On Closure under Complementation of Equational Tree Automata for Theories Extending AC ⋆

Kumar Neeraj Verma

LSV/CNRS UMR 8643 & INRIA Futurs projet SECSI & ENS Cachan, France
verma@lsv.ens-cachan.fr

Abstract. We study the problem of closure under complementation of languages accepted by one-way and two-way tree automata modulo equational theories. We deal with the equational theories of commutative monoids (ACU), idempotent commutative monoids ($ACUI$), Abelian groups ($ACUM$), and the theories of exclusive-or ($ACUX$), generalized exclusive-or ($ACUX_n$), and distributive minus symbol ($ACUD$). While the one-way automata for all these theories are known to be closed under intersection, the situation is strikingly different for complementation. We show that one-way ACU and $ACUD$ automata are closed under complementation, but one-way $ACUX$, $ACUX_n$, $ACUM$ and $ACUI$ automata are not. The same results hold for the two-way automata, except for the theory $ACUI$, as the two-way automata modulo all these theories except $ACUI$ are known to be as expressive as the one-way automata. The question of closure under intersection and complementation of two-way $ACUI$ automata is open.

1 Introduction

Tree automata [4, 2] enjoy many good properties: emptiness is decidable, the class of languages accepted is closed under Boolean operations notably. *Two-way* tree automata extend ordinary tree automata (call them *one-way* tree automata to distinguish them from two-way tree automata) by allowing transitions that not only construct terms, as in one-way automata, but also destruct terms (see [2], Chapter 7, *Alternating Tree Automata*). More specifically, one-way automata have transitions of the form "if terms $t_1, ..., t_n$ are accepted at states $q_1, ..., q_n$ then term $f(t_1, ..., t_n)$ is accepted at state q". The new transitions allowed in two-way automata are of the form "if term $f(t_1, ..., t_n)$ is accepted at state q and terms $t_{i_1}, ..., t_{i_k}$ are accepted at states $q'_1, ..., q'_k$ then term t_i is accepted at state q'" where $1 \leq i, i_1, ..., i_k \leq n$. Two-way tree automata are more conveniently represented by the so called *definite clauses* of first order logic which provide a uniform mechanism for representing various kinds of transitions, like *alternation* ("if term t is accepted at states q_1 and q_2 then t is accepted at state q"), transitions with equality constraints between subterms, etc. Two-way tree automata can be effectively reduced to one-way tree automata. As a result they also enjoy the good properties of decidability of emptiness and closure under Boolean operations.

⋆ Partially supported by the ACI "cryptologie" PSI-Robuste, ACI VERNAM, the RNTL project EVA and the ACI jeunes chercheurs "Sécurité informatique, protocoles cryptographiques et détection d'intrusions".

In spite of having the same expressiveness as one-way automata, two-way automata are sometimes more convenient to work with because of the new kinds of transitions. An important area in which two-way tree automata have recently been applied is verification of cryptographic protocols [11, 6], where terms are used to represent messages and tree languages to represent sets of messages known to an intruder. The transitions of two-way tree automata allow us to model the ability of the intruder to encrypt and decrypt messages. This approach works fine when the cryptographic primitives are assumed to be perfect, in which case the messages are the members of the free term algebra. However often complex cryptographic primitives have additional algebraic properties, which need to be modeled using equational theories. The theory of associativity and commutativity, and its extensions like the theories of exclusive-or and Abelian groups are some examples that occur often. As an example the original version of the Bull's recursive authentication protocol presented in [14] was formally proved correct using the assumption of perfect cryptographic primitives. However this protocol uses the exclusive-or operation for encryption. By taking into account the properties of the exclusive-or operation, an attack against this protocol was found in [15]. Also protocols like the group Diffie-Hellman protocol [16], use algebraic properties of modular exponentiation [3]. This motivates exploring variants of tree automata which take into account equational properties of the function symbols.

Several notions of tree automata accepting terms modulo equational theories have been proposed recently [9, 12, 18], which extend the tree automata concept. The notions of one-way and two-way *equational tree automata* have been studied extensively in [18] which deals with several variants of the equational theory ACU of an associative, commutative operator $+$ with unit 0. It shows the decidability of emptiness and closure under intersection of the one-way automata for all these theories, as well as the equivalence between two-way and one-way automata for most of these theories. However the problem of closure under complementation has been left open. In this paper we examine this question.

More specifically, we study one-way and two-way equational tree automata, modulo the theory ACU and several of its extensions. The theory ACU of an associative-commutative symbol $+$ with unit 0 consists of the three axioms:

(A) $x + (y + z) = (x + y) + z$
(C) $x + y = y + x$
(U) $x + 0 = x$

The extensions of ACU that we deal with are obtained by adding one of the following axioms to the base theory ACU:

(X) $x + x = 0$ (xor axiom)
(X_n) $\underbrace{x + \ldots + x}_{n \text{ times}} = 0$ (generalization of xor axiom, where $n \geq 2$)
(M) $x + (-x) = 0$ (minus axiom)
(D) $-(x + y) = (-x) + (-y)$, $-(-x) = x$, $-0 = 0$ (distributivity of '$-$' symbol)
(I) $x + x = x$ (idempotence)

where $-$ is an additional unary symbol. We name a theory by the names of its axioms; e.g., $ACUM$ is the theory of Abelian groups and $ACUX$ the theory of xor. Note that

D is implied by $ACUM$, so $ACUD$ is a (strictly) weaker theory than $ACUM$. The theories dealt with in this paper are ACU, $ACUX$, $ACUX_n$, $ACUD$, $ACUM$ and $ACUI$.

We show that modulo ACU and $ACUD$ the one-way automata are closed under complementation. On the other hand, modulo $ACUI$, $ACUM$ (Abelian groups), $ACUX$ (the theory of xor) and $ACUX_n$, the one-way automata are not closed under complementation. Except for the theory $ACUI$, the same results hold for two-way tree automata. The case of two-way $ACUI$ automata is open. An interesting pattern visible here is the coincidence between closure under complementation of the one-way equational tree automata and the linearity of the equational theory involved. These results on complementation are in sharp contrast to the results on intersection.

Plan. After some material on related work, we define our one-way and two-way equational tree automata in Section 2, recalling some important results, notably connections of ACU automata with semilinear sets. These results are used in Section 3 to show closure under complementation of (one-way and two-way) ACU automata. The (one-way and two-way) $ACUX$ and $ACUX_n$ automata are not closed under complementation, as shown in Section 4. To deal with the $ACUD$ and Abelian groups case, we first generalize the results of Section 1 on ACU automata to the $ACUD$ case in Section 5. These are then used to show closure under complementation of (one-way and two-way) $ACUD$ automata in Section 6. The results and proofs in the Abelian groups case ($ACUM$) are exactly as in the $ACUX$ case, as we argue in Section 7. Lastly we show that the one-way $ACUI$ automata are not closed under complementation in Section 8, leaving open the question for two-way $ACUI$ automata. We conclude in Section 9.

Related work. Tree automata modulo AC have been considered several times, though not all these notions coincide. Lugiez [9] considers one-way AC tree automata which have additional sort restrictions, but are also extended with a rich constraints language. Recent work by Lugiez [10] extends it to include equality and counting constraints, providing a rich framework that includes most known proposals for *one-way AC* tree automata with decidable emptiness problems. The resulting class of automata is also shown to be closed under Boolean operations. This framework is however incomparable to ours: while Lugiez's automata accommodate equality tests naturally, our two-way automata cannot avoid undecidability in the presence of equality constraints [7, 18].

Ohsaki [12, 13] considers a larger framework of (one-way) $\mathcal{E}$ tree automata, where $\mathcal{E}$ is an equational theory. Ohsaki's *regular* $\mathcal{E}$ automata coincide with our one-way $\mathcal{E}$ tree automata when $\mathcal{E}$ is linear (like AC, ACU), but this does *not* hold in theories like $ACUX$, $ACUM$, $ACUI$. For example, in Ohsaki's case, with transition rules $a \to q$, $b \to q$ and $0 \to q'$, and with the theory $ACUX$, we have $a + b \to^* q'$, meaning $a + b$ is accepted at q'. In our case, with the corresponding clauses $q(a)$, $q(b)$ and $q'(0)$, and with the theory $ACUX$, $a + b$ is not accepted at q'. For arbitrary $\mathcal{E}$ we do not know the relation between our automata and Ohsaki's automata, and the two notions appear rather dissimilar. One of the key differences is that while we extend the traditional correspondence between tree automata and logic programs from the non-equational case to the equational case, Ohaski's automata do not preserve this correspondence, as they mix equality on terms with equality on states, as illustrated by the above example.

While we consider tree automata as logic programs and extend them by adding equational theories, there has also been much work on *equational logic programs* [8] as such, which are logic programs in which the special equality predicate also appears, so that the equational theory can be coded in the logic program itself. In this context, complementation of equational tree automata corresponds to negation in equational logic programs. However our equational tree automata differ in that the equality symbol does not occur in the logic programs, and we separately consider an equational theory in which the equality predicate occurs. Moreover the author is not aware of any work on finding decision procedures or closure properties for subclasses of equational logic programs, especially for the ones corresponding to our automata.

The notion of *two-way* equational tree automata has been studied by Goubault-Larrecq and the author [7, 18] for the theory AC and its variants. Note that the specific theories $ACUX$, $ACUM$, etc., that we consider have traditionally not been considered in the framework of (one-way) tree automata; they give rise to specific technical problems and solutions. The notion of alternating two-way AC tree automata is treated in detail in [7], which also considers some additional push clause formats which are not relevant in this paper. Finally we have found that closure under Boolean operations has also been shown for the *multiset automata* of Colcombet [1], which correspond to the subclass of our one-way ACU automata in which all symbols other than $+, 0$ are unary, and were introduced for studying process rewrite systems.

To avoid confusion we clarify that the "two-way automata" of [17] are an entirely different notion from ours.

2 Two-Way $\mathcal{E}$-Tree Automata

Fix a signature Σ of function symbols, each coming with a fixed arity, and let $\mathcal{E}$ be an equational theory, inducing a congruence $=_{\mathcal{E}}$ on the terms built from Σ. We will use clauses of first order logic as a general means of representing various classes of automata.

A *definite clause* is an implication of the form:

$$P(t) \Leftarrow P_1(t_1) \wedge ... \wedge P_n(t_n) \qquad (1)$$

where $P, P_1, ..., P_n$ are predicates and $t, t_1, ..., t_n$ are terms built from Σ and variables. Given a finite set $\mathcal{C}$ of such definite clauses we define *derivations* of ground atoms using the following two rules:

$$\boxed{\frac{P_1(t_1\sigma) ... P_n(t_n\sigma)}{P(t\sigma)} \text{if } P(t) \Leftarrow P_1(t_1) \wedge ... \wedge P_n(t_n) \in \mathcal{C} \qquad \frac{P(s)}{P(t)} \text{if } s =_{\mathcal{E}} t}$$

where σ is a ground substitution. Thus a derivation is a tree-like structure, which should not be confused with the trees which are the terms built from Σ. Accordingly, a *subderivation* of a derivation δ is a subtree of δ. The *last step* of a derivation δ refers to the step at the root of δ. The connection of definite clauses with automata is as follows: predicates are states, finite sets of definite clauses are automata, and an atom $P(t)$ is derivable using $\mathcal{C}$, iff the term t is *accepted* at state P in the automaton $\mathcal{C}$. The derivations using $\mathcal{C}$ are sometimes called *runs* of the automaton $\mathcal{C}$.

It is also easy to see that the set of derivable atoms is exactly the least Herbrand model of the set of clauses modulo $\mathcal{E}$.

The language $\mathcal{L}_P(\mathcal{C}/\mathcal{E})$ is the set of terms t such that $P(t)$ is derivable. When $\mathcal{E}$ is the empty theory, we shall call it $\mathcal{L}_P(\mathcal{C})$. If in addition some state P_f is specified as being *final* then the language *accepted* by $\mathcal{C}$ is $\mathcal{L}(\mathcal{C}/\mathcal{E}) = \mathcal{L}_{P_f}(\mathcal{C}/\mathcal{E})$. Also, given a language L and an equational theory $\mathcal{E}$, $\mathcal{E}(\mathcal{L})$ denotes the set of terms t such that $t =_{\mathcal{E}} s$ for some $s \in \mathcal{L}$.

We define *one-way automata* as consisting of clauses:

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

$$P(x) \Leftarrow P_1(x) \quad (3)$$

which we shall call *pop clauses* and *epsilon clauses* respectively. In (2), the variables $x_1, ..., x_n$ are distinct.

These automata (without equations) are exactly the classical tree automata usually described in the literature. It may be helpful to read the pop clause as: "if terms x_1,...,x_n are accepted at states P_1,...,P_n respectively, then $f(x_1, ..., x_n)$ is accepted at P". This is usually denoted by the rewrite rule $f(P_1, ..., P_n) \to P$. The epsilon clause corresponds to the rewrite rule $P_1 \to P$. We have the following easy result:

Lemma 1 ([18]). *For any one-way automaton $\mathcal{C}$ and equational theory $\mathcal{E}$, $\mathcal{L}_P(\mathcal{C}/\mathcal{E}) = \mathcal{E}(\mathcal{L}_P(\mathcal{C}))$. In particular emptiness of one-way $\mathcal{E}$ tree-automata is decidable.*

For Ohsaki's regular $\mathcal{E}$ tree automata, this result holds if $\mathcal{E}$ is linear, but not in general. A counter-example is the automaton modulo $ACUX$ given in Section 1.

Since we are dealing with theories extending ACU, we assume that Σ contains symbols $+$, 0 and in presence of equations D or M the symbol $-$. Note that we don't consider the case of Σ containing several $+$ (resp. $-$, 0) symbols. Symbols in $\Sigma_f = \Sigma \setminus \{+, -, 0\}$, are called *free*. Free symbols of zero arity are called *constants*. Terms of the form $f(t_1, ..., t_n)$ where f is free are called *functional terms*. Accordingly the pop clauses in our automata are of the following form:

$$P(x + y) \Leftarrow P_1(x) \wedge P_2(y) \quad (4) \qquad P(a) \text{ where } a \text{ is a constant} \quad (6)$$

$$P(0) \quad (5) \qquad P(-x) \Leftarrow P_1(x) \quad (7)$$

$$P(f(x_1, ..., x_n)) \Leftarrow P_1(x_1) \wedge ... \wedge P_n(x_n), f \text{ being free} \quad (8)$$

Clauses (6) are special cases of clauses (8).

One-way ACU (resp. $ACUI$, $ACUX$, $ACUX_n$) automata are sets of clauses (3), (4–6) and (8); one-way $ACUM$ and $ACUD$ automata in addition contain clauses (7).

We define two-way automata by adding the following kind of clauses

$$Q(x_i) \Leftarrow P(f(x_1, ..., x_n)) \wedge Q_1(x_{i_1}) \wedge ... \wedge Q_k(x_{i_k}), \quad (9)$$
$$f \text{ being free}, i \in \{1, ..., n\} \setminus \{i_1, ..., i_k\}, 1 \leq i_1 < ... < i_k \leq n$$

called *push clauses*, to one-way automata (the variables $x_1, ..., x_n$ are distinct.) Hence *two-way* automata are sets of clauses (3) and (4–9)—with the proviso that (7) is only included when $- \in \Sigma$. The side-conditions "$i \in \{1, ..., n\} \setminus \{i_1, ..., i_k\}, 1 \leq i_1 < ... < i_k \leq n$" have been introduced in clause (9) to avoid undecidability problems. This is discussed in [18], which also argues that removing the side-condition "f being free" from clause (9) has no impact in the case of theories $ACUX$ and $ACUM$ but is problematic in the case of other theories like ACU and $ACUD$. Hence, while the expression "two-way automata" is normally used in a general sense, without the restrictions imposed by the side-conditions in clause (9), however in the rest of this paper, this expression will be used in the specialized sense defined above. *Constant-only* automata are one-way automata which contain clauses (6) instead of the general clauses (8) (the only free symbols in the signature are constants.) Given a one-way or two-way automaton $\mathcal{C}$, we define $\mathcal{C}_{eq}$ to be the part of $\mathcal{C}$ with clauses (3), (4), (5) and (7) (the equational part), and $\mathcal{C}_{free}$ is the remaining part.

Example 1. Consider a signature having constants a and b, a unary symbol f and a binary symbol g. Define two-way automaton $\mathcal{A}$ modulo $ACUX$ to consist of the clauses

(i) $P_1(a)$
(ii) $P_2(b)$
(iii) $P_3(f(x)) \Leftarrow P_2(x)$
(iv) $P_4(0)$
(v) $P_5(g(x,y)) \Leftarrow P_4(x) \wedge P_3(y)$
(vi) $P_6(x+y) \Leftarrow P_1(x) \wedge P_3(y)$
(vii) $P_7(x+y) \Leftarrow P_6(x) \wedge P_1(y)$
(viii) $P_8(x) \Leftarrow P_5(g(x,y)) \wedge P_7(y)$
(ix) $P_9(x+y) \Leftarrow P_8(x) \wedge P_1(y)$

Then the following is an example of a derivation in the automaton. At each step, the clause applied or the equational rewriting used is indicated on the right side.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P_4(0)}\,(iv) \quad \dfrac{\dfrac{}{P_2(b)}\,(ii)}{P_3(f(b))}\,(iii)}{P_5(g(0,f(b)))}\,(v) \quad \dfrac{\dfrac{\dfrac{\dfrac{}{P_1(a)}\,(i) \quad \dfrac{\dfrac{}{P_2(b)}\,(ii)}{P_3(f(b))}\,(iii)}{P_6(a+f(b))}\,(vi) \quad \dfrac{}{P_1(a)}\,(i)}{P_7(((a+f(b))+a))}\,(vii)}{P_7(f(b))}\,(\text{ACUX})}{P_8(0)}\,(viii) \quad \dfrac{}{P_1(a)}\,(i)}{P_9(0+a)}\,(ix)}{P_9(a)}\,(\text{ACUX})$$

The languages accepted by all our one-way or two-way automata are trivially closed under union. It has been shown in [18] that the one-way automata for all these theories (ACU, $ACUD$, $ACUX$, $ACUX_n$, $ACUM$, $ACUI$) are closed under intersection. There it is also shown that two-way automata are as expressive as one-way automata, for all these theories, except $ACUI$ for which the problem is currently open. The emptiness test for one-way automata has already been dealt with (Lemma 1). In this paper, we shall concentrate on closure under complementation of these automata. Note also that closure under intersection and decidability of emptiness imply decidability of membership of one-way equational tree automata for all theories considered in this paper since, using Lemma 1, for any theory $\mathcal{E}$ and term t, the language $\mathcal{E}(\{t\})$ is easily recognized by a one-way $\mathcal{E}$ tree automaton.

First we recall some important results on constant-only ACU automata. Let the constants in our signature be $a_1, ..., a_p$. Modulo ACU, the ground terms are of the form $\sum_{i=1}^{p} n_i a_i$ with $n_i \in \mathbb{N}$: equivalently p-tuples of natural numbers. Recall that a *linear set* is a set of the form $\{\nu + n_1\nu_1 + ... + n_k\nu_k \mid n_1, ..., n_k \in \mathbb{N}\}$ for some $\nu, \nu_1, ..., \nu_k \in \mathbb{N}^p$. A *semilinear set* is a finite union of linear sets. The semilinear sets are exactly the sets definable in Presburger arithmetic [5]. In particular they are closed under union, intersection, complementation and projection. Then we have:

Lemma 2 **([18]).** *Constant-only ACU automata accept exactly semilinear sets.*

To prepare ourselves, we need another special property of ACU automata which allows us to 'reuse' parts of derivations. This will be required very often in the paper.

Lemma 3 **([18]).** *Let $\mathcal{E}$ be any set of equations containing ACU. Consider a derivation δ of an atom $P(t)$ modulo $\mathcal{E}$. Let $\delta_1, ..., \delta_n$ be non-overlapping subderivations of δ such that outside the δ_i's, the only equations used are ACU and the set S of clauses used contains only clauses of kind (3), (4) and (5) (see Figure 1, noting that the derivation trees are shown with the root at the top.) Suppose the conclusions of $\delta_1, ..., \delta_n$ are $P_1(t_1), ..., P_n(t_n)$. Then*

1. $t =_{ACU} t_1 + ... + t_n$
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 δ' of $P(s_1 + ... + s_n)$ modulo $\mathcal{E}$, containing δ'_i's as subderivations, such that outside the δ'_i's, the only equations used are ACU, and all clauses used belong to S.*

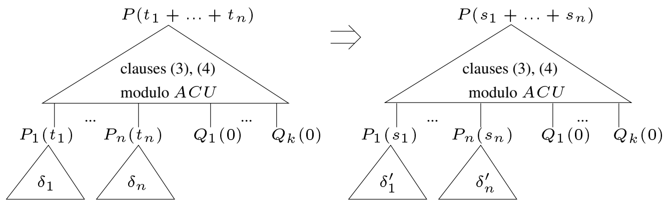

Fig. 1. Reuse of ACU derivations

The following definition gives one way of computing such δ_i's and $P_i(t_i)$'s:

Definition 1. *Consider a derivation δ of an atom $P(t)$ in a one-way automaton modulo ACU. Let $\delta_1, ..., \delta_n$ be the set of maximal subderivations of δ in which the last step used is an application of clause (8) (or clause (6)). 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 δ. (From Lemma 3 we have $t =_{ACU} t_1 + ... + t_n$.)*

3 Complementation of ACU Automata

Let $\mathcal{C}$ be a one-way ACU automaton with predicates from some finite set $\mathbb{P}$. We introduce new predicate symbols $\breve{S}$ and $\widehat{S}$ for each $S \subseteq \mathbb{P}$ and constants a_S for each $S \subseteq \mathbb{P}$. We intend $\breve{S}$ to accept terms accepted at all the predicates in S but nowhere else. $\widehat{S}$ is intended to accept the functional terms accepted at $\breve{S}$.

Define automaton $\mathcal{C}^*_{eq} = \mathcal{C}_{eq} \cup \{P(a_S) \mid P \in S\}$. The idea in defining $\mathcal{C}_{eq}$ is to compute all possible derivations using the clauses of the equational part. The constant a_S acts as abstraction for the terms accepted at $\widehat{S}$ for $S \subseteq \mathbb{P}$.

From Lemma 2, $\mathcal{L}_P(\mathcal{C}^*_{eq}/ACU)$ is a semilinear set for every $P \in \mathbb{P}$. Given $S \subseteq \mathbb{P}$, define $\mathcal{L}^S = \bigcap_{P\in S} \mathcal{L}_P(\mathcal{C}^*_{eq}/ACU) \setminus \bigcup_{P\in\mathbb{P}\setminus S} \mathcal{L}_P(\mathcal{C}^*_{eq}/ACU)$. This is a semilinear set. By Lemma 2, we can construct a constant-only automaton $\mathcal{C}_S$ with final state F_S such that $\mathcal{L}(\mathcal{C}_S/ACU) = \mathcal{L}^S$. We assume that automata $\mathcal{C}_S$'s are built from mutually disjoint sets of (fresh) states.

Define automaton $\mathcal{C}^\dagger$ to consist of the following clauses:

- for each $S \subseteq \mathbb{P}$ the clause $\breve{S}(x) \Leftarrow F_S(x)$.
- clauses (3), (4) and (5) (but not (6)) from each $\mathcal{C}_S$.
- for each clause $R(a_S)$ in some $\mathcal{C}_S$, the clause $R(x) \Leftarrow \widehat{S}(x)$.
- for each free function symbol f of arity n, and each $S_1, ..., S_n \subseteq \mathbb{P}$, the clause $\widehat{S}(f(x_1, ..., x_n)) \Leftarrow \breve{S}_1(x_1) \wedge ... \wedge \breve{S}_n(x_n)$ where $S = \{P \mid \exists P_1 \in S_1.\exists...\exists P_n \in S_n.P(f(x_1, ..., x_n)) \Leftarrow P_1(x_1) \wedge ... \wedge P_n(x_n) \in \mathcal{C}_{free}\}$.

This construction plays the role of the usual determinization procedure in standard tree automata. Note that it is however more complex because of the ACU symbol. We have the following result:

Lemma 4. *For any $S \subseteq \mathbb{P}$ and any term t, $\breve{S}(t)$ is derivable in $\mathcal{C}^\dagger$ modulo ACU iff $S = \{P \in \mathbb{P} \mid P(t)$ is derivable in $\mathcal{C}$ modulo $ACU\}$.*

Proof. By induction on the size of t. Let $t =_{ACU} t_1 + ... + t_n (n \geq 0)$ where for $1 \leq i \leq n$, $t_i = f_i(t_i^1, ..., t_i^{k_i})$, for some free f_i. Let $S = \{P \in \mathbb{P} \mid P(t)$ is derivable in $\mathcal{C}\}$. First we show that *(a)* $\breve{S}(t)$ is derivable in $\mathcal{C}^\dagger$ modulo ACU; then we show that *(b)* if $\breve{S}'(t)$ is derivable in $\mathcal{C}^\dagger$ modulo ACU for some $S' \subseteq \mathbb{P}$ then $S' = S$.

Proof of (a). For $1 \leq i \leq n, 1 \leq j \leq k_i$ let $S_i^j = \{Q \mid Q(t_i^j)$ is derivable in $\mathcal{C}$ modulo $ACU\}$. By induction hypothesis $\breve{S}_i^j(t_i^j)$ is derivable in $\mathcal{C}^\dagger$ modulo ACU. For $1 \leq i \leq n$ let $S_i = \{Q \mid \exists Q^1 \in S_i^1.\exists...\exists Q^{k_i} \in S_i^{k_i}.Q(f_i(x_1, ..., x_{k_i})) \Leftarrow Q^1(x_i) \wedge ... \wedge Q^{k_i}(x_{k_i}) \in \mathcal{C}\}$. Then the clause $\widehat{S}_i(f_i(x_1, ..., x_{k_i})) \Leftarrow \breve{S}_i^1(x_1) \wedge ... \wedge \breve{S}_i^{k_i}(x_{k_i}) \in \mathcal{C}^\dagger$. So $\widehat{S}_i(t_i)$ is derivable in $\mathcal{C}^\dagger$ modulo ACU for $1 \leq i \leq n$.

For each $P \in S$, the derivation of $P(t)$ has a functional support of the form $Q_1^P(t_1), ..., Q_n^P(t_n)$. For $P \in S, 1 \leq i \leq n$, the derivation of $Q_i^P(t_i)$ uses at the last step, a clause $Q_i^P(f_i(x_1, ..., x_{k_i})) \Leftarrow Q_i^{P,1}(x_1) \wedge ... \wedge Q_i^{P,k_i}(x_{k_i})$ and derivations of $Q_i^{P,j}(t_i^j)(1 \leq j \leq k_i)$. Then $Q_i^{P,j} \in S_i^j$ for $P \in S, 1 \leq i \leq n, 1 \leq j \leq k_i$. Hence $Q_i^P \in S_i$. By Lemma 3, *(*)* $P(a_{S_1} + ... + a_{S_n})$ is derivable in $\mathcal{C}^*_{eq}$ modulo ACU for each $P \in S$.

(**) If $P \in \mathbb{P} \setminus S$ then $a_{S_1} + ... + a_{S_n} \notin \mathcal{L}_P(\mathcal{C}^*_{eq}/ACU)$. Otherwise the corresponding derivation would have a functional support of the form $Q_1(a_{S_1}), ..., Q_n(a_{S_n})$ where $Q_i \in S_i$ for $1 \leq i \leq n$. Then by construction of S_i, $Q_i(t_i)$ would be derivable for $1 \leq i \leq n$. Then by Lemma 3, $P(t_1 + ... + t_n)$ would be derivable in $\mathcal{C}$ modulo ACU. Hence we would have a derivation of $P(t)$ in $\mathcal{C}$ modulo ACU which contradicts the definition of S.

Combining *(*)* and *(**)*, $a_{S_1} + ... + a_{S_n} \in \mathcal{L}^S$. The derivation of $F_S(a_{S_1} + ... + a_{S_n})$ in $\mathcal{C}_S$ modulo ACU has a functional support of the form $R_1(a_{S_1}), ..., R_n(a_{S_n})$. Then $R_i(t_i)$ is derivable in $\mathcal{C}^\dagger$ modulo ACU using the clause $R_i(x) \Leftarrow \widehat{S}_i(x)$ for $1 \leq i \leq n$. Hence by Lemma 3 we get a derivation of $F_S(t_1 + ... + t_n)$ in $\mathcal{C}^\dagger$ modulo ACU. Finally we use the clause $\breve{S}(x) \Leftarrow F_S(x)$ to get a derivation of $\breve{S}(t_1 + ... + t_n)$, i.e. of $\breve{S}(t)$. This proves *(a)*.

Proof of (b). Now suppose $S' \subseteq \mathbb{P}$ such that $\breve{S}'(t)$ is derivable in $\mathcal{C}^\dagger$ modulo ACU. We show that $S' = S$. The derivation of $\breve{S}'(t)$ uses at the last step a derivation of $F_{S'}(t)$. From the construction of $\mathcal{C}^\dagger$, the latter derivation has a functional support of the form $\widehat{S'_1}(t_1), ..., \widehat{S'_n}(t_n)$ and the clauses used immediately above the derivation of $\widehat{S}'_i(t_i)$ are of the form $R'_i(x) \Leftarrow \widehat{S}'_i(x)$ corresponding to the clause $R'_i(a_{S'_i})$ of $\mathcal{C}_{S'}$. Also the clauses used above the clauses $R'_i(x) \Leftarrow \widehat{S}'_i(x)$ are from $\mathcal{C}_{S'}$. Then by Lemma 3, $F_{S'}(a_{S'_1} + ... + a_{S'_n})$ is derivable in $\mathcal{C}_{S'}$ modulo ACU. For $1 \leq i \leq n$ the derivation of $\widehat{S'_i}(t_i)$ uses at the last step a clause of the form $\widehat{S}'_i(f_i(x_1, ..., x_{k_i})) \Leftarrow \breve{S}'^1_i(x_1) \wedge ... \wedge \breve{S}'^{k_i}_i(x_{k_i}) \in \mathcal{C}^\dagger$ and derivations of $\breve{S}'^j_i(t^j_i)(1 \leq j \leq k_i)$. Then by induction hypothesis $S'^j_i = S^j_i$. Then by definition of $\mathcal{C}^\dagger$, $S'_i = S_i$. Hence $a_{S_1} + ... + a_{S_n} \in \mathcal{L}_{S'}$. We have already seen that $a_{S_1} + ... + a_{S_n} \in \mathcal{L}^S$. Since the $\mathcal{L}^S$'s are mutually disjoint, so $S' = S$. $\square$

As a consequence we obtain:

Theorem 1. *One-way ACU automata are closed under complementation.*

Proof. Let $\mathcal{C}$ be a one-way ACU automaton with predicates from $\mathbb{P}$ and with final state F. Define automaton $\mathcal{C}^\dagger$ as above. Pick a new predicate $F^\dagger$. Add to $\mathcal{C}^\dagger$ the clauses $F^\dagger(x) \Leftarrow \breve{S}(x)$ for every $S \subseteq \mathbb{P}, F \notin S$. Call this new automaton $\mathcal{C}^\ddagger$. Then by Lemma 4, $t \in \mathcal{L}_{F^\dagger}(\mathcal{C}^\ddagger/ACU)$ iff for some $S \subseteq \mathbb{P}, F \notin S$ we have $t \in \mathcal{L}_{\breve{S}}(\mathcal{C}^\dagger/ACU)$, iff $F \notin \{P \in \mathbb{P} \mid t \in \mathcal{L}_P(\mathcal{C}/ACU)\}$ iff $t \notin \mathcal{L}_F(\mathcal{C}/ACU)$. Hence $F^\ddagger$ accepts the complement of the language accepted by F. $\square$

Since two-way ACU automata have the same expressiveness [18] as one-way ACU automata, we also have:

Theorem 2. *Two-way ACU automata are closed under complementation.*

The idea of using constants as abstractions for the functional terms accepted at certain states, and the correspondence between constant-only ACU automata and Presburger definability, have already been used in [18] to show closure under intersection of various classes of automata. This may give the impression that the arguments for closure under complementation are similar to the ones for intersection. There are indeed some common ideas, which led the author to conjecture that one-way tree automata modulo all theories considered here are closed under complementation. However it was surprisingly found that while these ideas work for showing closure under intersection of the one-way automata modulo all theories under consideration, they don't work for showing closure under complementation of the one-way automata modulo the theories $ACUX$, $ACUX_n$, $ACUM$ and $ACUI$. We will show that the latter are not closed under complementation. Unlike in the case of intersection, our results show a strong correlation between closure under complementation of one-way equational tree automata and the linearity of the equational theory involved, at least as far as the theories dealt with in this paper are concerned.

4 Counter-Example for $ACUX$ and $ACUX_n$ Automata

Contrary to the ACU case, $ACUX$ automata are not closed under complementation, as we show in this section. The result generalizes to the $ACUX_n$ case for $n \geq 2$, i.e. we can show that one-way or two-way $ACUX_n$ automata are not closed under complementation for any $n \geq 2$. (We get the $ACUX$ case by making $n = 2$.) To show this fix some $n \geq 2$. Define languages $\mathcal{L}_1 = \{f^{k_1}(a) + ... + f^{k_n}(a) \mid k_1, ..., k_n \geq 0\}$ (assuming a signature that has at least a unary symbol f and a constant a) and $\mathcal{L}_2 = \{0\}$. Let us clarify that we are considering languages modulo $ACUX_n$, so that, for example, when we write $\{0\}$, we actually mean the language $\{t \mid t =_{ACUX_n} 0\}$. Similar remarks hold for other theories considered later. Then $\mathcal{L}_1 \setminus \mathcal{L}_2 = \{f^{k_1}(a) + ... + f^{k_n}(a) \mid k_1, ..., k_n \geq 0$ and for some $1 \leq i, j \leq n, i \neq j$ and $k_i \neq k_j\}$. $\mathcal{L}_1$ and $\mathcal{L}_2$ are clearly accepted by one-way $ACUX_n$ automata. However $\mathcal{L}_1 \setminus \mathcal{L}_2$ is not:

Lemma 5. *The language $\mathcal{L} = \{f^{k_1}(a)+...+f^{k_n}(a) \mid k_1, ..., k_n \geq 0$ and for some $1 \leq i, j \leq n, i \neq j$ and $k_i \neq k_j\}$ is not accepted by any one-way $ACUX_n$ automaton.*

Proof. Assume on the contrary that there is a one-way automaton $\mathcal{C}$ with final state P which modulo $ACUX_n$ accepts $\mathcal{L}$. Let $\mathbb{P}$ be the set of predicates in $\mathcal{C}$. For each $p \geq 0$ define $S_p = \{Q \in \mathbb{P} \mid Q(f^p(a))$ is derivable in $\mathcal{C}$ modulo $ACUX_n\}$. Since $\mathbb{P}$ is finite, it has finitely many subsets. Hence we have some $p \neq q$ such that $S_p = S_q$. Since $p \neq q$, the term $f^p(a) + \sum_{i=1}^{n-1} f^q(a) \in \mathcal{L}$. From Lemma 1, we have some $t =_{ACUX_n} f^p(a) + \sum_{i=1}^{n-1} f^q(a)$ such that $P(t)$ is derivable in $\mathcal{C}$ modulo ACU. We have $t =_{ACU} t_1 + t_2 + ... + t_n + u_1^1 + ... + u_1^n + ... + u_k^1 + ... + u_k^n (k \geq 0)$ such that t_j and u_i^j are functional for $1 \leq i \leq k, 1 \leq j \leq n$, $t_1 =_{ACUX_n} f^p(a), t_j =_{ACUX_n} f^q(a)$ for $2 \leq j \leq n$, and $u_i^j =_{ACUX_n} u_i^{j'}$ for $1 \leq i \leq k, 1 \leq j, j' \leq n$. The derivation of $P(t)$ has functional support of the form $I_1(t_1), I_2(t_2), ..., I_n(t_n), I_1^1(u_1^1), ..., I_1^n(u_1^n)$, $..., I_k^1(u_k^1), ..., I_k^n(u_k^n)$. Then $I_1 \in S_p$ and $I_2, ..., I_n \in S_q$. However $S_p = S_q$, hence $I_2, ..., I_n \in S_p$. Thus $I_1(f^p(a)), I_2(f^p(a)), ..., I_n(f^p(a))$ are derivable in $\mathcal{C}$ modulo $ACUX_n$. By Lemma 3, $P(\sum_{i=1}^{n} f^p(a) + u_1^1 + ... + u_1^n + ... + u_k^1 + ... + u_k^n)$ (or $P(0)$) is derivable in $\mathcal{C}$ modulo $ACUX_n$ leading to contradiction. $\square$

Since $\mathcal{L}_1 \setminus \mathcal{L}_2 = \mathcal{L}_1 \cap \complement(\mathcal{L}_2)$ where $\complement(\mathcal{L}')$ denotes the complement of $\mathcal{L}'$, we have:

Theorem 3. *One-way $ACUX_n$ automata are not closed under complementation for any $n \geq 2$. This holds for one-way $ACUX$ automata in particular.*

The key idea, also used in the further counter-examples, is that in presence of non-linear equations like X, doing complementation involves putting disequality constraints on subterms, which is beyond the power of our automata.

Again, since two-way $ACUX_n$ automata have the same expressiveness [18] as one-way $ACUX_n$ automata, we get:

Theorem 4. *Two-way $ACUX_n$ automata are not closed under complementation for any $n \geq 2$. This holds for one-way $ACUX$ automata in particular.*

We remark that the situation is different for the theory $ACUX_n$ when $n = 1$. In this case the axiom $x = 0$ means that every term is equal to 0. Trivially the automata are closed under intersection and complementation. This is clearly not an interesting case.

5 Constant-Only $ACUD$ Automata and Reuse of $ACUD$ Derivations

In this section we generalize Lemmas 2 and 3 to the $ACUD$ case. These will be useful in dealing with the $ACUD$ and $ACUM$ automata.

First we consider constant only $ACUD$ automata. Recall that Σ_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:

Lemma 6 ([18]). *The language accepted by a constant-only $ACUD$ automata with constants from Σ_f is a semilinear set with constants from $\Sigma_f \cup \overline{\Sigma_f}$. Conversely, a semilinear set with constants from $\Sigma_f \cup \overline{\Sigma_f}$ can be represented as accepted by a constant-only $ACUD$ automaton with constants from Σ_f.*

Also we have the following generalization of Lemma 3 to the $ACUD$ case.

Lemma 7. *Let $\mathcal{E}$ be any set of equations implying $ACUD$. Consider a derivation δ of an atom $P(t)$ modulo $\mathcal{E}$. Let $\delta_1, ..., \delta_n$ be non-overlapping subderivations of δ such that outside the δ_i's, the only equations used are $ACUD$ and the set S of clauses used contains only clauses of kind (3), (4), (5) and (7). Suppose the conclusions of $\delta_1, ..., \delta_n$ are $P_1(t_1), ..., P_n(t_n)$. Then*

1. *$t =_{ACUD} \pm_1 t_1 \pm_2 ... \pm_n t_n$ for some $\pm_1, ..., \pm_n \in \{+, -\}$. ($+t_1$ means t_1.)*
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 δ' of $P(\pm_1 s_1 \pm_2 ... \pm_n s_n)$ (the $\pm_i$'s here are the same as above) modulo $\mathcal{E}$, containing δ'_i's as subderivations, such that outside the δ'_i's, the only equations used are $ACUD$, and all clauses used belong to S.*

Finally the notion of functional support is generalized to $ACUD$ derivations.

Definition 2. *Consider a derivation δ of an atom $P(t)$ in a one-way automaton modulo $ACUD$. Let $\delta_1, ..., \delta_n$ be the set of maximal subderivations of δ in which the last step used is an application of clause (8) (or clause (6)). 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 δ. (From Lemma 7, t must be of the form $\pm_1 t_1 \pm_2 ... \pm_n t_n$.)*

6 Complementation of $ACUD$ Automata

In the $ACUD$ case the situation is similar to the ACU case: the automata are closed under complementation. The arguments being similar to the ACU case, we give the full construction with a sketch of the proof.

Let $\mathcal{C}$ be a one-way $ACUD$ automaton with predicates from $\mathbb{P}$. Introduce predicate symbols $\breve{S}$ and $\widehat{S}$ for each $S \subseteq \mathbb{P}$, and sets of constants $A = \{a_S \mid S \subseteq \mathbb{P}\}$ and $\overline{A} = \{\overline{a_S} \mid S \subseteq \mathbb{P}\}$. Define automaton $\mathcal{C}_{eq}^* = \mathcal{C}_{eq} \cup \{P(a_S) \mid P \in S\}$. From Lemma 6, for each P, $\mathcal{L}_P(\mathcal{C}_{eq}^*/ACUD)$ is a semilinear set on constants from $A \cup \overline{A}$. Given $S \subseteq \mathbb{P}$, define $\mathcal{L}^S = \bigcap_{P \in S} \mathcal{L}_P(\mathcal{C}_{eq}^*/ACUD) \setminus \bigcup_{P \in \mathbb{P} \setminus S} \mathcal{L}_P(\mathcal{C}_{eq}^*/ACUD)$. This is a semilinear set on constants from $A \cup \overline{A}$. By Lemma 6 we can construct a constant-only $ACUD$ automaton $\mathcal{C}_S$, using constants from A, and with final state F_S such that $\mathcal{L}(\mathcal{C}_S/ACUD) = \mathcal{L}^S$. We assume that automata $\mathcal{C}_S$'s are all built from mutually disjoint sets of (fresh) predicates.

Define automaton $\mathcal{C}^\dagger$ to consist of the following clauses:

- for each $S \subseteq \mathbb{P}$ the clause $\breve{S}(x) \Leftarrow F_S(x)$.
- clauses (3), (4), (5) and (7) (but not (6)) from each $\mathcal{C}_S$.
- for each clause $R(a_S)$ in some $\mathcal{C}_S$, the clause $R(x) \Leftarrow \widehat{S}(x)$.
- for each free functional symbol f of arity n, and each $S_1, ..., S_n \subseteq \mathbb{P}$, the clause $\widehat{S}(f(x_1, ..., x_n)) \Leftarrow \breve{S}_1(x_1) \wedge ... \wedge \breve{S}_n(x_n)$ where $S = \{P \mid \exists P_1 \in S_1.\exists ... \exists P_n \in S_n.P(f(x_1, ..., x_n)) \Leftarrow P_1(x_1) \wedge ... \wedge P_n(x_n) \in \mathcal{C}_{free}\}$.

Lemma 8. *For any $S \subseteq \mathbb{P}$ and any term t, $\breve{S}(t)$ is derivable in $\mathcal{C}^\dagger$ modulo $ACUD$ iff $S = \{P \in \mathbb{P} \mid P(t)$ is derivable in $\mathcal{C}$ modulo $ACUD\}$.*

Proof. By induction on the size of t. Let $t =_{ACUD} s_1 + ... + s_m - t_1 - ... - t_n (m, n \geq 0)$ where for $1 \leq i \leq m$ $s_i = f_i(s_i^1, ..., s_i^{k_i})$ for some free f_i, and for $1 \leq i \leq n$ $t_i = g_i(t_i^1, ..., t_i^{l_i})$ for some free g_i. Let $S = \{P \in \mathbb{P} \mid P(t)$ is derivable in $\mathcal{C}\}$. First we show that *(a)* $\breve{S}(t)$ is derivable in $\mathcal{C}^\dagger$ modulo $ACUD$; then we show that *(b)* if $\breve{S}'(t)$ is derivable in $\mathcal{C}^\dagger$ modulo $ACUD$ for $S' \subseteq \mathbb{P}$ then $S' = S$.

Proof of (a). For $1 \leq i \leq m, 1 \leq j \leq k_i$ let $S_i^j = \{Q \mid Q(s_i^j)$ is derivable in $\mathcal{C}$ modulo $ACUD\}$. By induction hypothesis $\breve{S}_i^j(s_i^j)$ is derivable in $\mathcal{C}^\dagger$ modulo $ACUD$. For $1 \leq i \leq n, 1 \leq j \leq l_i$ let $T_i^j = \{Q \mid Q(t_i^j)$ is derivable in $\mathcal{C}$ modulo $ACUD\}$. Then by induction hypothesis $\breve{T}_i^j(t_i^j)$ is derivable in $\mathcal{C}^\dagger$ modulo $ACUD$. For $1 \leq i \leq m$ let $S_i = \{Q \mid \exists Q^1 \in S_i^1.\exists ... \exists Q^{k_i} \in S_i^{k_i}.Q(f_i(x_1, ..., x_{k_i})) \Leftarrow Q^1(x_i) \wedge ... \wedge Q^{k_i}(x_{k_i}) \in \mathcal{C}\}$. Then the clause $\widehat{S}_i(f_i(x_1, ..., x_{k_i})) \Leftarrow S_i^1(x_1) \wedge ... \wedge S_i^{k_i}(x_{k_i}) \in \mathcal{C}^\dagger$. So $\widehat{S}_i(s_i)$ is derivable in $\mathcal{C}^\dagger$ for $1 \leq i \leq n$. Similarly for $1 \leq i \leq m$ let $T_i = \{Q \mid \exists Q^1 \in T_i^1.\exists ... \exists Q^{l_i} \in T_i^{l_i}.Q(g_i(x_1, ..., x_{l_i})) \Leftarrow Q^1(x_i) \wedge ... \wedge Q^{l_i}(x_{l_i}) \in \mathcal{C}\}$. Then the clause $\widehat{T}_i(g_i(x_1, ..., x_{l_i})) \Leftarrow T_i^1(x_1) \wedge ... \wedge T_i^{l_i}(x_{l_i}) \in \mathcal{C}^\dagger$. So $\widehat{T}_i(t_i)$ is derivable in $\mathcal{C}^\dagger$ for $1 \leq i \leq n$.

Arguing as in the ACU case, but using Lemma 7 instead of Lemma 3 we show that $P(a_{S_1} + ... + a_{S_m} - a_{T_1} - ... - a_{T_n})$ is derivable in $\mathcal{C}_{eq}^*$ modulo $ACUD$ for each $P \in S$. Also we show that if $P \in \mathbb{P} \setminus S$ then $a_{S_1} + ... + a_{S_m} - a_{T_1} - ... - a_{T_n} \notin \mathcal{L}_P(\mathcal{C}_{eq}^*/ACUD)$.

Hence $a_{S_1} + ... + a_{S_m} - a_{T_1} - ... - a_{T_n} \in \mathcal{L}^S$. The derivation of $F_S(a_{S_1} + ... + a_{S_m} - a_{T_1} - ... - a_{T_n})$ in $\mathcal{C}$ modulo $ACUD$ has a functional support of the form $R_1(a_{S_1}), ..., R_m(a_{S_m}), R_1'(a_{T_1}), ..., R_n'(a_{T_n})$. As in the ACU case we use Lemma 7 to get a derivation of $F_S(s_1 + ... + s_m - t_1 - ... - t_n)$ in $\mathcal{C}^\dagger$ modulo $ACUD$. Finally we use the clause $\breve{S}(x) \Leftarrow F_S(x)$ to get a derivation of $\breve{S}(s_1 + ... + s_m - t_1 - ... - t_n)$, i.e. of $\breve{S}(t)$. This proves *(a)*.

Proof of (b). Now suppose $S' \subseteq \mathbb{P}$ such that $\breve{S}'(t)$ is derivable in $\mathcal{C}^\dagger$ modulo $ACUD$. We show that $S' = S$. The derivation of $\breve{S}'(t)$ uses at the last step a derivation of $F_{S'}(t)$. The latter derivation has a functional support of the form $\widehat{S'_1}(s_1), ..., \widehat{S'_m}(s_m), \widehat{T'_1}(t_1), ..., \widehat{T'_n}(t_n)$. Arguing as in the ACU case, we use Lemma 7 to show that $F_{S'}(a_{S'_1} + ... + a_{S'_m} - a_{T'_1} - ... - a_{T'_n})$ is derivable in $\mathcal{C}_{S'}$ modulo $ACUD$. For $1 \leq i \leq m$ the derivation of $\widehat{S'_i}(s_i)$ uses at the last step, a clause $\widehat{S'_i}(f_i(x_1, ..., x_{k_i})) \Leftarrow S'^1_i(x_1) \wedge ... \wedge S'^{k_i}_i(x_{k_i}) \in \mathcal{C}^\dagger$ and derivations of $S'^j_i(s^j_i)(1 \leq j \leq k_i)$. By induction hypothesis $S'^j_i = S^j_i$. By definition of $\mathcal{C}^\dagger$, $S'_i = S_i$. Similarly for $1 \leq i \leq n$, $T'_i = T_i$. Hence $a_{S_1} + ... + a_{S_m} - a_{T_1} - ... - a_{T_n} \in \mathcal{L}_{S'}$. We have already seen that $a_{S_1} + ... + a_{S_m} - a_{T_1} - ... - a_{T_n} \in \mathcal{L}^S$. Since the $\mathcal{L}^S$'s are mutually disjoint, we have $S' = S$. $\square$

Then as in the ACU case, and using the fact that two-way $ACUD$ automata have the same expressiveness [18] as one-way $ACUD$ automata, we conclude:

Theorem 5. *One-way $ACUD$ automata are closed under complementation. Also two-way $ACUD$ automata are closed under complementation.*

7 Counter-Example for $ACUM$ Automata

The $ACUM$ (Abelian groups theory) case is similar to the $ACUX$ case. Define languages $\mathcal{L}_1 = \{f^n(a) - f^m(a) \mid n, m \geq 0\}$, $\mathcal{L}_2 = \{0\}$. Then $\mathcal{L}_1 \setminus \mathcal{L}_2 = \{f^n(a) - f^m(a) \mid n, m \geq 0$ and $n \neq m\}$ is not accepted by any one-way $ACUM$ automaton:

Lemma 9. *The language $L = \{f^n(a) - f^m(a) \mid n, m \geq 0$ and $n \neq m\}$ is not accepted by any one-way $ACUM$ automaton.*

Proof. The proof is similar to that of Lemma 5, the difference being that we need the $ACUD$ case (Lemma 7) instead of the ACU case (Lemma 3). Assume on the contrary that there is a one-way automaton $\mathcal{C}$ with final state P which modulo $ACUM$ accepts $\mathcal{L}$. Let $\mathbb{P}$ be the set of predicates in $\mathbb{C}$. For each $n \geq 0$ define $S_n = \{Q \in \mathbb{P} \mid Q(f^n(a))$ is derivable in $\mathcal{C}$ modulo $ACUM\}$. Then we have some $n \neq m$ such that $S_n = S_m$. Then $f^n(a) - f^m(a) \in \mathcal{L}$. We have some $t =_{ACUM} f^n(a) - f^m(a)$ such that $P(t)$ is derivable in $\mathcal{C}$ modulo $ACUD$. $t =_{ACUD} t_1 - t_2 + u_1 - v_1 + ... + u_k - v_k (k \geq 0)$ such that t_1 and t_2 are functional, $t_1 =_{ACUM} f^n(a)$, $t_2 =_{ACUM} f^m(a)$, u_i, v_i are functional and $u_i =_{ACUM} v_i$ for $1 \leq i \leq k$. The derivation of $P(t)$ has functional support of the form $I(t_1), J(t_2), I_1(u_1), J_1(v_1), ..., I_k(u_k), J_k(v_k)$. As before, $I(f^n(a))$ and $J(f^n(a))$ are derivable in $\mathcal{C}$ modulo $ACUM$. Then by Lemma 7, $P(f^n(a) - f^n(a) + u_1 - v_1 + ... + u_k - v_k)$ (or $P(0)$) is derivable in $\mathcal{C}$ modulo $ACUM$ leading to contradiction. $\square$

Since one-way $ACUM$ automata are closed [18] under intersection and are as expressive [18] as two-way $ACUM$ automata, we have as for the $ACUX$ case:

Theorem 6. *One-way $ACUM$ automata are not closed under complementation. Neither are two-way $ACUM$ automata closed under complementation.*

8 Counter-Example for $ACUI$ Automata

We now show that one-way $ACUI$ automata are not closed under complementation either. Define languages $\mathcal{L}_1 = \{f^n(a) + f^m(a) \mid n, m \geq 0\}$ and $\mathcal{L}_2 = \{f^n(a) \mid n \geq 0\}$. We have $\mathcal{L}_1 \setminus \mathcal{L}_2 = \{f^n(a) + f^m(a) \mid n, m \geq 0$ and $n \neq m\}$.

Lemma 10. *The language $\mathcal{L} = \{f^n(a) + f^m(a) \mid n, m \geq 0$ and $n \neq m\}$ is not accepted by any one-way $ACUI$ automaton.*

Proof. Assume on the contrary that there is a one-way automaton $\mathcal{C}$ with final state P which accepts $\mathcal{L}$. Let $\mathbb{P}$ be the set of predicates in $\mathcal{C}$. For each $n \geq 0$ define $S_n = \{Q \in \mathbb{P} \mid Q(f^n(a))$ is derivable in $\mathcal{C}$ modulo $ACUI\}$. We must have some $n \neq m$ such that $S_n = S_m$. Then $f^n(a) + f^m(a) \in \mathcal{L}$. From Lemma 1, we must have some $t =_{ACUI} f^n(a) + f^m(a)$ such that $P(t)$ is derivable in $\mathcal{C}$ modulo ACU. t must be of the form $s_1 + ... + s_p + t_1 + ... + t_q (p, q \geq 1)$ such that for $1 \leq i \leq p, 1 \leq j \leq q$, s_i and t_j are functional, $s_i =_{ACUI} f^n(a)$ and $t_j =_{ACUI} f^m(a)$. Then arguing as in the $ACUX_n$ case, we get a derivation of $P(\sum_{i=1}^{p+q} f^n(a))$ (or $P(f^n(a))$) modulo $ACUI$ leading to contradiction. □

Since one-way $ACUI$ automata are closed [18] under intersection we have:

Theorem 7. *One-way $ACUI$ automata are not closed under complementation.*

However this does not answer whether two-way $ACUI$ automata are closed under complementation since the question of equivalence between two-way and one-way $ACUI$ automata is currently open [18]. Closure under intersection of two-way $ACUI$ automata is also left open in [18], but it is mentioned that the two-way $ACUI$ automata are powerful enough to encode alternation. This suggests that this case is difficult, because adding alternation to one-way automata is already known to produce undecidability in the case of theories $ACU, ACUD, ACUM$, while in the case of theories $ACUX, ACUX, ACUI$ the question is open.

9 Conclusion

We have studied the problem of closure under complementation of one-way and two-way tree automata modulo the equational theories ACU (associativity, commutativity, unit), $ACUD$ (ACU with a distributive '$-$' symbol), $ACUM$ (Abelian groups), $ACUX$ (exclusive-or), $ACUX_n$ (generalized exclusive-or), and $ACUI$ (ACU with idempotence.)

We have shown that for the theories ACU and $ACUD$, the one-way and two-way automata are closed under complementation. For the theories $ACUM$, $ACUX$ and $ACUX_n$, we have shown that neither one-way nor two-way automata are closed under complementation. In the case of $ACUI$, while the one-way automata are not closed under complementation, the question in open for two-way automata. An interesting pattern visible here is the coincidence between closure under complementation of the one-way automata and the linearity of the equational theory.

These results are in sharp contrast with the results on closure under intersection. The one-way automata for all the above theories are known to be closed under intersection. Also, the two-way automata for all theories except $ACUI$ are known to be closed under intersection. (The question is open for two-way $ACUI$ automata.)

The results about the two-way automata are obtained by reducing them to one-way automata. The equivalence between one-way and two-way $ACUI$ automata is however open. A positive answer would also mean that two-way $ACUI$ automata are closed under intersection but not under complementation.

Acknowledgements: I thank Jean Goubault-Larrecq for discussions and suggestions, as well as the anonymous referees for helpful comments.

References

1. 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.
2. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. www.grappa.univ-lille3.fr/tata, 1997.
3. W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.
4. F. Gécseg and M. Steinby. Tree languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 1–68. Springer Verlag, 1997.
5. S. Ginsburg and E. H. Spanier. Semigroups, Presburger formulas and languages. *Pacific Journal of Mathematics*, 16(2):285–296, 1966.
6. J. Goubault-Larrecq. A method for automatic cryptographic protocol verification. In *FMPPTA'2000, 15th IPDPS Workshops*, pages 977–984. Springer-Verlag LNCS 1800, 2000.
7. J. Goubault-Larrecq and K. N. Verma. Alternating two-way AC-tree automata. In preparation.
8. M. Hanus. The integration of functions into logic programming: From theory to practice. *Journal of Logic Programming*, 19 & 20:583–628, 1994.
9. D. Lugiez. A good class of tree automata. Application to inductive theorem proving. In *ICALP'98*, pages 409–420. Springer-Verlag LNCS 1443, 1998.
10. D. Lugiez. Counting and equality constraints for multitree automata. In *FOSSACS'03*, pages 328–342. Springer-Verlag LNCS 2620, 2003.
11. D. Monniaux. Abstracting cryptographic protocols with tree automata. In *SAS'99*, pages 149–163. Springer-Verlag LNCS 1694, 1999.
12. H. Ohsaki. Beyond regularity: Equational tree automata for associative and commutative theories. In *CSL'01*, pages 539–553. Springer-Verlag LNCS 2142, 2001.
13. H. Ohsaki and T. Takai. Decidability and closure properties of equational tree languages. In *RTA'02*, pages 114–128. Springer-Verlag LNCS 2378, 2002.
14. L. C. Paulson. Mechanized proofs for a recursive authentication protocol. In *10th IEEE Computer Security Foundations Workshop*, pages 84–95, 1997.
15. P. Ryan and S. Schneider. An attack on a recursive authentication protocol: A cautionary tale. *Information Processing Letters*, 65(1):7–10, 1998.
16. M. Steiner, G. Tsudik, and M. Waidner. Key agreement in dynamic peer groups. *IEEE Transactions on Parallel and Distributed Systems*, 11(8):769–780, 2000.
17. M. Y. Vardi. Reasoning about the past with two-way automata. In *ICALP'98*, pages 628–641. Springer Verlag LNCS 1443, 1998.
18. 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.