A Resolution-Based Approach for Deciding Equality Logic Formulas

Olga Tveretina and Hans Zantema

Department of Computer Science, TU Eindhoven, P.O. Box 513,
5600 MB Eindhoven, The Netherlands
o.tveretina@tue.nl, h.zantema@tue.nl

Equality logic with *uninterpreted functions (UIFs)* is used for hardware verification [3]. The problem of deciding the validity of the formula in equality logic with UIFs can be reduced to checking satisfiability of equality formulas [1].

Many current approaches for deciding satisfiability of equality formulas use a transformation of equality logic to propositional logic and then they use standard SAT checker. Goel et al. [4] and Bryant et al. [2] reduce an equality formula to a propositional one by adding *transitivity constraints*. While another approach called *range allocation* [6] a formula structure is analyzed to define a small domain for each variable. Then a standard BDD based tool is used to check satisfiability of the formula under the domain. Yet another approach [5] is based on BDD computation, with some extra rules for dealing with transitivity.

We propose a resolution-based approach for deciding satisfiability of equality logic formulas (E-SAT) in conjunctive normal form. For this single rule we prove soundness and completeness.

1 E-CNF

Similar to propositional logic every equality formula can be transformed to an *equality formula in conjunctive normal form (E-CNF)* such that the original formula is satisfiable if and only if the E-CNF is satisfiable. We shall consider satisfiability of E-CNFs.

An E-CNF is defined as follows.

<variable> ::= $x \mid y \mid z \ldots$
<literal> ::= *<variable = variable>* | *<variable ≠ variable>*
<clause> ::= *<literal>* | *<clause ∨ literal>*
<E-CNF> ::= *<clause>* | *<E-CNF ∧ clause>*

Let V be a set of variables. A *domain* D is defined to be a non-empty set. For any domain we define an *assignment* as a function $A : V \to D$. For an assignment A we define the corresponding *interpretation* I_A on literals by:

$I_A(x = y) = true$ if $A(x) = A(y)$
$I_A(x = y) = false$ if $A(x) \neq A(y)$
$I_A(x \neq y) = \neg I_A(x = y)$.

An assignment A satisfies a literal l on some domain D if $I_A(l) = true$. An assignment satisfies a clause if an interpretation of at least one of its literals yields *true*. An assignment satisfies an E-CNF F if and only if it satisfies each of its clauses and we say that F is mapped to *true*. An E-CNF F is called *satisfiable* if there is a domain D and an assignment A such that F is mapped to *true*. If there is no assignment that maps an E-CNF F to *true* then F is called *unsatisfiable*.

2 Removing Redundant Clauses

We give the approaches for removing clauses from the E-CNF without influencing on its satisfiability.

Definition 1. *A set of literals in a form* $\{x_1 = x_2, \ldots, x_{n-1} = x_n, x_1 \neq x_n\}$ *is called a* contradictory cycle. *A contradictory cycle of* F *is a contradictory cycle such that all its literals occur in* F.

Theorem 2. *Let F be an E-CNF; a literal l not occur in any contradictory cycle of F . Then every clause containing l can be removed from F without influencing its satisfiability.*

3 A Proof Rule and a Decision Procedure

We give a single proof rule called *equality resolution (ER)*:

$$\frac{(x_1 = x_2) \vee C_1, \dots,\ (x_{n-1} = x_n) \vee C_{n-1}, (x_1 \neq x_n) \vee C_n}{C_1 \vee \dots \vee C_n}$$

The rule is sound since no assignment exist satisfying all literals $x_1 = x_2, \dots, x_{n-1} = x_n, x_1 \neq x_n$.

Theorem 3. *An E-CNF is unsatisfiable iff the empty clause can be derived by ER rule.*

We give a decision procedure based on the ER rule. It takes as an input an E-CNF F. The sign $\perp$ denotes the empty clause; Θ_F is the set of all contradictory cycles contained in F.

Procedure E-SAT (F);

```
begin
   Θ := ∅;
   while ((Θ_F\Θ) ≠ ∅) do
   begin
      RemoveRedundant (F);
      θ := ShortestContrCycle (F,Θ);
      Θ := Θ ∪ {θ};
      if ⊥ ∈ F return(unsatisfiable);
      if F = ∅ return(satisfiable);
      F := F∪ ER(F,θ);
   end
   return(satisfiable);
end
```

The procedure `ShortestContrCycle` (F, Θ) chooses a contradictory cycle of the shortest length not contained in Θ. The procedure `RemoveRedundant`(F) repeatedly removes clauses from F by Theorem 2 until nothing can be removed any more. The function `ER`(F, θ) forms the set of clauses derived from F by all possible applications of ER rule for a contradictory cycle θ. If the empty clause is derived then the output is "unsatisfiable". If either the set of clauses is empty or no contradictory cycles left then the output is "satisfiable".

References

1. Ackermann, W. *Solvable cases of the decision problem.* Studies in Logic and the Foundations of Mathematics. North-Holland, Amsterdam, 1954.
2. Bryant, R., and Velev, M. Boolean satisfiability with transitivity constraints. *ACM Transactions on Computational Logic 3*, 4 (October 2002), 604–627.
3. Burch, J., and Dill, D. Automated verification of pipelined microprocesoor control. In *Computer-Aided Verification (CAV'94)* (June 1994), D. Dill, Ed., vol. 818 of *LNCS*, Springer-Verlag, pp. 68–80.
4. Goel, A., Sajid, K., Zhou, H., Aziz, A., and Singhal, V. BDD based procedures for a theory of equality with uninterpreted functions. In *Computer-Aided Verification (CAV'98)* (1998), A. J. Hu and M. Y. Vardi, Eds., vol. 1427 of *LNCS*, Springer-Verlag, pp. 244–255.
5. Groote, J., and van de Pol, J. Equational binary decision diagrams. In *Logic for Programming and Reasoning (LPAR'2000)* (2000), M. Parigot and A. Voronkov, Eds., vol. 1955 of *LNAI*, pp. 161–178.
6. Rodeh, Y., and Shtrichman, O. Finite instantiations in equivalence logic with uninterpreted functions. In *Computer Aided Verification (CAV'01)* (July 2001), vol. 2102 of *LNCS*, Springer-Verlag, pp. 144–154.