Automatic Decidability

Christopher Lynch and Barbara Morawska
Department of Mathematics and Computer Science Box 5815
Clarkson University, Potsdam, NY 13699-5815, USA
E-mail: clynch@clarkson.edu,morawskb@clarkson.edu *

Abstract

We give a set of inference rules with constant constraints. Then we show how to extend a set of equational clauses, so that if the application of these inference rules halts on these clauses, then the theory is decidable by applying a standard set of Paramodulation inference rules. In addition, we can determine the number of clauses generated in this decision procedure. For some theories, such as the theory of lists, there are $O(n \times lg(n))$ clauses. For others it is polynomial. And for others it is simply exponential such as the theory of (extensional) arrays.

1 Introduction

Verification problems often involve proving the validity of a universally quantified equational implication[1] modulo a background theory, such as the theory of lists or arrays or some other data structure. For example, in the theory of lists, we may want to prove that $\forall x_1, x_2, x_3, x_4, x_5, x_6(x_1 = cons(x_3, x_4) \wedge x_2 = cons(x_5, x_6) \wedge car(x_1) = car(x_2) \wedge cdr(x_1) = cdr(x_2) \rightarrow x_1 = x_2)$. By skolemization, validity of a universally quantified equational implication modulo a theory can be shown equivalent to proving the unsatisfiability of a ground conjunction of equations and disequations modulo that theory, which is the problem we will concern ourselves with in this paper.

In particular we have developed a set of inference rules P that takes a theory E in first order logic with equality, represented as a set of clauses. If our procedure halts, then it shows that the standard set of Paramodulation inference rules halts for any ground set of equations and disequations along with E[3]. Therefore, the standard set of inference rules is a decision procedure for E. Unfortunately, it is undecidable, in general, to decide the validity of any set of ground equations and disequations modulo E. Therefore, it cannot be the case that P will always halt. However, it often does halt, and we give examples of theories where it does halt, such as the theory of lists and the theory of arrays.[2]

When P halts, it not only shows that the theory is decidable, it gives a bound on the number of clauses that will be generated in the inference procedure. For example, the theory of lists is an example of a class of theories that we call *noncyclic*. In the theory of lists, only $n \times log(n)$ clauses are generated, where n is the size of the set of ground equations and disequations. In any theory where P halts and no clause has more than one equation, then only polynomially many clauses are generated by the inference rules. In other theories where P halts, there are simply exponentially many clauses generated.

This paper is based heavily on the work of [1], which provides a good background for the usefulness of this work. Our paper is basically an automation of their result, with some additional complexity results. They showed that the standard Paramodulation inference will decide the validity of a conjunction of ground equations and disequations modulo the theory of lists and (extensional) arrays. They proved the completeness of their method. Our improvement of their result is to give an automated method of proving that the standard Paramodulation inference rules will decide such theories. In addition, our inference rules P automatically give a bound on the number of clauses generated by the decision procedure.

Our procedure P is itself a set of inference rules, applied to clauses containing *constant constraints*, which is a conjunction of constraints of the form $const(t)$, true if and only if t is a constant. Our inference rules are an extension of the congruence closure algorithm given in [4] (see also [7]). The paper [4] proves a complexity bound of $O(n \times lg(n))$, based on the way the precedence of the constants is chosen. We use the same technique as they do, and show how to

*This work was supported by NSF grant number CCR-0098270 and ONR grant number N00014-01-1-0435. Contact the first author at (315) 268-2384, fax number (315) 268-2371.

[1]Predicates can be encoded with equations. There may be function symbols.

[2]See [1] to show how this also applies to the theory of extensional arrays.

extend the congruence closure results to *noncylic* theories. This gives us better complexity results than those given in [1].

Our results are in the spirit of automated decidability and complexity results of theories saturated by ordered paramodulation [5].

2 Preliminaries

We use standard definitions as in [2].

We consider *terms* built out of function symbols, constants and variables. We will have a special designated set of constants, called *extension constants*, which can only be created by the **Extension** inference rule given later in this section. If s and t are terms, then $s \approx t$ is an *equation*. An equation between two extension constants is called an *extension equation*. The negation of an equation is called a *disequation* and written in the form $s \not\approx t$. A *clause* is a disjunction of equations and disequations. Predicates can be encoded by equations, by treating predicate symbols as function symbols, and adding a new constant symbol $\top$. If A is a predicate, then we encode A as the equation $A \approx \top$, and $negA$ is encoded as $A \neq \top$. Therefore, for simplicity, we assume that $\approx$ is our only predicate symbol. If E is a set of clauses, then we designate the set of function symbols and constants appearing in E as Σ_E.

An equation $s \approx t$ is called a *real equation* if s is not $\top$ and t is not $\top$. A clause which contains at most one real equation is called a *single equation clause*. A set of single equation clauses is called a *single equation theory*. Note that any form of disequations are allowed in a single equation clause.

We define a substitution as usual to be a mapping from variables to terms which is almost everywhere the identity, and we also identify a substitution with the homomorphic extension of itself. In an abuse of notation, we also define a substitution whose domain is the set of constants and write $C[c \mapsto d]$ to mean that every occurrence of c in C is replaced by d.

We assume a reduction ordering $\succ$, which is total on ground terms, for example, the lexicographic path ordering[6]. This ordering is extended to equations by considering them as multisets of terms, and then to clauses by considering them as multisets of equations. We define the ordering in such a way that the negative literals are always slightly bigger than the positive ones, and the smallest terms are the constant $\top$ and the extension constants. Since the ordering is total, we can always orient the ground equations, hence we treat them often as rewrite rules and write $s \rightarrow t$ instead of $s \approx t$.

We want to to decide the satisfiability of a set of ground equations and disequations, modulo a theory. First we provide an algorithm to make the ground equations and disequations *flat*.

Definition 1 *Define a new set of constants called extension constants. A* flat f-rule *is a rule of the form* $f(c_1, \cdots, c_n) \rightarrow c_0$*, where each* c_i *is an extension constant and* $n \geq 0$*. An equation is* flat *if it is a flat* f*-rule or an extension equation. A disequation is* flat *if it contains only extension constants.*

Let S be a set of equations and disequations. Then S can be flattened by repeated application of the following **Extension** rule[4]:

$$\frac{S \cup \{A[t]\}}{S \cup \{A[c],\, t \rightarrow c\}}$$

where c is a fresh extension constant, $A[t]$ is not flat, and t is either

1. a constant a which is not an extension constant, or
2. of the form $f(c_1, \ldots, c_n)$ such that $c_1, \ldots, c_n$ are extension constants,

The number of rules created by this will be the number of function (and constant) symbols in S.

From now on, we assume that S is a set of ground, flat equations.

3 Saturation

In this section we will define a sound and complete set of inference rules for determining the unsatisfiability of a set of clauses. First we define some closure rules in Figure 1, and some deletion rules in Figure 2. This are the same as the standard inference rules given in [3], except for an additional rule called **Orient**.

The inference rules use a *selection rule*, which is a function that is applied to a clause C and returns a set of one or more literals in C. We write $Sel(C)$ as the set of literals *selected* in C. The selection rule is very important for our purposes, because it restricts the literals needed to be involved in an inference, and a clever selection rule may yield a decision procedure where another selection rule does not. For completeness, the selection rule must have the following properties:

Selection Rule

For all clauses C, $Sel(C)$ contains

1. a negative literal in C ("don't care non-determinism") or
2. all maximal literals in C wrt $\prec$.

Right Paramodulation:

$$\frac{G \cup \{u[s'] \approx v \vee C, s \approx t \vee D\}}{G' \cup \{(u[t] \approx v \vee C \vee D)\sigma\}}$$

where $G' = G \cup \{u[s'] \approx v \vee C, s \approx t \vee D\}$, $\sigma = mgu(s, s')$, $u[s'] \approx v$ is selected in its clause, $s \approx t$ is selected in its clause, $u[s']\sigma \not\preceq v\sigma$, $s\sigma \not\preceq t\sigma$, and s' is not a variable.

Left Paramodulation:

$$\frac{G \cup \{u[s'] \not\approx v \vee C, s \approx t \vee D\}}{G' \cup \{(u[t] \not\approx v \vee C \vee D)\sigma\}}$$

where $G' = G \cup \{u[s'] \not\approx v \vee C, s \approx t \vee D\}$, $\sigma = mgu(s, s')$, $u[s'] \approx v$ is selected in its clause, $s \approx t$ is selected in its clause, $u[s']\sigma \not\preceq v\sigma$, $s\sigma \not\preceq t\sigma$, and s' is not a variable.

Equational Resolution

$$\frac{G \cup \{u \not\approx v \vee C\}}{G' \cup \{C\sigma\}}$$

where $G' = G \cup \{u \not\approx v \vee C\}$, $\sigma = mgu(u, v)$ and $u \not\approx v$ is selected.

Equational Factoring

$$\frac{G \cup \{s \approx t \vee s' \approx u \vee C\}}{G' \cup \{(s \approx u \vee t \not\approx u \vee C)\sigma\}}$$

where $G' = G \cup \{s \approx t \vee s' \approx u \vee C\}$, $\sigma = mgu(s, s')$, $s \approx t$ is selected, $t\sigma \not\preceq u\sigma$, $s\sigma \not\preceq t\sigma$ and $s'\sigma \not\preceq u\sigma$.

Orient

$$\frac{G \cup \{c \approx d\}}{G[c \mapsto d]}$$

where c and d are extension constants and $depth(c) \leq depth(d)$. Then we set the precedence $c \succ d$ and set $depth(d) = max\{depth(d), 1 + depth(c)\}$.

Figure 1. Rules for Closure

Subsumption:

$$\frac{G \cup \{C, C'\}}{G \cup \{C\}}$$

where $C\sigma \subseteq C'$ for some substitution σ.

Simplification:

$$\frac{G \cup \{C[l'], l \approx r\}}{G \cup \{C[r\sigma], l \approx r\}}$$

where $l' = l\sigma$, $l\sigma \succ r\sigma$, and $C[l'] \succ (l\sigma \approx r\sigma)$.

Tautology Deletion:

$$\frac{G \cup \{t \approx t \vee C\}}{G} \quad , \quad \frac{G \cup \{s \approx t \vee s \not\approx t \vee C\}}{G}$$

Figure 2. Rules for Deletion

Closure rules are necessary for completeness. Deletion rules are optional but recommended, because they decrease the search space. All of the closure rules except for **Orient** add a new clause. **Orient** is also exceptional, because it is a global, rather than a local rule. We require that **Orient** is performed eagerly. In other words, as soon as an extension equation is created, **Orient** is performed immediately.

We will define a function called $depth$, which will be applied to extension constants. When an extension constant c is created by the **Extension** rule, we set $depth(c) = 0$. The **Orient** rule will modify the depth of an extension constant. The precedence of extension constants is not determined by the original precedence. Precedence between these constants will be determined by the **Orient** rule, by analyzing depth. It will be determined in such a way that every extension constant has only logarithmically many extension constants smaller than it, just as in the union-find algorithm.

An *inference system* I is a set of closure and deletion rules. A (possibly infinite) set of clauses T is considered to be *saturated* by an inference system I if any application of a closure rule from I to T results in a clause C such that either

1. C is in T, or
2. a sequence of deletion rules from I to $T \cup \{C\}$ will result in T (i.e., C could be removed, so the inference is not necessary).[3]

A set of clauses S has the *finite saturation property* if T is a saturation of S, and T is finite. If T is a set of clauses,

[3]This nonstandard definition of redundancy is needed in the next section.

then T' is a *saturation of T by I* if T' is saturated by I and T' is obtained by a (possibly infinite) sequence of closure and deletion rules from I applied to T. This mimics what happens in a real theorem prover, where closure rules **must** be applied and deletion rules **may** be applied. In practice, it is better to apply deletion rules whenever possible to cut down on the size of the search space.

Let I be the inference system consisting of the closure rules in Figure 1. and the deletion rules of Figure 2. We have the following theorem[3].

Theorem 1 *A set T of clauses is unsatisfiable if and only if the empty clause is in a saturation of T by I if and only if the empty clause is in all saturations of T by I.*

In particular, we will consider a set of clauses E that is already saturated by I, and a set of flat ground equations and disequations S. We would like to know whether I halts on $E \cup S$.

4 Meta-saturation

In this section, we consider E to be a single equation theory and show a sufficient condition for $E \cup S$ to have the finite saturation property for any set S of ground equations and disequations. To do this, we define an inference system P, and then a certain set of equations is saturated by P. If this set of equations has the finite saturation property under P, then any such $E \cup S$ will have the finite saturation property under S. This means that any saturation of $E \cup S$ will halt, and moreover we show that it halts in polynomial time. The case where E is not a single equation theory is defined in the next section by a different inference system, but in that case it is only shown to halt in exponential time.

First we will define a set of constraint equations and disequations G_E, that will in a sense represent any possible set of flat, ground equations and disequations S.

The only constraints that we are going to impose on terms in the equations in G_E, are *constant constraints*. An *atomic constant constraint* is of the form $const(t)$, and it is true if t is an extension constant. A *constant constraint* is a conjunction of zero or more atomic constant constraints, which is true if each atomic constant constraint in the conjunction is true. A substitution σ *satisfies* a constant constraint φ if $\varphi\sigma$ is true. A *constrained clause* is of the form $C\llbracket \varphi \rrbracket$, where C is an (unconstrained) clause and φ is a constant constraint. $C\sigma$ is an *instance* of $C\llbracket \varphi \rrbracket$ if σ is a substitution satisfying φ. Orderings on constrained clauses can be defined so that a clause C is bigger than a clause D if all ground instances of C are bigger than all ground instances of D.

G_E is defined as the following set of equations:

$$G_E := \{f(x_1, \ldots, x_n) \approx x_0 \llbracket const(x_0) \wedge \cdots \wedge const(x_n) \rrbracket | f \in \Sigma_E\} \cup \{x \approx y \llbracket const(x) \wedge const(y) \rrbracket, x \not\approx y \llbracket const(x) \wedge const(y) \rrbracket\}$$

Now we present an inference system P for $E \cup G_E$, that will simulate closure rules of I applied to $E \cup S$. We cannot simulate all deletion rules, because we cannot assume that there will always exist ground clauses in the saturation of $E \cup S$, on which such a deletion depends. The inference system P is really the same as I, expect that all clauses now have constraints (unconstrained clauses have the empty constraint), which are inherited by the conclusion of an inference. Also, we no longer have the **Orient** rule, and the Deletion rules are more restrictive.

The closure rules of P are given in Figure 3 and the deletion rules are in Figure 4.

Before presenting the theorem first important theorem of this paper, we need a couple more definitions.

Definition 2 *Let $C\llbracket \varphi \rrbracket$ be a constrained clause. $C\sigma$ is a* constraint instance *of $C\llbracket \varphi \rrbracket$ if $Dom(\sigma) = Vars(\varphi)$ and the range of σ only contains extension constants.*

Notice that if $C\sigma$ is a constraint instance of C, then it is also an instance of $C\llbracket \varphi \rrbracket$, because φ is satisfied. However, $C\sigma$ may not be ground, because unconstrained variables are not substituted for.

Now we show that for a single equation theory, a finite saturation of $E \cup G(E)$ implies a finite saturation of $E \cup S$ for any set of ground equations and disequations S, which gives a decidability result for the theory E.

Theorem 2 *Let E be a single equation theory, finitely saturated by I. Let G be the set of all clauses generated in a finite saturation of $E \cup G_E$ by P (i.e., E has finite saturation property). Let S be a set of flat ground equations and disequations. Then we can saturate $E \cup S$ by I such that every clause generated in the saturation is*

- *a constraint instance of a clause in G, or*
- *a flat f-rule where $f \notin \Sigma_E$*

Proof. The proof is by induction on the length of saturation inferences in I. For the base case, we show that all clauses in $E \cup S$ satisfy the theorem. This is true, because, since the equations and disequations in S are flat, then each equation and disequation in S is a constraint instance of a member of G_E, or is a flat f-rule where f does not appear in E.

Now, in the inductive part we have to show 2 facts:

1. Every clause added to $E \cup S$ in the process of saturation is a constraint instance of some clause in the saturation of $E \cup G_E$ or a flat f-rule.

Right Paramodulation:

$$\frac{G \cup \{u[s'] \approx v \vee C\,[\![\,\varphi\,]\!], s \approx t \vee D\,[\![\,\Psi\,]\!]\}}{G' \cup \{(u[t] \approx v \vee C \vee D\,[\![\,\varphi \wedge \Psi\,]\!])\sigma\}}$$

where $G' = G \cup \{u[s'] \approx v \vee C\,[\![\,\varphi\,]\!], s \approx t \vee D\,[\![\,\Psi\,]\!]\}$, $\sigma = mgu(s, s')$, $u[s'] \approx v$ is selected in its clause, $s \approx t$ is selected in its clause, $u[s']\sigma \not\preceq v\sigma$, $s\sigma \not\preceq t\sigma$, and s' is not a variable.

Left Paramodulation:

$$\frac{G \cup \{u[s'] \not\approx v \vee C\,[\![\,\varphi\,]\!], s \approx t \vee D\,[\![\,\Psi\,]\!]\}}{G' \cup \{(u[t] \not\approx v \vee C \vee D\,[\![\,\varphi \wedge \Psi\,]\!])\sigma\}}$$

where $G' = G \cup \{u[s'] \not\approx v \vee C\,[\![\,\varphi\,]\!], s \approx t \vee D\,[\![\,\Psi\,]\!]\}$, $\sigma = mgu(s, s')$, $u[s'] \approx v$ is selected in its clause, $s \approx t$ is selected in its clause, $u[s']\sigma \not\preceq v\sigma$, $s\sigma \not\preceq t\sigma$, and s' is not a variable.

Equational Resolution

$$\frac{G \cup \{u \not\approx v \vee C\,[\![\,\varphi\,]\!]\}}{G' \cup \{(C\,[\![\,\varphi\,]\!])\sigma\}}$$

where $G' = G \cup \{u \not\approx v \vee C\,[\![\,\varphi\,]\!]\}$, $\sigma = mgu(u, v)$ and $u \not\approx v$ is selected.

Equational Factoring

$$\frac{G \cup \{s \approx t \vee s' \approx u \vee C\,[\![\,\varphi\,]\!]\}}{G' \cup \{(s \approx u \vee t \not\approx u \vee C\,[\![\,\varphi\,]\!])\sigma\}}$$

where $G' = G \cup \{s \approx t \vee s' \approx u \vee C\,[\![\,\varphi\,]\!]\}$, $\sigma = mgu(s, s')$, $s \approx t$ is selected, $t\sigma \not\preceq u\sigma$, $s\sigma \not\preceq t\sigma$ and $s'\sigma \not\preceq u\sigma$.

Figure 3. Constrained Rules for Closure

Subsumption:

$$\frac{G \cup \{C, C'\,[\![\,\varphi\,]\!]\}}{G \cup \{C\}}$$

where C is a member of E, and $C\sigma \subseteq C'$ for some substitution σ, or if C and C' are renamings of each other.

Simplification:

$$\frac{G \cup \{C[l']\,[\![\,\varphi\,]\!], l \approx r\}}{G \cup \{C[r\sigma]\,[\![\,\varphi\,]\!], l \approx r\}}$$

where $l \approx r$ is a member of E, $l' = l\sigma$, $l\sigma \succ r\sigma$, and $C[l'] \succ (l\sigma \approx r\sigma)$.

Tautology Deletion:

$$\frac{G \cup \{t \approx t \vee C\,[\![\,\varphi\,]\!]\}}{G},$$

$$\frac{G \cup \{s \approx t \vee s \not\approx t \vee C\,[\![\,\varphi\,]\!]\}}{G}$$

Also, a clause with an unsatisfiable constraint may be removed.

Figure 4. Constrained Rules for Deletion

2. A deletion of a clause in the saturation of $E \cup G_E$ will not leave some clauses in the saturation of $E \cup S$ that were constraint instances of the deleted clause, with no representative in the saturation of $E \cup G_E$ (i.e. that all these constraint instances will also be deleted from the saturation of $E \cup G_E$).

Hence in Part 1 of the argument we start by considering the rule **Orient (Fig.1)**. Orient is an exceptional rule in the set of closure rules, because it does not really add any clause to the set (rather deletes one), but modifies the rest of the set.

Assume that all the clauses in G in the premise of the rule are constraint instances of some clauses in the saturation of $E \cup G_E$ or flat f-rules where $f \notin \Sigma_E$. Hence in particular let C be such a clause which is a constraint instance of some C' in the saturation of $E \cup G_E$. This means that $C'\sigma = C$. Hence there is a substitution σ' such that $x\sigma' = d$, whenever $x\sigma = c$ and $y\sigma' = y\sigma$ in all other cases. Therefore $C[c \mapsto d]$ will also be a constraint instance of C'.

Now let C be a flat f-rule, where f does not appear in E. Then Orient converts C to another flat f-rule with extension constants replaced.

Now we consider **Right Paramodulation (Fig.1.)**. Assume that the first clause in the premise of the rule is a constraint instance of some clause in the saturation of $E \cup G_E$. Then the other clause must be also.

Since $u[s'] \approx v \vee C$ is a constraint instance of a clause D_1 in the saturation of $E \cup G_E$, $D_1\sigma = u[s'] \approx v \vee C$. Similarly, $D_2\sigma = s \approx t \vee D$.
We have two cases to consider:

1. If s' is not an extension constant, then there is a term s_1 in D_1, such that $s_1\sigma = s'$ is not a variable, and hence there is a Right Paramodulation inference from D_1 and D_2 to D_3 such that $D_3\sigma = u[t] \approx v \vee C \vee D$.

2. If s' is an extension constant, then, since $u[s'] \approx v$ is a constraint instance of D_1, $D_1 = u[x] \approx v' \vee C' \llbracket const(x) \wedge \varphi \rrbracket$. $s \approx t$ must then be an extension equation. Since $s \approx t$ is selected in its clause, the rest of the clause must also be extension equations. But then D must be empty, since otherwise the clause $s \approx t \vee D$ is not a single equation clause, and the property is being a single equation clause is preserved by the inferences. So Orient applies.

Right Paramodulation involving flat f-rules
Assume that the equation in the premise $u[s'] \approx v$ of the rule is a flat f-rule. In this case C is empty. We have two cases here: either $s \approx t$ is another flat f-rule or $s \approx t$ is an extension equation.

1. Consider the case where $s \approx t$ is an extension equation. Then D must be empty. So Orient applies.

2. If $s \approx t$ is another flat f-rule, then C and D are empty and by Right Paramodulation we get an extension equation, which is an instance of $x \approx y \llbracket const(x) \wedge const(y) \rrbracket$ in the saturation of $E \cup G_E$.

The inferences **Left Paramodulation (Fig.1.)**, **Equational Resolution (Fig.1.)** and **Equational Factoring (Fig.1.)** can also be lifted in a similar way.

For the second part of the inductive argument, let us consider deletion rules in the saturation of $E \cup G_E$. Assume that there is a clause D in the saturation of $E \cup G_E$, which is deleted by the first case of **Subsumption (Fig.4.)** and a clause C is a constraint instance of D. Notice that according to the assumptions of the rule, there must be a clause $D' \in E$, such that $D'\sigma \subseteq D$. Since all members of E are in the saturation of $E \cup S$, there will be a substitution σ' such that $D'\sigma' \subseteq C$, and C will be also deleted by Subsumption from the saturation of $E \cup S$.

A similar argument may be made for **Simplification (Fig.4.)**. For **Tautology Deletion (Fig.4.)**, it is enough to notice that any instance of tautology is also a tautology, hence will be deleted from the saturation of $E \cup S$. The other case of **Subsumption** is just a matter of deleting duplicates. Hence the second part of the proof is done. □

Next we want to prove a result about the complexity. We have shown that, for a single equation theory E, if $E \cup G_E$ has the finite saturation property, then the theory of E is decidable. We will now show that it is decidable in polynomial time. First we define the *width* of a constrained clause.

Definition 3 *For a constrained clause $C \llbracket \varphi \rrbracket$, define the* width *of $C \llbracket \varphi \rrbracket$, written as $w(C \llbracket \varphi \rrbracket)$ to be the number of constrained variables in C. For a set of clauses S, define the* width *of S to be $w(S) = max\{w(C) \mid C \in S\}$.*

As a consequence of Theorem 2 we have the following.

Theorem 3 *Let E be a single equation theory, saturated by I. Let G be the set of all clauses generated in a finite saturation of $E \cup G_E$ by P. Let S be a set of flat ground equations and disequations. Then the saturation of $E \cup S$ will generate $O(|S|^{w(G)})$ clauses.*

Proof. Theorem 2 shows that every clause generated in the saturation of $E \cup S$ will be a constraint instance of a clause in G or a flat f-rule where f does not appear in E.
There are at most $|S|$ extension constants. Therefore, if C is a clause in G, then there are at most $|S|^{w(C)}$ constraint instances of C. If all symbols of S appear in E, then this immediately shows that there are $O(|S|^{w(G)})$ clauses appearing in the saturation of $E \cup S$, since the number of clauses in G is constant in $|S|$.

But suppose there is a symbol f which appears in E but not in S. Then $E \cup S$ will contain some flat f-rules. But the only way a new rule containing the symbol f can be created is if an extension constant is reduced in a flat f-rule, and then a new flat f-rule results. These flat f-rules are the only clauses generated that are not constraint instances of G. So we need to count how many times the extension constants of a flat f-rule can be reduced. Note that the precedence of the extension constants has been designed specifically so that there is no reduction sequence $c_1 \succ c_2 \succ \cdots \succ c_n$ of extension constants that is longer than $lg(|S|)$. Each clause contains no more than a constant number of extension constants (we are considering arity to be constant). Therefore, the extension constants in that clause will be reduced by an Orient rule $O(lg(|S|))$ times. Since there are at most $|S|$ flat f-rules, this gives us $O(|S| \times lg(|S|))$ clauses that originate from flat f-rules.

We have shown that there are $O(|S|^{w(G)})$ constraint instances of G, and there are $O(|S| \times lg(|S|))$ clauses originating from flat f-rules. So there are $O(|S|^{w(G)} + (|S| \times lg(|S|)))$ clauses generated in the saturation of $E \cup S$. Since there is a clause $x \approx y \llbracket const(x) \wedge const(y) \rrbracket$ in G_E, then $w(G)$ is at least 2, so the first term dominates the second, and therefore there are $O(|S|^{w(G)})$ clauses generated. □

This shows that in this case the theory is decidable in polynomial time, but we can get a better bound on the polynomial in some cases. We now give some definitions which lead to a theorem giving a better bound on the decision procedure in some cases, for example the theory of lists.

A saturation can be viewed as a graph, so that each clause generated (modulo renaming) labels a node in the graph. If there is an inference in the saturation with $C_1, \cdots, C_n$ as the premises and C as the conclusion (modulo renaming), then there is an edge from each node labeled with C_i to the node labeled with C. We call this graph a *saturation graph*. The saturation is called *cyclic* if the saturation graph contains a cycle, otherwise the saturation is called *noncyclic*.

We define a *proof* of C inductively, as a tree of nodes labeled with clauses, whose root is labeled C. If $C \in S$ then a single node labeled with C is a *proof* of C. Suppose there is an inference with premises $C_1, \cdots, C_n$ and conclusion C, and suppose that $\Pi_1, \cdots, \Pi_n$ are proofs of $C_1, \cdots, C_n$ respectively. Then the tree containing $\Pi_1, \cdots, \Pi_n$, plus the node labeled by C and the edges from each C_i to C is a proof of C. Note that a proof of C is just one way of deriving C.

Note that a proof could be also defined as a subgraph in the saturation graph, that can be directed from the root to the leaves (hence a dag) and then the tree is the expanded dag with all nodes having at most one parent. If a saturation graph is noncyclic, there are only a finite number of proofs of C.

Definition 4 *We call a clause D an* initial ancestor *of a clause C in a proof Π in the saturation of $E \cup G_E$, if D labels a leaf in the proof Π of C, D is not of a form $x \approx y \llbracket const(x) \wedge const(y) \rrbracket$ and $D \notin E$.*

In a proof of a clause C in the saturation of $E \cup G_E$, let $k(C, E)$ be the number of initial ancestors of C which are not in E. Define $k(E)$ as $max\{k(C, E) \mid C$ in the saturation of $E \cup G(E)$ and C is not of the form $x \approx y \llbracket const(x) \wedge const(y) \rrbracket\}$.

Theorem 4 *Let E be a single equation theory, saturated by I. Let G be the set of all clauses generated in a finite saturation of $E \cup G_E$ by P. Let S be a set of flat ground equations and disequations. If the saturation of $E \cup G_E$ is noncyclic. Then the saturation of $E \cup S$ will generate $O(|S|^{k(E)} \times lg(|S|))$ clauses.*

Proof. Consider a clause C in the saturation graph of $E \cup G_E$. Assume first that C is not of the form $x \approx y \llbracket const(x) \wedge const(y) \rrbracket$.

We need to consider how many constraint instances of C will appear in the saturation of $E \cup S$. Since the saturation graph is noncyclic, there is only a finite number of proofs of C in the saturation graph. And the since saturation graph is not dependent on S, this number is not dependent on S.

Let's consider one of the proofs, Π, of C. Then C has at most $k(C, E)$ initial ancestors which are constrained equations in Π. Consider an equation $e[x_1, \ldots, x_m]$ appearing in a clause in the proof Π. Note that if we consider an constraint instance $\Pi\sigma$ of Π, then the value of $x_i\sigma$ is the same as the value of one of the parents of e, leading eventually back to an identical value of one of the initial ancestors of C. Even if we consider an instance of C that is reduced by some applications of the Orient rule, there is a proof where all of its ancestors are similarly reduced by Orient. So the number of potential instances of C in a proof Π is the same as the number of possible initial ancestors in a constraint instance of Π. In Π, there are $k(C, E)$ initial ancestors of C. At the beginning of the saturation of $E \cup S$, there are $O(|S|^{k(C,E)})$ instances of the initial ancestors E. During the saturation, each of these sets of ancestors can be reduced by Orient $O(lg(|S|)$ times. Therefore, there are $O(S^{k(C,E)} \times lg(|S|)$ instances of C derived from instances of Π. And since there are a constant number of proofs of C, there are $O(S^{k(C,E)} \times lg(|S|)$ instances of C in the saturation of $E \cup S$. And since there are a constant number of clauses in the saturation of $E \cup G_E$, there are $O(S^{k(C,E)} \times lg(|S|)$ clauses generated in the saturation of $E \cup S$.

Now we have to count the instances of clauses in the saturation of $E \cup G_E$ of the form $x \approx y \llbracket const(x) \wedge const(y) \rrbracket$. There are only S of them, because every time one of them is created, it will be immediately used in an Orient rule, and one extension constant is deleted. Since constants can only be deleted $O(|S|)$ times, then only $O(|S|)$ extension equations can be created.

In order to complete the proof, we have to count the number of flat f-rules with $f \notin \Sigma_E$. But since each of them can be reduced only $O(lg(|S|))$, there will be $O(|S| \times lg(|S|))$ flat f-rules in the saturation of $E \cup S$.

Since $(|S|^{k(E)} \times lg(|S|))$ dominates over $(|S| \times lg(|S|))$ and S, this means that there are $O(|S|^{k(E)} \times lg(|S|))$ clauses in the saturation of $E \cup S$. □

Note that any time a clauses is reduced by Orient, it cannot ever appear again, because the every occurrence of the extension constant that is reduced is removed from the set of clauses.

5 Extending the Procedure

The results in the previous section are for single equation theories. In order to directly simulate more than single equation theories, we would need to allow inferences into variables in certain restricted cases. This would be possible, but it ends up exploding the search space. If there are two real variables in a clause, then inferences with flat rules can reduce the clause to another clause containing two extension equations. These extension equations can be used in inferences with other clause, which have the effect of continuing to create larger clauses with more and more extension equations. Therefore, we would lose the finite saturation property.

Therefore, we choose a different approach to handling general clauses. We use the same inference system P as in the previous section, but we now allow a clause to represent constraint instances as before, but now we also allow constraint instances with some any possible disjunction of extension equations disjoined to them. This makes the complexity of the procedure go from polynomial to exponential.

Theorem 5 *Let E be a set of clauses, finitely saturated by I. Let G be the set of all clauses generated in a finite saturation of $E \cup G_E$ by P. (i.e. E has the finite saturation property wrt P.) Let S be a set of flat ground equations and disequations. Then we can saturate $E \cup S$ by I such that every clause generated in the saturation is*

- *a disjunction of extension equations $c_1 \approx d_1 \vee \ldots \vee c_n \approx d_n$, where $n \geq 1$, or*
- *a clause of the form $C \vee c_1 \approx d_1 \vee \ldots \vee c_n \approx d_n$, where C is a constraint instance of a clause C' in G and $n \geq 0$, or*
- *a clause of the form $C \vee c_1 \approx d_1 \vee \ldots \vee c_n \approx d_n$, where C is a flat f-rule where $f \notin \Sigma_E$ and $n \geq 0$.*

Proof. The proof, as in the previous completeness theorem, is by induction on the length of saturation inferences in I. The base case is the same as in the previous case.

For the first part we show that every clause added to $E \cup S$ in the process of saturation is one of the three kinds named in the theorem.

Let us consider an inference with the rule **Orient (Fig.1)**. Assume that a clause in the saturation of $E \cup S$ is of the form $c_1 \approx d_1 \vee \ldots \vee c_n \approx d_n$, where $n \geq 1$, and all the constants are extension constants. Then Orient may reduce one constant, but the modified clause will be of the same form.

Assume now that there is a clause of the form $C \vee c_1 \approx d_1 \vee \ldots \vee c_n \approx d_n$, where C is a constraint instance of a clause C' in G, $n \geq 0$ and $c_1 \approx d_1, \ldots c_n \approx d_n$ are extension equations. Since C is a constraint instance of C', there is a substitution σ, such that $C'\sigma = C$. Hence if Orient modifies the clause in the C-part, we can see that just as in the previous section, there will be a substitution σ' such that $x\sigma' = d$, whenever $x\sigma = c$ and $y\sigma' = y\sigma$ in all other cases. Therefore $C[c \mapsto d]$ will also be a constraint instance of C'. If Orient modifies the clause in one of the extension equations, we will get still a clause of the same form.

Assume now that there is a clause of the form $C \vee c_1 \approx d_1 \vee \ldots \vee c_n \approx d_n$, where C is a flat f-rule where $f \notin \Sigma_E$, $n \geq 0$ and $c_1 \approx d_1, \ldots, c_n \approx d_n$ are extension equations. If Orient affects the C-part of the clause, it will change it into another flat f-rule, and if it affects the part with extension equations, it will also preserve the form of the clause.

Now we consider **Right Paramodulation (Fig.1.)**.

Assume that $u[s'] \approx v \vee C$ is a disjunction of extension equations. Then $u[s']$ is just an extension constant c, and $s = c$. $s \approx t$ must be an extension equation. Then $s \approx t \vee D$ must be a disjunction of extension equations, because otherwise there would be some bigger literal in the clause, and $s \approx t$ would not be selected. Hence the conclusion of the rule will be: $t \approx v \vee C \vee D$ which is also a disjunction of extension equations.

Assume that $u[s'] \approx v \vee C$ is a clause of the form $D_2 \vee c_1 \approx d_1 \vee \ldots \vee c_n \approx d_n$, where D is a constraint instance of a clause D' in G, $n \geq 0$, and $c_1 \approx d_1 \vee \ldots \vee c_n \approx d_n$ are extension equations.
If $u[s'] \approx v$ is an extension equation, then C must also be a disjunction of such equations, because otherwise $u[s'] \approx v$ would be too small to be selected (the previous case applies). Hence assume that $u[s'] \approx v$ is in the D_2-part. The clause has the form $u[s'] \approx v \vee C_1 \vee c_1 \approx d_1 \vee \ldots \vee c_n \approx d_n$, where $u[s'] \approx v \vee C_1$ is a constraint instance of D'. We have two cases to consider here: $s \approx t$ is an extension equation or not. If $s \approx t$ is an extension equation, then either D is empty, but then Orient applies, or D must be a disjunction of extension constants (because otherwise $s \approx t$ would be too small to be selected). Then Right Paramodulation generates the clause: $u[t] \approx v \vee C_1 \vee c_1 \approx d_1 \vee \ldots \vee c_n \approx d_n$. Since $D'\sigma = u[s'] \approx v \vee D_1$, there is a substitution σ' such that $x\sigma' = d$, whenever $x\sigma = c$ and $y\sigma' = y\sigma$ in all other cases. Therefore $u[t] \approx v \vee D_1$ will also be a constraint instance of D' and the form of the clause will be preserved.

Consider now that $s \approx t$ is not an extension equation. Let $s \approx t \vee D$ be of the form $s \approx t \vee D_1 \vee c'_1 \approx d'_1 \vee \ldots \vee c'_m \approx d'_m$ where $s \approx t \vee D_1$ is a constraint instance of a clause D'' in the saturation of $E \cup G_E$. There is a substitution σ such that $D'\sigma = u[s'] \approx v \vee C_1$, $D''\sigma = s \approx t \vee D_1$ and there is an inference in the saturation of $E \cup G_E$ from D' and D'' to D''' such that $D'''\sigma = u[t] \approx v' \vee C'_1$. Hence the conclusion of the Right Paramodulation in the saturation of $E \cup S$ is still of the right form (the second case of the theorem).

It is not possible for $s \approx t$ to be a flat f-rule with $f \notin \Sigma_E$, because s' cannot contain the symbol f, since D_2 is a constraint instance of a clause in the saturation of $E \cup G_E$.

Assume then that $u[s'] \approx v \vee C$ is of the form $f(c'_1, \ldots, c'_m) \approx c' \vee c_1 \approx d_1 \vee \ldots \vee c_n \approx d_n$, where $u[s'] \approx v$ is $f(c'_1, \ldots, c'_m) \approx c'$ is a flat f-rule where $f \notin \Sigma_E$ and $n \geq 0$. If s' is an extension constant ($s' = c'_i$ for some i), then $s \approx t$ must be an extension equation, and if D is empty, Orient applies, otherwise $s \approx t \vee D$ must be a disjunction of extension equations. From the Right Paramodulation we get then another clause which is a flat f-

rule disjoined with extension equations. Now consider the case where s' is not an extension constant, then s' is u itself, and $s \approx t$ must be another flat f-rule. By Right Paramodulation we get $v \approx t \vee c_1 \approx d_1 \vee \ldots \vee c_n \approx d_n \vee D$, hence the disjunction of extension equations.

The inferences **Left Paramodulation (Fig.1.)**, **Equational Resolution (Fig.1.)** and **Equational Factoring (Fig.1.)** can also be lifted in a similar way.

For the second part of the inductive argument, let us consider deletion rules in the saturation of $E \cup G_E$. Hence assume that there is a clause D in the saturation of $E \cup G_E$, which is deleted by the first case of **Subsumption (Fig.4.)** and there is a clause $C \vee c_1 \approx d_1 \vee \ldots \vee c_n \approx d_n$ in the saturation of $E \cup S$, such that C is a constraint instance of D and $c_1 \approx d_1, \ldots, c_n \approx d_n$ are extension equations (with $n \geq 0$). Hence there must be a clause $D' \in E$, such that $D'\sigma \subseteq D$. Since all members of E are in the saturation of $E \cup S$, there will be a substitution σ' such that $D'\sigma' \subseteq C$, hence also $D'\sigma' \subseteq C \vee c_1 \approx d_1 \vee \ldots c_n \approx d_n$, and $C \vee c_1 \approx d_1 \vee \ldots c_n \approx d_n$ will be also deleted by Subsumption from the saturation of $E \cup S$.

A similar argument may be made for **Simplification (Fig.4.)**. For the **Tautology Deletion (Fig.4.)**, it is enough to notice that any instance of tautology is also a tautology, hence will be deleted from the saturation of $E \cup S$. The other case of **Subsumption** is just a matter of deleting duplicates. Hence, the second part of the proof is done. □

That shows that these are decidable theories. Next we shown that the complexity of the decision procedure is simply exponential

As a consequence of Theorem 5 we have the following.

Theorem 6 *Let E be a finite set of clauses, saturated by I. Let G be the set of all clauses generated in a finite saturation of $E \cup G_E$ by P. Let S be a set of flat ground equations and disequations. Then the satisfiability of $E \cup S$ will generate $O(2^{|S|^2})$ clauses.*

Proof. From the previous section we know that there are $O(|S|^{w(G)})$ constraint instances of clauses in the saturation of G plus flat f-rules, where f does not appear in E.

In this case we must also count all disjunctions of such clauses with extension equations, and also disjunctions of extension equations by themselves. There are $O(|S|)$ extension constants in the saturation of $E \cup S$, hence there are $O(|S|^2)$ extension equations, and there are $O(2^{|S|^2})$ disjunctions of such equations. Hence there are $O(|S|^{w(G)} \times 2^{|S|^2}) = O(2^{|S|^2})$ clauses generated in the saturation of $E \cup S$. □

6 Example

We illustrate our results with some examples from the paper [1]. First is the theory of lists.

Let E be the list theory defined by the following equations:

$$\begin{array}{c} car(cons(x,y)) \approx x \\ cdr(cons(x,y)) \approx y \end{array}$$

Then G_E consists of the following constrained equations and disequations:

$$\begin{array}{l} car(x) \approx y \, [\![\, const(x) \wedge const(y) \,]\!] \\ cdr(x) \approx y \, [\![\, const(x) \wedge const(y) \,]\!] \\ cons(x,y) \approx z \, [\![\, const(x) \wedge const(y) \wedge const(z) \,]\!] \\ x \approx y \, [\![\, const(x) \wedge const(y) \,]\!] \\ x \not\approx y \, [\![\, const(x) \wedge const(y) \,]\!] \end{array}$$

Consider a Right Paramodulation inference between the first member of E and the third member of G_E:

$$\frac{\begin{array}{l} car(cons(x,y)) \approx x \\ \quad cons(x',y') \approx z' \, [\![\, const(x') \wedge const(y') \wedge const(z') \,]\!] \end{array}}{car(z') \approx x' \, [\![\, const(x') \wedge const(z') \,]\!]}$$

We removed the constraint $const(y')$ from the conclusion, since y' does not appear in the clause. The conclusion will be immediately deleted because it is a renaming of a member of G_E. However, this inference will determine two edges in the saturation graph, one from the node labeled by the first premise to the node labeled by the conclusion, and the another one from the node labeled by the second premise to the node labeled by the conclusion.

There is also an inference between the second member of E and the third member of G_E, but that is also deleted because it is a renaming of something in G_E. There are no other possible inferences. Therefore $E \cup G_E$ is the saturation of itself. There were some edges added to the saturation graph, but it did not create a cycle. Therefore, this shows that the problem of the satisfiability of $E \cup S$ for any set of equations and disequations S can be decided in time $O(|S| \times lg(|S|))$.

Now let us consider the case where E is the theory of arrays, also from [1]. In this example, $select$ is a function that takes an array and an index position, and returns the element at that index position. $store$ is a function that takes an array, an index position and en element, and returns the array that results when that element is put in that position in the array.

$$\begin{array}{c} select(store(x,y,z),y) \approx z \\ y \approx w \vee select(store(x,y,z),w) = select(x,w) \end{array}$$

Then G_E consists of the following constrained equations and disequations:

$$\begin{array}{l} select(x,y) \approx z \, [\![\, const(x) \wedge const(y) \wedge const(z) \,]\!] \\ store(x,y,z) \approx w \, [\![\, const(x) \wedge const(y) \wedge const(z) \\ \wedge const(w) \,]\!] \\ x \approx y \, [\![\, const(x) \wedge const(y) \,]\!] \\ x \not\approx y \, [\![\, const(x) \wedge const(y) \,]\!] \end{array}$$

The only inference which generates a new clause is an inference between the second member of E and the second member of G_E, which gives the clause $select(x, w) \approx select(z, w) \vee (y \approx w)[\![const(x) \wedge const(y) \wedge const(z)]\!]$

The only inference which generates a new clause is an inference between this new clause and the first member of G_E, which gives $select(x, w) \approx z \vee (y \approx w)[\![const(x) \wedge const(y) \wedge const(z) \wedge const(w)]\!]$

Then there is an inference between this new clause and the first member of G_E, plus an inference between this new clause and itself. These two inference generate the two clauses $x_1 \approx x_2 \vee x_3 \approx x_4 [\![const(x_1) \wedge \cdots \wedge const(x_4)]\!]$ and $x_1 \approx x_2 \vee x_3 \approx x_4 \vee x_5 \approx x_6 [\![const(x_1) \wedge \cdots \wedge const(x_6)]\!]$

All inference using these two clauses are inferences into a variable position, so they are disallowed, and there are no more inference. Therefore, E has the finite saturation property.

7 Conclusion

We consider our results to be an extension of previous results. In [9], there is an algorithm for deciding the theory of lists. In [8] there is a decision procedure for the theory of arrays, and in [10] a decision procedure for the theory of arrays with extensionality. In [1], a general result is given which subsumes the previous results and gives a general procedure which fits everything into a saturation theorem proving framework. What we have done is to automate and generalize the results of [1]. Given a theory for a data structure in first order logic with equality, we provide a set of inference rules such that when they halt, then the standard Paramodulation rules will decide the theory of that data structure. Not only will they decide the theory, but they will also give a bound on the number of clauses generated. For certain *noncyclic* theories like lists, the complexity is $O(n \times lg(n))$. For some other theories, including those that have no disjunctions, the complexity is polynomial. For other theories, like the theory of arrays, it is simply exponential. Our results are an extension of the congruence closure algorithm given in [4], which gives a nice proof of an $O(n \times lg(n))$ bound for the empty theory. We showed that their algorithm can be extended to *noncyclic* theories, like the theory of arrays. Our work is also in the spirit of [5], because it gives an automatic method to derive decision procedures and complexity results, that of saturation of a particular theory.

Acknowledgments

Many of these results were obtained while the first author was visiting MPI Saarbrücken, and we thank Harald Ganzinger for many helpful discussions. Also, this work is based heavily on the work of [1]. We thank Alessandro Armando, Silvio Ranise and Michaël Rusinowitch for discussions of their paper. We would also like to thank the anonymous referees for helping us improve the quality of this paper.

References

[1] A. Armando, S. Ranise and M. Rusinowitch. Uniform Derivation of Decision Procedures by Superposition. In *Proceedings 15th Workshop on Computer Science Logic.*, LNCS vol. 2142, 513-527, 2001.

[2] F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge, 1998.

[3] L. Bachmair and H. Ganzinger. Rewrite-based equational theorem proving with selection and simplification. In *Journal of Logic and Computation* 4(3), 1-31, 1994.

[4] L. Bachmair and A. Tiwari. Abstract Congruence Closure and Specializations. In *Proceedings 17th Conference on Automated Deduction.* LNCS vol. 1831, 64-78, 2000.

[5] D. Basin and H. Ganzinger. Automated complexity analysis based on ordered resolution. In *J. Association for Computing Machinery* 48(1), 70-109, 2001.

[6] N. Dershowitz. Termination of Rewriting. In *Journal of Symbolic Computation* 3, 69-116, 1987.

[7] D. Kapur. Shostak's Congruence Closure as Completion. In *Proceedings 8th Conference on Rewriting Techniques and Applications.* LNCS vol. 1232, 23-37, 1997.

[8] G. Nelson. Techniques for Program Verification. In *Technical Report CSL-81-10*, Xerox Palo Alto Research Center, June 1981.

[9] G. Nelson and D. C. Oppen. Fast Decision Procedures Based on Congruence Closure. In *J. Association for Computing Machinery* 27(2), 356-364, 1980.

[10] A. Stump, D. L. Dill, C. W. Barrett and J. Levitt. A Decision Procedure for an Extensional Theory of Arrays. In *Proceedings 16th IEEE Symposium on Logic in Computer Science,* IEEE Computer Society Press, 29-37, 2001.