage from conflicts?
  - Learning
  - Backjumping
- Can we reuse something from a previous computation?
  - Progressive SAT

Exercises (1)

(From *Logic for Computer Science: Foundations of Automatic Theorem Proving*)

1. Show that the following set of clauses are unsatisfiable using the resolution method:
   1. $\{\{A, B, \neg C\}, \{A, B, C\}, \{A, \neg B\}, \{\neg A\}\}$
   2. $\{\{A, \neg B, C\}, \{B, C\}, \{\neg A, C\}, \{B, \neg C\}, \{\neg B\}\}$
   3. $\{\{A, \neg B\}, \{A, C\}, \{\neg B, C\}, \{\neg A, B\}, \{B, \neg C\}, \{\neg A, \neg C\}\}$
   4. $\{\{A, B\}, \{\neg A, B\}, \{A, \neg B\}, \{\neg A, \neg B\}, \}$
2. Find all resolvents of the following pairs of clauses:
   1. $\{A, B\}, \{\neg A, \neg B\}$
   2. $\{A, \neg B\}, \{B, C, D\}$
   3. $\{\neg A, B, \neg C\}, \{B, C\}$
   4. $\{A, \neg A\}, \{A, \neg A\}$
3. Find all resolvents of the following sets of clauses:
   1. $\{\{A, \neg B\}, \{A, B\}, \{\neg A\}\}$
   2. $\{\{A, B, C\}, \{\neg B, \neg C\}, \{\neg A, \neg C\}\}$
   3. $\{\{\neg A, \neg B\}, \{B, C\}, \{\neg C, A\}\}$
   4. $\{\{A, B, C\}, \{A\}, \{B\}\}$

Exercises (2)

1. Show using resolution whether the following statements hold:
   1. $x \vee y \vee \neg z \models (x \vee z) \leftrightarrow (\neg y \rightarrow x)$
   2. $((\neg X \vee \neg Y) \rightarrow \neg(\neg Y \vee X))$ is satisfiable

Exercises (3)

(From *Logic for Computer Science: Foundations of Automatic Theorem Proving*)

1. Show that the following set of clauses are unsatisfiable using the DPLL algorithm:
   1. $\{\{A, B, \neg C\}, \{A, B, C\}, \{A, \neg B\}, \{\neg A\}\}$
   2. $\{\{A, \neg B, C\}, \{B, C\}, \{\neg A, C\}, \{B, \neg C\}, \{\neg B\}\}$
   3. $\{\{A, \neg B\}, \{A, C\}, \{\neg B, C\}, \{\neg A, B\}, \{B, \neg C\}, \{\neg A, \neg C\}\}$
   4. $\{\{A, B\}, \{\neg A, B\}, \{A, \neg B\}, \{\neg A, \neg B\}, \}$
2. Show using DPLL whether the following statements hold:
   1. $x \vee y \vee \neg z \models (x \vee z) \leftrightarrow (\neg y \rightarrow x)$
   2. $((\neg X \vee \neg Y) \rightarrow \neg(\neg Y \vee X))$ is satisfiable

Exercises (3)

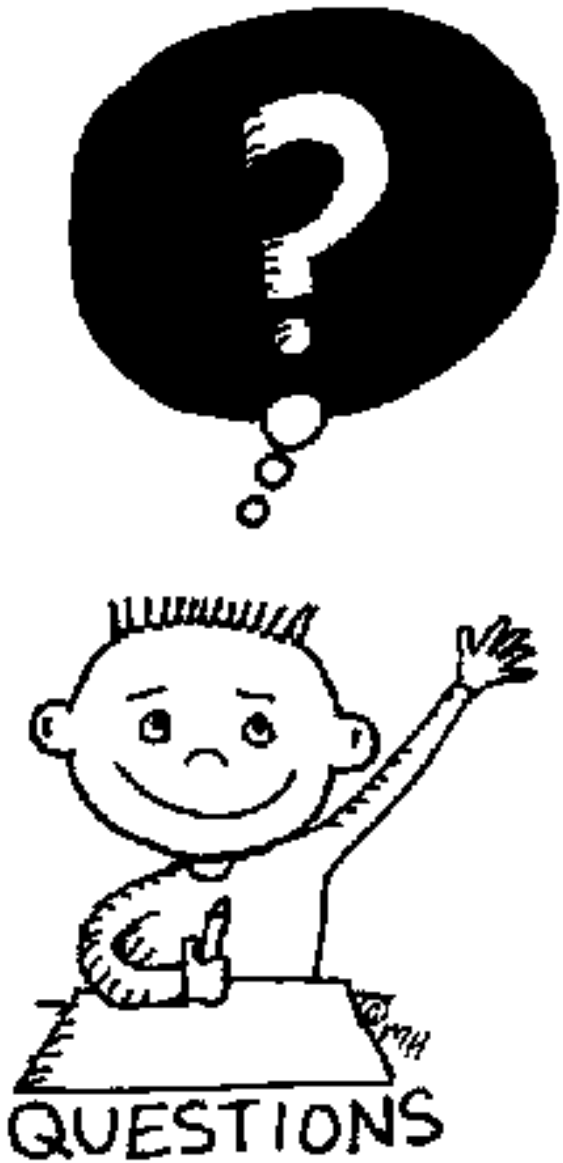
1. Find formulas in CNF and DNF having the following truth table:

A	B	C	D	ϕ
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1

A	B	C	D	ϕ
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

2. Decide whether the following formula is satisfiable:

$$A_1 \wedge (\neg A_1 \vee \neg A_2) \wedge (A_2 \vee A_3) \wedge (\neg A_3 \vee \neg A_4) \wedge (A_4 \vee A_5)$$



END OF THE LECTURE