Satisfiability Checking

Lazy SMT-Solving for Equality Logic

Prof. Dr. Erika Ábrahám

RWTH Aachen University
Informatik 2
LuFG Theory of Hybrid Systems

WS 14/15

Reminder: Equality logic with uninterpreted functions

We extend the propositional logic with

- equalities and
- uninterpreted functions (UFs).

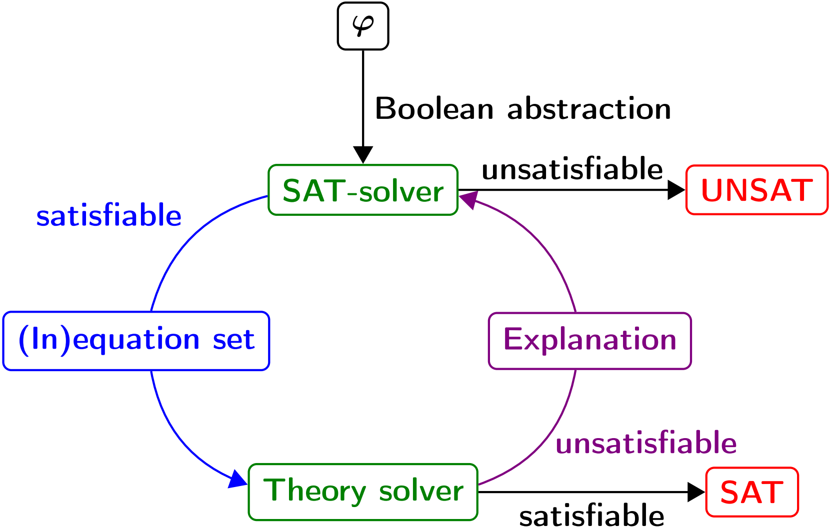
Syntax:

- variables x over an arbitrary domain D,
- constants c from the same domain D,
- function symbols F for functions of the type $D^n \to D$, and
- equality as predicate symbol.

$$\begin{array}{lllllll} \textit{Terms:} & t & := & c & \mid & x & \mid & F(t,\ldots,t) \\ \textit{Formulas:} & \varphi & := & t = t & \mid & (\varphi \land \varphi) & \mid & (\neg \varphi) \end{array}$$

Semantics: straightforward

Full lazy SMT-solving



First: Conjunction of equalities without UF

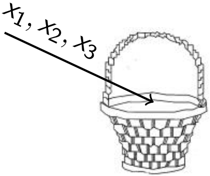
Input: A conjunction φ of equalities and disequalities without UF

Algorithm

1. Define an equivalence class for each variable in φ.
2. For each equality $x = y$ in φ: merge the equivalence classes of x and y.
3. For each disequality $x \neq y$ in φ:
   if x is in the same class as y, return 'UNSAT'.
4. Return 'SAT'.

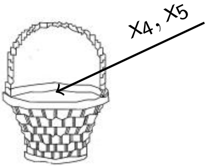
Example

$$\varphi^E : \quad x_1 = x_2 \wedge x_2 = x_3 \wedge x_4 = x_5 \wedge x_5 \neq x_1$$



Equivalence class 1

SAT



Equivalence class 2

Next: Add uninterpreted functions

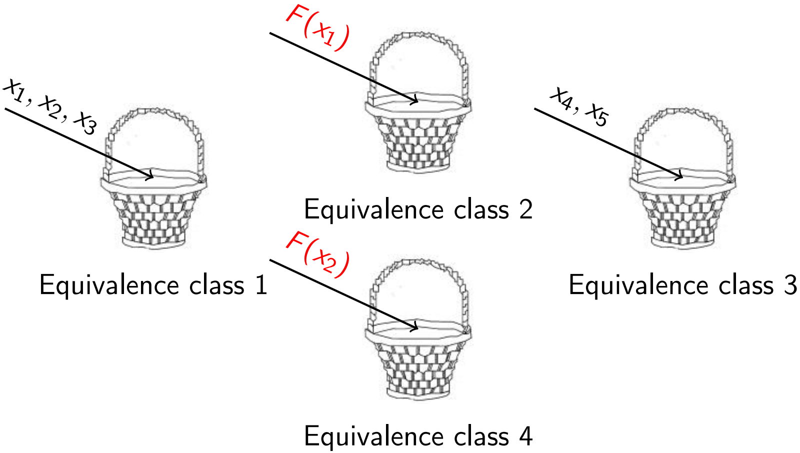
How do they relate?

- $x = y$, $F(x) = F(y)$: $\models (x = y) \rightarrow (F(x) = F(y))$
- $x = y$, $F(x) \neq F(y)$: conjunction unsatisfiable
- $x \neq y$, $F(x) = F(y)$: unrelated (conjunction satisfiable)
- $x \neq y$, $F(x) \neq F(y)$: $\models (F(x) \neq F(y)) \rightarrow (x \neq y)$

- $x = y$, $F(G(x)) = F(G(y))$: $\models (x = y) \rightarrow (F(G(x)) = F(G(y)))$

Next: Add uninterpreted functions

$$\varphi^E : \quad x_1 = x_2 \land x_2 = x_3 \land x_4 = x_5 \land x_5 \neq x_1 \land F(x_1) \neq F(x_2)$$

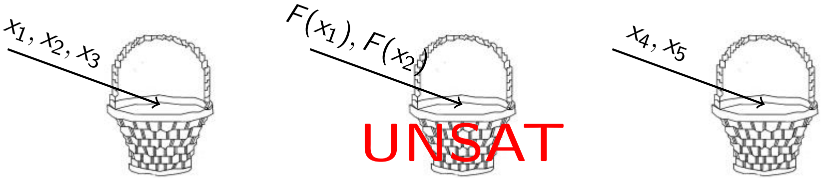


Next: Compute the *congruence closure*

$$\varphi^E : \quad x_1 = x_2 \wedge x_2 = x_3 \wedge x_4 = x_5 \wedge x_5 \neq x_1 \wedge F(x_1) \neq F(x_2)$$

Congruence closure:

If all the arguments of two function applications are in the same class, merge the classes of the applications!



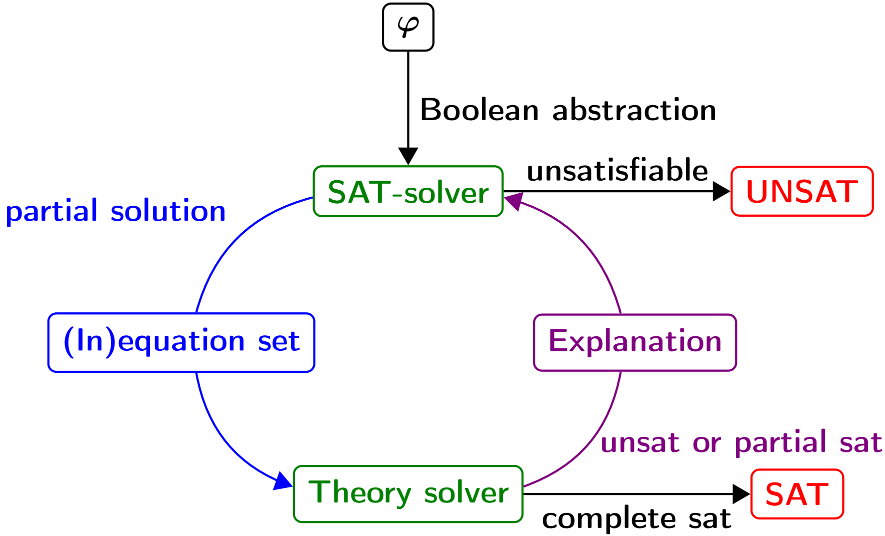
Equivalence class 1 | Equivalence class 2 | Equivalence class 3

Input: A conjunction φ of equalities and disequalities with UFs of type $D \to D$

Algorithm

1. $\mathcal{C} := \{\{t\} \mid t$ occurs as subexpression in an (in)equation in $\varphi\}$;
2. for each equality $t = t'$ in φ with $[t] \neq [t']$
   $\mathcal{C} := (\mathcal{C} \setminus \{[t], [t']\}) \cup \{[t] \cup [t']\}$;
   while exists $F(t), F(t')$ in φ with $[t] = [t']$ and $[F(t)] \neq [F(t')]$
   $\mathcal{C} := (\mathcal{C} \setminus \{[F(t)], [F(t')]\}) \ \cup \ \{[F(t)] \cup [F(t')]\}$;
3. for each inequality $t \neq t'$ in φ
   if $[t] = [t']$ return "UNSAT";
4. return "SAT";

Less lazy SMT-solving



Requirements on the theory solver

Needed for lass lazy SMT solving:

1. Incrementality: In less lazy solving we extend the set of constraints. The solver should make use of the previous satisfiability check for the check of the extended set.
2. (Preferably minimal) infeasible subsets: Compute a reason for unsatisfaction
3. Backtracking: The theory solver should be able to remove constraints in inverse chronological order.

Requirements on the theory solver

Solution:

1. Incrementality:
   - When a new equation is added, update the equivalence relation and check the previously added inequalities for satisfiability.
   - When a new inequation is added, check its satisfiability under the current equivalence relation.
2. (Preferably minimal) infeasible subsets: A conflict appears when an inequation $a \neq b$ cannot be true together with the current equalities; build the set of this inequation $a \neq b$ and (a minimal number of) equations that imply $a = b$ by transitivity.
3. Backtracking: Remember computation history.