# On the Complexity of Equational Horn Clauses

Kumar Neeraj Verma[1] Helmut Seidl[1] Thomas Schwentick[2]

[1] Institut für Informatik, Technische Universität München, Germany
{verma,seidl}@in.tum.de
[2] Fachbereich Mathematik und Informatik, Philipps-Universität Marburg, Germany
tick@informatik.uni-marburg.de

**Abstract.** Security protocols employing cryptographic primitives with algebraic properties are conveniently modeled using Horn clauses modulo equational theories. We consider clauses corresponding to the class $\mathcal{H}3$ of Nielson, Nielson and Seidl. We show that modulo the theory ACU of an associative-commutative symbol with unit, as well as its variants like the theory XOR and the theory AG of Abelian groups, unsatisfiability is NP-complete. Also membership and intersection-non-emptiness problems for the closely related class of one-way as well as two-way tree automata modulo these equational theories are NP-complete. A key technical tool is a linear time construction of an existential Presburger formula corresponding to the Parikh image of a context-free language. Our algorithms require deterministic polynomial time using an oracle for existential Presburger formulas, suggesting efficient implementations are possible.

## 1 Introduction

In [1], Blanchet proposes to use first-order Horn clauses for verifying secrecy of cryptographic protocols. Among others, this approach has later-on also been advocated by Goubault-Larrecq and Parrennes [10], Comon-Lundh and Cortier [5] and Seidl and Verma [19] who consider rich decidable fragments of clauses which still allow us to model many useful protocols. While traditional methods for verifying cryptographic protocols have been based on the *perfect cryptography* assumption, a more accurate analysis of these protocols requires us to take into account algebraic properties of cryptographic primitives, modeled using equational theories. For example modeling of protocols based on modular exponentiation must account for properties like associativity and commutativity [11]. In general what we require most often in protocols are the associative and commutative theories ACU, XOR and AG (i.e., the theory of Abelian groups) [6]. While the case of protocols with bounded numbers of sessions has already received considerable attention [2], there exist very few decidability results in the case of unbounded number of sessions, in the presence of equational theories. Horn clauses modulo equational theories provide a suitable framework for modeling such protocols. A decidable class of clauses with the theory XOR is studied in [5] where a non-elementary upper bound is proposed. In [11, 20], this problem has been attacked by forms of Horn clauses corresponding to two-way automata (see e.g. [4], Chapter 7), in the presence of several variants of the theory of associativity and commutativity. In this framework, automata-theoretic problems like membership and intersection-non-emptiness correspond to the unsatisfiability problem for clauses.

Dealing with first order clauses in the presence of equational theories, in particular associative-commutative theories, is also of more general interest, and has received considerable interest in the past [16]. While most work has focused on obtaining sound and complete inference systems for general forms of clauses, very little work has been done on obtaining decidable fragments of clauses in the presence of such theories.

In this paper, we start from the class $\mathcal{H}3$ of Horn clauses which has been proposed in [13] for control-flow analysis and also is used by Goubault-Larrecq for cryptographic analysis of C programs [10]. This class is closely related to two-way automata [20, 4] and has a polynomial unsatisfiability problem. We extend this class by operators satisfying associative-commutative theories. We show that unsatisfiability then becomes NP-complete for the theories $\mathrm{XOR}$ and $\mathrm{AG}$. For the theory $\mathrm{ACU}$ of an associative-commutative symbol with unit, the same holds true under suitable restrictions.

Independently of the application to cryptographic protocols, related notions of tree automata have also been studied by others [14, 3], notably for applications to XML document processing [17, 18, 12]. The languages accepted by unordered Presburger tree automata [17], for example, are essentially those accepted by our one-way $\mathrm{ACU}$ automata. While very general classes of these automata have been shown to be decidable, their complexity remains mostly unknown. A common idea underlying all these classes is their connection to *Parikh images* of context free languages [15], i.e. *semilinear* or Presburger-definable sets [9] which are closed under Boolean operations. For example in [20], to decide intersection-non-emptiness of two $\mathrm{ACU}$ automata, the product automaton is computed by first computing semilinear sets corresponding to the automata and then computing intersection of the semilinear sets. Both steps are expensive, and such ideas are unlikely to give us optimal algorithms.

In this paper we show how to obtain optimal algorithms for $\mathcal{H}3$ clause sets, or two-way tree automata, modulo associative-commutative theories, without computing product automata. A key technique we rely on is a linear time construction of an existential Presburger formula corresponding to the Parikh image of a context-free language, allowing us to show that membership and intersection-non-emptiness for one-way as well as two-way tree automata modulo the theories $\mathrm{ACU}, \mathrm{XOR}$ and $\mathrm{AG}$ are NP-complete. NP-completeness of the membership problem for one-way $\mathrm{ACU}$ automata is also shown in [14], although the complexity of the intersection-non-emptiness problem is left open there. We resolve both questions for each of the theories $\mathrm{ACU}, \mathrm{XOR}$ and $\mathrm{AG}$, besides others like the theory $\mathrm{XOR}_p$ which contains the axiom $\sum_{i=1}^{p} x = 0$ besides the axioms of $\mathrm{ACU}$ ($\mathrm{XOR}$ is the special case $p = 2$). We further extend these complexity results to two-way automata as well. As a consequence the non-emptiness problem, which requires linear time in the one-way case, is also NP-complete in the two-way case. Further we obtain NP-completeness of the unsatisfiability problem for $\mathcal{H}3$ modulo $\mathrm{ACU}, \mathrm{XOR}, \mathrm{AG}$ and $\mathrm{XOR}_p$. Note that the technique of [14] is not useful here since it is specific to the membership problem and to one-way automata.

**Outline.** We start in Section 2 by introducing our classes of automata and clauses, and demonstrate how they model cryptographic protocols. To deal with these classes, we give in Section 3 a linear time construction of existential Presburger formulas corresponding to Parikh images of context free languages. This is used to deal with one-way $\mathrm{ACU}$ automata in Section 4. The one-way $\mathrm{XOR}$ and $\mathrm{AG}$ cases are similarly dealt with

in Section 5. These results are used to deal with two-way automata and $\mathcal{H}3$ in Section 6. The readers are also referred to [20] which uses similar techniques to show decidability of most of the problems which we show here to be NP-complete.

## 2 Clauses, Automata and Cryptographic Protocols

Fix a signature $\Sigma$ of function symbols. Since we deal with variants of the ACU theory, we assume that $\Sigma$ contains at least the symbols $+$ and $0$, and additionally the symbol $-$ when dealing with the theories ACUD or AG, defined below. Symbols in $\Sigma_f = \Sigma \setminus \{+, -, 0\}$, are *free*. Free symbols of zero arity are *constants*. Terms of the form $f(t_1, ..., t_n)$ where $f$ is free are *functional terms*. The equational theory ACU consists of the equations $x + (y + z) = (x + y) + z, x + y = y + x$ and $x + 0 = x$. The other theories we deal with are obtained by adding equations to this theory. The theory XOR is obtained by adding the equation $x + x = 0$. More generally, the theory $\mathrm{XOR}_p$ for $p \geq 2$ is obtained by the equation $\sum_{i=1}^{p} x = 0$. The theory AG is obtained by the equation $x + (-x) = 0$. The theory ACUD, obtained by the equations $-(x + y) = (-x) + (-y)$ and $-(-x) = x$, is weaker than AG and is introduced as a tool to deal with the theory AG which is of interest to us. The equation $x + x = x$ gives the theory ACUI of idempotent commutative monoids. Throughout this paper, if $s =_{\mathrm{ACU}} t$ or $s =_{\mathrm{ACUD}} t$ then $s$ and $t$ are treated as the same object.

A *clause* is a finite set of *literals* $A$ (a *positive* literal) or $-A$ (a *negative* literal), where $A$ is an *atom* $P(t_1, \ldots, t_n)$. A *Horn clause* contains at most one positive literal. The clause $A \vee -A_1 \vee \ldots \vee -A_n$ is written as $A \Leftarrow A_1 \wedge \ldots \wedge A_n$ and called a *definite* clause. The clause $-A_1 \vee \ldots \vee -A_n$ is written as $\bot \Leftarrow A_1 \wedge \ldots \wedge A_n$ and called a *goal* clause. $A$ is *head* of the first clause, while $-A_1 \vee \ldots \vee -A_n$ is the *tail* of both clauses. Satisfiability of clauses modulo equational theories is defined as usual. To every clause $C$ we can associate a variable dependence graph $G_C$ whose nodes are the literals of $C$, and two literals are adjacent if they share a variable. A clause $C$ is called H3 if

(1) the head, if any, of $C$ is linear, i.e. no variable occurs twice in it

(2) $G_C$ is acyclic, and adjacent literals in $G_C$ share at most one variable.

(3) the symbol $+$ does not occur in non-ground negative literals, except in case of theories XOR and AG.

The class $\mathcal{H}3$ consists of finite sets of H3 clauses. The first two conditions above are as in [13] in the non-equational case. In the equational case, we now also impose the third condition. Without this restriction, the unsatisfiability problem in the ACU case would subsume [7, 22] the provability problem in MELL (Multiplicative Exponential Linear Logic) which itself subsumes the reachability problem in VASS (Vector Addition Systems with States). The latter is decidable and EXPSPACE-hard, while decidability of the former is still open. Examples of H3 clauses are one-way and two-way equational tree automata clauses defined below.

An *(equational) tree automaton* $\mathcal{A}$ is a finite set of definite clauses involving only unary predicates. We read an atom $P(t)$ as "term $t$ is accepted at state $P$". We write $\mathcal{A}/\mathcal{E}$ to emphasize the equational theory $\mathcal{E}$ modulo which the automaton is considered. *Derivations* of ground atoms in the automaton are defined using the following two rules:

$$\frac{P_1(t_1\sigma)\ldots P_n(t_n\sigma)}{P(t\sigma)}\,(P(t) \Leftarrow P_1(t_1)\wedge\ldots\wedge P_n(t_n)\in\mathcal{A}) \qquad \frac{P(s)}{P(t)}\,(s =_{\mathcal{E}} t)$$

where substitution $\sigma$ maps all variables to ground terms, and $=_{\mathcal{E}}$ is the congruence on terms induced by $\mathcal{E}$. Hence the derivable atoms are exactly the elements of the least Herbrand model modulo $\mathcal{E}$. We define the language $L_P(\mathcal{A}/\mathcal{E}) = \{t \mid P(t) \text{ is derivable}\}$. When $\mathcal{E}$ is the empty theory, we also write it as $L_P(\mathcal{A})$. If in addition some state $P$ is designated as *final* then the language accepted by the automaton is $L_P(\mathcal{A}/\mathcal{E})$. For a language $L$ we define $\mathcal{E}(L) = \{s \mid \exists t \in L \cdot s =_{\mathcal{E}} t\}$. Note that automata-theoretic problems are closely related to the unsatisfiability problem of Horn clauses:

**Lemma 1.** *Let $\mathcal{A}$ be a tree automaton and $\mathcal{E}$ an equational theory.*
*(i) $L_P(\mathcal{A}/\mathcal{E}) \neq \emptyset$ iff $\mathcal{A}\cup\{\bot \Leftarrow P(x)\}$ is unsatisfiable modulo $\mathcal{E}$.*
*(ii) $t \in L_P(\mathcal{A}/\mathcal{E})$ iff $\mathcal{A}\cup\{\bot \Leftarrow P(t)\}$ is unsatisfiable modulo $\mathcal{E}$, where $t$ is ground.*
*(iii) $L_P(\mathcal{A}/\mathcal{E})\cap L_Q(\mathcal{A}/\mathcal{E}) \neq \emptyset$ iff $\mathcal{A}\cup\{\bot \Leftarrow P(x)\wedge Q(x)\}$ is unsatisfiable modulo $\mathcal{E}$.*

Hence the results in this paper can be interpreted from an automata-theoretic viewpoint as well as from a logical viewpoint. *One-way automata* consist of clauses:

$$P(f(x_1,...,x_n)) \Leftarrow P_1(x_1)\wedge...\wedge P_n(x_n) \quad (1) \qquad P(x) \Leftarrow P_1(x) \quad (2)$$

which we call *pop clauses* and *$\epsilon$-clauses* respectively. In (1), the variables $x_1,...,x_n$ are mutually distinct. In the non-equational case, one-way automata are exactly the tree automata usually found in the literature, and which accept regular tree languages. We also recall from [20]:

**Lemma 2.** *We have $\mathcal{L}_P(\mathcal{A}/\mathcal{E}) = \mathcal{E}(\mathcal{L}_P(\mathcal{A}))$ for any one-way automaton $\mathcal{A}$ and equational theory $\mathcal{E}$. In particular, emptiness for one-way $\mathcal{E}$ tree-automata is decidable in linear time.*

Because of the form of signatures that we consider, the pop clauses in our automata are of the following form,

$$P(x+y) \Leftarrow P_1(x)\wedge P_2(y) \quad (3) \qquad P(a) \text{ where } a \text{ is a constant} \quad (5)$$
$$P(0) \quad (4) \qquad P(-x) \Leftarrow P_1(x) \quad (6)$$
$$P(f(x_1,...,x_n)) \Leftarrow P_1(x_1)\wedge...\wedge P_n(x_n) \qquad (f \text{ is free}) \quad (7)$$

called *$+$-pop clauses*, *zero clauses*, *constant clauses*, *minus clauses* and *free pop clauses*. Clauses (5) are special cases of clauses (7). For $\mathcal{E} \in \{\mathrm{ACU}, \mathrm{XOR}, \mathrm{AG}, \mathrm{ACUD}\}$, one-way $\mathcal{E}$-tree automata are sets of clauses (2–7) (clause (6) is present only when $- \in \Sigma$). We define two-way automata by adding the following kind of clauses

$$Q(x_i) \Leftarrow P(f(x_1,...,x_n))\wedge \bigwedge_{j\in\{1,...,n\}\setminus\{i\}} Q_j(x_j) \qquad (f \text{ is free}, 1\leq i\leq n) \quad (8)$$

called *push clauses*, to one-way automata (the variables $x_1,...,x_n$ are mutually distinct.) Hence *two-way* automata are sets of clauses (2–8) (clause (6) is included only when $- \in \Sigma$). For a two-way automaton $\mathcal{A}$, we let $\mathcal{A}_{eq}$ denote the set of $\epsilon$-clauses, $+$-pop clauses, zero clauses and $-$ clauses in $\mathcal{A}$. These are the *equational clauses* of $\mathcal{A}$. Let

us discuss the side-conditions in the above push clause. We first prohibit the variable $x_i$ to occur twice in the tail of the clause. Removing this restriction allows us to encode alternating tree automata. In the non-equational case, this leads to an EXPTIME-complete emptiness problem. In case of the theories $\mathrm{ACU}, \mathrm{AG}$ and $\mathrm{ACUD}$, the emptiness problem becomes undecidable [20]. The $\mathrm{XOR}$ case is decidable [21], though the complexity seems high. Secondly we have restricted $f$ to be free. In the $\mathrm{ACU}$ case, the justification is as for $\mathcal{H}3$ clauses. In case of the theory $\mathrm{XOR}$ the clause $P(x) \Leftarrow Q(x+y) \wedge R(y)$ is equivalent to the clause $P(x+y) \Leftarrow Q(x) \wedge R(y)$. In the $\mathrm{AG}$ case, the former clause is equivalent to the clauses $P(x+y) \Leftarrow Q(x) \wedge R'(y)$ and $R'(-y) \Leftarrow R(y)$ for fresh $R'$. Push clauses $P(x) \Leftarrow Q(-x)$ involving the $-$ symbol are equivalent to the minus clause $P(-x) \Leftarrow Q(x)$ modulo our equational theories.

Note that we have restricted each $x_j \neq x_i$ to occur exactly twice in the tail. Our complexity results hold even if we allow more atoms of the form $Q_j^1(x_j), Q_j^2(x_j) \ldots$ in the tail provided the number of repetitions of each variable is bounded by a constant. Allowing the variables $x_j$ to occur an arbitrarily large number of times in the tail makes the non-emptiness problem subsume the intersection-non-emptiness problem for a sequence of tree automata, leading to EXPTIME-hardness already in the non-equational case. In the equational case, the complexity is likely to be higher.

### 2.1 Modeling Cryptographic Protocols

To demonstrate the modeling of protocols using equational H3 clauses, we take the following variant of the Needham-Schroeder-Lowe protocol from [2], in standard notation. The operator $+$ obeys $\mathrm{XOR}$ laws. Note that the analysis of [2] is for bounded number of sessions whereas our interest is in the analysis for unbounded number of sessions.

$$\begin{aligned} A &\rightarrow B : \{N_A, A\}_{K_B} \\ B &\rightarrow A : \{N_B, N_A + B\}_{K_A} \\ A &\rightarrow B : \{N_B\}_{K_B} \end{aligned}$$

We model it using Horn clauses as in [5]. We let the function symbols $\{\_\}\_$ and $\langle \_, \_ \rangle$ denote encryption and pairing. Each protocol step is repeated arbitrarily many times, although only finitely many nonces are used. For every pair of distinct agents $a$ and $b$ we chose constants $n_{ab}^1$ and $n_{ab}^2$ representing the nonces $N_A$ and $N_B$ which are used in sessions between $A$ and $B$. We choose a predicate $\mathsf{known}$ to represent messages known to the adversary. For every pair of agents $a$ and $b$, we have the following three clauses corresponding to the three protocol steps:

$$\begin{aligned} \mathsf{known}(\{\langle n_{ab}^1, a\rangle\}_{K_b}) \\ \mathsf{known}(\{\langle n_{ab}^2, x+b\rangle\}_{K_a}) &\Leftarrow \mathsf{known}(\{\langle x, a\rangle\}_{K_b}) \\ \mathsf{known}(\{x\}_{K_b}) &\Leftarrow \mathsf{known}(\{\langle x, n_{ab}^1 + b\rangle\}_{K_a}) \end{aligned}$$

This is based on the well known assumption that the adversary has full control over the network. All messages sent by agents are sent to him and all messages received by agents are received from him. The second clause for example represents the fact that if $b$ receives the message $\{\langle x, a\rangle\}_{K_B}$ for any $x$ then he will send the message $\{\langle n_{ab}^2, x + b\rangle\}_{K_a}$. In place of $x$, $b$ expects some nonce generated by $a$, however the adversary can fool $b$ by sending a message with something else in place of $x$. We need other clauses to represent the ability of the adversary to compute new messages from existing messages. We have the clause $\mathsf{known}(\{x\}_y) \Leftarrow \mathsf{known}(x) \wedge \mathsf{known}(y)$ to represent the

ability of the adversary to encrypt messages. His decryption ability is represented by clauses $\mathsf{known}(x) \Leftarrow \mathsf{known}(\{x\}_k) \wedge \mathsf{known}(k^{-1})$ for every key $k$, where $k^{-1}$ is the inverse of $k$. The ability of the adversary to pair and unpair messages is represented by the clauses $\mathsf{known}(\langle x, y\rangle) \Leftarrow \mathsf{known}(x) \wedge \mathsf{known}(y)$, $\mathsf{known}(x) \Leftarrow \mathsf{known}(\langle x, y\rangle)$ and $\mathsf{known}(y) \Leftarrow \mathsf{known}(\langle x, y\rangle)$. The clause $\mathsf{known}(x + y) \Leftarrow \mathsf{known}(x) \wedge \mathsf{known}(y)$ represents the ability of the adversary to apply the $+$ operation on known messages. The adversary's knowledge of other messages $m$ like identities of agents, public keys, private keys of dishonest agents, is represented by clauses $\mathsf{known}(m)$. Finally to check secrecy of a message $S$ we add the clause $\bot \Leftarrow \mathsf{known}(S)$ and check that the resulting clause set is satisfiable. All these clauses are H3. Our modeling used only finitely many nonces in infinitely many sessions. This is a safe abstraction: we detect all attacks against the protocols. However the secrecy problem is still undecidable. Indeed not all protocols can be modeled using H3 clauses without further safe abstractions.

As further examples, note that the modeling in [11] of the IKA.1 initial key agreement protocol requires only H3, or two-way automata clauses, modulo ACU. The verification in [11] is done using approximation techniques since the known algorithms for two-way ACU automata were too expensive. Our improved algorithms in this paper should let us dispense with approximation techniques in dealing with clauses required for such protocols. The clauses required for the modeling of the example protocol using XOR in [5] are also H3, whereas the upper bound provided for their class of clauses modulo XOR is non-elementary. The results in this paper are likely to provide efficient techniques for dealing with a large number of protocols in practice.

## 3 Parikh Images of Context-Free Grammars

Our equational tree automata are closely related to Parikh images of context free languages and Presburger formulas, as we show in this section. The *Parikh image* $\mathcal{P}(x)$ of a string $x$ on some alphabet maps each symbol $a$ to the number of occurrences of $a$ in $x$. The Parikh image of a set of strings is the set of Parikh images of its members. It is well-known [15] that Parikh images of context-free languages are *semilinear* sets, which are exactly the sets definable by (existential) Presburger formulas. We first improve this result by showing that for every context-free grammar $G$ one can compute in *linear time* an *existential* Presburger formula $\phi_G$ which characterizes the Parikh image of the language $L(G)$ generated by $G$. The proof combines a result from [8] with techniques from [18]. Recall that existential Presburger formulas $\phi$ are defined by the following grammar and interpreted over natural numbers:

$$t ::= 0 \mid 1 \mid x \mid t_1 + t_2 \qquad \phi ::= t_1 = t_2 \mid t_1 > t_2 \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \exists x \cdot \phi_1$$

The result in [8] is formulated in terms of communication-free Petri nets whose definition we recall next. A *Net* $N = (S, T, W)$ consists of a set $S$ of *places*, a set $T$ of *transitions* and a weight function $W : (S \times T) \cup (T \times S) \to \mathbb{N}$. If $W(x, y) > 0$ we say that there is an edge from $x$ to $y$ of weight $W(x, y)$. A net is *communication-free*, if for each transition $t$ there is at most one place $s$ with $W(s, t) > 0$ and furthermore $W(s, t) = 1$. A *marking* $M$ associates a number of *tokens* which each place, formally it is simply a function $S \to \mathbb{N}$. A *communication-free Petri net* is a pair $(N, M_0)$, where $N$ is a communication-free net and $M_0$ is a marking. A marking $M$ *enables* a transition

$t$ in a communication-free Petri net if $M(s) > 0$, for the place $s$ with $W(s,t) > 0$. If a transition $t$ is enabled for a marking $M$, then it can *occur* resulting in the marking $M'$ defined by $M'(s) = M(s) + W(t,s) - W(s,t)$, for every place $s$. A marking $M'$ is *reachable* from a marking $M$, if there is a sequence $\sigma = t_1 \cdots t_m$ of transitions and a sequence of markings $M_0 = M, M_1, \ldots, M_m = M'$ such that, for each $i$ the occurrence of $t_i$ in $(N, M_{i-1})$ results in $M_i$. We say also that $\sigma$ *can occur at* $M$ in that case. The following result was shown in [8] (Lemma 3.1 and Theorem 3.1).

**Theorem 3.** *Let $(N, M_0)$ be a communication-free Petri net with transition set $T$ and let $X$ be a function from $T$ to $\mathbb{N}$. There exists a sequence $\sigma$ of transitions with $\mathcal{P}(\sigma) = X$ that can occur in $(N, M_0)$ if and only if the following two conditions hold.*

*(a) For each place $s$, it holds $M_0(s) + \sum_{t \in T}[(W(t,s) - W(s,t))X(t)] \geq 0$, and*
*(b) in the subgraph of $N$ which is induced by the transitions $t \in T$ with $X(t) > 0$, every place is reachable (in the graph-theoretical sense) from some place $s$ with $M_0(s) > 0$.*

The intimate relationship between context-free grammars and communication-free Petri nets can be seen as follows. Let $G$ be a grammar with non-terminal set $V$, terminal set $U$, start symbol $A_0$ and set $P$ of productions. With $G$ we associate a net $N_G = (V \cup U, P, W)$. If $A \to \alpha$ is a production $p$ from $P$ then $W(A,p) = 1$ and $W(p,B)$ is the number of times which $B$ occurs in $\alpha$, for each $B \in V \cup U$. The Petri net $(N_G, M_G)$ is then obtained by setting $M_G(A_0) = 1$ and $M_G(A) = 0$ for all other $A$. Note that it is communication-free. An application of a production $p$ now corresponds to the occurrence of the transition $p$ in the net. Hence, it is not hard to see that $X$ is is the Parikh image of a sequence that can occur in $(N_G, M_G)$ if and only if there is a derivation of $G$ in which each production $p$ is used exactly $X(p)$ times.

Given a context-free grammar $G$ on terminals $a_1, \ldots, a_p$, we now compute an existential Presburger formula $\phi_G(x_{a_1}, \ldots, x_{a_p})$ representing its Parikh image, i.e. such that $\phi_G(n_1, \ldots, n_p)$ holds iff some string in $L(G)$ contains each $a_i$ exactly $n_i$ times. For this it basically remains to express requirement (b) of Theorem 3. This can be done analogously as in [18].

**Theorem 4.** *Given a context-free grammar $G$, one can compute an existential Presburger formula $\phi_G$ for the Parikh image of $L(G)$ in linear time.*

*Proof.* Let $G = (V, U, P, A_0)$ be context-free and let $N_G$ and $M_G$ be defined as above. Let, for each $A \in U$, $x_A$ be a variable, and for each $p \in P$, let $y_p$ be a variable. Clearly, the free variables of $\phi_G$ will be the variables $x_A$ with $A \in U$. We need three kinds of quantifier-free subformulas.

– First, for each $A \in V$ there is one equation which is directly determined from requirement (a) of Theorem 3. To this end, let $p_1, \ldots, p_k$ be all productions with $A$ on the left-hand side and let, for each production $p$, $A(p)$ denote the number of occurrences of $A$ on the right hand side of $p$. Then $\phi_G$ contains the equation $M_G(A) + \sum_{p \in P} A(p) y_p - \sum_{i=1}^{k} y_{p_i} = 0$. Note that we have $= 0$ instead of $\geq 0$ here, as we have to make sure that the derivation under consideration is complete, i.e., there are no remaining non-terminals. Note further, that we do not need such subformulas for $A \in U$ as such $A$ only occur on right-hand sides of productions.

– Next, we have to make sure that the values $x_A$ are consistent with the $y_p$. To this end, we have, for each $A \in U$ an equation $x_A = \sum_{p \in P} A(p) y_p$.
– Finally, it remains to express requirement (b) of Theorem 3. For this purpose, we use additional variables $z_A$, for each $A \in U \cup V$. The idea is that the $z_A$ reflect the distance of $A$ from $A_0$ in a spanning tree on the subgraph of $N_G$ induced by those $p$ with $y_p > 0$. To this end, we use the following kinds of formulas.
  - We have $x_A = 0 \vee z_A > 0$, for each $A \in U$.
  - If $p_1, \ldots, p_l$ are the productions with $A$ on the right-hand side and $B_1, \ldots, B_l$ are their corresponding left-hand sides then we have a formula $(z_A = 0) \vee \bigvee_{i=1}^{l} (z_A = z_{B_i} + 1 \wedge y_{p_i} > 0 \wedge z_{B_i} > 0)$. If one of the $B_i$ is the start symbol $A_0$ the corresponding disjunct is replaced by $z_A = 1 \wedge y_{p_i} > 0$.

It is not hard to prove that the resulting formula $\phi_G$, in which all variables, except the $x_A$ with $A \in U$, are existentially quantified, characterizes exactly the Parikh image of $L(G)$. For the one direction, if a vector is in the Parikh image of $L(G)$, the variables can be chosen such that they correspond to a derivation of $G$. Otherwise, if a vector satisfies $\phi_G$ then the equations for the $z_A$ make sure that condition (b) of Theorem 3 is fulfilled and the remaining equations verify that there is a derivation of a string with the corresponding numbers of symbols.

Finally, the size of $\phi_G$ is linear in the size of $G$, i.e., basically the size of $P$. Note that, in the sums over $p \in P$ only summands with $A(p) > 0$ are taken. Implemented thoroughly on a register machine, the construction of $\phi_G$ is possible in linear time. □

Recall also that to check satisfiability of an existential Presburger formula, we can first move quantifiers to the top, then non-deterministically replace subformulas $\phi_1 \vee \phi_2$ by $\phi_1$ or $\phi_2$, and check satisfiability of the resulting formula $\exists x_1 \cdot \ldots \exists x_n \cdot \phi$ where $\phi$ is a conjunction of equations and inequations. As satisfiability for formulas in the latter form is NP-complete, we have:

**Lemma 5.** *Satisfiability of existential Presburger formulas is NP-complete.*

It remains to relate equational tree automata to context free grammars. Let $\mathcal{A}$ be a constants-only ACU automaton on constants $a_1, \ldots, a_p$. Modulo ACU, the terms are then of the form $n_1 a_1 + \ldots + n_p a_p$, equivalently tuples $(n_1, \ldots, n_p) \in \mathbb{N}^p$. We consider $\mathcal{A}$ as a context-free grammar. States of $\mathcal{A}$ are non-terminals and constants are terminals. Clauses $P(0)$, $P(a)$, $P(x) \Leftarrow Q(x)$ and $P(x + y) \Leftarrow P_1(x) \wedge P_2(y)$ are productions $P \to \lambda$, $P \to a$, $P \to Q$ and $P \to P_1 P_2$ respectively where $\lambda$ is the empty string. The final state $P$ is the start symbol. From Theorem 4 we obtain an existential Presburger formula, which we denote as $\phi_{\mathcal{A},P}(x_{a_1}, \ldots, x_{a_p})$, such that $\phi_{\mathcal{A},P}(n_1, \ldots, n_p)$ holds iff $n_1 a_1 + \ldots + n_p a_p \in L_P(\mathcal{A}/\mathrm{ACU})$.

**Lemma 6.** *For any constants-only* ACU *automaton $\mathcal{A}$ and state $P$ we can compute in linear time an existential Presburger formula $\phi_{\mathcal{A},P}$ describing $L_P(\mathcal{A}/\mathrm{ACU})$.*

## 4 One-Way ACU Automata

We show in this section that membership and intersection-non-emptiness for one-way ACU automata are NP-complete. We first make the following observation from [20] about derivations in ACU automata, allowing us to reuse certain parts of derivations.

**Lemma 7.** *Let $\mathcal{E}$ be any set of equations containing* ACU*. Consider a derivation $\delta$ of an atom $P(t)$ modulo $\mathcal{E}$. Let $\delta_1, ..., \delta_n$ be non-overlapping subderivations of $\delta$ such that outside the $\delta_i$'s, the only equations used are* ACU *and the set $S$ of clauses used contains only equational clauses (see Figure 1.) Suppose the conclusions of $\delta_1, ..., \delta_n$ are $P_1(t_1), ..., P_n(t_n)$. Then*

1. $t =_{\mathrm{ACU}} t_1 + ... + t_n$
2. *If there are derivations $\delta'_1, ..., \delta'_n$ of atoms $P_1(s_1), ..., P_n(s_n)$ modulo $\mathcal{E}$ then there is a derivation $\delta'$ of $P(s_1 + ... + s_n)$ modulo $\mathcal{E}$, containing $\delta'_i$'s as subderivations, such that outside the $\delta'_i$'s, the only equations used are* ACU*, and all clauses used belong to $S$.*

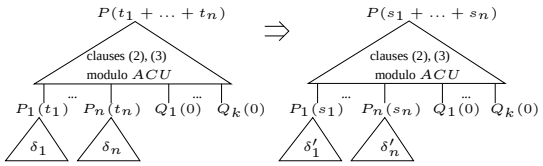


**Fig. 1.** Reuse of ACU derivations

The following definition from [20] gives one way of computing such $\delta_i$'s and $P_i(t_i)$'s:

**Definition 8.** *Consider a derivation $\delta$ of an atom $P(t)$ in a one-way automaton modulo* ACU*. Let $\delta_1, ..., \delta_n$ be the set of maximal subderivations of $\delta$ in which the last step used is an application of a free pop clause (or base clause). Suppose the conclusions of $\delta_1, ..., \delta_n$ are $P_1(t_1), ..., P_n(t_n)$ (in which case $t_1, ..., t_n$ must be functional). Then we will say that the (unordered) list of atoms $P_1(t_1), ..., P_n(t_n)$ is the* functional support *of the derivation $\delta$. (From Lemma 7 we have $t =_{\mathrm{ACU}} t_1 + ... + t_n$).*

Lemma 7 tells us how to reuse an arbitrarily large derivation involving only $+$-pop clauses, zero clauses and $\epsilon$-clauses. Sets of such derivations can also be represented by Presburger-formulas, by using some constants to represent the effect of other clauses. Formally consider a set $S$ of $+$-pop clauses, zero clauses, $\epsilon$-clauses on some set of predicates $\mathbb{P}$. Introduce constants $a_{P,Q}$ for states $P$ and $Q$. Intuitively $a_{P,Q}$ represents terms accepted by both $P$ and $Q$. For a set $Z \subseteq \mathbb{P}^2$, define constants-only ACU automaton $S[Z] = S \cup \{P(a_{P,Q}), Q(a_{P,Q}) \mid (P, Q) \in Z\}$. Lemma 6 then allows us to represent the languages $L_P(S[Z]/\mathrm{ACU})$ by existential Presburger formulas $\phi_{S[Z],P}$.

We now show how to decide intersection-non-emptiness of one-way ACU automata. The procedure can be thought of as marking of non-empty states in the product automaton computed in [20]. The relations $\Rrightarrow_{\mathrm{ACU}}$ and $\Rrightarrow_{free}$ below allow us to mark new states. The proof of the NP upper bound then involves guessing an increasing sequence of marked states. Formally, consider one-way automata $\mathcal{A}$ and $\mathcal{B}$ on sets of states $\mathbb{P}$ and $\mathbb{Q}$. Given $Z \subseteq \mathbb{P} \times \mathbb{Q}$ and $(P, Q) \in \mathbb{P} \times \mathbb{Q}$, if $L_P(\mathcal{A}_{eq}[Z]/\mathrm{ACU}) \cap L_Q(\mathcal{B}_{eq}[Z]/\mathrm{ACU}) \neq \emptyset$ then we say that $Z \Rrightarrow_{\mathrm{ACU}} (P, Q)$. Intuitively this means that if the pairs of states in $Z$ have non-empty intersection, then on the basis of the equational clauses in $\mathcal{A}$ and $\mathcal{B}$ we

can conclude that $P$ and $Q$ have non-empty intersection. A term in $L_P(\mathcal{A}_{eq}[Z]/\mathrm{ACU})$ represents the effect of an arbitrarily large derivation using equational clauses of $\mathcal{A}$, starting from derivations of terms in intersections of pair of states of $Z$, and ending at $P$. Formally:

**Lemma 9.** *If* $L_{P'}(\mathcal{A}/\mathrm{ACU}) \cap L_{Q'}(\mathcal{B}/\mathrm{ACU}) \neq \emptyset$ *for all* $(P', Q') \in Z$ *and* $Z \Rrightarrow_{\mathrm{ACU}} (P, Q)$ *then* $L_P(\mathcal{A}/\mathrm{ACU}) \cap L_Q(\mathcal{B}/\mathrm{ACU}) \neq \emptyset$.

*Proof.* For each $(P', Q') \in Z$ we have terms $t_{P',Q'}$ such that $P'(t_{P',Q'})$ and $Q'(t_{P',Q'})$ are derivable in $\mathcal{A}/\mathrm{ACU}$ and $\mathcal{B}/\mathrm{ACU}$ respectively. As $Z \Rrightarrow_{\mathrm{ACU}} (P, Q)$ hence we have some $a_{P_1,Q_1} + \ldots + a_{P_n,Q_n} \in L_P(\mathcal{A}_{eq}[Z]/\mathrm{ACU}) \cap L_Q(\mathcal{B}_{eq}[Z]/\mathrm{ACU})$. From the definition of $\mathcal{A}_{eq}[Z]$ the derivation of $P(a_{P_1,Q_1} + \ldots + a_{P_n,Q_n})$ in $\mathcal{A}_{eq}[Z]/ACU$ has a functional support $P_1(a_{P_1,Q_1}), \ldots, P_n(a_{P_n,Q_n})$. Also $P_i(t_{P_i,Q_i})$ are derivable in $\mathcal{A}/\mathrm{ACU}$. By Lemma 7 $P(t_{P_1,Q_1} + \ldots + t_{P_n,Q_n})$ is derivable in $\mathcal{A}/\mathrm{ACU}$. Similarly $Q(t_{P_1,Q_1} + \ldots + t_{P_n,Q_n})$ is derivable in $\mathcal{B}/\mathrm{ACU}$. Hence $L_P(\mathcal{A}/\mathrm{ACU}) \cap L_Q(\mathcal{B}/\mathrm{ACU}) \neq \emptyset$. □

We write $Z \Rrightarrow_{free} (P, Q)$ to mean that $\mathcal{A}$ has some free pop clause $P(f(x_1, \ldots, x_n)) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n)$ and $\mathcal{B}$ has some free pop clause $Q(f(x_1, \ldots, x_n)) \Leftarrow Q_1(x_1) \wedge \ldots \wedge P_n(x_n)$ such that $\{(P_i, Q_i)\} \in Z$ for $1 \leq i \leq n$. Intuitively this means that if the pairs of states in $Z$ have non-empty intersection, then on the basis of the free pop clauses in $\mathcal{A}$ and $\mathcal{B}$ we can conclude that $P$ and $Q$ have non-empty intersection. We write $Z \Rrightarrow (P, Q)$ to say that there are some $(P_1, Q_1), \ldots, (P_n, Q_n) \in \mathbb{P} \times \mathbb{Q}$ such that $(P_n, Q_n) = (P, Q)$ and for $1 \leq i \leq n$ we have

$$Z \cup \{(P_1, Q_1), \ldots, (P_{i-1}, Q_{i-1})\} \; (\Rrightarrow_{\mathrm{ACU}} \cup \Rrightarrow_{free}) \; (P_i, Q_i).$$

Intuitively this represents the effect of a sequence of conclusions using the $\Rrightarrow_{\mathrm{ACU}}$ and $\Rrightarrow_{free}$ rules. This rule suffices for detecting all pairs having non-empty intersection:

**Lemma 10.** *If* $L_P(\mathcal{A}/\mathrm{ACU}) \cap L_Q(\mathcal{B}/\mathrm{ACU}) \neq \emptyset$ *then* $\emptyset \Rrightarrow (P, Q)$.

*Proof.* We do induction on the size of the given term $t \in L_P(\mathcal{A}/\mathrm{ACU}) \cap L_Q(\mathcal{B}/\mathrm{ACU})$. Let $t = t_1 + \ldots + t_m$ where $t_i = f_i(t_i^1, \ldots, t_i^{k_i})$ is functional for $1 \leq i \leq m$. The derivation of $P(t)$ has some functional support $P_1(t_1), \ldots, P_m(t_m)$ where for $1 \leq i \leq m$ the derivation of $P_i(t_i)$ uses a clause $P_i(f_i(x_1, \ldots, x_{k_i})) \Leftarrow P_i^1(x_1) \wedge \ldots \wedge P_i^{k_i}(x_{k_i})$ and the derivations of $P_i^1(t_i^1), \ldots, P_i^{k_i}(t_i^{k_i})$. Similarly the derivation of $Q(t)$ has some functional support $Q_1(t_1), \ldots, Q_m(t_m)$ where for $1 \leq i \leq m$ the derivation of $Q_i(t_i)$ uses a clause $Q_i(f_i(x_1, \ldots, x_{k_i})) \Leftarrow Q_i^1(x_1) \wedge \ldots \wedge Q_i^{k_i}(x_{k_i})$ and the derivations of $Q_i^1(t_i^1), \ldots, Q_i^{k_i}(t_i^{k_i})$. Hence $t_i^j \in L_{P_i^j}(\mathcal{A}/\mathrm{ACU}) \cap L_{Qj_i}(\mathcal{B}/\mathrm{ACU})$ for $1 \leq i \leq m, 1 \leq j \leq k_i$. By induction hypothesis we have $\emptyset \Rrightarrow (P_i^j, Q_i^j)$ for $1 \leq i \leq m, 1 \leq j \leq k_i$. For $1 \leq i \leq m$, $\{(P_i^j, Q_i^j) \mid 1 \leq j \leq k_i\} \Rrightarrow_{free} (P_i, Q_i)$. Hence $\emptyset \Rrightarrow (P_i, Q_i)$ for $1 \leq i \leq m$. Also because of the above two functional supports we know from Lemma 7 that $a_{(P_1,Q_1)} + \ldots + a_{(P_m,Q_m)} \in L_P(\mathcal{A}_{eq}[Z]/\mathrm{ACU}) \cap L_Q(\mathcal{B}_{eq}[Z]/\mathrm{ACU})$ where $Z = \{(P_i, Q_i) \mid 1 \leq i \leq m\}$. Hence $Z \Rrightarrow_{\mathrm{ACU}} (P, Q)$. Hence $\emptyset \Rrightarrow (P, Q)$. □

**Lemma 11.** *Intersection-non-emptiness for one-way* ACU *automata is in NP.*

*Proof.* Let $P$ and$Q$ be the final states of $\mathcal{A}$ and $\mathcal{B}$ respectively. From Lemmas 9 and 10 $L_P(\mathcal{A}/\mathrm{ACU}) \cap L_Q(\mathcal{B}/\mathrm{ACU}) \neq \emptyset$ iff $\emptyset \Rrightarrow (P, Q)$. The latter is equivalent to existence

of (mutually distinct) pairs $(P_1, Q_1), \ldots, (P_n, Q_n) \in \mathbb{P} \times \mathbb{Q}$ such that $(P_n, Q_n) = (P, Q)$ and for $1 \leq i \leq n$, we have

$$Z_i = \{(P_1, Q_1), \ldots, (P_{i-1}, Q_{i-1})\}(\Rrightarrow_{\mathrm{ACU}} \cup \Rrightarrow_{free})(P_i, Q_i).$$

There are polynomially many such pairs. Checking $Z_i \Rrightarrow_{free} (P_i, Q_i)$ requires polynomial time. To check that $Z_i \Rrightarrow_{\mathrm{ACU}} (P_i, Q_i)$ we check satisfiability (Lemma 5) of

$$\phi_{\mathcal{A}_{eq}[Z_i],P_i}(x_{a_{P_1,Q_1}}, \ldots, x_{a_{P_{i-1},Q_{i-1}}}) \wedge \phi_{\mathcal{B}_{eq}[Z_i],Q_i}(x_{a_{P_1,Q_1}}, \ldots, x_{a_{P_{i-1},Q_{i-1}}})$$

which can be computed in polynomial time using Lemma 6. □

Hence the membership problem is also in NP. It is in fact NP-complete since the membership problem for Parikh images of languages generated by context free grammars is NP-hard [8]. This is also shown in [14], but the complexity of the intersection-non-emptiness problem is left open there. Indeed the technique of [14] is too specific to the membership problem. We now have:

**Theorem 12.** *The membership and intersection-non-emptiness problems for one-way* ACU *automata are NP-complete.*

## 5 Theories XOR and AG

We now show NP-completeness of membership and intersection-non-emptiness for one-way automata modulo other theories including XOR and AG. First we describe the XOR case. For $n \in \mathbb{N}$ if $n$ is odd then define $n^* = 1$ otherwise define $n^* = 0$. If $a_1, \ldots, a_k$ are mutually distinct constants then define $(n_1 a_1 + \ldots + n_k a_k)^* = n_1^* a_1 + \ldots + n_k^* a_k$. For a set $L$ of such terms define $L^* = \{t^* \mid t \in L\}$. Because of the cancellation axiom, we also need to decide intersection-non-emptiness of all pairs of states in the same automaton. Hence we consider a single one-way automaton $\mathcal{A}$ on set of states $\mathbb{P}$, instead of two different automata as in the ACU case. Given $Z \subseteq \mathbb{P}^2$ and $(P, Q) \in \mathbb{P}^2$, we say $Z \Rrightarrow_{\mathrm{XOR}} (P, Q)$ to mean that $(L_P(\mathcal{A}_{eq}[Z]/\mathrm{ACU}))^* \cap (L_Q(\mathcal{A}_{eq}[Z]/\mathrm{ACU}))^* \neq \emptyset$. This is the counterpart of the $\Rrightarrow_{\mathrm{ACU}}$ relation in the ACU case. Intuitively, because of the cancellation axiom of XOR, we only need to check whether a constant occurs an even or odd number of times. The relations $\Rrightarrow_{free}$ and $\Rrightarrow$ are redefined as expected. As in the ACU case, states $P$ and $Q$ have non-empty intersection iff $\emptyset \Rrightarrow (P, Q)$. The construction is easily generalized to the $\mathrm{XOR}_p$ theory for any (fixed) $p \geq 2$: we need to consider intersection-non-emptiness for $p$-tuples of states.

**Lemma 13.** *Intersection-non-emptiness for one-way* $\mathrm{XOR}_p$ *automata is in NP. This holds in particular for the theory* XOR*.*

*Proof.* (Sketch:) The algorithm works as in the ACU case. To check that $Z = \{(P_1, Q_1), \ldots, (P_n, Q_n)\} \Rrightarrow_{\mathrm{XOR}} (P, Q)$ we check that the formula

$$\phi_{\mathcal{A}_{eq}[Z],P}(x_{a_{P_1,Q_1}}, \ldots, x_{a_{P_n,Q_n}}) \wedge \phi_{\mathcal{B}_{eq}[Z],Q}(y_{a_{P_1,Q_1}}, \ldots, y_{a_{P_n,Q_n}})$$
$$\wedge \textstyle\bigwedge_{1 \leq i \leq n} (even(x_{a_{P_i,Q_i}}) \wedge even(y_{a_{P_i,Q_i}}) \vee odd(x_{a_{P_i,Q_i}}) \wedge odd(y_{a_{P_i,Q_i}}))$$

is satisfiable where $even(x) \equiv \exists y \cdot x = 2y$ and $odd(x) \equiv \exists y \cdot x = 2y + 1$. □

Next we prove NP-hardness of membership.

**Lemma 14.** *Membership for one-way* $\mathrm{XOR}_p$ *automata is NP-hard for* $p \geq 2$*.*

*Proof.* (Sketch:) We use a reduction from the 3-colorability problem for undirected graphs. For a graph $G = (V, E)$ we define the automaton $\mathcal{A}_G$ as follows. We have the set $\{r, b, g\}$ of three colors. For every vertex $v$, color $c$ and edge $e$ adjacent on $v$, introduce a fresh constant $a_{e,v,c}$, representing the assignment of color $c$ to $v$. For every edge $e$ joining $u$ and $v$, introduce a fresh predicate $P_e$. For every pair $(c_1, c_2)$ of distinct colors, add clause $P_e(a_{e,u,c_1} + a_{e,v,c_2})$ to $\mathcal{A}_G$. For every vertex $v$ introduce a fresh predicate $P_v$. For every color $c$, add the clause $P_v(\sum_{e \in E, e \text{ adjacent on } v}(p-1)a_{e,v,c})$ to $\mathcal{A}_G$. Finally add the clause $P(\sum_{v \in V} x_v + \sum_{e \in E} x_e) \Leftarrow \bigwedge_{v \in V} P_v(x_v) \wedge \bigwedge_{e \in E} P_e(x_e)$ to $\mathcal{A}_G$. Then 3-colorability of $G$ is equivalent to $0 \in L_P(\mathcal{A}_G/\mathrm{XOR}_p)$. □

**Theorem 15.** *The membership and intersection-non-emptiness problems for one-way* $\mathrm{XOR}_p$ *automata are NP-complete for* $p \geq 2$*. This holds in particular for* XOR.

To deal with the AG case, we use the ACUD theory as a tool. This is similar to the way the XOR case was dealt with using the ACU theory. The ACUD theory allows us to normalize terms by pushing the $-$ symbol downwards to functional terms. First we consider constant only ACUD automata. $\Sigma_f$ is the set of constants in our signature. Let $\overline{\Sigma}_f$ be a set of fresh constants $\{\overline{a} \mid a \in \Sigma_f\}$. Terms built from $\Sigma_f \cup \{+, -, 0\}$ modulo ACUD are of the form $a_1 + ... + a_m - b_1 - ... - b_n$ $(m, n \geq 0)$ while those built from $\Sigma_f \cup \overline{\Sigma}_f \cup \{+, 0\}$ modulo ACU are of the form $a_1 + ... + a_m + \overline{b_1} + ... + \overline{b_n}$ $(m, n \geq 0)$. Hence there is a natural 1-1 correspondence between terms (languages) on $\Sigma_f \cup \{+, -, 0\}$ modulo ACUD and terms (languages) on $\Sigma \cup \overline{\Sigma}_f \cup \{+, 0\}$ modulo ACU. Then modulo this correspondence of languages we have [20]:

**Lemma 16.** *The language accepted by a constants-only* ACUD *automaton* $\mathcal{A}$ *with constants from* $\Sigma_f$ *is also accepted by a constants-only* ACU *automaton* $\mathcal{B}$ *with constants from* $\Sigma_f \cup \overline{\Sigma_f}$*.* $\mathcal{B}$ *is computable in linear time from* $\mathcal{A}$*.*

For a set $S$ of $+$-pop clauses, minus clauses, zero clauses and $\epsilon$-clauses, and a set $Z$ of pairs of states, the automaton $S[Z]$ is defined as before. If the elements of $Z$ are $(P_1, Q_1), \ldots, (P_k, Q_k)$ then using Lemmas 6 and 16 we can compute in polynomial time an existential Presburger formula $\phi_{S[Z],P}(x_{a_{P_1,Q_1}}, \ldots, x_{a_{P_k,Q_k}}, x'_{a_{P_1,Q_1}}, \ldots, x'_{a_{P_k,Q_k}})$ such that $\phi_{S[Z],P}(n_1, \ldots, n_k, m_1, \ldots, m_k)$ holds iff $n_1 a_{P_1,Q_1} + \ldots + n_k a_{P_k,Q_k} - m_1 a_{P_1,Q_1} - \ldots - m_k a_{P_k,Q_k} \in L_P(S[Z]/\mathrm{ACUD})$.

We now show how to decide intersection-non-emptiness of one-way AG automata. Consider a one-way automaton $\mathcal{A}$ on set of states $\mathbb{P}$. Given $Z \subseteq \mathbb{P}^2$ and $(P, Q) \in \mathbb{P}^2$, we say $Z \Rrightarrow_{\mathrm{AG}} (P, Q)$ to mean that some $n_1 a_{P_1,Q_1} + \ldots + n_p a_{P_p,Q_p} - m_1 a_{P_1,Q_1} - \ldots - m_p a_{P_p,Q_p} \in L_P(\mathcal{A}/\mathrm{ACUD})$ and some $n'_1 a_{P_1,Q_1} + \ldots + n'_p a_{P_p,Q_p} - m'_1 a_{P_1,Q_1} - \ldots - m'_p a_{P_p,Q_p} \in L_Q(\mathcal{A}/\mathrm{ACUD})$ where $(P_1, Q_1), \ldots, (P_p, Q_p)$ are the mutually distinct elements of $Z$ and $n_i - m_i = n'_i - m'_i$ for $1 \leq i \leq p$. This corresponds to the relations $\Rrightarrow_{\mathrm{ACU}}$ and $\Rrightarrow_{\mathrm{XOR}}$ in the ACU and XOR cases respectively, and takes care of possible cancellations using the AG axioms. Note the use of ACUD above instead of ACU. The relations $\Rrightarrow_{free}$ and $\Rrightarrow$ are defined as expected. We use a generalization of Lemma 7 to the ACUD case. The rest works as in the previous cases, and we have:

**Theorem 17.** *The membership and intersection-non-emptiness problems for one-way* AG *automata are NP-complete.*

Note that the above lower bound is inherited from the ACU case. Also, while we introduced the ACUD theory only as a tool to deal with the AG theory, these techniques clearly also work in the ACUD case. In fact the proofs become simpler because we do not have to deal with cancellations. The lower bound is also inherited from the ACU case. We merely state the result:

**Theorem 18.** *The membership and intersection-non-emptiness problems for one-way* ACUD *automata are NP-complete.*

## 6 $\mathcal{H}3$ and Two-Way Automata

We now show how to deal with two-way automata and $\mathcal{H}3$. First note that membership and intersection-non-emptiness modulo our equational theories are NP-hard, as for the one-way case. While non-emptiness in the one-way case is decidable in linear time, in the two-way case it becomes NP-hard because the intersection-non-emptiness problem reduces to the non-emptiness problem. To decide intersection-non-emptiness of states $P_1$ and $P_2$ we create fresh states $P_3, P_4$ and $P_5$ and add clauses $P_3(0)$, $P_4(f(x,y)) \Leftarrow P_1(x) \wedge P_3(y)$ and $P_5(x) \Leftarrow P_4(f(x,y)) \wedge P_2(y)$. Then non-emptiness of $P_5$ is equivalent to intersection-non-emptiness of $P_1$ and $P_2$. We now first show that the NP-upper bound holds also for two-way automata modulo our equational theories. We illustrate the techniques for the XOR case, the other cases are similar.

The key idea is to add new $\epsilon$-clauses to the automata till the push clauses become redundant. For example the push clause $P(x) \Leftarrow Q(f(x))$ and the pop clause $Q(f(x)) \Leftarrow R(x)$ can be "short-cut" to produce the $\epsilon$-clause $P(x) \Leftarrow R(x)$. This is the main idea in the non-equational case. In the XOR case, there can be arbitrarily many applications of $+$-pop clauses in between the applications of the free pop clause and the push clause. However after cancellations using the XOR axioms, only a functional term should be left before application of the push clause. This is formalized as follows, as in [20]. Consider a two-way automaton $\mathcal{A}$. For any two-way automaton $\mathcal{A}'$, $\mathcal{A}'_{ow}$ denotes the subset of $\mathcal{A}$ without the push clauses. If there is some $Z \subseteq \mathbb{P}^2$ such that

(1) $R(x_i) \Leftarrow P(f(x_1, \ldots, x_n)) \wedge \bigwedge_{j \in \{1,\ldots,n\} \setminus \{i\}} R_j(x_j) \in \mathcal{A}$
(2) $Q(f(x_1, \ldots, x_n)) \Leftarrow Q_1(x_1) \wedge \ldots \wedge Q_n(x_n) \in \mathcal{A}$
(3) $a_{Q,Q} + 2n_1 a_{P_1,P'_1} + \ldots + 2n_m a_{P_m,P'_m} \in L_P(\mathcal{A}_{eq}[Z \cup \{(Q,Q)\}]/\mathrm{ACU})$
(4) $L_{P_j}(\mathcal{A}_{ow}/\mathrm{XOR}) \cap L_{P'_j}(\mathcal{A}_{ow}/\mathrm{XOR}) \neq \emptyset$ for $1 \leq j \leq m$
(5) for $j \in \{1, \ldots, n\} \setminus \{i\}$, $L_{Q_j}(\mathcal{A}_{ow}/\mathrm{XOR}) \cap L_{R_j}(\mathcal{A}_{ow}/\mathrm{XOR}) \neq \emptyset$

then we write $\mathcal{A} \rhd \mathcal{A} \cup \{R(x_i) \Leftarrow Q_i(x_i)\}$ provided $R(x_i) \Leftarrow Q_i(x_i) \notin \mathcal{A}$. Note that the pairs $(P_j, P'_j)$ in (3) are necessarily from $Z \cup \{(Q,Q)\}$. The relation $\rhd$ is one-step of our saturation procedure. It does not affect the set of derivable atoms modulo XOR[20]. From Theorem 4 we now know that the validity of a saturation step is in NP.

Note that there are only polynomially many $\epsilon$-clauses possible. Given a two-way automaton $\mathcal{A}$, we can keep on adding new $\epsilon$-clauses as long as possible. In the end we have an automaton $\mathcal{B}$. We then remove all push clauses to get one-way automaton $\mathcal{B}_{ow}$. This last step does not affect the set of derivable atoms modulo XOR[20]. This gives us a way of deciding intersection-non-emptiness in NP. We guess the saturated automaton by choosing a sequence of saturation steps. We then remove all push clauses and check

intersection-non-emptiness on the resulting one-way automaton. The same techniques work in the case of other theories, and we have:

**Theorem 19.** *Let $\mathcal{E} \in \{\mathrm{ACU}, \mathrm{XOR}, \mathrm{AG}, \mathrm{ACUD}, \mathrm{XOR}_p \mid p \geq 2\}$. The membership, non-emptiness and intersection-non-emptiness problems for two-way $\mathcal{E}$ tree automata are NP-complete.*

This also allows us to deal with the class $\mathcal{H}3$. For this we apply the transformation given in [13] which takes polynomial time and produces a set of H3 clauses of the form

(1) $P(x_i) \Leftarrow Q(f(x_1, \ldots, x_n)) \wedge \bigwedge_{j \in \{1,\ldots,n\} \setminus \{i\}} P_j(x_j)$
(2) $P(x_1, \ldots, x_n) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n) \wedge \bigwedge_{1 \leq i \leq m} Q_i()$
(3) $P(x_i) \Leftarrow Q(x_1, \ldots, x_n) \wedge \bigwedge_{j \in \{1,\ldots,n\} \setminus \{i\}} P_j(x_j)$
(4) $P() \Leftarrow Q(x_1, \ldots, x_n) \wedge \bigwedge_{1 \leq i \leq n} P_i(x_i)$

besides one-way automata clauses. The transformation preserves satisfiability. Also condition (3) of the definition of H3 clauses ensures that $+$ does not occur in the tail of clauses except in case of theories $\mathrm{XOR}$ or $\mathrm{AG}$. In these cases, $+$ can be removed from tails as in Section 2. The symbol $-$ is also removed from tails as in Section 2. Next to get rid of predicates with arbitrary arities, we encode atoms $P(t_1, \ldots, t_n)$ as $P'(f_n(t_1, \ldots, t_n))$. Then we are left with just two-way automata clauses, except for the presence of nullary predicates. The saturation procedure above is easily generalized to take care of nullary predicates: it can now generate clauses of the form $P()$. Clauses $\bot \Leftarrow A_1 \wedge \ldots \wedge A_n$ are also treated as definite clauses by treating $\bot$ as a nullary predicate. To check unsatisfiability we check that the clause $\bot$ can be generated using saturation steps.

**Theorem 20.** *Unsatisfiability for $\mathcal{H}3$ modulo each of the theories $\mathrm{ACU}, \mathrm{XOR}, \mathrm{AG}, \mathrm{ACUD}$ and $\mathrm{XOR}_p$ for $p \geq 2$ is NP-complete.*

In case of the related theory $\mathrm{ACUI}$ of idempotent commutative monoids, it is known that two-way automata are powerful enough to encode alternation [20]. Hence the associated problems become EXPTIME-hard, though they are decidable using the techniques of [21]. This is similar to the $\mathrm{XOR}$ case, whereas alternating automata are undecidable in the case of the theories $\mathrm{ACU}, \mathrm{AG}$ and $\mathrm{ACUD}$.

## 7 Conclusion

We have shown that the polynomially decidable class $\mathcal{H}3$ of Horn clauses can be extended with the theories $\mathrm{ACU}, \mathrm{XOR}, \mathrm{AG}, \mathrm{ACUD}$ and $\mathrm{XOR}_p$ for $p \geq 2$, to obtain NP-complete unsatisfiability problems. This improves known algorithms for one-way as well as two-way tree automata modulo these theories: essentially all problems are NP-complete. Our algorithms require deterministic polynomial time using an oracle for existential Presburger formulas, suggesting efficient implementations are possible.

While these decidability results can also be extended to the theory $\mathrm{ACUI}$ of idempotent commutative monoids, the complexity is EXPTIME-hard and probably higher. In the $\mathrm{ACU}$ case, we have forbidden the symbol $+$ to appear in non-ground negative literals. Without this restriction, the decidability question is still open. However the problems involved subsume other problems which are known to be EXPSPACE-hard and whose decidability is still open.

## References


1. B. Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *CSFW'01*, pages 82–96. IEEE Computer Society Press, 2001.
2. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP decision procedure for protocol insecurity with XOR. In *LICS'03*, pages 261–270, 2003.
3. T. Colcombet. Rewriting in the partial algebra of typed terms modulo AC. In *Electronic Notes in Theoretical Computer Science*, volume 68. Elsevier Science Publishers, 2002.
4. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. `http://www.grappa.univ-lille3.fr/tata`, 1997.
5. H. Comon-Lundh and V. Cortier. New decidability results for fragments of first-order logic and application to cryptographic protocols. In *RTA'03*, pages 148–164. Springer-Verlag LNCS 2706, 2003.
6. V. Cortier, S. Delaune, and P. Lafourcade. A survey of algebraic properties used in cryptographic protocols. *Journal of Computer Security*, 2005. To appear.
7. P. de Groote, B. Guillaume, and S. Salvati. Vector addition tree automata. In *LICS'04*, pages 64–73. IEEE Computer Society Press, 2004.
8. J. Esparza. Petri nets, commutative context-free grammars, and basic parallel processes. *Fundam. Inform.*, 31(1):13–25, 1997.
9. S. Ginsburg and E. H. Spanier. Semigroups, Presburger formulas and languages. *Pacific Journal of Mathematic*, 16(2):285–296, 1966.
10. J. Goubault-Larrecq and F. Parrennes. Cryptographic protocol analysis on real C code. In *VMCAI'05*, pages 363–379. Springer-Verlag LNCS 3385, 2005.
11. J. Goubault-Larrecq, M. Roger, and K. N. Verma. Abstraction and resolution modulo AC: How to verify Diffie-Hellman-like protocols automatically. *Journal of Logic and Algebraic Programming*, 2005. To Appear. Available as Research Report LSV-04-7, LSV, ENS Cachan.
12. D. Lugiez. Counting and equality constraints for multitree automata. In *FOSSACS'03*, pages 328–342. Springer-Verlag LNCS 2620, 2003.
13. F. Nielson, H. R. Nielson, and H. Seidl. Normalizable Horn clauses, strongly recognizable relations and Spi. In *SAS'02*, pages 20–35. Springer-Verlag LNCS 2477, 2002.
14. H. Ohsaki and T. Takai. Decidability and closure properties of equational tree languages. In *RTA'02*, pages 114–128. Springer-Verlag LNCS 2378, 2002.
15. R. J. Parikh. On context-free languages. *Journal of the ACM*, 13(4):570–581, October 1966.
16. M. Rusinowitch and L. Vigneron. Automated deduction with associative-commutative operators. *Applicable Algebra in Engineering, Communication and Computation*, 6:23–56, 1995.
17. H. Seidl, T. Schwentick, and A. Muscholl. Numerical document queries. In *PODS'03*, pages 155–166, 2003.
18. H. Seidl, T. Schwentick, A. Muscholl, and P. Habermehl. Counting in trees for free. In *ICALP'04*, pages 1136–1149. Springer-Verlag LNCS 3142, 2004.
19. H. Seidl and K. N. Verma. Flat and one-variable clauses: Complexity of verifying cryptographic protocols with single blind copying. In *LPAR'04*, pages 79–94. Springer-Verlag LNCS 3452, 2004.
20. K. N. Verma. Two-way equational tree automata for AC-like theories: Decidability and closure properties. In *RTA'03*, pages 180–196. Springer-Verlag LNCS 2706, 2003.
21. K. N. Verma. Alternation in equational tree automata modulo XOR. In *FSTTCS'04*, pages 518–530. Springer-Verlag LNCS 3328, 2004.
22. K. N. Verma and J. Goubault-Larrecq. Karp-Miller trees for a branching extension of VASS. Research Report LSV-04-3, LSV, ENS Cachan, France, January 2004.