Risoluzione Proposizionale
Propositional Resolution

Wolfgang Faber

University of Calabria, Italy

2007

Satisfiability

- Consequence and validity can be reduced to satisfiability testing (SAT).
- SAT: Long tradition
- Cook's Theorem
- Very efficient method unlikely.
- But one can try to be as efficient as possible!

Satisfiability – Truth Tables

- Construct Truth Tables
- Simple Method
- But usually quite inefficient
- Semantic level (try interpretations)

Satisfiability – Truth Tables

```
function sat_truthtable(φ: formula)
    {
    foreach interpretation I
        {
        if( evaluate(φ,I) == true)
            return true;
        }
    return false;
    }
```

Syntactic Method

- For all unsatisfiable formula ϕ: $\phi \equiv \perp$
- Find transformation which takes ϕ to $\perp$ for each unsatisfiable formula ϕ.
- Use a normal form (CNF).
- $\phi \Rightarrow \phi^{CNF} \Rightarrow \perp$ for unsatisfiable ϕ
- $\phi \Rightarrow \phi^{CNF} \Rightarrow \rho \neq \perp$ for satisfiable ϕ

Outline

Propositional Resolution

Main Observation:

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

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

Propositional Resolution

Main Observation:

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

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

Propositional Resolution

More general:

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

Resolution

Propositional Factorization

Additional observation:

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

Factorization

Propositional Factorization

Additional observation:

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

Factorization

Propositional Resolution

Formulas in CNF, write them as sets:

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

Factorization comes “for free”!

Propositional Resolution

Formulas in CNF, write them as sets:

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

Factorization comes “for free”!

Resolvent

Definition

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

Outline

Derivation

Definition

Given a set of clauses S, a derivation by resolution of a clause C from S is a sequence $C_1, \ldots, C_n$, such that $C_n = C$ and for each C_i $(0 \leq i \leq n)$ we have

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

If a derivation by resolution of C from S exists, we write $S \vdash_R C$.

Derivation

Definition

Given a set of clauses S, a derivation by resolution of a clause C from S is a sequence $C_1, \ldots, C_n$, such that $C_n = C$ and for each C_i $(0 \leq i \leq n)$ we have

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

If a derivation by resolution of C from S exists, we write $S \vdash_R C$.

Example Derivation

$$(rain \rightarrow streetwet) \land rain$$
$$(\neg rain \lor streetwet) \land (rain)$$
$$\{\{\neg rain, streetwet\}, \{rain\}\}$$

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

$$(rain \rightarrow streetwet) \land rain \vdash_R streetwet$$

Example Derivation

$$(rain \rightarrow streetwet) \wedge rain$$
$$(\neg rain \vee streetwet) \wedge (rain)$$
$$\{\{\neg rain, streetwet\}, \{rain\}\}$$

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

$$(rain \rightarrow streetwet) \wedge rain \vdash_R streetwet$$

Example Derivation

$$(rain \rightarrow streetwet) \wedge rain$$
$$(\neg rain \vee streetwet) \wedge (rain)$$
$$\{\{\neg rain, streetwet\}, \{rain\}\}$$

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

$$(rain \rightarrow streetwet) \wedge rain \vdash_R streetwet$$

Example Derivation

$$(rain \rightarrow streetwet) \wedge rain$$
$$(\neg rain \vee streetwet) \wedge (rain)$$
$$\{\{\neg rain, streetwet\}, \{rain\}\}$$

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

$$(rain \rightarrow streetwet) \wedge rain \vdash_R streetwet$$

Example Derivation

$$(\mathit{rain} \rightarrow \mathit{streetwet}) \wedge \mathit{rain}$$
$$(\neg \mathit{rain} \vee \mathit{streetwet}) \wedge (\mathit{rain})$$
$$\{\{\neg \mathit{rain}, \mathit{streetwet}\}, \{\mathit{rain}\}\}$$

$$\begin{aligned} C_1 &= \{\neg \mathit{rain}, \mathit{streetwet}\} \\ C_2 &= \{\mathit{rain}\} \\ C_3 &= \{\mathit{streetwet}\} \end{aligned}$$

$$(\mathit{rain} \rightarrow \mathit{streetwet}) \wedge \mathit{rain} \vdash_R \mathit{streetwet}$$

Resolution

Theorem

If $S \vdash_R C$, then $S \models C$.

Proof.

Prove that $\{C_1, C_2\} \models C$ for two clauses C_1, C_2 and their resolvent C: Case distinction over the pair of resolved literals. $\square$

Outline

Empty Clause

Definition

Let $\square$ be the empty clause.

$\square$ is like $\perp$. $\square$ is different from an empty CNF!

Empty Clause

Definition

Let $\Box$ be the empty clause.

$\Box$ is like $\bot$. $\Box$ is different from an empty CNF!

Empty Clause

Definition

Let $\square$ be the empty clause.

$\square$ is like $\perp$. $\square$ is different from an empty CNF!

Refutation

Definition

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

Example Refutation

$$(rain \rightarrow streetwet) \wedge rain \wedge \neg streetwet$$
$$(\neg rain \vee streetwet) \wedge (rain) \wedge (\neg streetwet)$$
$$\{\{\neg rain, streetwet\}, \{rain\}, \{\neg streetwet\}\}$$

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

$$(rain \rightarrow streetwet) \wedge rain \wedge \neg streetwet \vdash_R \Box$$

Example Refutation

$$(rain \rightarrow streetwet) \wedge rain \wedge \neg streetwet$$
$$(\neg rain \vee streetwet) \wedge (rain) \wedge (\neg streetwet)$$
$$\{\{\neg rain, streetwet\}, \{rain\}, \{\neg streetwet\}\}$$

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

$$(rain \rightarrow streetwet) \wedge rain \wedge \neg streetwet \vdash_R \square$$

Example Refutation

$$(rain \rightarrow streetwet) \wedge rain \wedge \neg streetwet$$
$$(\neg rain \vee streetwet) \wedge (rain) \wedge (\neg streetwet)$$
$$\{\{\neg rain, streetwet\}, \{rain\}, \{\neg streetwet\}\}$$

$$\begin{aligned} C_1 &= \{\neg rain, streetwet\} \\ C_2 &= \{rain\} \\ C_3 &= \{\neg streetwet\} \\ C_4 &= \{streetwet\} \\ C_5 &= \{\} = \square \end{aligned}$$

$$(rain \rightarrow streetwet) \wedge rain \wedge \neg streetwet \vdash_R \square$$

Example Refutation

$$(rain \rightarrow streetwet) \wedge rain \wedge \neg streetwet$$
$$(\neg rain \vee streetwet) \wedge (rain) \wedge (\neg streetwet)$$
$$\{\{\neg rain, streetwet\}, \{rain\}, \{\neg streetwet\}\}$$

$$\begin{aligned} C_1 &= \{\neg rain, streetwet\} \\ C_2 &= \{rain\} \\ C_3 &= \{\neg streetwet\} \\ C_4 &= \{streetwet\} \\ C_5 &= \{\} = \square \end{aligned}$$

$$(rain \rightarrow streetwet) \wedge rain \wedge \neg streetwet \vdash_R \square$$

Example Refutation

$$(rain \rightarrow streetwet) \wedge rain \wedge \neg streetwet$$
$$(\neg rain \vee streetwet) \wedge (rain) \wedge (\neg streetwet)$$
$$\{\{\neg rain, streetwet\}, \{rain\}, \{\neg streetwet\}\}$$

$$\begin{aligned} C_1 &= \{\neg rain, streetwet\} \\ C_2 &= \{rain\} \\ C_3 &= \{\neg streetwet\} \\ C_4 &= \{streetwet\} \\ C_5 &= \{\} = \square \end{aligned}$$

$$(rain \rightarrow streetwet) \wedge rain \wedge \neg streetwet \vdash_R \square$$

Example Refutation

$$(\mathit{rain} \rightarrow \mathit{streetwet}) \wedge \mathit{rain} \wedge \neg\mathit{streetwet}$$
$$(\neg\mathit{rain} \vee \mathit{streetwet}) \wedge (\mathit{rain}) \wedge (\neg\mathit{streetwet})$$
$$\{\{\neg\mathit{rain}, \mathit{streetwet}\}, \{\mathit{rain}\}, \{\neg\mathit{streetwet}\}\}$$

$$\begin{aligned} C_1 &= \{\neg\mathit{rain}, \mathit{streetwet}\} \\ C_2 &= \{\mathit{rain}\} \\ C_3 &= \{\neg\mathit{streetwet}\} \\ C_4 &= \{\mathit{streetwet}\} \\ C_5 &= \{\} = \square \end{aligned}$$

$$(\mathit{rain} \rightarrow \mathit{streetwet}) \wedge \mathit{rain} \wedge \neg\mathit{streetwet} \vdash_R \square$$

Resolution

Theorem

$S \vdash_R \square$ if and only if S is unsatisfiable.

Proof.

Soundness follows from $S \models \square$.

Completeness by induction over the number of variables in the formula. $\square$

Resolution

Theorem

$S \vdash_R \Box$ if and only if S is unsatisfiable.

Proof.

Soundness follows from $S \models \Box$.
Completeness by induction over the number of variables in the formula. $\Box$

Resolution

- Validity of F: Test whether $\neg F \vdash_R \square$.
- Satisfiability of F: Test whether $F \nvdash_R \square$.
- Entailment of G by F ($F \models G$): Test whether $F \wedge \neg G \vdash_R \square$.

Resolution

- Validity of F: Test whether $\neg F \vdash_R \square$.
- Satisfiability of F: Test whether $F \nvdash_R \square$.
- Entailment of G by F ($F \models G$): Test whether $F \wedge \neg G \vdash_R \square$.

Resolution

- Validity of F: Test whether $\neg F \vdash_R \square$.
- Satisfiability of F: Test whether $F \nvdash_R \square$.
- Entailment of G by F ($F \models G$): Test whether $F \wedge \neg G \vdash_R \square$.

Outline

Resolve All Clauses

function resolve_all(F: cnf)
{
foreach $C1 \in F$
foreach $C2 \in F$
foreach $a \in V$ such that $a \in C1 \wedge \neg a \in C2$
$F := F \cup \{C1 \setminus \{a\} \cup C2 \setminus \{\neg a\}\}$;
return F;
}

This could be parallelized.

Resolve All Clauses

function resolve_all(F: cnf)
{
foreach $C1 \in F$
foreach $C2 \in F$
foreach $a \in V$ such that $a \in C1 \land \neg a \in C2$
$F := F \cup \{C1 \setminus \{a\} \cup C2 \setminus \{\neg a\}\}$;
return F;
}

This could be parallelized.

SAT via Resolution

```
function sat_resolution_breadth_first(ϕ: formula)
    {
    cnf F := transform_to_cnf(ϕ);
    cnf Fold;
    repeat
        {
        Fold := F;
        F := resolve_all(F);
        if( □ ∈ F )
            return false;
        }
    until( F == Fold );
    return true;
    }
```

Complexity

- Deciding $F \vdash_R \Box$ requires up to an exponential number of steps (with respect to the size of the formula).
- Since unsatisfiability of a formula is $co - NP$ complete, this is "reasonable".

Complexity

- Deciding $F \vdash_R \square$ requires up to an exponential number of steps (with respect to the size of the formula).
- Since unsatisfiability of a formula is $co - NP$ complete, this is "reasonable".

Example

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

Simple Refinements

- Drop tautological clauses (i.e. clauses C for which $\exists a \in V : a \in C \wedge \neg a \in C$).
- Drop subsumed clauses (clause C_1 subsumes clause C_2 if $C_1 \subseteq C_2$).

Linear Resolution

- Linear Resolution: Any intermediate derivation uses a clause obtained in the previous step.

Theorem

Linear resolution is refutation complete; i.e. if a formula is unsatisfiable, a refutation by linear resolution exists.

Example

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

Linear Input Resolution

- Linear Input Resolution: Any intermediate derivation uses a clause obtained in the previous step and a clause of the original formula.

Definition

A clause is called Horn clause if it contains at most one positive atom. A formula in CNF is a Horn formula if it contains only Horn clauses.

Theorem

Linear input resolution is refutation complete for Horn formulas; i.e. if a Horn formula is unsatisfiable, a refutation by linear input resolution exists.

Examples

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

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

Examples

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

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

Infinite CNFs

Theorem (Compactness)

An infinite set of clauses is satisfiable if and only if each finite subset is satisfiable.

Theorem (Compactness)

An infinite set of clauses is unsatisfiable if and only if there exists a finite subset, which is unsatisfiable.

Infinite CNFs

Theorem (Compactness)

An infinite set of clauses is satisfiable if and only if each finite subset is satisfiable.

Theorem (Compactness)

An infinite set of clauses is unsatisfiable if and only if there exists a finite subset, which is unsatisfiable.