Proving Theorems and Verifying Programs Automatically

Applied Logic for Computer Science

UWO – December 3, 2017

Plan

1. Introduction to SMT solving
2. Using Yices for checking assertions
3. Equality Reasoning
4. Theory Reasoning

Plan

1. Introduction to SMT solving
2. Using Yices for checking assertions
3. Equality Reasoning
4. Theory Reasoning

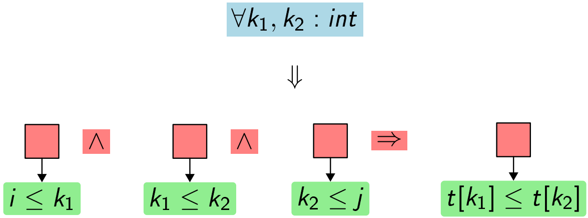
A logical formula . . .

$$sorted(t, i, j) =$$

$$\forall k_1, k_2 : int \;.\; i \leq k_1 \;\wedge\; k_1 \leq k_2 \;\wedge\; k_2 \leq j \Rightarrow t[k_1] \leq t[k_2]$$

...as seen by an SMT solver

$sorted(t, i, j) =$



- Instantiation
- Logic reasoning
- Theory reasoning (here: Arithmetic)

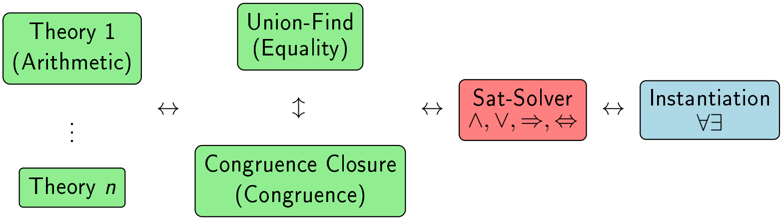
Satisfiability Modulo Theories

SMT provers divide the problem in three parts

- The theory part: equality reasoning, arithmetic reasoning, ...
- The satisfiability part: deals with logical connectors $\wedge$, $\vee$, $\Rightarrow$, $\neg$,...
- The instantiation of quantified axioms

We will look at each of the three parts in turn

The different parts of an SMT solver



A more detailed example

Hypotheses

- $H_1 : a > 0$
- $H_2 : \forall xy. x \geq y \rightarrow max(x, y) = x$

Goal

$G : f(max(a, 0)) = f(a)$

Solved by an SMT Solver (1)

Negate the Goal

$H_1 \wedge H_2 \rightarrow G$ becomes $H_1 \wedge H_2 \wedge \neg G$

Launch Sat-Solver

Assume H_1, H_2 and $\neg G$ and try to derive a contradiction

- Assume the inequality $a > 0$
- Register the lemma: $\forall xy . x \geq y \rightarrow max(x, y) = x$
- Assume the inequality $f(max(a, 0)) \neq f(a)$
- Currently no contradiction!

Instantiation

Specialize the lemma by applying it to a and 0 and replace $\rightarrow$:

$a \geq 0 \rightarrow max(a, 0) = a \quad \Leftrightarrow \quad a < 0 \vee max(a, 0) = a$

Solved by an SMT Solver (2)

Split the disjunction

First assume $a < 0$, then assume $\neg(a < 0)$, try to find a contradiction in both cases

Assuming $a < 0$

Direct contradiction with H_1 (using knowledge about the symbols $<$ and $\geq$)

Assuming $\neg(a < 0)$

- It follows $max(a, 0) = a$
- Deduce $f(max(a, 0)) = f(a)$
- Contradiction with $\neg G$

We have obtained a contradiction in all cases, the negated formula is unsatisfiable, that means the input formula is valid!

Plan

Using yices interactively

```
moreno@gorgosaurus:~$ yices -i
Yices (version 1.0.40). Copyright SRI International.
GMP (version 5.1.1). Copyright Free Software Foundation, Inc.
Build date: Wed Dec  4 09:42:16 PST 2013
Type `(exit)' with parentheses to exit.
Type `(help)' with parentheses for help.
yices > (define f::(-> int int))

yices > (define i::int)

yices > (define j::int)

yices > (assert (= (- i 1) (+ j 2)))

yices > (assert (/= (f (+ i 3)) (f (+ j 6))))
unsat

yices >
```

Using yices interactively

```
moreno@gorgosaurus:~$ yices -i
Yices (version 1.0.40). Copyright SRI International.
GMP (version 5.1.1). Copyright Free Software Foundation, Inc.
Build date: Wed Dec  4 09:42:16 PST 2013
Type '(exit)' with parentheses to exit.
Type '(help)' with parentheses for help.
yices > (define x::int)

yices > (define y::int)

yices > (define z::int)

yices > (assert (= (+ (* 3 x) (* 6 y) z) 1))

yices > (assert (= z 2))

yices > (check)
unsat
```

Using `yices` interactively

Input file smt.ys

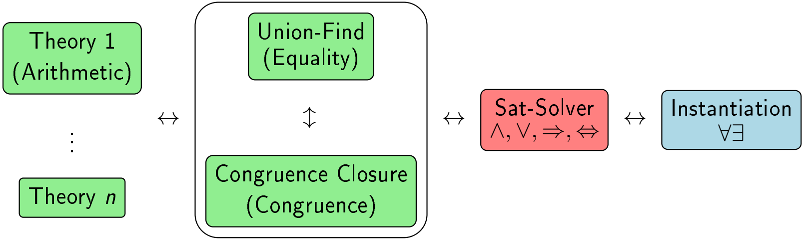
```
(define  x::int)
(define  y::int)
(define  f::(-> int int))
(assert  (/= (f (+ x 2)) (f (- y 1))))
(assert  (= x (- y 4)))
(check)
```

Call on the command line

```
moreno@gorgosaurus:~$ yices -e smt.ys
sat
(= x 0)
(= y 4)
(= (f 2) 1)
(= (f 3) 5)
```

Plan

Equality Reasoning



Equality reasoning - The problem

Terms

$t ::= c \mid f(t_1, \cdots, t_n)$

Given

a list of equations $t = t'$

We want to know

Does the equation $t_1 \stackrel{?}{=} t_2$ follow?

Using the axioms

Reflexivity	$t = t$
Symmetry	$t_1 = t_2 \to t_2 = t_1$
Transitivity	$t_1 = t_2 \wedge t_2 = t_3 \to t_1 = t_3$
Congruence	$t_1 = t_2 \to f(t_1) = f(t_2)$

Example

Given

- $f^2(a) = f(f(a)) = a$
- $f^5(a) = f(f(f(f(f(a))))) = a$

We want to prove

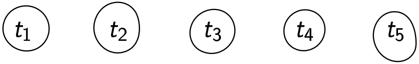
$f(a) = a$

Proof

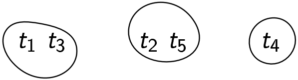
1. $f^5(a) = f^3(a)$ (Congruence)
2. $f^2(a) = f^3(a) = a$ (Transitivity, Symmetry)
3. $f^3(a) = f(f^2(a)) = f(a)$ (Congruence)
4. $f(a) = a$ (Transitivity of (2) and (3))

Disjoint Sets

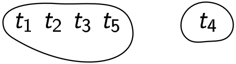
- Goal: deal with the first three axioms efficiently
- Idea: put all terms into disjoint sets
- When two terms are in the same set, they are equal
- Initial state: every term is in his own set:



- After treating $t_1 = t_3$ and $t_2 = t_5$:



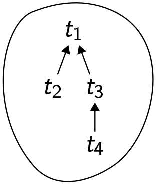
- After treating $t_1 = t_2$:



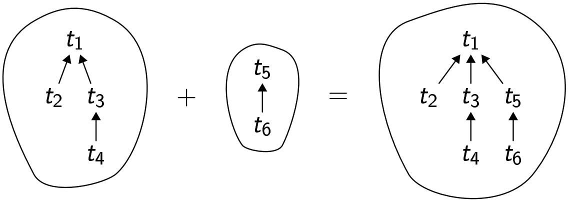
- Deciding $t \stackrel{?}{=} t'$ amounts to checking if t, t' are in the same set

Union-Find (1975)

- Represent each set by a tree with upward pointers:



- The root is the representative
- Operation find to find the representative of any term: just follow the arrows
- Operation union to treat an equality: simply point one root to the other



Two important optimizations

- Keep trees small: let point root of smaller tree to root of larger tree
- Path compression: "flatten" trees, each time we are searching for a root r starting from t, let t point directly to r afterwards
- Result: Algorithm is quasi-linear (optimal)
- Incrementality: we can add equations one by one, interleave equations $t_1 = t_2$ with queries $t_1 \stackrel{?}{=} t_2$

Inequalities $t_1 \neq t_2$

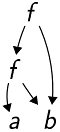
- Simply maintain the information that two sets of terms must be different
- Merging sets for which an inequality was registered leads to an inconsistency

Congruence Closure (1980)

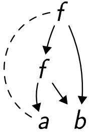
- Deal with the fourth axiom: Congruence

$$\forall xy.x = y \rightarrow f(x) = f(y)$$

  for any function symbol f
- Solution: represent a term by a directed acyclic graph (DAG) with sharing. Example: $f(f(a, b), b)$



- Add an equivalence relation to this graph (using union-find):



  represents $f(f(a, b), b) = a$

Finding new equalities

- Build a reverse dictionary mapping nodes to their fathers:

$$\begin{aligned} a &\mapsto f(a,b), g(a) \\ b &\mapsto f(a,b) \end{aligned}$$

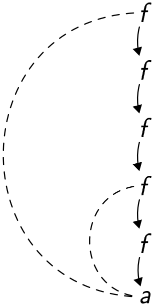
- Two new operations: find and merge.

```
merge(t1,t2) =
  union(t1,t2);
  F1,F2 = fathers(t1), fathers(t2);
  for each x in F1, y in F2 do
    if congruent(x,y) then merge(x,y);
  done
```

Congruence Closure — Example

Given

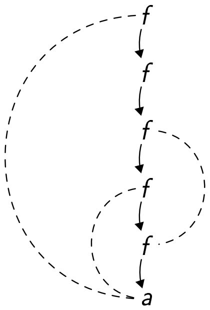
- $f^2(a) = f(f(a)) = a$
- $f^5(a) = f(f(f(f(f(a))))) = a$



Congruence Closure — Example

Given

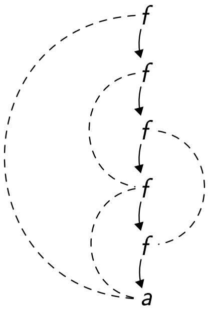
- $f^2(a) = f(f(a)) = a$
- $f^5(a) = f(f(f(f(f(a))))) = a$



Congruence Closure — Example

Given

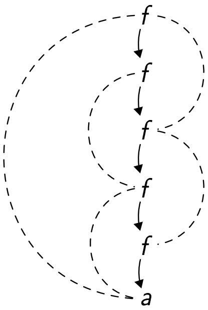
- $f^2(a) = f(f(a)) = a$
- $f^5(a) = f(f(f(f(f(a))))) = a$



Congruence Closure — Example

Given

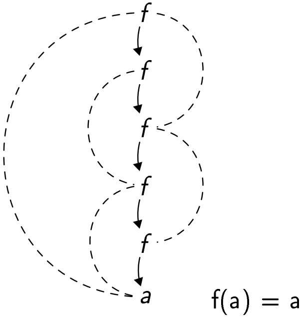
- $f^2(a) = f(f(a)) = a$
- $f^5(a) = f(f(f(f(f(a))))) = a$



Congruence Closure — Example

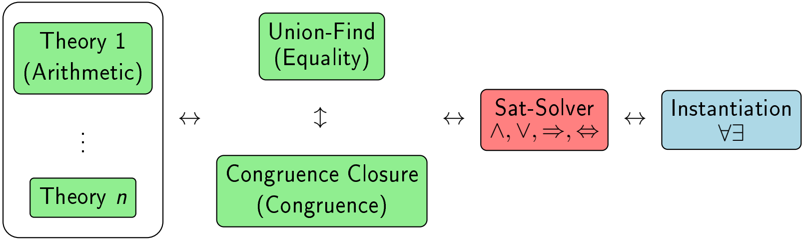
Given

- $f^2(a) = f(f(a)) = a$
- $f^5(a) = f(f(f(f(f(a))))) = a$



Plan

Theory Reasoning (Arithmetic)



Arithmetic reasoning

Arithmetic

- Interprets the function symbols $+$, $-$, $\times$, $\div$, and the arithmetic constants
- But also the relation symbols $\leq$, $<$, $\geq$, $>$

There are a few algorithms to deal with Linear Arithmetic

- Gauss Elimination (Equality only)
- Fourier-Motzkin
- Simplex Algorithm

We will look more closely at these methods

Gauss Elimination

Goal: deal with equalities in linear arithmetics

- Transform term into sums of monomials: $\sum_i^k c_i t_i$
- When treating an equality between such polynomes

$$\sum_i^k c_i t_i = \sum_j^k d_i s_i$$

isolate a monomial, say, t_1, and build the equation

$$t_1 = \sum_j^k \frac{d_i}{c_1} s_i - \sum_{i \neq 1}^k \frac{c_i}{c_1} t_i$$

Fourier-Motzkin Algorithm (1)

Goal: deal with inequalities in linear arithmetics

basic notions

- An inequality C in canonical form:

$$\sum_{i=1}^{n} a_i x_i \leq a_0 \qquad a_i \in \mathbb{Q}$$

- Note αC the multiplication of an inequation with a coefficient α:

$$\sum_{i=1}^{n} \alpha a_i x_i \leq \alpha a_0$$

- Note $C_1 + C_2$ the addition of two inequations :

$$\sum_{i=1}^{n} (a_i + b_i) x_i \leq a_0 + b_0$$

Fourier-Motzkin Algorithm (2)

Set $I = \{C_1 \cdots C_n\}$ the starting set of inequations. Each step of the algorithm will eliminate a variable from the set of the equations.

- Let I^+ (I^-) be the set of equations where x appears with positive (negative) coefficient
- Compute

$$I_x = \bigcup_{C \in I^-, D \in I^+} \beta C + \alpha D \qquad \alpha x \in C, -\beta x \in D$$

- Let I_0 the set of inequations in I without x
- Replace I par $I' = I_0 \cup I_x$
- In particular, if x appears only with coefficients of the same sign in I, suppress all inequations where x appears
- When I does not contain variables any more, either we have satisfiable inequalities (like $1 \leq 2$) or an inconsistency

Fourier-Motzkin Algorithm (3)

- Complexity: double exponential
- Not incremental
- Still behaves well in practice
- Can be easily extended to deduce equations between terms

References

- http://yices.csl.sri.com/old/download-yices1-full.html (The yices software)
- http://www.cs.cornell.edu/gomes/papers/SATSolvers-KR-Handbook.pdf SAT solvers handbook
- https://en.wikipedia.org/wiki/Boolean_satisfiability_problem (SAT)
- https://en.wikipedia.org/wiki/Satisfiability_modulo_theories (SMT)
- https://en.wikipedia.org/wiki/DPLL_algorithm (DPLL)
- http://0a.io/boolean-satisfiability-problem-or-sat-in-5-minutes/

This lecture follows partly a presentation by Hans Zantema (Eindhoven University of Technology), another by Luciano Serafini (Fondazione Bruno Kessler, Trento) and another by David L. Dill (Stanford University).