Seminar on Decision Procedures
By Ruzica Piskac

Problem 1 Congruence Closure Algorithm

Apply the congruence closure algorithm to the following formulas. Use a simple version of the algorithm, without DAGs. Provide a detailed description of constructed partitions and their mergings.

- \( f(x, y) = f(y, x) \land f(a, y) \neq f(y, a) \)
- \( f(g(x)) = g(f(x)) \land f(g(f(y))) = x \land f(y) = x \land g(f(x)) \neq x \)
- \( f(f(f(a))) = f(f(a)) \land f(f(f(a)))) = a \land f(a) = a \land f(a) \neq a \)
- \( f(f(f(a))) = f(a) \land f(f(a)) = a \land f(a) \neq a \)
- \( p(x) \land f(f(x)) = x \land f(f(f(x))) = x \land \neg p(f(x)) \)

Problem 2 Congruence Closure with DAGs

Apply the DAG-based congruence closure algorithm to the formulas given in Problem 1. Provide a detailed description of congruences and mergings.

Problem 3 Cooper’s Method

Apply the Cooper’s quantifier elimination method to the following linear integer arithmetic formulas.

- \( \forall y.3 < x + 2y \lor 2x + y < 3 \)
- \( \exists y.x = 2y \lor 2x + y < 3 \)
- \( \exists y.x = 2y \land y < x \)
- \( \forall x.(\exists y.x = 2y) \rightarrow (\exists y.3x = 2y) \)
Problem 4  Ferrante-Rackoff’s Method

Apply the Ferrante-Rackoff’s quantifier elimination method to the formulas given in Problem 3 but treat them as rational arithmetic formulas.

Problem 5  Multisets with Cardinality Constraints

Prove that the following multiset formula is valid:

\[ S \oplus C \subseteq L \land S \neq L \rightarrow |S| < |L| \]

This formula was introduced in the lectures as a verification condition expressing that a list without duplicates will contain less elements, as long as there is at least one duplicate.

To prove this formula, the easiest way is to follow the description of an algorithm given in my doctoral thesis (Chapters 2.4 - 2.6). Here are certain milestones that will assure you that you are on the good track

1. \( \exists e. F \) can be turn into \( \neg \forall e. \neg F \)

2. \( \neg \forall e. F \) can be rewritten by \( 0 \neq \Sigma_{e \in E} \text{ite}(F, 0, 1) \)

3. this sum can be again flattened by introducing a fresh integer variable \( v \):
   
   \[ 0 \neq v \land v = \Sigma_{e \in E} \text{ite}(F, 0, 1) \]

4. let \( e \) be some complex expression. If in the sum normal form, you obtain a formula of the form \( P \land \vec{u} \in \{(e, v_1, \ldots, v_n) \mid F\}^* \), you can simplify this formula by again introducing a fresh integer variable \( x_f \): \( P \land \vec{u} \in \{(x_f, v_1, \ldots, v_n) \mid x_f = e \land F\}^* \)

5. expand the definition of the \( \text{ite}(F, x, y) \) operator

6. formula \( P \land \vec{u} \in \{\vec{v} \mid F_1 \lor F_2\}^* \) is equisatisfiable to

   \[ P \land \vec{u} = \vec{u}_1 + \vec{u}_2 \land \vec{u}_1 \in \{\vec{v} \mid F_1\}^* \land \vec{u}_2 \in \{\vec{v} \mid F_2\}^* \]

7. you do not need to compute semilinear sets - they are non-trivial in this case. It is enough if show that one of summands have to be zero, and the other will always compute values contradicting the linear integer arithmetic formula