CS 357: Advanced Topics in Formal Methods
Fall 2019

Lecture 16

Aleksandar Zeljić
Stanford University

Proofs

- Why proofs?
- What do we prove?
- What is the proof engine of SAT solvers?

Resolution Proof System

- Axioms: Clauses of the formula
- Inference rule:

$$\frac{c \vee l \qquad\qquad d \vee \neg l}{c \vee d}\ \textit{Resolution}$$

- Refutation ends with derivation of an empty clause - $\square$

Example

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

Example

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

Two representations:

- Annotated list
- DAG

Resolution complexity

- Number of clauses in a refutation is its size/length.
- Length of refuting ϕ - length of the shortest refutation
- Yields lower bound on the solving time using CDCL
- Upper bound: $\exp(O(N))$
- Lower bound: $\exp(\Omega(N))$

Known provably exponential classes

Pigeon-hole principle formulas / Dirichlet's box principle

- Place N+1 pigeons into N holes. No hole may hold more than one pigeon.
- Variables: $p_{i,j}$ - pigeon i belongs to hole j
- Every pigeon gets a hole

$$p_{i,1} \vee p_{i,2} \vee \ldots \vee p_{i,j}, \qquad \forall i \in \{1, 2, \ldots, N+1\}$$

- Every hole gets at most one pigeon

$$\neg p_{i,j} \vee \neg p_{i',j}, \qquad \forall i, i' \in \{1, 2 \ldots, N+1\}, \forall j \in \{1, 2 \ldots, N\}$$

PHP

Adding extra axioms:

- Functionality axioms - no pigeon gets two holes:

$$\neg p_{i,j} \vee \neg p_{i,j'}, \qquad \forall j, j' \in \{1, 2, \ldots, N+1\}$$

- Onto axioms - every hole gets a pigeon:

$$p_{1,j} \vee p_{2,j} \vee \ldots \vee p_{N+1,j}, \qquad \forall i \in \{1, 2, \ldots, N+1\}$$

Does not help - Resolution cannot count

Many other examples - Random k-CNF, Tseitin graphs, etc.

SAT solvers expect more

Extended Resolution [Tseitin]

- *extension rule* + resolution rule
- Extension:

$$x := a \wedge b \equiv (x \vee \neg a \vee \neg b) \wedge (\neg x \vee a) \wedge (\neg x \vee b)$$

- Exponentially stronger system than just resolution
- No known results on exponential lower bounds

- Pre-/in-processing steps are challenging to capture
- Not compact enough, keeps deriving consequences

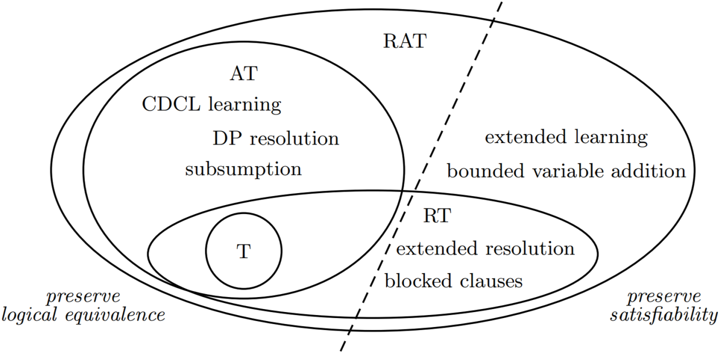
Redundancy-based clausal proofs

Proof:

Sequence of *clauses*, ending with the empty clause, that are redundant w.r.t. ϕ

- Allows addition and deletion of *redundant* clauses
- All derivations satisfy efficiently checkable syntactic criterion
- DRAT is the de facto standard nowadays
- Equivalent to Extended Resolution

Hierarchy of Redundant properties



Classes of redundant properties:

T - Tautology:

$$(p \vee \neg p)$$

Classes of redundant properties:

T - Tautology:

$$(p \vee \neg p)$$

AT - Asymmetric tautology

- $ALA(\phi, C)$ - Asymmetric literal addition, repeat until fix-point:

$$\exists (C \vee l) \in \phi \backslash \{C\} \quad \text{then} \quad C := C \vee \neg l$$

- *AT* - $ALA(\phi, C)$ has property T
- a.k.a. RUP - *reverse unit propagation*:

$$\square \in BCP(\phi, \neg C)$$

Classes of redundant properties:

RT - Resolution Tautology (a.k.a. blocked clauses):

1. $C = (l_1 \vee l_2 \vee ... \vee l_n \vee l)$ has property T or
2. exists $l \in C$ s.t. for each clause $C' \in \phi : \neg l \in C'$, every resolvent of C and C' over l has property T

Classes of redundant properties:

RT - Resolution Tautology (a.k.a. blocked clauses):

1. $C = (l_1 \vee l_2 \vee ... \vee l_n \vee l)$ has property T or
2. exists $l \in C$ s.t. for each clause $C' \in \phi : \neg l \in C'$, every resolvent of C and C' over l has property T

RAT - Resolution Asymmetric Tautology

1. $C = (l_1 \vee l_2 \vee ... \vee l_n \vee l)$ has property AT or
2. exists $l \in C$ s.t. for each clause $C' \in \phi : \neg l \in C'$, every resolvent of C and C' over l has property AT

Example

$$\phi : (a \vee b) \wedge (b \vee c) \wedge (\neg b \vee \neg c)$$

Which redundant properties have the following clauses:

- $a \vee \neg a$

Example

$$\phi : (a \vee b) \wedge (b \vee c) \wedge (\neg b \vee \neg c)$$

Which redundant properties have the following clauses:

- $a \vee \neg a$ *T (AT, RT, RAT)*
- $a \vee \neg c$

Example

$$\phi : (a \vee b) \wedge (b \vee c) \wedge (\neg b \vee \neg c)$$

Which redundant properties have the following clauses:

- $a \vee \neg a$ — *T (AT, RT, RAT)*
- $a \vee \neg c$ — *AT, RT, (RAT)*
- $\neg a \vee c$

Example

$$\phi : (a \vee b) \wedge (b \vee c) \wedge (\neg b \vee \neg c)$$

Which redundant properties have the following clauses:

- $a \vee \neg a$ *T (AT, RT, RAT)*
- $a \vee \neg c$ *AT, RT, (RAT)*
- $\neg a \vee c$ *RAT*

  Note: For RAT, all resolvents over **one literal** should have AT property, this is the case for resolvents over a

DRAT - Deletion RAT

- RAT clauses are expressive enough!
- Adding RAT clauses preserves *satisfiability*
- Deleting RAT clause preserves *unsatisfiability*
- Clauses are *efficiently* checkable (using BCP)
- Overall process is still expensive
- Allows trimming of formulas
- Optimized proofs
- Pythagorean Triples: 200TB resolution proof takes 67GB in DRAT

Example

Consider the problem of:

Avoiding monochromatic solutions of the equation:

$$a + b = c, \quad \text{with } a < b < c,$$

while coloring the natural numbers with two colors.

- Smallest counter-example: $\{1, 2, 3, \ldots, 9\}$
- Encode into sat using 9 variables:

$$v_i = \begin{cases} T, & \text{if } i \text{ is red} \\ F, & \text{if } i \text{ is blue} \end{cases} \qquad i \in \{1, 2, \ldots, 9\}$$

Example

```
p cnf 9 32
   1  2  3 0      -1 -2 -3 0
   1  3  4 0      -1 -3 -4 0
   1  4  5 0      -1 -4 -5 0
   2  3  5 0      -2 -3 -5 0
   1  5  6 0      -1 -5 -6 0
   2  4  6 0      -2 -4 -6 0
   1  6  7 0      -1 -6 -7 0
   2  5  7 0      -2 -5 -7 0
   3  4  7 0      -3 -4 -7 0
   1  7  8 0      -1 -7 -8 0
   2  6  8 0      -2 -6 -8 0
   3  5  8 0      -3 -5 -8 0
   1  8  9 0      -1 -8 -9 0
   2  7  9 0      -2 -7 -9 0
   3  6  9 0      -3 -6 -9 0
   4  5  9 0      -4 -5 -9 0
```

Example

```
p cnf 9 32
  1  2  3 0      -1 -2 -3 0
  1  3  4 0      -1 -3 -4 0
  1  4  5 0      -1 -4 -5 0
  2  3  5 0      -2 -3 -5 0
  1  5  6 0      -1 -5 -6 0
  2  4  6 0      -2 -4 -6 0
  1  6  7 0      -1 -6 -7 0
  2  5  7 0      -2 -5 -7 0
  3  4  7 0      -3 -4 -7 0
  1  7  8 0      -1 -7 -8 0
  2  6  8 0      -2 -6 -8 0
  3  5  8 0      -3 -5 -8 0
  1  8  9 0      -1 -8 -9 0
  2  7  9 0      -2 -7 -9 0
  3  6  9 0      -3 -6 -9 0
  4  5  9 0      -4 -5 -9 0
```

DRAT proof:

```
1 4 0
  1 0
  4 0
    0
```

Example

```
p cnf 9 32
  1  2  3 0      -1 -2 -3 0
  1  3  4 0      -1 -3 -4 0
  1  4  5 0      -1 -4 -5 0
  2  3  5 0      -2 -3 -5 0
  1  5  6 0      -1 -5 -6 0
  2  4  6 0      -2 -4 -6 0
  1  6  7 0      -1 -6 -7 0
  2  5  7 0      -2 -5 -7 0
  3  4  7 0      -3 -4 -7 0
  1  7  8 0      -1 -7 -8 0
  2  6  8 0      -2 -6 -8 0
  3  5  8 0      -3 -5 -8 0
  1  8  9 0      -1 -8 -9 0
  2  7  9 0      -2 -7 -9 0
  3  6  9 0      -3 -6 -9 0
  4  5  9 0      -4 -5 -9 0
```

DRAT proof:

```
1 4 0
  1 0
  4 0
    0
```

- 512 possible partitions
- 4 line proof

Unsat cores

Unsatisiable core of formula ϕ

A subset of ϕ that is still unsatisafiable.

- A core is *minimal* if removing any conjunct turns it satisfiable.
- How can we extract unsat cores?
- How can we minimize them?
- What can they be used for?
- In practice: `https://rise4fun.com/Z3/smtc_core`

Craig interpolation

Craig interpolant:

Suppose formula $\alpha \wedge \beta$ is unsatisafiable. There exists a formula I over literal in both α and β s.t.:

1. $\alpha \rightarrow I$ and
2. $I \wedge \beta$ is unsatisafiable.

Craig interpolation

Craig interpolant:

Suppose formula $\alpha \wedge \beta$ is unsatisafiable. There exists a formula I over literal in both α and β s.t.:

1. $\alpha \rightarrow I$ and
2. $I \wedge \beta$ is unsatisafiable.

- Explanation genralization / Conflict minimization
- In Model Checking: discovering relevant predicates and abstractions