Formal Methods and Specification (SS 2021)
Lecture 11: Abstraction, Decision Procedures

Stefan Ratschan

Katedra číslicového návrhu
Fakulta informačních technologií
České vysoké učení technické v Praze

Evropský sociální fond Praha & EU: Investujeme do vaší budoucnosti

Motivation

Methods already discussed need checks of

- verification conditions
- conditions for execution paths (symbolic execution)

Can, to a large extent, be done automatically.

Powerful solvers reason for success in recent years.

Prelude: Validity vs. Satisfiability

For example: real numbers

$$\models x + 1 \geq x$$

or, equivalently

$$\neg(x + 1 \geq x) \text{ that is } x + 1 < x \text{ unsatisfiable}$$

and, in general:

$$\models \phi$$

iff

formula $\neg\phi$ is unsatisfiable

Analogy:

Prove a formula ϕ

or, equivalently

assume $\neg\phi$ and find a contradiction

Intuition: satisfiability = solvability

Potential confusion:

we want to prove $\models \phi$, but algorithms usually decide satisfiability.

Example:

$Q \leftarrow \bot$
if $200 + 314x \geq 2 \wedge \mathrm{end\_of\_the\_world} > 2073$ **then**
  $Q \leftarrow \top$
if $[\neg[\mathrm{end\_of\_the\_world} > 2073] \vee 200 + 314x < 2] \wedge Q$ **then**
  explosion!

Two catastrophic execution paths:

1. skipping first **if**
2. executing first **if**

Check whether executable (see symbolic execution formulas):

$$\neg Q \wedge \neg[200 + 314x \geq 2 \wedge \mathrm{end\_of\_the\_world} > 2073] \wedge$$
$$[\neg[\mathrm{end\_of\_the\_world} > 2073] \vee 200 + 314x < 2] \wedge Q$$

$$\neg Q \wedge 200 + 314x \geq 2 \wedge \mathrm{end\_of\_the\_world} > 2073 \wedge Q_1 \wedge$$
$$[\neg[\mathrm{end\_of\_the\_world} > 2073] \vee 200 + 314x < 2] \wedge Q_1$$

Everything o.k., if both formulas unsat

Unsat Proof of First Formula

$Q \leftarrow \bot$
if $200 + 314x \geq 2 \wedge \mathrm{end\_of\_the\_world} > 2073$ **then**
  $Q \leftarrow \top$
if $[\neg[\mathrm{end\_of\_the\_world} > 2073] \vee 200 + 314x < 2] \wedge Q$ **then**
  explosion!

To prove unsatisfiability of

$$\neg Q \wedge \neg[200 + 314x \geq 2 \wedge \mathrm{end\_of\_the\_world} > 2073] \wedge$$
$$[\neg[\mathrm{end\_of\_the\_world} > 2073] \vee 200 + 314x < 2] \wedge Q$$

we assume this formula, which immediately results in a contradiction.

We only used Boolean constants, ignored everything else.

Unsat Proof of Second Formula

$Q \leftarrow \bot$
if $200 + 314x \geq 2 \wedge \mathrm{end\_of\_the\_world} > 2073$ **then**
    $Q \leftarrow \top$
if $[\neg[\mathrm{end\_of\_the\_world} > 2073] \vee 200 + 314x < 2] \wedge Q$ **then**
    explosion!

To prove unsatisfiability of

$$\neg Q \wedge 200 + 314x \geq 2 \wedge \mathrm{end\_of\_the\_world} > 2073 \wedge Q_1$$
$$[\neg[\mathrm{end\_of\_the\_world} > 2073] \vee 200 + 314x < 2] \wedge Q_1$$

we assume this formula, and try to arrive at a contradiction.

No contradiction from properties of Boolean constants alone.

Unsat Proof of Second Formula

$$\neg Q \land 200 + 314x \geq 2 \land \mathrm{end\_of\_the\_world} > 2073 \land Q_1 \land$$
$$[\neg[\mathrm{end\_of\_the\_world} > 2073] \lor 200 + 314x < 2] \land Q_1$$

We ignore details, but consider the Boolean structure

$$\neg Q \land S \land R \land Q_1 \land [\neg R \lor T] \land Q_1$$

leads to a contradiction?

satisfiable, satisfied by

$$\{Q \mapsto \bot, S \mapsto \top, R \mapsto \top, Q_1 \mapsto \top, T \mapsto \top\}$$

$200 + 314x \geq 2$, $\mathrm{end\_of\_the\_world} > 2073$, $200 + 314x < 2$

$$\neg Q \land S \land R \land Q_1 \land [\neg R \lor \neg S] \land Q_1$$

Unsat Proof of Second Formula

Assume $\neg Q \wedge S \wedge R \wedge Q_1 \wedge [\neg R \vee \neg S] \wedge Q_1$

$\neg Q, S, R, Q_1, [\neg R \vee \neg S], Q_1$

Case 1: $\neg Q, S, R, Q_1, \neg R, Q_1$ contradiction

Case 2: $\neg Q, S, R, Q_1, \neg S, Q_1$ contradiction

We only used Boolean structure, ignored everything else.

Summary

We started by ignoring details,
and incrementally used more and more properties:

1. properties of Boolean constants
2. properties of Boolean operations
3. $<$ is the negation of $\geq$

Ignoring details that are not relevant for the given problem:

abstraction

If abstraction is unsat then original formula is unsat,
but not the other way round

If abstraction is satisfiable, and we want to prove unsatisfiability,
we have to add additional properties

The same holds analogically, if we want to prove that a formula holds

Program Abstraction

Application of same principle directly to programs

$Q \leftarrow \bot$
if $S \wedge R$ **then**
    $Q \leftarrow \top$
if $[\neg R \vee T] \wedge Q$ **then**
    explosion!

Every execution of original program, has a
  corresponding execution of the abstracted program
    but not the other way round

For example, $\{S \mapsto \top, R \mapsto \top, T \mapsto \top\}$ leads to explosion.

etc.

Solvers for Propositional Logic

SAT: "satisfiability" (splnitelnost)

- Input: propositional logical formula, usually in conjunctive normal form
- Output:
  - satisfying assignment, if it exists,
  - `unsat`, if the formula is not satisfiable.

Example: $[\neg P \vee Q \vee R] \wedge [\neg Q \vee R] \wedge [\neg Q \vee \neg R] \wedge [P \vee Q]$

Reminder:

We can prove a formula by showing unsatisfiability of its negation.

NP-hard problem, but huge efficiency improvements in recent years

Today's solvers can solve huge formulas

Further Example

if x=1 **then**
  $y \leftarrow 1$
  if $\mathrm{adfxg7}(x) \neq \mathrm{adfxg7}(y)$ **then**
    @ $\perp$

$$x = 1 \wedge y = 1 \wedge \mathrm{adfxg7}(x) \neq \mathrm{adfxg7}(y) \quad \text{satisfiable?}$$

Boolean abstraction?

$$P \wedge Q \wedge R$$

satisfied by

$$\{P \mapsto \top, Q \mapsto \top, R \mapsto \top\}$$

We have to use knowledge about equality,
but we can abstract from the precise behavior of functions

Example Continued

We want to check

$$x = 1 \land y = 1 \land \text{adfxg7}(x) \neq \text{adfxg7}(y)$$

After abstraction

$$a = c \land b = c \land f(a) \neq f(b)$$

This implies

$$f(c) \neq f(c)$$

which is in contradiction with the law of reflexivity.

So, using the law of reflexivity we can show that the abstraction is unsat

Equality

What do we know about $=$?

- $\forall x \,.\, x = x$ (reflexivity)
- $\forall x, y \,.\, x = y \Rightarrow y = x$ (symmetry)
- $\forall x, y, z \,.\, [x = y \wedge y = z] \Rightarrow x = z$ (transitivity)
- for every function symbol f of arity n,
  $\forall x_1, \dots, x_n \,.\, [x_1 = y_1, \dots, x_n = y_n] \Rightarrow f(x_1, \dots, x_n) = f(y_1, \dots, y_n)$

first three axioms: equivalence relation

all axioms: congruence relation

result: *theory of free function symbols*

Example [Bradley and Manna, 2007]:
How to prove $\neg[f(a, b) = a \wedge f(f(a, b), b) \neq a]$?
Show unsatisfiability of negation, that is, add

$$f(a, b) = a \wedge f(f(a, b), b) \neq a$$

to assumptions and find a contradiction.

Congruence Closure Algorithm [Nelson and Oppen, 1980]

$$f(a,b) = a \wedge f(f(a,b),b) \neq a$$

$=$ is an equivalence relation, it defines equivalence classes on sub-terms.

instead of individual equalities we work with whole equivalence classes

Initial assumption: every sub-term is in a separate class:

$$\{\{a\},\{b\},\{f(a,b)\},\{f(f(a,b),b)\}\}$$

from $f(a,b) = a$:

$$\{\{a, f(a,b)\},\{b\},\{f(f(a,b),b)\}\}$$

congruence propagation: from $\{a, f(a,b)\}$:
$f(f(a,b),b)$ and $f(a,b)$ in same class:

$$\{\{a, f(a,b), f(f(a,b),b)\},\{b\}\}$$

Contradiction with disequality! This implies unsatisfiability.

Congruence Closure Algorithm [Nelson and Oppen, 1980]

- Input: formula $s_1 = t_1 \wedge \cdots \wedge s_m = t_m \wedge s_{m+1} \neq t_{m+1} \wedge \cdots \wedge s_n \neq t_n$
- Output: `unsatisfiable`, `satisfiable`
  (in the theory of free function symbols)

Let τ be the set of sub-terms of

$$s_1 = t_1 \wedge \cdots \wedge s_m = t_m \wedge s_{m+1} \neq t_{m+1} \wedge \cdots \wedge s_n \neq t_n$$

$\sim \leftarrow$ equivalence relation on τ s.t. for all $u, v \in \tau$, $u \neq v$ implies $u \not\sim v$

while there is $i \in \{1 \ldots m\}$, $u, v \in \tau$ s.t. $u \not\sim v$,
    $u[s_i \leftarrow t_i] = v$ or $u[t_i \leftarrow s_i] = v$ **do**
  merge equivalence classes of u and v in $\sim$
  while there is $u, v \in \tau$, $p, q \in \tau$, s.t. $u \not\sim v$, $p \sim q$,
      $u[p \leftarrow q] = v$ or $u[q \leftarrow p] = v$ **do**
    merge equivalence classes of u and v in $\sim$
if there is an $i \in \{m+1, \ldots, n\}$ s.t. $s_i \sim t_i$ **then**
  return `unsatisfiable`
else
  return `satisfiable`

General Formulas: Disjunctions, Quantifiers

A formula $\phi_1 \vee \cdots \vee \phi_n$ is satisfiable iff
one of the formulas $\phi_1, \ldots, \phi_n$ is satisfiable.

Example: To prove

$$a = a \wedge [a = b \Rightarrow f(a) = f(b)]$$

we show unsatisfiability of

$$\neg[a = a \wedge [a = b \Rightarrow f(a) = f(b)]]$$

that is, unsatisfiability of

$$a \neq a \vee [a = b \wedge f(a) \neq f(b)]$$

that is, unsatisfiability of both $a \neq a$ and $a = b \wedge f(a) \neq f(b)$.

Hence: formulas with conjunctions and disjunctions:
disjunctive normal form

General case with quantifiers: undecidable

Using the Theory of Free Function Symbols

Example: proving unsatisfiability of

$$a[x] = 0 \wedge x = y \wedge a[y] = 1$$

Abstracted formula:

$$f(a, x) = c \wedge x = y \wedge f(a, y) = d$$

Sub-terms: $\{a, c, d, x, y, f(a, x), f(a, y)\}$

Result, satisfiable, equivalence classes: $\{a\}, \{x, y\}, \{c, d, f(a, x), f(a, y)\}$

Refinement of abstraction:

$$f(a, x) = c \wedge x = y \wedge f(a, y) = d \wedge c \neq d$$

Further example: pointers

Application to data structures:
  abstract from detailed properties,
    and try a proof just using equality
      i.e., in the theory of free function symbols

Generalization, Further Theories

Quantifiers, axiom of equality of arrays

[Bradley et al., 2006]

[Bradley and Manna, 2007]

General case: undecidable

Theory	Description	Full	QFF
T_{E}	equality	no	yes
T_{PA}	Peano arithmetic	no	no
$T_{\mathbb{N}}$	Presburger arithmetic	yes	yes
$T_{\mathbb{Z}}$	linear integers	yes	yes
$T_{\mathbb{R}}$	reals (with $\cdot$)	yes	yes
$T_{\mathbb{Q}}$	rationals (without $\cdot$)	yes	yes
T_{RDS}	recursive data structures	no	yes
T_{RDS}^{+}	acyclic recursive data structures	yes	yes
T_{A}	arrays	no	yes
$T_{\mathsf{A}}^{=}$	arrays with extensionality	no	yes

Combination of theories [Nelson and Oppen, 1979]

Witnesses and Their Usage

Decision procedures sometimes also return satisfying assignments
(*a witness for satisfiability*)

For example, in the theory of integers:

Input: $x + y \geq 0 \wedge x - 2y \leq 1$
Output: `sat`, $\{x \mapsto -1, y \mapsto 1\}$

This allows automatic execution of I/O specifications:

Spec:

- Input: $I(x)$
- Output: y s.t. $O(x, y)$

For given input a, corresponding output:
witness of satisfiability of $I(a) \wedge O(a, y)$

How to ensure efficiency of automatic execution of specifications?

Real Numbers

Real numbers:

- satisfiability of conjunctions of linear equations: Gaussian elimination
- satisfiability of conjunctions of linear equations and inequalities: simplex algorithm

Sometimes it is even possible to compute something in undecidable cases:

`http://rsolver.sourceforge.net`

Summary

Levels of abstraction:

- Boolean: SAT solvers
- equality between function symbols: congruence closure
- individual theories: decision procedures for arrays, real numbers etc.

In theory (decidability) level of individual theories would be enough

In practice, choice of level very important.

SAT modulo theory (SMT): tight integration:

- SAT-solver
- solver for individual theories

Examples: CVC4, z3, openSAT

Open Research Questions

Only 20-30 years ago: paper and pencil.

Today: speed, speed, speed, speed!

https://smt-comp.github.io/

Conclusion

Basis for automatization of software verification: decision procedures

If one chooses the right level of abstraction,
it is possible to prove properties of extremely complex systems.

Even the source code of a program is
an abstraction of the resulting system:

The compiler may be incorrect, the operating system, or
we might even have the end of the world

Literature I

Aaron Bradley and Zohar Manna. *The calculus of computation*. Springer, 2007.

Aaron Bradley, Zohar Manna, and Henny Sipma. What's decidable about arrays? In *Verification, Model Checking, and Abstract Interpretation*, volume 3855 of *Lecture Notes in Computer Science*, pages 427–442. Springer Berlin / Heidelberg, 2006.

Greg Nelson and Derek C. Oppen. Simplification by cooperating decision procedures. *ACM Trans. Program. Lang. Syst.*, 1(2):245–257, 1979. ISSN 0164-0925. doi: http://doi.acm.org/10.1145/357073.357079.

Greg Nelson and Derek C. Oppen. Fast decision procedures based on congruence closure. *J. ACM*, 27(2):356–364, April 1980. doi: 10.1145/322186.322198.