DPLL-based Procedure for Equality Logic with Uninterpreted Functions

Olga Tveretina

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

Abstract. The logic of equality with uninterpreted functions (EUF) has been proposed for processor verification. We describe EDPLL, a calculus for proving satisfiability of formulas in this kind of logic. Being based on the DPLL procedure, EDPLL can adopt heuristics developed for this method.

Keywords: equality logic with uninterpreted functions, satisfiability, DPLL procedure.

1 Introduction

The logic of equality with uninterpreted functions (EUF) has been proposed for verifying hardware [3]. This type of logic is mainly used for proving the equivalence between systems. When verifying the equivalence between two formulas it is often possible to abstract away functions replacing them with uninterpreted functions.

EUF formula is a boolean formula over atoms that are equalities between ground terms. For example, the formula:

$$f(a) \neq f(b) \wedge a = c \wedge c = b$$

is unsatisfiable.

Current approaches mostly use either a transformation of a formula in EUF to a propositional one and apply SAT checker [5, 2] or integrate SAT with a congruence closure-based decision procedures for equational theories.

The Davis-Putnam-Logemann-Loveland procedure (DPLL) [4] was introduced in the early 60s as a proof procedure for first-order logic. Nowadays, only its propositional logic core component is widely used in efficient propositional logic provers. The success was a motivation for developing DPLL-based procedure for equality logic with uninterpreted functions. The main idea of the DPLL method is to choose an atom from the formula and proceed with two recursive calls: one obtained by assuming this atom and another by assuming the negation of the atom. The procedure terminates with the answer "unsatisfiable" if the empty clause is derived for all branches, and otherwise it returns "satisfiable". Different techniques based on the DPLL method are introduced in [1, 6, 8]. We present the DPLL-based procedure, which differs from the approaches mentioned above.

Brief description of EDPLL. The idea of EDPLL is to split a literal that occurs in purely positive clauses of length greater than one, apply unit propagation to all unit clauses and term reduction. The difference of our method that after assigning true to unit clause this clause is not removed from the formula. A part of out procedure is a term reduction which means replacement terms satisfying some conditions by constants not occurring in the formula. The goal of this replacement is to simplify the formula.

Our paper is organized as follows. In Section 2 we give basic definitions. In Section 3 we introduce the EDPLL procedure. We present some theorems in Section 4. Some concluding remarks are in Section 5.

2 Notation

In CNF, the formula ϕ is a conjunction of clauses, each clause C is a disjunction of literals, each literal l is an atom or a negated atom. An empty formula is true, and $\perp$, the empty clause is false. An atom A is an equality of the form $s = t$, where s and t are terms. A constant (0-arity function symbol) is a term, and $f(t_1, \ldots, t_n)$ is a term if f is n-arity function symbol, and $t_1, \ldots, t_n$ are terms. We say that $t_1, \ldots, t_n$ are subterms of a term $f(t_1, \ldots, t_n)$, and all subterms of $t_1, \ldots, t_n$ are also subterms of $f(t_1, \ldots, t_n)$.

We say that l is a *positive literal* if it is coincides with some atom, C is a *positive clause* if it contains only positive literals.

The number of literals in a clause is called the *length* of the clause.

The *depth* of a term t is denoted by $\mathsf{depth}(t)$ and inductively defined: $\mathsf{depth}(a) = 1$ if a is a constant, and $\mathsf{depth}(f(t_1, \ldots, t_n)) = 1 + \max(\mathsf{depth}(t_1, \ldots, t_n))$.

The positive clauses of length greater than one contained in ϕ are called *core* of ϕ. It is denoted as $\mathsf{Core}(\phi)$.

We say that a term t is *free* in ϕ if for each its subterm s a literal $s = u$ does not occur in $\mathsf{Core}(\phi)$ for any term u.

We say that a term t is *reducible* in ϕ if $\mathsf{depth}(t) > 1$ and all its subterms are free terms in ϕ. We say that a constant a is *fresh* for ϕ if it does not occur in ϕ.

Let At be a set of atoms. We define an *interpretation* as a function $\mathsf{I} : \mathsf{At} \to \{\mathsf{true}, \mathsf{false}\}$. A literal l is true in I if either l is an atom a and $\mathsf{I}(a) = \mathsf{true}$ or l is a negated atom $\neg a$ and $\mathsf{I}(a) = \mathsf{false}$. We write $\mathsf{I} \models l$, if a literal l is true in I. Then an *admissible* interpretation is one satisfying the following conditions.

- $\mathsf{I} \models t = t$;
- if $\mathsf{I} \models s = t$ then $\mathsf{I} \models t = s$;
- if $\mathsf{I} \models s = u$ and $\mathsf{I} \models u = t$ then $\mathsf{I} \models s = t$;
- if $\mathsf{I} \models s_i = t_i$ for all $i \in \{1, \ldots, n\}$ then $\mathsf{I} \models f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n)$.

Definition 1. *A formula ϕ is called satisfiable if it is* true *in some admissible interpretation* I*. Otherwise ϕ is called unsatisfiable.*

We will use throughout the paper the following notations: $\phi[t/s]$ denotes a formula ϕ that is obtained by substituting the term t for the term s; we define $\phi|_l = \{C - \{\neg l\} | \ C \in \phi, l \notin C\neg\}$.

3 EDPLL

EDPLL can be used to decide the satisfiability of equality logic formulas with uninterpreted functions in conjunctive normal form. The main operations of the method are *unit propagation*, *term reduction*, *tautology atom removing* and recursive reduction to smaller problems. If the empty clause for all branches is derived then the procedure returns "unsatisfiable". If for at least for one branch the formula is derived with the empty core then the procedure returns "satisfiable".

Term Reduction: if t is a reducible term occurring in ϕ and a is a fresh constant in ϕ then

$$\frac{\phi}{\phi[t/a]}$$

Unit propagation I: if a and b are constants then

$$\frac{\{a=b\}, \phi}{\phi|_{a=b}[b/a]}$$

Unit propagation II: if for some terms s and t, $depth(t) > 1$ and $depth(t) \geq depth(s)$ then

$$\frac{\{s=t\}, \phi}{\{s=t\}, \phi|_{s=t}[t/s]}$$

Tautology atoms removing: if t is a term

$$\frac{\phi}{\phi|_{t=t}}$$

Branching means choosing a literal l and checking satisfiability of $\phi \wedge l$ and $\phi \wedge \neg l$. ϕ is satisfiable only if both $\phi \wedge l$ and $\phi \wedge \neg l$ are satisfiable.

A CNF ϕ is called *reduced* if it is closed under term reduction, unit propagation and tautology atoms removing rules.

The procedure Reduce taks as an input a CNF ϕ and returns a reduced CNF.

```
EDPLL(φ) :
    begin
        φ := Reduce(φ);
        if (φ contains the empty clause)
            return "unsatisfiable";
        if (Core(φ) = ∅)
            return "satisfiable";
        choose a literal l in Core(φ);
        if (EDPLL(φ ∧ l) is "satisfiable")
            return "satisfiable";
        else
            return EDPLL(φ ∧ ¬l);
    end
```

Fig. 1. The EDPLL procedure

Example 2. As am example we consider the formula raised during translation validation [7], where concrete functions replaced by uninterpreted function symbols.

$$\neg\phi : u_1 = f(x_1, y_1) \wedge u_2 = f(x_2, y_2) \wedge z = g(u_1, u_2) \wedge z \neq g(f(x_1, y_1), f(x_2, y_2)).$$

After applying the term reduction rule, we obtain

$$\neg\phi' : u_1 = v_1 \wedge u_2 = v_2 \wedge z = g(u_1, u_2) \wedge z \neq g(v_1, v_2).$$

After applying the unit propagation I rule, we obtain

$$\neg\phi'' : z = g(u_1, u_2) \wedge z \neq g(u_1, u_2).$$

After applying unit resolution II , we obtain

$$\neg\phi''' : z = g(u_1, u_2) \wedge \bot.$$

4 Soundness and Completeness

In this section we will give arguments that the EDPLL procedure is sound and complete. One can see that both rules for unit propagation and the term reduction rule preserve (un)satisfiability of a formula.

Theorem 3. *(SAT criterion) Let ϕ be a reduced CNF. If* $\mathsf{Core}(\phi) = \emptyset$ *then ϕ is satisfiable.*

The termination of the procedure follows from the fact that at every step the total depth of all terms in the formula is decreased. Based on the above arguments we can conclude soundness and completeness of our approach.

Theorem 4. (*Soundness and Completeness*) *A CNF ϕ is unsatisfiable iff the* EDPLL(ϕ) *returns "unsatisfiable".*

5 Conclusions and future work

We have presented a new approach for checking satisfiability of formulas in the EUF logic. A part of our method is technique for reducing a formula that can be of interest itself. Our procedure can incorporate some optimization techniques developed by the SAT community for the DPLL method. We are going to implement our procedure and to compare it with existing techniques. We considered example, where after few steps we could prove unsatisfiability of the formula. Traditional approach would lead at first to transformation of the formula to propositional formula of bigger size and then applying any standard SAR checker. We can see from the considered example that our approach can be efficient for some formulas. Although at present we cannot make general conclusions about efficiency of the procedure.

References

1. Baumgartner, P. FDPLL – A First-Order Davis-Putnam-Logeman-Loveland Procedure. In *CADE-17 – The 17th International Conference on Automated Deduction* (2000), D. McAllester, Ed., vol. 1831 of *Lecture Notes in Artificial Intelligence*, Springer, pp. 200–219.
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. Davis, M., Logemann, G., and Loveland, D. A machine program for theorem proving. *Communications of the Association for Computing Machinery 7* (July 1962), 394–397.
5. 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.
6. Nieuwenhuis, R., and Oliveras, A. Congruence closure with integer offsets. In *10h Int. Conf. Logic for Programming, Artif. Intell. and Reasoning (LPAR)* (2003), M. Vardi and A. Voronkov, Eds., LNAI 2850, pp. 78–90.
7. Pnueli, A., Rodeh, Y., Shtrichman, O., and Siegel, M. Deciding equality formulas by small domains instantiations. In *Computer Aided Verification (CAV'99)* (1999), vol. 1633 of *LNCS*, Springer-Verlag, pp. 455–469.
8. Tinelli, C. A DPLL-based calculus for ground satisfiability modulo theories. In *Proceedings of the 8th European Conference on Logics in Artificial Intelligence (Cosenza, Italy)* (2002), G. Ianni and S. Flesca, Eds., vol. 2424 of *Lecture Notes in Artificial Intelligence*, Springer, pp. 308–319.