Theoretical Computer Science 12 (1980) 291–302
© North-Holland Publishing Company

COMPLEXITY, CONVEXITY AND COMBINATIONS OF THEORIES*

Derek C. OPPEN

Computer Science Department, Stanford University, Stanford, CA 94305, U.S.A.

Communicated by Z. Manna
Received February 1979
Revised September 1979

Abstract. We restrict our attention to decidable quantifier-free theories, such as the quantifier-free theory of integers under addition, the quantifier-free theory of arrays under storing and selecting, or the quantifier-free theory of list structure under **cons, car** and **cdr**. We consider *combinations* of such theories: theories whose sets of symbols are the union of the sets of the symbols of the individual theories and whose set of axioms is the union of the sets of axioms of the individual theories. We give a general technique for determining the complexity of decidable combinations of theories, and show, for example, that the satisfiability problem for the quantifier-free theory of integers, arrays, list structure and uninterpreted function symbols under $+$, $\leqslant$, **store, select, cons, car** and **cdr** is NP-complete. We next consider the complexity of the satisfiability problem for formulas already in disjunctive normal form: why some combinations of theories admit deterministic polynomial time decision procedures while for others the problem is NP-hard. Our analysis hinges on the question of whether the theories being combined are *convex*; that is, whether any conjunction of literals in the theory can entail a proper disjunction of equalities between variables. This leads to a discussion of the role that case analysis plays in deciding combinations of theories.

1. Introduction

In many applications of theorem proving, particularly those involving program verification, program manipulation and program optimization, we would like to be able to very quickly decide formulas or simplify expressions involving the common data structures of programming languages: numbers, arrays, records, list structure, sets, multisets.

The first-order theories of these data structures are either undecidable or of very high complexity. For this reason, most 'practical' theorem provers restrict their attention to quantifier-free formulas over these data structures. Empirically this restriction seems reasonable: it admits a large and useful class of formulas, yet theorem provers which handle this class generally do so reasonably efficiently.

The purpose of this paper is to explore the complexity of reasoning in quantifier-free theories. We are particularly interested in combinations of quantifier-free theories, such as the theory which 'combines' the quantifier-free theories of integers

* This research was supported by the National Research Foundation under contract MCS 78-02835.

under addition, arrays under storing and selecting, and list structure under **car, cdr, cons**. The reason is that the formulas which arise in practice tend to be 'mixed' formulas containing symbols from various theories (formulas such as $A[I+1] < A[I]$) rather than from just one of the base theories. Thus, we are interested in, for instance, the quantifier-free theory of integers, arrays, list structure and uninterpreted function symbols under $+$, $\leq$, **store**, **select, cons, car** and **cdr**. (A decision procedure for this theory is implemented as part of the simplifier in the Stanford Pascal Verifier.)

Implementers of theorem provers for a theory such as this one have generally shied away from implementing an actual decision procedure, and have instead relied on ad hoc techniques designed to catch 'most' cases. There are at least two reasons for this. One is that, until recently, there has been little research done on what a decision procedure looks like for 'mixed' theories such as these (or even if one can exist). The main reason, however, is the common belief that any decision procedure for such a theory must be slow and impractical, that the complexity of such an apparently rich theory must be very high. However, as we shall show, the satisfiability problem for the above theory is in fact (only!) NP-complete.

In Section 2, we describe a nondeterministic procedure for deciding combinations of quantifier-free theories which generalizes the deterministic procedures given in [11, 17, 18]. We analyze the running time of this procedure. In Section 3, we review some existing results on decidability and complexity of various theories. In Section 4, we use the results of the previous sections to analyze the complexity of several combinations of theories. In Section 5, we consider quantifier-free DNF combinations of theories, that is, combinations of theories all of whose formulas are in disjunctive normal form. Some of these theories have polynomial time solutions while others are NP-hard. In Section 6, we discuss the cost of the case-analysis inherent in any (deterministic) implementation of the procedure described in Section 2.

2. Nondeterministic combinations of theories

Assume we have several quantifier-free theories formalized in classical first-order logic with equality, extended to include the three-argument conditional function **if–then–else**. The symbols $=$, $\wedge$, $\vee$, $\neg$, $\supset$, **if–then–else**, $\forall$ and $\exists$ are common to all theories; we call them the *logical symbols*. Each theory is characterized in the usual way by its set of *non-logical symbols* and *non-logical axioms*.

If $\mathcal{S}$ is a theory, then a term is an $\mathcal{S}$-term if each non-logical symbol occurring in the term is a non-logical symbol of $\mathcal{S}$. We define *$\mathcal{S}$-literal* and *$\mathcal{S}$-formula* similarly. (A *literal* is an atomic formula or its negation.) If $\mathcal{S}$ is a theory, then a *decision procedure* for $\mathcal{S}$ is an algorithm for determining whether a formula is valid in $\mathcal{S}$. A *satisfiability program* for $\mathcal{S}$ is a program which determines whether a conjunction $L_1 \wedge \cdots \wedge L_k$ of $\mathcal{S}$-literals is satisfiable in $\mathcal{S}$. (The general decision problem for $\mathcal{S}$ can easily be reduced to this problem.)

Our goal is to construct a (nondeterministic) satisfiability program for the quantifier-free theory whose set of non-logical symbols is the union of the sets of non-logical symbols of the individual theories and whose set of axioms is the union of the sets of axioms of the individual theories. We will assume that we have just two theories; the generalization to more than two is straightforward. We also assume that we have a satisfiability program for each of our quantifier-free theories, and that each theory is *stably-infinite*, that is, that any quantifier-free formula in the theory has an infinite model if it has any model. We will use the name of a theory to denote also its satisfiability program and the conjunction of its axioms.

A formula F *entails* a formula G if G is a logical consequence of F, that is, if $F \supset G$ is a theorem of first-order logic. A formula F entails a formula G *within a theory* $\mathcal{S}$ if $F \supset G$ is a theorem in $\mathcal{S}$. If the context is clear, we will omit specifying the theory – for instance, we will say that $x - y = 0$ entails $x = y$ without specifying 'in the quantifier-free theory of integers under addition'.

A *parameter* of a formula is any non-logical symbol which occurs free in the formula. Thus the parameters of $a = b \vee \forall x\, P(x, f(x)) = c$ are a, b, P, f and c.

A *simple* formula is one whose only parameters are individual variables. For instance, $x \neq y \vee z = y$ and $\forall x\, x \neq y$ are simple, but $x < y$ and $f(x) = y$ are not. Thus an unquantified simple formula is a propositional combination of equalities between individual variables.

A formula F is *non-convex* if there exist variables $x_1, y_1, \ldots, x_n, y_n$, $n \geq 2$, such that $F \supset x_1 = y_1 \vee \cdots \vee x_n = y_n$ but for no i between 1 and n does $F \supset x_i = y_i$. Otherwise, F is *convex*. That is, a formula is non-convex if it entails a disjunction of equalities between variables without entailing any of the equalities alone; otherwise it is convex. For instance, the formula $1 \leq x \leq 2 \wedge y = 1 \wedge z = 2$ is non-convex over the integers because it entails the disjunction $x = y \vee x = z$ without entailing either equality alone.

The proof of the following lemmas appears in [11].

Lemma 1. *If F is any formula, then there exists an unquantified simple formula* Res(F), *the residue of F, which is the strongest simple formula that F entails; that is, if H is any simple formula entailed by F, then* Res(F) *entails H.* Res(F) *can be written so that its only variables are free variables of F.*

Some examples of residues can be found in Table 1.

Lemma 2. *If A and B are formulas whose only common parameters are individual variables, then* $\text{Res}(A \wedge B) \equiv \text{Res}(A) \wedge \text{Res}(B)$.

Notice that the condition of the lemma is satisfied when A and B are from different theories which have no non-logical symbols in common.

Lemma 3. *Let $F_1, F_2, \ldots, F_n$ be simple, convex formulas and V be the set of all variables appearing in any F_i. Suppose that for all x, y in V and for all i, j from* 1 *to n,*

Table 1

Formula	Residue
$x = f(a) \wedge y = f(b)$	$a = b \supset x = y$
$x = \mathbf{store}(v, i, e)[j]$	$i = j \supset x = e$
$x = \mathbf{store}(v, i, e)[j] \wedge y = v[j]$	**if** $i = j$ **then** $x = e$ **else** $x = y$
$x \neq x$	**false**

either both F_i and F_j entail $x = y$, or neither do. Then $F_1 \wedge F_2 \wedge \cdots \wedge F_n$ is satisfiable if and only if each F_i is satisfiable.

2.1. *The nondeterministic satisfiability procedure*

We assume we have two theories $\mathcal{S}$ and $\mathcal{T}$ which have no common non-logical symbols, that we have satisfiability programs for $\mathcal{S}$ and $\mathcal{T}$, and that $\mathcal{S}$ and $\mathcal{T}$ are the axioms for $\mathcal{S}$ and $\mathcal{T}$. We are given an unquantified formula F whose non-logical symbols are among those of $\mathcal{S}$ and $\mathcal{T}$, and wish to determine whether F is satisfiable in the theory $\mathcal{S} \cup \mathcal{T}$, that is, whether $\mathcal{S} \wedge \mathcal{T} \wedge F$ is satisfiable.

Consider first the disjunctive normal form of F. Each disjunct is a conjunction of literals; F is satisfiable if and only if one of these disjuncts is satisfiable. Our first step is therefore to guess which literals in F make up a satisfiable disjunct. Call the conjunction of these literals F'.

F' may contain 'mixed' literals, literals which are neither $\mathcal{S}$-literals nor $\mathcal{T}$-literals. For instance, suppose that $\mathcal{S}$ is quantifier-free Presburger arithmetic, that $\mathcal{T}$ is the quantifier-free theory of list structure and that F' is the single literal $y = \mathbf{car}(x) + 3$. This literal is neither a $\mathcal{S}$-literal nor a $\mathcal{T}$-literal. We wish to divide F' into two formulas, one which can be handled by the satisfiability program for $\mathcal{S}$ and one which can be handled by the satisfiability program for $\mathcal{T}$. That is, we want to construct two formulas F_S and F_T so that F_S is a conjunction of $\mathcal{S}$-literals, F_T is a conjunction of $\mathcal{T}$-literals, and $F_S \wedge F_T$ is satisfiable if and only if F' is. In our example, $y = \mathbf{car}(x) + 3$ is equivalent to $z = \mathbf{car}(x) \wedge y = z + 3$, where z is a new (existential) variable; $z = \mathbf{car}(x)$ is a $\mathcal{T}$-literal and $y = z + 3$ is a $\mathcal{S}$-literal. We can therefore let F_S be $y = z + 3$ and F_T be $z = \mathbf{car}(x)$. (Variables are typeless.) In general, we construct F_S and F_T from arbitrary F' in similar fashion: for each literal appearing in F', if the literal is an $\mathcal{S}$-literal, we add it to F_S; if it is a $\mathcal{T}$-literal, we add it to F_T; otherwise we introduce new variables to replace terms of the wrong 'type' and add equalities defining these variables.

We now wish to determine if $\mathcal{S} \wedge F_S \wedge \mathcal{T} \wedge F_T$ is satisfiable. If $x_1, \ldots, x_k$ are all the variables in F_S and F_T, we guess the equalities and disequalities that hold among the x_i, and let E be a conjunction of equalities and disequalities of variables describing our guess. For instance, if there are four variables x_1, x_2, x_3 and x_4, E might be $x_1 = x_2 \wedge x_3 = x_4 \wedge x_1 \neq x_3$.

We add E to our conjunction and now wish to determine the satisfiability of $\mathcal{S} \wedge F_S \wedge \mathcal{T} \wedge F_T \wedge E$. It suffices to show that $\mathrm{Res}(\mathcal{S} \wedge F_S \wedge \mathcal{T} \wedge F_T \wedge E)$ is satisfiable (that is, not **false**). By Lemma 2, this residue is equivalent to $\text{Res}(\mathcal{S} \wedge F_S \wedge E) \wedge \text{Res}(\mathcal{T} \wedge F_T \wedge E)$. Since E already expresses precisely the qualities and disequalities that hold between the variables, $\text{Res}(\mathcal{S} \wedge F_S \wedge E)$ and $\text{Res}(\mathcal{T} \wedge F_T \wedge E)$ are simple and convex, and entail the same set of equalities among variables. Hence, by Lemma 3, to verify that F' is satisfiable, it suffices to show that both $\text{Res}(\mathcal{S} \wedge F_S \wedge E)$ and $\text{Res}(\mathcal{T} \wedge F_T \wedge E)$ are satisfiable. This will be the case if and only if $F'_S \equiv F_S \wedge E$, which is a conjunction of $\mathcal{S}$-literals, is satisfiable in $\mathcal{S}$, and $F'_T \equiv F_T \wedge E$, which is a conjunction of $\mathcal{T}$-literals, is satisfiable in $\mathcal{T}$. We can use the satisfiability programs of $\mathcal{S}$ and $\mathcal{T}$ to determine their satisfiability.

The essential idea behind this nondeterministic procedure thus is to guess all the qualities that hold between the variables and then to use the individual satisfiability programs to decide whether the formula with these equalities is satisfiable.

2.2. Analysis of the algorithm

What is the running time of this nondeterministic satisfiability procedure? Let n be the length of the incoming formula F.

We can guess F', the satisfiable disjunct of literals of F, in nondeterministic polynomial time. The size of F' is linear in n since F' is a subformula of F.

We now have to construct F_S and F_T. There are at most n subterms in F' of the wrong 'type'; for each such subterm we have to replace it by a new variable and add the equation defining the variable. We can certainly construct F_S and F_T in deterministic time polynomial in n. Each is of length polynomial in n, allowing for the n new variable symbols.

We can guess the equalities that hold between the variables of F' in nondeterministic polynomial time. The size of E is polynomial in n, and so the sizes of F'_S and F'_T are also polynomial in n.

We have thus shown that the problem of constructing the formulas F'_S and F'_T is in NP. The remaining time required is whatever time is required by the satisfiability programs for $\mathcal{S}$ and $\mathcal{T}$ to verify that F'_S and F'_T are satisfiable.

Consider now the satisfiability problem for $\mathcal{S} \cup \mathcal{T}$. Since arbitrary boolean structure is allowed in formulas, the problem is certainly NP-hard. If the problems of determining the satisfiability of conjunctions of $\mathcal{S}$-literals and of $\mathcal{T}$-literals are also in NP, then the satisfiability problem for $\mathcal{S} \cup \mathcal{T}$ is in NP, and hence NP-complete. Otherwise, the complexity of the satisfiability problem is dominated by the complexity of the satisfiability problem for $\mathcal{S}$ or for $\mathcal{T}$.

The results given above for two theories generalize in a straightforward fashion to more than two theories.

The following summarizes these results.

Theorem 1. *Let $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k$ be decidable, stably-infinite, quantifier-free theories with no common non-logical symbols. Then $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \cdots \cup \mathcal{T}_k$ is decidable; if the satisfiability problem for each of the $\mathcal{T}_i$ is in* NP, *then the satisfiability problem for $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \cdots \cup \mathcal{T}_k$ is in* NP *and hence* NP-*complete. Otherwise the complexity of the* *satisfiability problem is dominated by the maximum of the complexities of the satisfiability problems for* $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \cdots \cup \mathcal{T}_k$.

3. Review of existing complexity results

Before using the results of the previous section to analyze the complexity of various combinations of theories, we first summarize some existing results. In the following, the quantifier-free DNF theory is the theory in which every formula is already in disjunctive normal form. This restriction is of interest because the complexity of its satisfiability problem is just the complexity of determining the satisfiability of a conjunction of literals.

3.1. *Theory of integers under addition*

The first order theory was shown decidable by Presburger [15]. Fischer and Rabin [6] prove that the theory has a double-exponential lower bound on nondeterministic time. Oppen [12] proves that the theory has a triple-exponential upper bound on deterministic time; Berman [2] has refined this bound. Reddy and Loverland [16] prove that the bounded quantifier subtheory has a double-exponential upper bound. The satisfiability problems for the quantifier-free theory and the quantifier-free DNF theory are NP-complete; this follows from [4].

An interesting subtheory of quantifier-free Presburger arithmetic is the quantifier-free theory of integers under successor; thus addition of variables is not allowed, only addition of a variable and a constant. This theory is also NP-complete [19]. (However, if disequalities are also not allowed, then one can determine the satisfiability of a formula of length n in the quantifier-free DNF theory in time $O(n^3)$ [14].)

3.2. *Theory of rationals under addition*

The quantifier-free DNF theory has a polynomial time satisfiability problem. This follows from Khachian's recent discovery of a polynomial time algorithm for linear programming ([9]; see also [7]).

3.3. *Theory of equality with function symbols*

A proof of the decidability of the quantifier-free theory appears in [1]. (An example of a valid formula in this theory is $x = y \supset f(x, y) = f(y, x)$.) Nelson and Oppen [10] give an $O(n^2)$ decision procedure for the DNF quantifier-free theory (see also [20–22]). It follows that the satisfiability problem for the quantifier-free theory is NP-complete.

3.4. *Theory of list structure under* **car**, **cdr** and **cons**

There are several possible axiomatizations for this theory:

$$\begin{array}{l}\mathbf{car}(\mathbf{cons}(x, y)) = x\\ \mathbf{cdr}(\mathbf{cons}(x, y)) = y\\ \mathbf{atom}(x) \supset \mathbf{cons}(\mathbf{car}(x), \mathbf{cdr}(x)) = x\\ \mathbf{atom}(\mathbf{cons}(x, y)),\end{array} \tag{1}$$

$$\begin{array}{l}\mathbf{car}(\mathbf{cons}(x, y)) = x\\ \mathbf{cdr}(\mathbf{cons}(x, y)) = y\\ x \neq \mathbf{nil} \supset \mathbf{cons}(\mathbf{car}(x), \mathbf{cdr}(x)) = x\\ \mathbf{cons}(x, y) \neq \mathbf{nil}\\ \mathbf{car}(\mathbf{nil}) = \mathbf{cdr}(\mathbf{nil}) = \mathbf{nil},\end{array} \tag{2}$$

$$\begin{array}{l}\mathbf{car}(\mathbf{cons}(x, y)) = x\\ \mathbf{cdr}(\mathbf{cons}(x, y)) = y\\ \mathbf{cons}(\mathbf{car}(x), \mathbf{cdr}(x)) = x\\ \mathbf{car}(x) \neq x\\ \mathbf{cdr}(x) \neq x\\ \mathbf{car}(\mathbf{car}(x) \neq X,\\ \dots\end{array} \tag{3}$$

Nelson and Oppen [10] show that the satisfiability problem for the quantifier-free DNF theory axiomatized by (1) has an $O(n^2)$ solution, but that the problem for (2) is NP-complete. Oppen [13] gives a linear algorithm for (3). Therefore, for any of these axiom schemata, the quantifier-free theory is NP-complete. The first order theory was shown decidable but not elementary recursive by Oppen [13].

These results generalize easily to data structures with one constructor c and k selector functions $s_1, \dots, s_k$. Such data structures are called *recursively defined data structures.*

3.5. *Theory of arrays under selecting and storing*

The axioms for this theory are as follows:

$$\begin{array}{l}\mathbf{select}(\mathbf{store}(A, I, E), I) = E\\ I \neq J \supset \mathbf{select}(\mathbf{store}(A, I, E), J) = \mathbf{select}(A, J)\\ \mathbf{store}(A, I, \mathbf{select}(A, I)) = A\\ \mathbf{store}(\mathbf{store}(A, I, E), I, F) = \mathbf{store}(A, I, F)\\ I \neq J \supset \mathbf{store}(\mathbf{store}(A, I, E), J, F) = \mathbf{store}(\mathbf{store}(A, J, F), I, E).\end{array}$$

$\mathbf{select}(A, I)$ is the Ith component of the one-dimensional array A. $A[I]$ abbreviates $\mathbf{select}(A, I)$. $\mathbf{store}(A, I, E)$ is the array whose Ith component is E and whose Jth component, for $J \neq I$, is $A[J]$. A two-dimensional array is considered a vector of vectors, so $A[I, J]$ abbreviates $A[I][J]$. The last three axioms are only needed if equalities between array terms are allowed [8].

Downey and Sethi [5] show that the satisfiability problem for the DNF quantifier-free theory is NP-complete.

3.6. *Theory of integers with function symbols*

Shostak [17] shows that the quantifier-free theory of integers with uninterpreted functions under $+$ and $\leq$ is decidable. (This also follows immediately from Theorem 1.)

3.7. *Theory of integers and arrays*

Suzuki and Jefferson [18] show that the quantifier-free theory of arrays and integers under $+$, $\leq$, **store** and **select** is decidable. (This also follows immediately from Theorem 1.) They also extend their results to the quantifier-free theory all of whose formulas are of the form $P \supset Q \wedge PERM(A, B)$, where P and Q are conjunctions of literals over the theory of arrays and integers under $+$, $\leq$, **store** and **select** and A and B are array terms. $PERM(A, B)$ is interpreted to mean that array A is a permutation of array B.

4. Complexity of various combinations of theories

The results given in Section 3 lead immediately to the following corollaries of Theorem 1.

Corollary 1. *The satisfiability problem for the quantifier-free theory of rationals (or integers), arrays, list structure and uninterpreted function symbols under* $+$, $\leq$, **store, select, cons, car** *and* **cdr** *is* NP-*complete.*

A decision procedure for this theory is implemented in the Stanford Simplifier.

Corollary 2. *The satisfiability problem for the quantifier-free theory of integers and arrays under* $+$, $\leq$, **store** *and* **select** is NP-*complete.*

This is the theory considered by Suzuki and Jefferson [18]. It is easy to verify as well that the addition of the *PERM* predicate does not change the NP-completeness.

Corollary 3. *The satisfiability problem for the quantifier-free theory of integers and uninterpreted function symbols under* $+$ *and* $\leq$ *is* NP-*complete.*

This is the theory considered by Shostak [17].

5. Convexity

Since the theories we considered in the previous section were already NP-hard (because of the arbitrary boolean structure allowed in formulas), our analysis of the running time of our nondeterministic procedure could be fairly gross: it sufficed to show that each step required at most nondeterministic polynomial time. But what if we restrict our attention to formulas already in disjunctive normal form, that is, to quantifier-free DNF combinations of theories? The theories then are no longer automatically NP-hard, and so we may be able to improve the upper bound down to polynomial time.

The satisfiability problem for some quantifier-free DNF theories (such as the theory of integers under addition or of arrays under storing and selecting) is already NP-hard. So of course any theory including such a theory must therefore be at least as hard.

However, if we further restrict our attention to quantifier-free DNF theories with deterministic polynomial time satisfiability problems, we might hope that their quantifier-free DNF combinations also admit deterministic polynomial time solutions. For instance, we might consider combinations of the quantifier-free DNF theories of equality with uninterpreted function symbols, and list structure under **car**, **cons** and **cdr** (with axioms (1) or (3)) since each has a deterministic polynomial satisfiability problem. Nelson and Oppen [10] show that the satisfiability problem for the quantifier-free DNF theory of list structure with uninterpreted function symbols has an $O(n^2)$ solution. In fact, as we now show, any combination of theories with polynomial time decision problems also has a polynomial time solution as long as the theories are convex.

Recall that a formula is non-convex if it entails a disjunction of equalities between variables without entailing any of the equalities alone; otherwise it is convex. Define a theory $\mathcal{S}$ to be *convex* if every conjunction of $\mathcal{S}$-literals is convex; otherwise it is *non-convex*.

Some of the theories considered in this paper are convex, others non-convex. The theories of equality with uninterpreted function symbols and of list structure under **car**, **cdr** and **cons** are convex [10]. The theory of rationals under + and $\leqslant$ is convex: the solution set of a conjunction of linear inequalities is a convex set; the solution set of a disjunction of equalities is a finite union of hyperplanes; and a convex set cannot be contained in a finite union of hyperplanes unless it is contained in one of them. The theories of integers under addition and of integers under successor are non-convex. For instance, the formula $1 \leqslant x \leqslant 2 \wedge y = 1 \wedge z = 2$ entails the disjunction $x = y \vee x = z$ without entailing either equality alone. The theory of arrays is non-convex. For instance, the formula $x = \textbf{store}(a, i, e)[j] \wedge y = a[j]$ entails $i = j \vee x = y$. The theory of the reals under multiplication is not convex; for example, $xy = 0 \wedge z = 0$ entails the disjunction $x = z \vee y = z$. The theory of sets is also non-convex; for example, consider $\{a, b, c\} \cap \{c, d, e\} \neq \{\}$.

The results on theories of most interest to us are summarized in Table 2.

Suppose we have two convex theories $\mathcal{S}$ and $\mathcal{T}$, and that for each we have a deterministic polynomial time decision procedure for deciding satisfiability of conjunctions of literals. Then we can decide the satisfiability of a conjunction F in their union in polynomial time by the following procedure (see [11]).

Table 2

Convex Theories	Non-convex Theories
Theory of rationals under addition	Theory of integers under addition
Theory of equality with uninterpreted function symbols	Theory of integers under successor
Theory of list structure (with axiom schemes (1) and (3))	Theory of arrays

Step 1. Construct F_S and F_T from F as in the nondeterministic procedure in Section 2. Then F_S is a conjunction of $\mathcal{S}$-literals, F_T is a conjunction of $\mathcal{T}$-literals, and $F_S \wedge F_T$ is satisfiable if and only if F is.
Step 2. Using the satisfiability procedures for $\mathcal{S}$ and $\mathcal{T}$, check to see if either F_S or F_T is unsatisfiable. If so, return **unsatisfiable** since F must be unsatisfiable.
Step 3. If either F_S or F_T entail some equality between variables not entailed by the other, then add the equality as a new conjunct to the one that does not entail it and go to Step 2.
Step 4. If this step is reached, F is satisfiable.

This procedure is a specialized deterministic version of the non-deterministic procedure given in Section 2. Instead of guessing the equalities that hold between variables, we compute the equalities in Step 3. We can implement Step 3 by determining, for each x and y in the formula, if $x = y$ is entailed by F_S or F_T. We can do this by checking, using the satisfiability procedure for $\mathcal{S}$, if $F_S \wedge x \neq y$ is unsatisfiable; similarly for F_T. The algorithm is a complete satisfiability procedure only if the theories $\mathcal{S}$ and $\mathcal{T}$ are convex (if they are not, F_S or F_T may entail a disjunction of equalities not entailed by the other).

What is the running time of this deterministic procedure? Steps 2 and 3 can be executed at most n times, where n is the length of F, since there can be most n variables in F_S and F_T, and there can be at most $n-1$ non-redundant equalities between n variables. Since $\mathcal{S}$ and $\mathcal{T}$ have polynomial time DNF satisfiability problems, Steps 2 and 3 require polynomial time.

The procedure therefore runs in polynomial time, and leads to the following theorem:

Theorem 2. *Let $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k$ be decidable, convex, stably-infinite, quantifier-free theories with no common non-logical symbols and with deterministic polynomial time* DNF *satisfiability problems. Then $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \cdots \cup \mathcal{T}_k$ has a deterministic polynomial time* DNF *satisfiability problem.*

6. Case splitting

If the theories being combined are non-convex, the nondeterministic procedure given in Section 2 for combining satisfiability programs translates in the obvious fashion into a deterministic procedure. The incoming formula is converted into disjunctive normal form, each disjunct is massaged into one containing no literals of 'mixed' type, a case split is done on all the ways that the variables in each conjunct can be equal, and the individual satisfiability programs are used to determine the satisfiability of each branch of the split.

This simplistic way of combining satisfiability programs is of course rather inefficient: a superior method is given in [11]. However, it is interesting to briefly analyze the running time of this brute force algorithm since it illustrates the importance of case splitting.

Assume that the size of the original formula is n. The size of each disjunct in disjunctive normal form is $O(n)$ since each disjunct is a subformula of the original formula; there may of course be $O(2^n)$ disjuncts.

We now want to convert each disjunct into the form $F_S \wedge F_T$. There are at most n subterms in F of the wrong 'type'; for each such subterm we have to replace it by a new variable and add the equation defining the variable. As before, we can construct $F_S \wedge F_T$ in deterministic polynomial time; its length is $O(n)$. The total number of variables (including newly introduced ones) is at most n.

The next step is the case split on the equalities between the variables appearing in F_S and F_T. The number of branches in the case split will be the number of ways we can partition the set of variables into nonempty disjoint subsets. By definition, this number is $B(n)$, the Bell number of n. For each case split, the corresponding partition of the set of variables can be represented in space $O(n)$.

The satisfiability programs for $\mathcal{S}$ and $\mathcal{T}$ receive this partition and one of the formulas F_S or F_T, and determine the satisfiability of the formula F_S or F_T under this partition.

Let us suppose that the theories $\mathcal{S}$ and $\mathcal{T}$ admit exponential-time decision procedures (as do all the quantifier-free theories we have used as examples in this paper). Then the whole deterministic case-splitting procedure described above runs in exponential time except for the factor $B(n)$, the number of case splits. And unfortunately $B(n)$ grows faster than 2^n.

$B(n)$ grows faster than 2^n or $(n/2)!$ but slower than $n!$ $B(0) = B(1) = 1$; $B(2) = 2$, $B(3) = 5$, $B(4) = 15, \ldots$. de Bruijn [3] shows that

$$\ln(B(n)) = n * \ln(n) - n - n * \ln(\ln(n)) + O(n * \ln(\ln(n))/\ln(n));$$

he actually gives several more terms in the expansion. Since $\ln(n!) = n * \ln(n) - n + O(\ln(n))$, the quantity $n!/B(n)$ is approximately $\ln^n(n)$.

So the satisfiability problem is dominated by $B(n)$. Bell numbers appear naturally in any algorithm which does case analysis on equalities between variables. This is the case for the algorithms given in [17, 18, 11]. Suzuki and Jefferson [18] prove that their decision procedure is O(n!); the bound given above improves on their bound.

Acknowledgment

I acknowledge with thanks the helpful conversations I have had with Vaughan Pratt and Chris Goad.

References

[1] W. Ackerman, *Solvable Cases of the Decision Problem* (North-Holland, Amsterdam, 1954).
[2] L. Berman, The complexity of logical theories, *Theoret. Comput. Sci.* **11** (1980) 71–77.
[3] N. de Bruijn, *Asymptotic Methods in Analysis* (North-Holland, Amsterdam, 1970) 103–108.
[4] I. Borosh and L.B. Treybig, Bounds on positive integral solutions of linear diophantine equations, *Proc. AMS* **55** (2) (1976) 299–304.
[5] P. Downey and R. Sethi, Assignment commands with array references, *J. ACM* **25** (4) (1978).
[6] M. Fischer and M. Rabin, Super-exponential complexity of Presburger arithmetic, *Proc. Symposium on the Complexity of Real Computation Processes* (1973).
[7] P. Gacs and Laszlo Lovasz; Khachian's algorithm for linear programming, CS Report STAN-CS-79-750, Stanford University (1979).
[8] D.M. Kaplan, Some completeness results in the mathematical theory of computation, *J. ACM* **15** (1968).
[9] Khachian, Polynomial algorithm for linear programming, manuscript, Computing Center, Academy Sciences USSR, Moscow (1978).
[10] C.G. Nelson and D.C. Oppen, Fast decision algorithms based on congruence closure, *J. ACM*, to appear.
[11] C.G. Nelson and D.C. Oppen, Simplification by cooperating decision procedures, *ACM Trans. Programming Languages and Systems* **2** (1) (1979).
[12] D.C. Oppen, A $2^{2^{2^n}}$ upper bound on the complexity of Presburger arithmetic, *J. Comput. System Sci.* **16** (3) (1978)..
[13] D.C. Oppen, Reasoning about recursively defined data structures, *J. ACM*, to appear.
[14] V. Pratt, Two easy theories whose combination is hard, manuscript (1977).
[15] M. Presburger, Uber die Vollstandigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt, *Comptes Rendus du 1er Congres des Mathematiciens des Pays Slavs* (1929).
[16] C.R. Reddy and D.W. Loveland, Presburger arithmetic with bounded quantifier alternation, *Proc. 10th Annual ACM Symposium on Theory of Computing* (1978).
[17] R. Shostak, A practical decision procedure for arithmetic with function symbols, *J. ACM* **26** (2) (1979) 351–360.
[18] N. Suzuki and D. Jefferson, Verification decidability of Presburger array programs, *Proc. Conference on Theoretical Computer Science*, University of Waterloo (1977).
[19] T. Chan, An algorithm for checking PL/CV arithmetic inferences, Technical Report 77–326, Department of Computer Science, Cornell University (1977).
[20] P. Downey, R. Sethi and R. Tarjan, Variations on the common subexpression problem, *J. ACM*, to appear.
[21] D. Kozen, Complexity of finitely represented algebras, *Proc. Annual ACM Symposium on Theory of Computing* (1977).
[22] R. Shostak, An algorithm for reasoning about equality, *Comm. ACM* (July 1978) 583–585.