Propositional Logic
Resolution and DPLL

Mario Alviano

University of Calabria, Italy

A.A. 2013/2014

Outline

1 Introduction

2 Propositional resolution
- Resolution
- Refutations
- Refinements and examples

3 DPLL

4 Theories with infinitely many formulas

5 Exercises

Satisfiability

- Validity, equivalence, entailment, ...
- Can be reduced to satisfiability testing (SAT)
  - Long tradition
  - Cook's Theorem implies that very efficient methods are unlikely to exist
  - But one can try to be as efficient as possible!

Satisfiability by truth tables

```
Algorithm: SAT by truth table
Input   : a set Γ of wffs
Output: true if Γ is SAT; false otherwise
begin
    foreach interpretation I do
        if evaluate(Γ,I) then
            return true;
    return false
```

- Simple method
- Usually quite inefficient
- Works on the semantic level (by trying interpretations)

Syntactic method

Observation

For all unsatisfiable sets Γ of wffs:

$$\Gamma \equiv \bot$$

- Can we find a transformation which takes Γ to $\bot$ iff Γ is unsatisfiable?
- Can we use a normal form such as CNF?

Goal

1. Start with any set Γ of wffs
2. Transform (in linear time) to Γ^{CNF}
3. Transform to $\bot$ iff Γ is unsatisfiable

Propositional resolution (1)

Main observation

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

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

More general

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

This is known as **resolution**!

Propositional resolution (2)

Additional observation

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

- Remove duplicated literals in clauses (factorization)

Resolution

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

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 $C_1, \ldots, C_n$.

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:

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

$$\begin{array}{ccc} \{\neg rain, streetwet\} & & \{rain\} \\ & \{streetwet\} & \end{array}$$

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

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

Refutation

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

Resolution Theorem

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

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

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

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

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

$$
\begin{tikzcd}
\{\neg rain, streetwet\} \arrow[dr, dash] & & \{rain\} \arrow[dl, dash] & & \{\neg streetwet\} \arrow[ddl, dash] \\
& \{streetwet\} \arrow[drr, dash] & & & \\
& & & \Box &
\end{tikzcd}
$$

$$rain \to streetwet,\ rain,\ \neg streetwet \vdash_R \Box$$

Validity, equivalence, entailment, ...

Validity

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

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

Equivalence

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

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

Entailment

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

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

Satisfiability

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

- Test whether $\phi \nvdash_R \Box$

Implementing resolution (1)

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

begin
  $\Gamma_{res} := \emptyset$;
  foreach $C_1 \in \Gamma^{CNF}$ **do**
    foreach $C_2 \in \Gamma^{CNF}$ **do**
      $\Gamma_{res} := \Gamma_{res} \cup \{C1 \setminus \{a\} \cup C2 \setminus \{\neg a\}\}$;
  return Γ_{res}

Algorithm: SAT by resolution

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

begin
  $\Gamma^{CNF} := \mathit{trasformToCNF}(\Gamma)$;
  repeat
    if $\Box \in \Gamma^{CNF}$ **then**
      return *false*;
    $\Gamma_{old} := \Gamma^{CNF}$;
    $\Gamma^{CNF} := \Gamma^{CNF} \cup \mathit{resolveAll}(\Gamma^{CNF})$;
  until $\Gamma_{old} = \Gamma^{CNF}$;
  return *true*;

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.

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

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 $P = A \wedge B$ follow from $(\neg A \to B) \wedge (A \to B) \wedge (\neg A \to \neg B)$?

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

Procedure Simplify(Γ)

```
begin
    while {ℓ} ∈ Γ do
        foreach c ∈ Γ do
            if ℓ ∈ c then Γ := Γ \ {c} ;
            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

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

Theories with infinitely many formulas

Compactness Theorem

A set Γ of wffs is satisfiable if and only if
each finite subset of Γ is satisfiable.

Corollary

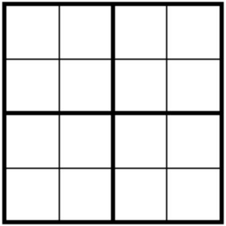
A set Γ of wffs is unsatisfiable if and only if
there exists a finite subset of Γ which is unsatisfiable.

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

1. Show that the following set of clauses are unsatisfiable using the resolution method and 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. 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. $\{\{A, B, C\}, \{A\}, \{B\}\}$

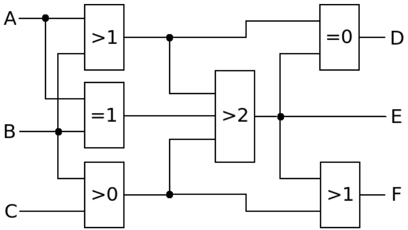
1. Show using resolution and DPLL whether the following statements hold
   1. $x \lor y \lor \neg z \models (x \lor z) \leftrightarrow (\neg y \to x)$
   2. $((\neg X \lor \neg Y) \to \neg(\neg Y \lor X))$ is satisfiable
2. Write a CNF formula generator for the Bishop Independence Problem

1. Represent a 4×4 Sudoku with 2×2 squares using a propositional logic formula. This means that there is a 4×4 grid of fields, each of which should be filled with exactly one number between 1 and 4. The grid is divided into 4 non-overlapping regions of dimension 2×2.



2. Generalize the formula for grids of $n^2 \times n^2$ fields, into each of which a number between 1 and n^2 should be written. The grid is divided into n^2 non-overlapping square regions of dimension $n \times n$. Here n is an arbitrary positive integer (so the previous example was a special case for $n = 2$).

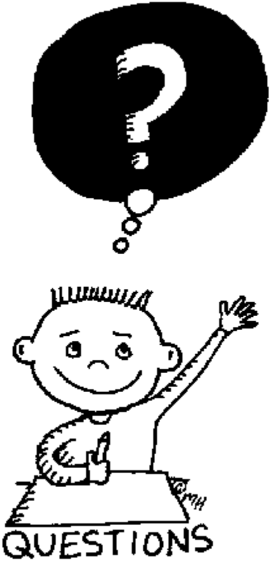
Represent the following Boolean circuit using a set of wffs:



Each component of the circuit has inputs at its left-hand side and output at its right-hand side. The output of a component is 1 if and only if the sum of its inputs satisfies the condition written inside the component; so if the sum of the inputs does not satisfy the condition, the output is 0. The inputs and outputs of the circuit (A, B, C, D, E, F) may have values 1 or 0. Bifurcations are indicated using a dot; crossing wires without a dot.

The modelled formula must have models that correspond exactly to the admissible input and output values of the circuit.

1. In the previous exercise, how can you find out whether the value of F can ever be 1 in an admissible state of the circuit?
2. Whether the value of E can ever be equal to the value of A?
3. Whether the value of F is 1 if and only if the value of D is 0?



END OF THE LECTURE