THE NELSON-OPPEN METHOD FOR COMBINING SPECIAL THEOREM PROVERS: IMPLEMENTATION AND APPLICATION TO PROGRAM VERIFICATION

DIPLOMARBEIT

ZUR ERLANGUNG DES AKADEMISCHEN GRADES

"DIPLOMINGENIEUR"

IN DER STUDIENRICHTUNG TECHNISCHE MATHEMATIK

eingereicht von

LEO BACHMAIR

angefertigt am Institut für Mathematik
der technisch-naturwissenschaftlichen Fakultät
der Johannes Kepler Universität Linz

eingereicht bei: o. Univ. Prof. Dr. B. Buchberger

Linz im November 1982

CHAPTER 2

COOPERATING DECISION PROCEDURES

In this chapter a method for combining decision procedures for several theories into a single decision procedure for the combination of the theories is described. The method, developed by Nelson and Oppen [18], is applicable to disjoint quantifier-free theories. We give a new and simpler proof of correctness of the algorithm. In fact, the original proof given by Nelson and Oppen is not totally correct.

2.1. First-order Theories

First, we introduce the basic logical notions which we will use in the subsequent chapters. We do not give an elaborated presentation of the subject but refer to Buchberger [06] for details. Precise definitions may also be found in any standard logic book, Hermes [12], for instance.

The theories we are dealing with are formalized in first-order predicate logic with equality. A first-order theory is a formal system T such that the language L(T) of T is a first-order language and the axioms of T are the logical axioms of L(T) and certain further axioms, called the nonlogical axioms. The logical symbols ~ (negation), v (disjunction), & (conjunction), => (implication), ≡ (equivalence), = (equality), the existential quantifier $\exists$ and the universal quantifier $\forall$ are common to all theories. In order to specify a theory we have only to specify its nonlogical symbols and its nonlogical axioms.

The description of the semantics of a first-order language L is done by specifying a nonempty set U, called the universe, and an interpretation function I, which is defined on variables, function symbols and predicate symbols. I assigns an element of U to each variable, an n-ary function over U to each n-ary function symbol and an n-ary predicate over U to each n-ary predicate symbol in L.

A model of a theory T is a universe U together with an interpretation I for L(T) in which all the nonlogical axioms of T are valid. A formula F is valid in T if it is valid in every model of T (if it is a "logical consequence" of the nonlogical axioms of T). A formula F is satisfiable in T if there is some model of T in which F is valid.

The decision problem for a theory T is as follows: given a formula F in T, decide whether F is valid in T or not. A decision procedure for T is an algorithm which determines whether a formula F is valid in T. We are concerned with decision procedures for quantifier-free theories. A quantifier-free theory is a theory whose formulas do not contain quantifiers. The decision problem for a quantifier-free theory T can be reduced to the problem of testing the satisfiability of a conjunction of T-literals (A literal is an atomic formula or a negation of an atomic formula, a T-literal is a literal of L(T). We will also use the notions T-term and T-formula in the same sense):

> A formula F is valid in T if and only if ~F is unsatisfiable in T. But ~F can be put into disjunctive normal form, that is, there are formulas $F_1, \ldots, F_n$ such that ~F is equivalent to $(F_1 \vee \ldots \ldots \vee F_n)$ and F_i is a conjunction of literals for each i such that $1 \leq i \leq n$. Then ~F is unsatisfiable if and only if F_i is unsatisfiable for each i such that $1 \leq i \leq n$. Thus, in order to test the satisfiability of ~F, it suffices to test the satisfiability of each conjunction F_i.

A decision procedure which determines the satisfiability of a conjunction of T-literals is called a satisfiability procedure for T.

In the next section we will delineate a method for combining satisfiability procedures for (quantifier-free) theories into a satisfiability procedure for their combination. If S and T are theories,

then the combination of S and T is the theory $S \cup T$ whose set of nonlogical symbols is the union of the sets of the nonlogical symbols of S and T and whose set of nonlogical axioms is the union of the sets of nonlogical axioms of S and T.

2.2. Examples of Equality Propagation

In the sequel R denotes the (quantifier-free) theory of real numbers under + and $\leq$, A the (quantifier-free) theory of arrays under store and select, L the (quantifier-free) theory of list structures under cons, car, cdr and atom and EQ the (quantifier-free) theory of equality with uninterpreted function symbols. These theories are treated in more detail in the following chapters. For the moment let us assume that we are given satisfiability procedures, denoted S(R), S(A), S(L) and S(EQ), respectively, for these theories. Besides we assume that each satisfiability procedure has the ability to detect certain equalities which are a logical consequence of the input formula. Of course, this is no restriction since determinining whether a formula G follows from F can be done by checking whether F & ~G is unsatsifiable (and this can be done by any satisfiability procedure). The essential point is, that a satisfiability procedure should have the ability to detect certain equalities efficiently.

In this section we illustrate how to combine S(R), S(A), S(L) and S(EQ) into satisfiability procedures for the combination of individual theories. Consider the conjunction F

$$x \leq y \;\&\; y \leq x+car(cons(0,x)) \;\&\; f(h(x)-h(y))=c \;\&\; f(0)\neq c.$$

This formula contains the arithmetic symbols +, $\leq$ and the arithmetic constant 0, the list symbols car and cons and uninterpreted function symbols f and h and thus falls within the theory $R \cup L \cup EQ$. Terms containing symbols of different theories are also called "mixed" terms. Since the formula above contains "mixed" terms, none of the satisfiability procedures S(R), S(L) or S(EQ) is applicable to it.

The first step in processing F is to make F homogeneous. We

construct three conjunctions F_R, F_L and F_{EQ} such that F_R contains only arithmetic literals, F_L only list expressions, F_{EQ} only uninterpreted function symbols and variables and such that F is satisfiable if and only if F_R & F_L & F_{EQ} is satisfiable. We do this by introducing new variables to replace terms of the wrong type and adding equalities defining these variables. The second conjunct in the formula above, for instance, would be an arithmetic literal except that it contains the term car(cons(0,x)). Therefore car(cons(0,x)) is replaced by a new variable, say v_1, and the equality v_1=car(cons(0,x)) is added to the conjunction. By continuing in this manner we eventually obtain the following three conjuncts:

F_R	F_L	F_{EQ}
$x \leq y$	$v_1=car(cons(v_2,x))$	$f(v_3)=c$
$y < x+v_1$		$v_4=h(x)$
$v_2=0$		$v_5=h(y)$
$v_3=v_4-v_5$		$f(v_2)\neq c$

Now we can test the satisfiability of each of the conjuncts by using the appropriate satisfiability procedure. In our example we find that each of the conjuncts is satisfiable for itself, whereas the original conjunction is unsatisfiable as one easily sees. Therefore some interaction between the satisfiability procedures has to take place. The individual decision procedures interact by "propagating equalities" (hence the name of the technique). Whenever some decision procedure deduces a new equality between variables it transmitts this equality to the other theories. Continuing in this way we may detect some inconsistency. In the example above S(L) deduces that $v_1=v_2$. The equality is propagated and added as a new conjunct to the other formulas. Then S(R) deduces x=y and propagates it. S(EQ) in continuation deduces $v_4=v_5$ which enables S(R) to infer that $v_2=v_3$. Finally S(EQ) detects an inconsistency and returns "unsatisfiable". The process of deducing and propagating equalities is shown below.

F_R	F_L	F_{EQ}
$x \leq y$	$v_1=car(cons(v_2,x))$	$f(v_3)=c$
$y \leq x+v_1$		$v_4=h(x)$
$v_2=0$		$v_5=h(y)$
$v_3=v_4-v_5$		$f(v_2)\neq c$
---	---	---
	$v_1=v_2$	
$x=y$		
		$v_4=v_5$
$v_2=v_3$		
		"unsatisfiable"

It is clear that if one of the conjunctions F_R, F_L or F_{EQ} becomes unsatisfiable, the original conjunction must be unsatisfiable too. The converse, that the original formula is satisfiable if the equality propagation process detects no inconsistency, need not hold as the following example shows.

Let F be the formula

$$select(store(v,i,e),j)=x \;\&\; select(v,j)=y \;\&\; \sim x \leq e \;\&\; \sim x \leq y.$$

Here v denotes a (one-dimensional) array, store(v,i,e) denotes the array with i-th component e and with j-th component select(v,j) for $j \neq i$ and select(v,j) denotes the element at location j in the array v. F is an $R \cup A$-formula. Splitting the formula into its homogeneous parts we obtain

F_A	F_R
$select(store(v,i,e),j)=x$	$\sim x \leq e$
$select(v,j)=y$	$\sim x \leq y$

Both formulas are satisfiable and none of the two conjunctions entails an equality between variables. Therefore propagation of equalities

does not yield an inconsistency although the original formula is unsatisfiable. However, the conjunction F_A implies a disjunction of equalities between variables, namely x=e v x=y. In this situation case-splitting is required. Each equality appearing in the disjunction is adjoined to F_R alone. First, we add x=e as a new conjunct to F_R and S(R) returns "unsatisfiable". Then we add x=y to F_R and again S(R) returns "unsatisfiable". In both cases S(R) has detected an inconsistency, thereby showing that the original formula is unsatisfiable. We are ready now to formulate the general equality propagation procedure.

2.3. The Equality Propagation Procedure

Let S and T be disjoint, decidable, stably-infinite, quantifier-free theories. (A theory is called stably-infinite if any quantifier-free formula in the theory has an infinite model if it has any model). Thus S and T have no common nonlogical symbols. Suppose furthermore, that we are given satisfiability procedures for both S and T. Besides, we presuppose that the satisfiability procedures have the additional ability to determine whether an equality x=y between variables follows from the input formula F. This is no restriction since deciding whether x=y follows from F is equivalent to determining whether F & x≠y is unsatisfiable (and each satisfiability procedure can do this). However, this method is very costly and for specific satisfiability procedures better methods of detecting equalities between variables can be found. We will describe a method, called equality propagation, which combines the satisfiability procedures for S and T into a satisfiability procedure for the combination S∪T of S and T. We describe the method for the combination of two theories but the algorithm can easily be generalized to more than two theories. The method is due to Nelson and Oppen [18].

Equality propagation procedure

Input: F, a conjunction of $S \cup T$-literals.

Output: "Unsatisfiable" if F is unsatisfiable (in $S \cup T$), "satisfiable" otherwise.

Method: We make use of the procedure split(F) described below. The algorithm is as follows.

1. $(F_S, F_T) := \text{split}(F)$.
2. Using the satisfiability procedures for S and T respectively decide whether F_S or F_T is unsatisfiable. If either F_S or F_T is unsatisfiable return "unsatisfiable".
3. [Equality propagation] If either F_S or F_T implies some equality between variables not implied by the other, then add the equality as a new conjunct to the one that does not imply it. Go to step 2.
4. [Case split necessary] If either F_S or F_T implies a disjunction $u_1=v_1 \vee \ldots \vee u_n=v_n$ of equalities between variables, without implying one of the equalities alone, then apply the procedure recursively to the n formulas $F_S \& F_T \& u_1=v_1$, ..., $F_S \& F_T \& u_n=v_n$. If any of these formulas are satisfiable, return "satisfiable". Otherwise return "unsatisfiable".
5. Return "satisfiable".

Splitting a formula into its homogeneous parts

Input: F, a conjunction of $S \cup T$-literals.

Output: A pair of formulas (F_S, F_T) such that F_S is a conjunction of S-literals, F_T a conjunction of T-literals and F is satisfiable if and only if F_S & F_T is satisfiable.

Method: The heart of the method is to replace terms of the wrong type by new variables. The algorithm uses the procedure tsymbol(F). L(S) denotes the language of a theory S, lit(F) denotes the set of literals ocurring in F. The algorithm is as follows.

```
split(F):
    F_S:=true
    F_T:=true
    while lit(F)≠∅ do
      begin
        select G∈lit(F)
        Lit(F):=lit(F)-{G}
        p:=tsymbol(G)
        wlg assume p∈L(S)
           otherwise interchange S and T in
            begin
              G':=G
              while G' contains some subterm t which is not
                                an S-term do
                  begin
                    G':=result of replacing each occurence
                        of t in G by a new variable v

                    lit(F):=lit(F)∪{v=t}
                  end
              add G' as new conjunct to F_S
            end
      end
    return (F_S,F_T)
```

```
tsymbol(F)
    if F=p(t1,...tn)
        then p
    elseif F=~G
        then tsymbol(G)
```

2.4. Analysis of the Algorithm

The complexity of the algorithm depends on the complexity of the individual decision algorithms and is dominated by the maximum of the complexities of the satisfiability procedures for S and for T. Detailed results may be found in Oppen [20].

The equality propagation in step 3 may be realized as follows. Take an equality x=y and determine whether it follows from F_S or F_T. If it follows from one formula but not from the other add it to the one which does not imply it. Otherwise take some other equality and repeat the process. The process terminates since there is only a finite number of equalities between variables. In practice more efficient methods of propagating equalities are possible. Specific methods depend on the particular theories under consideration.

A similar strategy is applicable in step 4 since there are also only finitely many disjunctions of equalities between variables. Step 4 is very costly and the fact whether case-splitting is required or not plays an important role. Case-splitting may only be caused by so-called non-convex formulas. A formula is called non-convex if it implies some disjunction of equalities between variables $u_1=v_1 \vee \ldots \vee u_n=v_n$ without implying one of the equalities alone, otherwise a formula is called convex. A theory S is called convex if every conjunction of S-literals is convex. If we combine convex theories case-splitting is not required. It might also be fruitful to analyze whether certain strategies of selecting the disjunctions minimize the expenses for case-splitting.

In Chapter 3 and Chapter 4, respectively, we will prove that the theories R and E are convex. The example in Section 2.2. shows that A is non-convex.

Opppen [20] proves that if S and T are convex (quantifier-free) theories with polynomial satisfiability problems, then testing the satisfiability of a conjunction of $S \cup T$-literals, by using the method described above, also requires polynomial time.

2.5. Correctness of the Procedure

In this section we give a proof of correctness of the procedure. Our proof is based on Robinson's Consistency Theorem and is simpler than the proof given by Nelson and Oppen [18]. Besides, Lemma 1 in Nelson and Oppen [18, p.253] is not correct.

The algorithm always terminates since each repetition of step 3 or recursive call in step 4 adjoins an equality to either F_S or F_T which was not a logical consequence of the formula before. This can happen at most n-1 times where n is the number of variables appearing in F_S or F_T initially.

It is clear that the procedure is correct if it returns "unsatisfiable". We have to prove that the procedure is also correct if it returns "satsifiable". The proof requires the lemma given below.

A parameter of a formula F is any nonlogical symbol which occurs in F or any variable which occurs free in F. For example, the parameters of $x \prec y \vee \exists\, z{:}f(z) \prec g(z)$ are x, y, f, g and $\prec$. A formula is called simple if its only parameters are variables. For example, $x=y \vee y \neq z$ and $\exists\, x{:}x \neq y$ are simple formulas but $x \prec y$ and $\exists\, x{:}f(x)=x$ are not. An unquantified simple formula is a propositional formula whose atomic formulas are equalities between variables. The next lemma characterizes quantified simple formulas.

Lemma 1: Let F be a quantified simple formula. Then there is some unquantified simple formula G such that F and G are equivalent in each infinite interpretation I. G may be chosen so that $V(G) \subset V(F)$. (V(G) denotes the set of all nonlogical symbols which occur in G).

Proof: Suppose F is of the form $\exists\, x{:}H$, where H is a formula such that the variable x occurs free in H. Let H' be the formula resulting

from H by first replacing each equality x=x by "true" and replacing each negation of an equality x≠x by "false", and by replacing any remaining equality involving x by "false". Let F' be the formula $(H' \vee H[x/v_1] \vee \ldots \vee H[x/v_n])$, where $V:=\{v_1,\ldots v_n\}= V(F)-\{x\}$. (Note that replacing x by some other variable v_i may cause some bounded occurence of v_i to be renamed. This does not effect the considerations below).

We claim that F and F' are equivalent in each infinite interpretation I. The proof proceeds in two steps:

1) We show that every infinite model of F is a model of F'. Assume that I is a model of F (abbreviated by $\models_I F$) and that I is infinite. We have

   $\models_I F$ iff $\models_I \exists x:H$ iff for some element a in the universe $\models_{I'} H$, where I' is such that $I(y)=I'(y)$ for $y\neq x$ and $I(x)=a$.

   We distinguish two cases:

   a) $I'(v_i)=I'(x)$ for some i, such that $1\leq i\leq n$.
   Then $\models_{I'} H$ implies that $\models_{I'} H[x/v_i]$. Since $I(v)=I'(v)$ for all variables in $H[x/v_i]$ we get $\models_I H[x/v_i]$. The assertion $\models_I F'$ follows immediately.

   b) $I'(x)\neq I'(v_i)$ for all i, such that $1\leq i\leq n$.
   In this case we can show that $\models_{I'} H'$ which again yields $\models_I F'$.

2) We now show that every infinite model of F' is a model of F. Assume that I is an infinite model of F'. Again we distinguish two cases in order to show that I satisfies F.

   a) $\models_I H[x/v_i]$ for some i, such that $1\leq i\leq n$.
   Define $I'(y):=I(y)$ for $y\neq x$ and $I'(x):=I(v_i)$. It is readily seen that I' satisfies H. This implies that I satisfies $\exists x:H$ since I' differs from I only in x.

   b) Assume that H' is valid in I.
   Define $I'(y):=I(y)$ for $y\neq x$ and $I'(x):=a$, where a is such that $I(v_i)\neq a$, for all i such that $1\leq i\leq n$. It is possible to select such an element a because the universe is infinite. Looking at the definition of H' it is easily seen that I' is a model for H. Thus I is a model for F.

By repeatedly eliminating quantifiers in that manner we eventually obtain a simple quantifier-free formula G which is equivalent to F in each infinite interpretation and has the additional property that $V(G) \subset V(F)$.

The proof of correctness is based on the following theorem.

Theorem 2: (Robinson's Consistency Theorem)

Let F_1 and F_2 be formulas. $F_1 \& F_2$ is unsatisfiable if and only if there is some formula A such that $V(A) \subset V(F_1) \cap V(F_2)$ and $F_1 \models A$ and $F_2 \models \sim A$.

For a proof of the theorem see Hermes [12, chapter VIII].

We now complete the proof of correctness. First consider the case that the procedure returns "satisfiable" from step 5. This signifies that F_S and F_T are both satisfiable and convex and imply the same equalities between variables. Assume that $F_S \& F_T$ is unsatisfiable. By Theorem 2 there must be some formula A such that $F_S \models A$, $F_T \models \sim A$ and $V(A) \subset V(F_S) \cap V(F_T)$: S and T are disjoint theories. Therefore A is a simple formula. (Hence we know that A contains only variables). In any infinite model A is equivalent to A', where A' is unquantified and $V(A') \subset V := V(F_S) \cap V(F_T)$ (by Lemma 1).

Define $E_1 := \{x=y : x \in V, y \in V, x=y \text{ follows from } F_S\}$ and $E_2 := \{x \neq y : x \in V, y \in V, x=y \text{ does not follow from } F_S\}$. Let E be the conjunction of all literals in E_1 and E_2, and let E' be the conjunction of all literals in E_2. We claim that either A' follows from E or $\sim A'$ follows from E and that both $F_S \& E'$ and $F_T \& E'$ are satisfiable.

First we prove that either A' or $\sim A'$ follows from E. We "evaluate" A' by replacing any literal of E_1 or E_2 by "true" and any other literal by "false". In that way we can decide whether $E \models A'$ or $E \models \sim A'$. Without loss of generality we assume that $E \models A'$.

Next we prove that $F_S \& E'$ is satisfiable, the assertion that $F_T \& E'$ is satisfiable is handled analogously. Assume that $F_S \& E'$ is unsatisfiable. This is equivalent to saying that $F_S \models \sim E'$, or otherwise stated that $F_S \models \sim(v_1 \neq u_1 \& \ldots \& v_k \neq u_k)$, where the literals $v_i \neq u_i$ are the literals in E'. We transform this further and get

$F_S \models (v_1=u_1 \vee \ldots \vee v_k=u_k)$. Since F_S is convex there has to be some i such that $1 \leq i \leq k$ and $F_S \models v_i=u_i$. This is a contradiction, thus F_S & E' has to be satisfiable.

S and T are stably-infinite theories. Thus there are infinite models I and J for F_S & E and F_T & E respectively. From $E \models A'$ we obtain that J satisfies A', but this contradicts that J satisfies ~A. Therefore F_S & F_T has to be satisfiable.

By induction on the depth of recursion it follows that the procedure is correct if it returns "satisfiable" from step 4. This completes the proof of correctness.

2.6. Implementation of the Procedure

The procedure has been implemented in Standard LISP on a IBM 370/155. The current implementation employs decision procedures for the quantifier-free theory of arithmetic under addition and order and for the quantifier-free theory of equality with uninterpreted function symbols. These decision procedures are described in Chapter 3 and Chapter 4, respectively. In Chapter 6 examples are given which illustrate the use of the procedure. The appendix contains a documentation of the LISP programs.