Decision Procedures for Verification

Combinations of Decision Procedures (2)

23.01.2023

Viorica Sofronie-Stokkermans

sofronie@uni-koblenz.de

Overview

Until now:

- The Nelson-Oppen Combination Procedure

Today:

- Proofs: Termination, Soundness, Completeness (additional assumptions)
- $DPLL(\mathcal{T})$
- Applications

The Nelson-Oppen algorithm

ϕ conjunction of literals

Step 1. Purification $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \phi \mapsto (\mathcal{T}_1 \cup \phi_1) \cup (\mathcal{T}_2 \cup \phi_2)$:

where ϕ_i is a pure Σ_i-formula and $\phi_1 \wedge \phi_2$ is equisatisfiable with ϕ.

Step 2. Propagation.

The decision procedure for ground satisfiability for $\mathcal{T}_1$ and $\mathcal{T}_2$ fairly exchange information concerning entailed unsatisfiability of constraints in the shared signature

i.e. clauses over the shared variables.

until an inconsistency is detected or a saturation state is reached.

The Nelson-Oppen algorithm

ϕ conjunction of literals

Step 1. Purification $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \phi \mapsto (\mathcal{T}_1 \cup \phi_1) \cup (\mathcal{T}_2 \cup \phi_2)$:

where ϕ_i is a pure Σ_i-formula and $\phi_1 \wedge \phi_2$ is equisatisfiable with ϕ.

not problematic; requires linear time

Step 2. Propagation.

The decision procedure for ground satisfiability for $\mathcal{T}_1$ and $\mathcal{T}_2$ fairly exchange information concerning entailed unsatisfiability of constraints in the shared signature
i.e. clauses over the shared variables.

until an inconsistency is detected or a saturation state is reached.

not problematic; termination guaranteed
Sound: if inconsistency detected input unsatisfiable
Complete: under additional assumptions

Implementation

ϕ conjunction of literals

Step 1. Purification: $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \phi \mapsto (\mathcal{T}_1 \cup \phi_1) \cup (\mathcal{T}_2 \cup \phi_2)$,
where ϕ_i is a pure Σ_i-formula and $\phi_1 \wedge \phi_2$ is equisatisfiable with ϕ.

Step 2. Propagation: The decision procedure for ground satisfiability for $\mathcal{T}_1$ and $\mathcal{T}_2$ fairly exchange information concerning entailed unsatisfiability of constraints in the shared signature
i.e. clauses over the shared variables.

until an inconsistency is detected or a saturation state is reached.

How to implement Propagation?

Guessing: guess a maximal set of literals containing the shared variables; check it for $\mathcal{T}_i \cup \phi_i$ consistency.

Backtracking: identify disjunction of equalities between shared variables entailed by $\mathcal{T}_i \cup \phi_i$; make case split by adding some of these equalities to ϕ_1, ϕ_2. Repeat as long as possible.

The Nelson-Oppen algorithm

Termination: only finitely many shared variables to be identified

The Nelson-Oppen algorithm

Termination: only finitely many shared variables to be identified

Soundness: If procedure answers "unsatisfiable" then ϕ is unsatisfiable

Proof: Assume that ϕ is satisfiable. Then $\phi_1 \wedge \phi_2$ satisfiable.

- The procedure cannot answer "unsatisfiable" in Step 2.
- Let $(\mathcal{M}, \beta) \models \phi_1 \wedge \phi_2$. Assume that $(\mathcal{M}, \beta) \models \bigwedge_{(c_i,c_j)\in E} c_i \approx c_j \wedge \bigwedge_{(c_i,c_j)\notin E} c_i \not\approx c_j$

Then
$$(\mathcal{M}_{|\Sigma_1}, \beta) \models \phi_1 \wedge \bigwedge_{(c_i,c_j)\in E} c_i \approx c_j$$
$$(\mathcal{M}_{|\Sigma_2}, \beta) \models \phi_2 \wedge \bigwedge_{(c_i,c_j)\in E} c_i \approx c_j$$

Guessing: $\bigwedge_{(c_i,c_j)\in E} c_i \approx c_j \wedge \bigwedge_{(c_i,c_j)\notin E} c_i \not\approx c_j$ "satisfiable arrangement".

Backtracking: Procedure answers satisfiable on the corresponding branch.

The Nelson-Oppen algorithm

Termination: only finitely many shared variables to be identified

Soundness: If procedure answers "unsatisfiable" then ϕ is unsatisfiable

Completeness: Under additional hypotheses

Completeness

Example:

E_1	E_2
$f(g(x), g(y)) \approx x$	$k(x) \approx k(x)$
$f(g(x), h(y)) \approx y$	
non-trivial	non-trivial

$g(c) \approx h(c) \wedge k(c) \not\approx c$

$g(c) \approx h(c)$	$k(c) \not\approx c$
satisfiable in E_1	satisfiable in E_2

no equations between shared variables; Nelson-Oppen answers "satisfiable"

Completeness

Example:

E_1	E_2
$f(g(x), g(y)) \approx x$	$k(x) \approx k(x)$
$f(g(x), h(y)) \approx y$	
non-trivial	non-trivial

$g(c) \approx h(c) \wedge k(c) \not\approx c$

$g(c) \approx h(c)$	$k(c) \not\approx c$
satisfiable in E_1	satisfiable in E_2

no equations between shared variables; Nelson-Oppen answers "satisfiable"

A model of E_1 satisfies $g(c) \approx h(c)$ iff $\exists e \in A$ s.t. $g(e) = h(e)$.

Then, for all $a \in A$: $a = f_A(g(a), g(e)) = f_A(g(a), h(e)) = e$

$g(c) \approx h(c) \wedge k(c) \not\approx c$ unsatisfiable

Completeness

Another example

$\mathcal{T}_1$ theory admitting models of cardinality at most 2

$\mathcal{T}_2$ theory admitting models of any cardinality

$$f_1 \in \Sigma_1, f_2 \in \Sigma_2 \quad \text{such that} \quad \mathcal{T}_i \not\models \forall x, y \quad f_i(x) = f_i(y).$$

$$\phi \;=\; f_1(c_1) \not\approx f_1(c_2) \quad \wedge \quad f_2(c_1) \not\approx f_2(c_3) \quad \wedge \quad f_2(c_2) \not\approx f_2(c_3)$$

$$\phi_1 \;=\; f_1(c_1) \not\approx f_1(c_2) \qquad \phi_2 \;=\; f_2(c_1) \not\approx f_2(c_3) \;\wedge\; f_2(c_2) \not\approx f_2(c_3)$$

The Nelson-Oppen procedure returns "satisfiable"

$$\mathcal{T}_1 \cup \mathcal{T}_2 \models \forall x, y, z (f_1(x) \not\approx f_1(y) \;\wedge\; f_2(x) \not\approx f_2(z) \;\wedge\; f_2(y) \not\approx f_2(z) \\ \rightarrow (x \not\approx y \wedge x \not\approx z \wedge y \not\approx z))$$

$f_1(c_1) \not\approx f_1(c_2) \quad \wedge \quad f_2(c_1) \not\approx f_2(c_3) \quad \wedge \quad f_2(c_2) \not\approx f_2(c_3)$ unsatisfiable

Completeness

Cause of incompleteness

There exist formulae satisfiable in finite models of bounded cardinality

Solution: Consider stably infinite theories.

$\mathcal{T}$ is stably infinite iff for every quantifier-free formula ϕ
ϕ satisfiable in $\mathcal{T}$ iff ϕ satisfiable in an infinite model of $\mathcal{T}$.

Note: This restriction is not mentioned in [Nelson Oppen 1979];

introduced by Oppen in 1980.

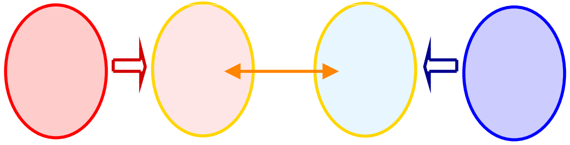
Completeness

Guessing version: C set of constants shared by ϕ_1, ϕ_2

R equiv. relation assoc. with partition of $C \mapsto ar(C, R) = \bigwedge_{R(c,d)} c \approx d \wedge \bigwedge_{\neg R(c,d)} c \not\approx d$

Lemma. Assume that there exists a partition of C s.t. $\phi_i \wedge ar(C, R)$ is $\mathcal{T}_i$-satisfiable. Then $\phi_1 \wedge \phi_2$ is $\mathcal{T}_1 \cup \mathcal{T}_2$-satisfiable.

Idea of proof: Let $\mathcal{A}_i \in \mathrm{Mod}(\mathcal{T}_i)$ s.t. $\mathcal{A}_i \models \phi_i \wedge ar(C, R)$. Then $c_{A_1} = d_{A_1}$ iff $c_{A_2} = d_{A_2}$.
Let $i : \{c_{A_1} \mid c \in C\} \to \{c_{A_2} \mid c \in C\}$, $i(c_{A_1}) = c_{A_2}$ well-defined; bijection.
Stable infinity: can assume w.l.o.g. that $\mathcal{A}_1, \mathcal{A}_2$ have the same cardinality
Let $h : \mathcal{A}_1 \to \mathcal{A}_2$ bijection s.t. $h(c_{A_1}) = c_{A_2}$



Use h to transfer the Σ_1-structure on $\mathcal{A}_2$.

Theorem. If $\mathcal{T}_1, \mathcal{T}_2$ are both stably infinite and the shared signature is empty then the Nelson-Oppen procedure is sound, complete and terminating. Thus, it transfers decidability of ground satisfiability from $\mathcal{T}_1, \mathcal{T}_2$ to $\mathcal{T}_1 \cup \mathcal{T}_2$.

Complexity

Main sources of complexity:

(i) transformation of the formula in DNF

(ii) propagation

(a) decide whether there is a disjunction of equalities between variables

(b) investigate different branches corresponding to disjunctions

Complexity

Main sources of complexity:

(i) transformation of the formula in DNF

(ii) propagation

$\mathcal{T}$ is convex	iff	for every quantifier-free formula ϕ,
		$\phi \models \bigvee_i x_i \approx y_i$ implies $\phi \models x_j \approx y_j$ for some j.

$\mapsto$ No branching

Complexity

Main sources of complexity:

(i) transformation of the formula in DNF

(ii) propagation

$\mathcal{T}$ is convex	iff	for every quantifier-free formula ϕ,
		$\phi \models \bigvee_i x_i \approx y_i$ implies $\phi \models x_j \approx y_j$ for some j.

$\mapsto$ No branching

Theorem.	Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be convex and stably infinite; $\Sigma_1 \cap \Sigma_2 = \emptyset$
	If satisfiability of conjunctions of literals in $\mathcal{T}_i$ is in PTIME
	Then satisfiability of conjunctions of literals in $\mathcal{T}_1 \cup \mathcal{T}_2$ is in PTIME

Complexity

In general: non-deterministic procedure

Theorem. Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be convex and stably infinite; $\Sigma_1 \cap \Sigma_2 = \emptyset$

If satisfiability of conjunctions of literals in $\mathcal{T}_i$ is in NP

Then satisfiability of conjunctions of literals in $\mathcal{T}_1 \cup \mathcal{T}_2$ is in NP

Extensions of the Nelson-Oppen procedure

- relax the stable infiniteness requirement

- relax the requirement that the theories have disjoint signatures

Extensions of the Nelson-Oppen procedure

- relax the stable infiniteness requirement

> [Tinelli,Zarba'03] One theory "shiny" (for each satisf. constraint we can compute a finite k s.t. the theory has models of every cardinality $\lambda \geq k$)
>
> [Fontaine'05] "Polite"; [Chocron,Fontaine,Ringeissen'14] "Gentle"

- relax the requirement that the theories have disjoint signatures

> [Tinelli,Ringeissen'03] Theories sharing absolutely free constructors
>
> [Ghilardi'04] Model theoretical conditions
>
> [Sofronie-Stokkermans, Ihlemann'10] Local extensions
>
> [Chocron, Fontaine, Ringeissen'16,'20] Theories with "bridging functions"

Main idea:

Find situations in which $\mathcal{T}_i$ models of $\phi_i, i = 1, 2$ can be "amalgamated" to a $\mathcal{T}_1 \cup \mathcal{T}_2$ model of $\phi_1 \wedge \phi_2$.

From conjunctions to arbitrary combinations

Until now:

check satisfiability for conjunctions of literals

Question:

how to check satisfiability of sets of clauses?

Overview

Satisfiability w.r.t. theories

- Propositional logic
  - resolution
  - DPLL
- First-order logic
  - resolution

- Ground formulae
  - conjunctions of literals: specialized methods
  - clauses: DPLL(T) $\Leftarrow$ TODAY
- Formulae with quantifiers
  - reduction to SAT for ground formulae instantiation $\Leftarrow$ NEXT WEEK (situations when sound and complete)
  - resolution (mod T)

3.6 The *DPLL*($\mathcal{T}$) algorithm

Reminder: Propositional SAT

The DPLL algorithm

A succinct formulation

State: $M||F$,

where:

- M partial assignment (sequence of literals),

  some literals are annotated (L^d: decision literal)

- F clause set.

A succinct formulation

UnitPropagation

$M||F, C \vee L \Rightarrow M, L||F, C \vee L$ if $M \models \neg C$, and L undef. in M

Decide

$M||F \Rightarrow M, L^d||F$ if L or $\neg L$ occurs in F, L undef. in M

Fail

$M||F, C \Rightarrow \text{Fail}$ if $M \models \neg C$, M contains no decision literals

Backjump

$$M, L^d, N||F \Rightarrow M, L'||F \quad \text{if} \begin{cases} \text{there is some clause } C \vee L' \text{ s.t.:} \\ F \models C \vee L', M \models \neg C, \\ L' \text{ undefined in } M \\ L' \text{ or } \neg L' \text{ occurs in } F. \end{cases}$$

Example

Assignment:	Clause set:	
$\emptyset$	$\|\|\neg P_1 \lor P_2, \neg P_3 \lor P_4, \neg P_5 \lor \neg P_6, P_6 \lor \neg P_5 \lor \neg P_2$	$\Rightarrow$ (Decide)
P_1^d	$\|\|\neg P_1 \lor P_2, \neg P_3 \lor P_4, \neg P_5 \lor \neg P_6, P_6 \lor \neg P_5 \lor \neg P_2$	$\Rightarrow$ (UnitProp)
$P_1^d P_2$	$\|\|\neg P_1 \lor P_2, \neg P_3 \lor P_4, \neg P_5 \lor \neg P_6, P_6 \lor \neg P_5 \lor \neg P_2$	$\Rightarrow$ (Decide)
$P_1^d P_2 P_3^d$	$\|\|\neg P_1 \lor P_2, \neg P_3 \lor P_4, \neg P_5 \lor \neg P_6, P_6 \lor \neg P_5 \lor \neg P_2$	$\Rightarrow$ (UnitProp)
$P_1^d P_2 P_3^d P_4$	$\|\|\neg P_1 \lor P_2, \neg P_3 \lor P_4, \neg P_5 \lor \neg P_6, P_6 \lor \neg P_5 \lor \neg P_2$	$\Rightarrow$ (Decide)
$P_1^d P_2 P_3^d P_4 P_5^d$	$\|\|\neg P_1 \lor P_2, \neg P_3 \lor P_4, \neg P_5 \lor \neg P_6, P_6 \lor \neg P_5 \lor \neg P_2$	$\Rightarrow$ (UnitProp)
$P_1^d P_2 P_3^d P_4 P_5^d \neg P_6$	$\|\|\neg P_1 \lor P_2, \neg P_3 \lor P_4, \neg P_5 \lor \neg P_6, P_6 \lor \neg P_5 \lor \neg P_2$	$\Rightarrow$ (Backtrack)
$P_1^d P_2 P_3^d P_4 \neg P_5$	$\|\|\neg P_1 \lor P_2, \neg P_3 \lor P_4, \neg P_5 \lor \neg P_6, P_6 \lor \neg P_5 \lor \neg P_2$	

DPLL with learning

The DPLL system with learning consists of the four transition rules of the Basic DPLL system, plus the following two additional rules:

Learn

$M||F \Rightarrow M||F, C$ if all atoms of C occur in F and $F \models C$

Forget

$M||F, C \Rightarrow M||F$ if $F \models C$

In these two rules, the clause C is said to be learned and forgotten, respectively.

SAT Modulo Theories (SMT)

Some problems are more naturally expressed in richer logics than just propositional logic, e.g:

- Software/Hardware verification needs reasoning about equality, arithmetic, data structures, ...

SMT consists of deciding the satisfiability of a ground 1st-order formula with respect to a background theory T

Example 1: $\mathcal{T}$ is Equality with Uninterpreted Functions (UIF):

$$f(g(a)) \not\approx f(c) \;\vee\; g(a) \approx d, \quad g(a) \approx c, \quad c \not\approx d$$

Example 2: for combined theories:

$$A \approx \mathrm{write}(B, a+1, 4), \quad \mathrm{read}(A, b+3) \approx 2 \;\vee\; f(a-1) \not\approx f(b+1)$$

SAT Modulo Theories (SMT)

The “very eager” approach to SMT

Method:

– translate problem into equisatisfiable propositional formula;

– use off-the-shelf SAT solver

- Why “eager”?

  Search uses all theory information from the beginning

- Characteristics:

  + Can use best available SAT solver
  – Sophisticated encodings are needed for each theory
  – Sometimes translation and/or solving too slow

Main Challenge for alternative approaches is to combine:

- DPLL-based techniques for handling the boolean structure

- Efficient theory solvers for conjunctions of $\mathcal{T}$-literals

SAT Modulo Theories (SMT)

"Lazy" approaches to SMT: Idea

Example: consider $\mathcal{T} = UIF$ and the following set of clauses:

$$\underbrace{f(g(a)) \not\approx f(c)}_{\neg P_1} \vee \underbrace{g(a) \approx d}_{P_2}, \quad \underbrace{g(a) \approx c}_{P_3}, \quad \underbrace{c \not\approx d}_{\neg P_4}$$

1. Send $\{\neg P_1 \vee P_2,\ P_3,\ \neg P_4\}$ to SAT solver

   SAT solver returns model $[\neg P_1, P_3, \neg P_4]$
   Theory solver says $\neg P_1 \wedge P_3 \wedge \neg P_4$ is $\mathcal{T}$-inconsistent

2. Send $\{\neg P_1 \vee P_2,\ P_3,\ \neg P_4,\ P_1 \vee \neg P_3 \vee P_4\}$ to SAT solver

   SAT solver returns model $[P_1, P_2, P_3, \neg P_4]$
   Theory solver says $P_1 \wedge P_2 \wedge P_3 \wedge \neg P_4$ is $\mathcal{T}$-inconsistent

3. Send $\{\neg P_1 \vee P_2, P_3, \neg P_4, P_1 \vee \neg P_3 \vee P_4, \neg P_1 \vee \neg P_2 \vee \neg P_3 \vee P_4\}$ to SAT solver

   SAT solver says UNSAT