

Automated Deduction

Laura Kovács

TU Wien

Satisfiability Modulo Theory (SMT)

- ▶ An interpretation I which makes all axioms of \mathcal{T} valid, that is $I \models A_{\mathcal{T}}$, is called a \mathcal{T} -interpretation.
- ▶ A formula F is **valid** in \mathcal{T} (or \mathcal{T} -valid) if F is valid in every \mathcal{T} -interpretation, written $\mathcal{T} \models F$.
- ▶ A formula F is **satisfiable** in \mathcal{T} , or \mathcal{T} -satisfiable, or **satisfiable modulo theory \mathcal{T}** if there exists a \mathcal{T} -interpretation which satisfies F .

Cesare Tinelli:

**A DPLL-Based Calculus for Ground
Satisfiability Modulo Theories.** JELIA 2002

Silvio Ranise and Cesare Tinelli.

The SMT-LIB Format: An Initial Proposal. PDPAR 2003.

Societal Impact of SMT – A Recent Example

SMT @ Amazon Web Services (AWS) 

- March 26, 2019 Quote by Rima Tanash, Security Engineer for AWS

”Using the Zelkova tool of AWS,
you can encode access policies [to the AWS cloud].

These policies are **logical statements** like,
’Only allow users from account ‘A’ to perform a read-action.’

Zelkova translates the policy to a **first order logic formula**,
then uses tools including **SMT solver** to find requests
that can **satisfy the formula** “

<https://www.cs.rice.edu/rima>

Related problems to SMT

- ▶ **Deciding a conjunction of literals:** How can we check whether a set of *literals* is satisfiable?
In particular, in \mathcal{T}_E , \mathcal{T}_A , and \mathcal{T}_Q .
- ▶ How can we put theory reasoning and SAT solving together?
- ▶ **Combination of theories:** Given decision procedures for theories, how can we build a decision procedure for formulas using several theories?

Deciding Theory: An Example

Question: Is $a = b \wedge b = c \wedge f(a) \neq f(c)$ satisfiable in \mathcal{T}_E ?

- ▶ From $a = b \wedge b = c$ and (transitivity), conclude $a = c$.
- ▶ From $a = c$ and (congruence), conclude $f(a) = f(c)$.
- ▶ $f(a) = f(c)$ contradicts $f(a) \neq f(c)$.

Question: Is $x = y \wedge f(f(x)) \neq f(f(y))$ satisfiable in \mathcal{T}_E ?

The reasoning made above is very different from splitting or DPLL.
It uses theory axioms.

We will now discuss **specialised decision procedures** for theories.

Deciding \mathcal{T}_E : Congruence Closure

Congruence closure

- ▶ is a method to decide satisfiability of formulas in \mathcal{T}_E ;
- ▶ a **decision procedure** for \mathcal{T}_E ;
- ▶ can be extended to a decision procedure for \mathcal{T}_A ;
- ▶ is the basis for combining theories;
- ▶ decides formulas in **the theory of equality and uninterpreted functions**.

What about formulas with **predicates** other than equality?

Formulas with uninterpreted predicates can be easily transformed to formulas without predicates other than $=$.

Example: Instead of $p(x) \wedge q(y, z) \wedge a = b \rightarrow \neg q(x, z)$

we use

$$f_p(x) = t \wedge f_q(y, z) = t \wedge a = b \rightarrow f_q(x, z) \neq t,$$

where f_p, f_q are fresh functions and t is a fresh constant.

Deciding \mathcal{T}_E : Congruence Closure – abstract definitions

Consider a set S and a binary relation R over S .

R is a **congruence relation** if:

- ▶ xRx
- ▶ $xRy \rightarrow yRx$
- ▶ $xRy \wedge yRz \rightarrow xRz$
- ▶ $x_1Ry_1 \wedge \dots \wedge x_nRy_n \rightarrow f(x_1, \dots, x_n)Rf(y_1, \dots, y_n)$ for all function symbols f .

Example: $=$ is a congruence relation.

The **congruence class** of $t \in S$ under the congruence relation R is

$$[t]_R = \{t' \in S \mid tRt'\}$$

The congruence relation R defines a **partition** on S :

$$\left(\bigcup_{[t]_R} [t]_R \right) = S \quad \text{and} \quad [t_1]_R \neq [t_2]_R \rightarrow [t_1]_R \cap [t_2]_R = \emptyset$$

Deciding \mathcal{T}_E : Congruence Closure – abstract definitions

The **congruence closure** R^c of R is the congruence relation such that:

- ▶ $R \subseteq R^c$
- ▶ for all other congruence relations R' with $R \subseteq R'$, either $R^c = R'$ or $R^c \subseteq R'$

R^c is the smallest congruence relation that includes R .

Example: Let $S = \{a, b, c, d\}$ and $R = \{aRb, bRc, dRd\}$. Then

$$R^c = \{aRb, bRc, dRd, aRa, bRb, cRc, bRa, cRb, aRc, cRa\} \quad \text{since}$$

- ▶ $aRb, bRc, dRd \in R^c$ by $R \subseteq R^c$
- ▶ $aRa, bRb, cRc \in R^c$ by reflexivity
- ▶ $bRa, cRb \in R^c$ by symmetry
- ▶ $aRc \in R^c$ by transitivity
- ▶ $cRa \in R^c$ by symmetry

Deciding \mathcal{T}_E : Congruence Closure Algorithm

Consider the formula F :

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

where terms s_i, t_i are terms. Is F \mathcal{T}_E -satisfiable?

Idea

1. set S is the set of subterms of F
2. construct the **congruence class of each subterm** of F under the binary relation $\{s_1 = t_1, \dots, s_n = t_n\}$.
Congruence closure R of $\{s_1 = t_1, \dots, s_n = t_n\}$!
3. if for some $i \in \{n+1, \dots, m\}$, we obtain that s_i and t_i are in the same congruence class, then F is **unsatisfiable**. Otherwise, F is **satisfiable**.

Deciding \mathcal{T}_E : Congruence Closure Algorithm

procedure *CongruenceClosure*(F)

input: F is $s_1 = t_1 \wedge \dots \wedge s_n = t_n \wedge s_{n+1} \neq t_{n+1} \wedge \dots \wedge s_m \neq t_m$

output: *satisfiable* or *unsatisfiable*

parameters: function *subterm_set*

begin

$S_F := \text{subterm_set}(F)$

$R := \{sRs \mid s \in S_F\}$ and $[s]_R := \{s\}$, defining the partition $\{[s]_R \mid s \in S_F\}$

for every $s_i = t_i$ in F , merge $[s_i]_R$ and $[t_i]_R$ by

forming the union $[s_i]_R \cup [t_i]_R$

propagate the new congruences that arise in this union (function congruence)

if $s_j R t_j$ for any $j \in \{n+1, \dots, m\}$ **then return** *unsatisfiable*

else return *satisfiable* **end**

Deciding \mathcal{T}_E : Congruence Closure by Example

Consider the formula $F : f(a, b) = a \wedge f(f(a, b), b) \neq a$.

Question: Is F \mathcal{T}_E -satisfiable?

Subterm set $S_F = \{a, b, f(a, b), f(f(a, b), b)\}$

Initial partition: $\{\{a\}, \{b\}, \{f(a, b)\}, \{f(f(a, b), b)\}\}$

Using $f(a, b) = a$, merge $\{f(a, b)\}$ and $\{a\}$ and form partition:

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

From $f(a, b) R a$ and $b R b$, by congruence we have: $f(f(a, b), b) R f(a, b)$

Thus, merge $\{a, f(a, b)\}$ and $\{f(f(a, b), b)\}$ and form partition:

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

This is the partition of the **congruence closure** of $\{f(a, b) = a\}$.

F contains $f(f(a, b), b) \neq a$, but we have $f(f(a, b), b) R a$ in the congruence closure. Hence, F is \mathcal{T}_E -unsatisfiable.

Deciding \mathcal{T}_E : Congruence Closure by Example

Consider the formula $F : f(a) = f(b) \wedge a \neq b$.

Question: Is F \mathcal{T}_E -satisfiable?

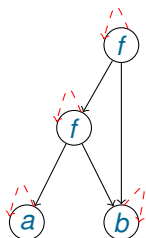
Consider the formula

$$F : f(f(f(a))) = a \wedge f(f(f(f(f(a)))))) = a \wedge f(a) \neq a.$$

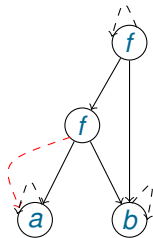
Question: Is F \mathcal{T}_E -satisfiable?

Congruence Closure and ... DAGs

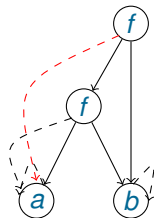
Consider the formula $F : f(a, b) = a \wedge f(f(a, b), b) \neq a$.



A node is a subterm of F .
Initial congruence classes.



A node is a subterm of F .
Union of congruence classes.



A node is a subterm of F .
Propagation of congruence classes.
Union-Find algorithm on the DAG for congruence closure.

Congruence closure is inexpensive – it is decidable in polynomial time.

Outline

Theory of Equality: Congruence Closure
Congruence Closure and DAGs

SMT and Non-Unit Clauses

Deciding \mathcal{T}_E : Problems Using Non-Unit Clauses

Example: Let's try to prove the validity of the formula:

$$F : (a = b \vee a = c) \wedge f(a) = a \rightarrow f(f(b)) = a \vee f(c) = a.$$

This is equivalent to establishing unsatisfiability of

$$a = b \vee a = c$$

$$f(a) = a$$

$$f(f(b)) \neq a$$

$$f(c) \neq a$$

We have a **non-unit clause**, so we can't use congruence closure.

Inputs of the congruence closure algorithm are conjunctions of \mathcal{T}_E -literals.