Two-Way Equational Tree Automata for AC-like Theories: Decidability and Closure Properties ⋆

Kumar Neeraj Verma

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

Abstract. We study two-way tree automata modulo equational theories. We deal with the theories of Abelian groups ($ACUM$), idempotent commutative monoids ($ACUI$), and the theory of exclusive-or ($ACUX$), as well as some variants including the theory of commutative monoids (ACU). We show that the one-way automata for all these theories are closed under union and intersection, and emptiness is decidable. For two-way automata the situation is more complex. In all these theories except $ACUI$, we show that two-way automata can be effectively reduced to one-way automata, provided some care is taken in the definition of the so-called push clauses. (The $ACUI$ case is open.) In particular, the two-way automata modulo these theories are closed under union and intersection, and emptiness is decidable. We also note that alternating variants have undecidable emptiness problem for most theories, contrarily to the non-equational case where alternation is essentially harmless.

1 Introduction

Tree automata [4, 2] enjoy many good properties: emptiness is decidable, the class of recognizable languages is closed under Boolean operations notably. This extends to so-called *two-way* tree automata, where transitions may not only construct terms as in ordinary tree automata (call them *one-way* to distinguish them from two-way automata), but also destruct terms (see [2], Chapter 7, *Alternating Tree Automata*). The presence of these latter transitions sometimes make two-way automata more convenient to work with than one-way automata, although they are equally expressive.

A recent and important extension of the tree automata concept is that of *equational* tree automata [8, 12] which recognize terms modulo an equational theory. Until now, all results on equational tree automata have been concerned with one-way automata only, see table below. The purpose of this paper is to fill this gap.

	one-way	two-way
non-equational	emptiness decidable, closed under $\cup$, $\cap$, $\complement$ [4, 2]	emptiness decidable, closed under $\cup$, $\cap$, $\complement$, reduce to one-way automata [2, 3]
equational	[8, 9, 12, 13]. (see related work section.)	This work.

⋆ 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".

More specifically, we study the notion of two-way equational tree automata, modulo several theories extending the theory ACU of one associative, commutative operation $+$ with unit 0, which is defined by the axioms (A) $x + (y + z) = (x + y) + z$, (C) $x+y = y+x$ and (U) $x+0 = x$. These theories will be obtained by adding axioms to the base theory ACU, taken from the following: idempotence (I) $x+x = x$, the xor axiom (X) $x+x = 0$, more generally the cancellation axiom (X_n) $nx = 0$ (where nx denotes $x + x + \ldots + x$); the minus axiom (M) $x + (-x) = 0$, where $-$ is an additional unary symbol; and the minus distributivity axioms $(D) -(x+y) = (-x)+(-y)$, $-(-x) = x$, $-0 = 0$. We name a theory by the names of its axioms; e.g., $ACUM$ is the theory of Abelian groups. Note that D is implied by $ACUM$, so $ACUD$ is a (strictly) weaker theory than $ACUM$. Except for the axiom $-(-x) = x$, the theory $ACUD$ resembles the theory $ACUh$ (ACU with homomorphism) considered in some papers.

We first show that modulo ACU, $ACUI$, $ACUD$, $ACUM$ (Abelian groups), $ACUX$ (the theory of exclusive-or), and $ACUX_n$, the one-way automata are closed under intersection, union, and emptiness is decidable. In particular membership is decidable since trivially, the equational closure of a singleton set is accepted by our one-way automata. The hard part is in showing closure under intersection; this is in particular much more involved than in the non-equational case, although the technique we use can be thought of as a kind of souped-up product construction.

As far as *two-way* equational tree automata are concerned, we show that modulo all theories $\mathcal{E}$ above except $ACUI$, they are exactly as expressive as the corresponding one-way automata, and we describe effective reduction processes from two-way to one-way automata modulo $\mathcal{E}$. (The $ACUI$ case is currently still open.) For these reductions to work, and in fact for any reduction from two-way to one-way automata to work, special care has to be taken in the definition of the so-called *push clauses*, which are the kinds of transitions that are added in the two-way case compared to the one-way case. Indeed, we show that, had we been more sloppy, then emptiness of two-way automata modulo ACU, $ACUD$, $ACUM$ would have been undecidable. We also show that the emptiness question for the *alternating* variants of one-way automata modulo the latter theories are undecidable, too. This is in sharp contrast with the non-equational case, where emptiness is decidable for alternating, two-way tree automata.

Two-way tree automata have recently been used for verification of cryptographic protocols [6, 11]. While they work in the case of perfect cryptographic primitives, more complex primitives with additional algebraic properties need to be modeled using equational theories. Two theories that occur often are the theories of exclusive-or and Abelian groups. This is not the topic of this paper and won't be pursued here.

Plan. After some material on related work, we define our one-way and two-way equational tree automata in Section 2, discussing basic properties and some undecidability results. To deal with decidable cases, we start with some easy results in Section 3, which shall form the basis for the rest of the paper; in particular, we show that the so-called *constant-only* one-way ACU automata recognize exactly the semilinear sets. These results are used in Section 4 to compute intersections of one-way automata modulo $ACUX$. From there, we deduce a procedure to convert two-way $ACUX$ automata to one-way $ACUX$ automata in Section 5; we also discuss generalizations to the theory $ACUX_n$ for any $n \geq 2$. The results and proofs in the Abelian groups case ($ACUM$)

are exactly as in the $ACUX$ case, as we argue in Section 7. This however requires us to first deal with the constant-only $ACUD$ case in Section 6. The general ACU and $ACUD$ cases are simplified versions of the $ACUX$ and Abelian groups cases respectively and are dealt with in Section 8. Lastly we show that intersections of one-way $ACUI$ automata are also computable in Section 9, and discuss why translating two-way $ACUI$ to one-way $ACUI$ automata is troublesome. We conclude in Section 10.

Related work. The multitree automata of Lugiez [9], which extend his earlier work [8], correspond to our one-way ACU automata but with a much richer set of constraints including equality constraints, still keeping decidability, and are closed under boolean operations. This is incomparable to ours: our two-way automata cannot avoid undecidability in the presence of equality constraints. 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 all theories. For example, in Ohsaki's case, with transition rules $a \to q, b \to q, q{+}q \to q'$ and $0 \to q'$, and with the theory $ACUX$, we have $a{+}b \to^* q'$. In our case, with the corresponding clauses $q(a)$, $q(b)$, $q'(x + y) \Leftarrow q(x) \wedge q(y)$ and $q'(0)$, and with the theory $ACUX$, $a + b$ is not accepted at q'. For arbitrary $\mathcal{E}$ we do not know the relationship between our automata and Ohsaki's automata, and the two notions appear rather dissimilar. The multiset automata of Colcombet [1] correspond to the subclass of our one-way ACU automata in which all symbols other than $+, 0$ are unary. Note that the specific theories $ACUX$, $ACUM$, etc., that we consider here have traditionally not been considered in the framework of (one-way equational) tree automata; they give rise to specific technical problems and solutions. The notion of alternating two-way AC tree automata will be treated in detail in [7], which also considers some additional push clause formats which are not relevant in this paper.

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. 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 $\mathcal{C}$ will be $\mathcal{L}(\mathcal{C}/\mathcal{E}) = \mathcal{L}_{P_f}(\mathcal{C}/\mathcal{E})$. Given a language $\mathcal{L}$ and an equational theory $\mathcal{E}$, $\mathcal{E}(\mathcal{L})$ denotes the set of terms t such that $t =_{\mathcal{E}} s$ for some $s \in \mathcal{L}$. A state or an automaton is called *empty* if it does not accept any term.

We will be especially interested in the following kind of clauses which we shall call *pop clauses*, *epsilon clauses* and *general push clauses* respectively.

$$P(f(x_1, ..., x_n)) \Leftarrow P_1(x_1) \wedge ... \wedge P_n(x_n) \quad (2)$$
$$P(x) \Leftarrow P_1(x) \quad (3)$$
$$P(x_i) \Leftarrow Q(f(x_1, ..., x_n)) \wedge P_1(x_{i_1}) \wedge ... \wedge P_k(x_{i_k}), \quad 1 \leq i, i_1, ..., i_k \leq n \quad (4)$$

In both clauses (2) and (4), the variables $x_1, ..., x_n$ are distinct. We define *one-way automata* as consisting of clauses of kind (2) and (3), whereas *general two-way automata* in addition contain clauses of kind (4). We shall see that the general push clauses are problematic, hence we shall consider restricted forms called push clauses below. Our one-way automata (without equations) are exactly the classical tree automata usually described in the literature. Clauses 2 and 3 correspond to transition rules $f(P_1, ..., P_n) \to P$ and $P_1 \to P$ of classical tree automata. Note that the 'two-way automata' of [15] are a different notion from ours. The following result is an easy consequence of the above definitions, and is shown by induction on the derivations.

Lemma 1. *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.*

The result does not hold for two-way automata in general.

We will sometimes need *extended epsilon clauses*:

$$P(x) \Leftarrow Q(x) \wedge P_1(x_1) \wedge ... \wedge P_n(x_n) \quad (5)$$

where the variables $x, x_1, ..., x_n$ are distinct. Intuitively, this is an epsilon clause $P(x) \Leftarrow Q(x)$ together with emptiness tests on the states $P_1, ..., P_n$. We have the easy:

Lemma 2. *For any one-way automaton $\mathcal{C}$ which in addition contains clauses (5), we can compute a one-way automaton $\mathcal{C}'$ such that for each P, $\mathcal{L}_P(\mathcal{C}/\mathcal{E}) = \mathcal{L}_P(\mathcal{C}'/\mathcal{E})$.*

Proof. Lemma 1 trivially extends to one-way automata with extra clauses (5), hence emptiness is decidable. Then remove all clauses (5) where some P_i is empty, $1 \leq i \leq n$, and remove all $P_i(x_i)$, $1 \leq i \leq n$, from the remaining clauses (5). $\square$

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

$$P(x+y) \Leftarrow P_1(x) \wedge P_2(y) \quad (6) \qquad P(a) \text{ where } a \text{ is a constant} \quad (8)$$
$$P(0) \quad (7) \qquad P(-x) \Leftarrow P_1(x) \quad (9)$$
$$P(f(x_1, ..., x_n)) \Leftarrow P_1(x_1) \wedge ... \wedge P_n(x_n), f \text{ being free} \quad (10)$$

$$Q(x_i) \Leftarrow P(f(x_1,...,x_n)) \wedge Q_1(x_{i_1}) \wedge ... \wedge Q_k(x_{i_k}), \quad (11)$$
$$f \text{ being free}, i \in \{1,...,n\} \setminus \{i_1,...,i_k\}, 1 \le i_1 < ... < i_k \le n$$

While clauses (6)–(10) are pop clauses, we call (11) *push clauses*. The side conditions in (11) will be discussed at the end of this section.

Clauses (8) are special cases of clauses (10). Also the general push clause $P(x) \Leftarrow P_1(-x)$ is equivalent to (9) in the theories $ACUM$ and $ACUD$. One-way ACU (resp. $ACUI$, $ACUX$, $ACUX_n$) automata are sets of clauses (3), (6–8) and (10); one-way $ACUM$ and $ACUD$ automata in addition contain clauses (9). We define two-way automata by adding clauses (11) to one-way automata: hence *two-way* automata are sets of clauses (3) and (6–11)—with the proviso that (9) is only included when $- \in \Sigma$. *Constant-only* automata are one-way automata which contain clauses (8) instead of the general clauses (10) (the only free symbols in the signature are constants.) Given a two-way automaton $\mathcal{C}$ we define $\mathcal{C}_{one-way}$ to be the part of $\mathcal{C}$ without clauses (11). Also, 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), (6), (7) and (9) (the equational part), and $\mathcal{C}_{free}$ is the remaining part.

The languages accepted by all our one-way or two-way automata are trivially closed under unions. As we have already dealt with emptiness of one-way automata (Lemma 1), we shall concentrate on intersection and reduction of two-way to one-way automata.

Negative Results. It is instructive to first consider constant-only ACU automata extended with *alternation clauses* of the form $P(x) \Leftarrow P_1(x) \wedge P_2(x)$. Then it is easy to encode reachable configurations of 2-counter automata $\mathcal{M}$ [10] using ACU automata (see [7].) It is not our purpose to replay the arguments of [7]. Instead here is the idea. We encode configurations $q(m,n)$ of $\mathcal{M}$ by atoms $q((x+m)a_1 + xa_2 + (y+n)b_1 + yb_2)$ for $x, y \ge 0$. To increment m we add a_1 and to decrement, we add a_2. Zero tests are done by alternation clauses, e.g., if $\mathcal{M}$ allows moving from state q to q' when $m = 0$, then we write a clause $q'(x) \Leftarrow q(x) \wedge zero_1(x)$ where $zero_1$ accepts all terms of the form $xa_1 + xa_2 + (y+n)b_1 + yb_2$. As reachability of states in two-counter automata is undecidable, emptiness of constant-only ACU automata with alternation clauses is undecidable. The same can be proved for $ACUM$ and $ACUD$ theories.

Now the above alternation clause can be coded as $Q(f(x)) \Leftarrow P_1(x), P(x) \Leftarrow Q(f(x)) \wedge P_2(x)$ for fresh Q. Accordingly, emptiness of constant-only ACU, $ACUM$, $ACUD$ automata with general push clauses is undecidable. This justifies the side-conditions $i \in \{1,...,n\} \setminus \{i_1,...,i_k\}, 1 \le i_1 < ... < i_k \le n$ in (11).

Note that we don't study automata with $+$*-push clauses*:

$$P(x) \Leftarrow P_1(x+y) \wedge P_2(y) \quad (12)$$

In the $ACUX$ case, this has no impact as (12) is equivalent to (6). In the $ACUM$ case (12) can be coded as $P(x+y) \Leftarrow P_1(x) \wedge Q(y), Q(-y) \Leftarrow P_2(y)$. In the ACU and $ACUD$ cases, they strictly increase expressiveness as they at least encode Petri nets; this is postponed to a later paper.

Finally adding equality constraints between brothers, i.e. dropping the condition that the free variables in the pop clauses should be distinct, also leads to undecidability: the alternation clause above can be coded as $P(x) \Leftarrow Q(f(x,y)), Q(f(x,x)) \Leftarrow P_1(x) \wedge P_2(x)$ for fresh Q.

3 Starting Up: The Constant-Only ACU Case, and an Easy Lemma

In this section, we recall some important results on constant-only ACU automata which will be useful throughout the paper. If the set of constants in the signature is $\Sigma_f = \{a_1, ..., a_p\}$ then 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 *semi-linear set* is a finite union of linear sets. The semi-linear sets are exactly the sets definable in Presburger arithmetic [5]. In particular they are closed under union, intersection, complementation and projection. We have the following result, also shown in [7] and corresponds to Corollary 6 of [13].

Lemma 3. *Constant-only ACU automata accept exactly semilinear sets.*

Proof. The proof uses Parikh's Theorem [14] which states that the commutative image of any context-free language is semi-linear, and in fact effectively so, together with the observation that the clauses (3), (6), (7) and (8) modulo ACU constitute exactly a context-free grammar modulo commutativity. $\square$

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 4. *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), (6) and (7) (see Figure 1.) 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.*

Proof. The first result follows from induction on δ. For the second, we replace the subderivations $\delta_1, ..., \delta_n$ by those of $P_1(s_1), ..., P_n(s_n)$ respectively. The atoms in the rest of the derivation need to be appropriately changed, keeping the clauses and equational rewritings applied at each step to be the same. $\square$

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 (10) (or clause (8)). 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 4 we have $t =_{ACU} t_1 + ... + t_n$.)*

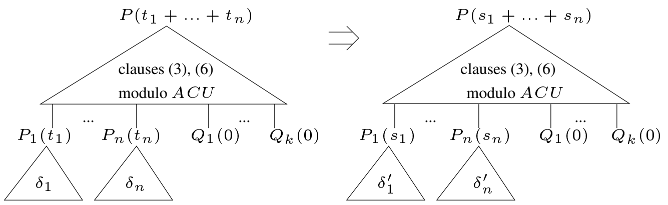

Fig. 1. Reuse of ACU derivations

4 One-Way $ACUX$ Case: Closure under Intersection

Consider a one-way $ACUX$ automaton $\mathcal{C}$ with predicates from some finite set $\mathbb{P}$. We introduce new predicate symbols (P, Q) and $\widehat{(P, Q)}$ for each $P, Q \in \mathbb{P}$, and sets of constants $S_1 = \{a_{P,Q} \mid P, Q \in \mathbb{P}\}$ and $S_2 = \{b_{P,Q} \mid P, Q \in \mathbb{P}\}$. The order of P, Q in all these is ignored. Instead of intersecting two distinct automata, we compute an automaton $\mathcal{C}_{inter}$ in which state (P, Q) represents intersection of P and Q for all P, Q. $\widehat{(P, Q)}$ accepts the functional terms among the terms accepted at (P, Q).

Consider the automaton $\mathcal{C}^*_{eq} = \mathcal{C}_{eq} \cup \{P(a_{P,Q}), P(b_{P,Q}) \mid P, Q \in \mathbb{P}\}$. The idea in defining $\mathcal{C}^*_{eq}$ and $\mathcal{C}_{P,Q,S,T}$ is to compute all possible derivations using clauses of the equational part. The $a_{P,Q}$'s and $b_{P,Q}$'s act as 'abstractions' for the functional terms accepted at both P and Q.

From Lemma 3 $\mathcal{L}_P(\mathcal{C}^*_{eq}/ACU)$ is a semilinear set for every P. For each $S \subseteq S_2$, we define $\mathcal{L}_{P,S}$ to be the set of those $t \in \mathcal{L}_P(\mathcal{C}^*_{eq}/ACU)$ such that each constant in S occurs in t a positive and even number of times and no constant from $S_2 \setminus S$ occurs in t. This operation is clearly Presburger-definable, and hence $\mathcal{L}_{P,S}$ is also a semilinear set. Define $\mathcal{L}'_{P,S}$ to be the language obtained from $\mathcal{L}_{P,S}$ by deleting all symbols of S_2, i.e., taking the image of $\mathcal{L}_{P,S}$ under the projection $\sum_{i,j} m_{ij} a_{P_i,Q_j} + \sum_{i,j} n_{ij} b_{P_i,Q_j} \mapsto \sum_{i,j} m_{ij} a_{P_i,Q_j}$. $\mathcal{L}'_{P,S}$ is again a semilinear set. Given $P, Q \in \mathbb{P}$ and $S, T \subseteq S_2$, clearly $\mathcal{L}'_{P,S} \cap \mathcal{L}'_{Q,T}$ is a semilinear set. By Lemma 3, we can construct a constant-only automaton $\mathcal{C}_{P,Q,S,T}$ with final state $F_{P,Q,S,T}$ such that $\mathcal{L}(\mathcal{C}_{P,Q,S,T}/ACU) = \mathcal{L}'_{P,S} \cap \mathcal{L}'_{Q,T}$. We assume that automata $\mathcal{C}_{P,Q,S,T}$'s are built from mutually disjoint sets of (fresh) states.

The required automaton $\mathcal{C}_{inter}$ has the following clauses:

- for each $P, Q \in \mathbb{P}$ and each $S, T \subseteq S_2$, the extended epsilon clause $(P, Q)(x) \Leftarrow F_{P,Q,S,T}(x) \wedge \bigwedge_{b_{R,R'} \in S \cup T} (R, R')(x_{R,R'})$.
- clauses (3), (6) and (7) (but not (8)) from each $\mathcal{C}_{P,Q,S,T}$.
- for each clause $R(a_{R',R''})$ in some $\mathcal{C}_{P,Q,S,T}$, the clause $R(x) \Leftarrow \widehat{(R', R'')}(x)$.
- for each pair of clauses $P(f(x_1, ..., x_n)) \Leftarrow P_1(x_1) \wedge ... \wedge P_n(x_n)$ and $Q(f(x_1, ..., x_n)) \Leftarrow Q_1(x_1) \wedge ... \wedge Q_n(x_n)$ in $\mathcal{C}_{free}$, the clause $\widehat{(P, Q)}(f(x_1, ..., x_n)) \Leftarrow (P_1, Q_1)(x_1) \wedge ... \wedge (P_n, Q_n)(x_n)$.

If $t =_{ACUX} t'$ then we must have $t =_{ACU} t_1 + ... + t_m + u_1 + v_1 + ... + u_n + v_n$, $t' = t'_1 + ... + t'_m + u'_1 + v'_1 + ... + u'_p + v'_p$, $t_i, t'_i, u_i, u'_i, v_i, v'_i$ being functional, such that $t_i =_{ACUX} t'_i, u_i =_{ACUX} v_i, u'_i =_{ACUX} v'_i$. The a's act as abstractions for the t_i, t'_i's and the b's for the u_i, v_i, u'_i, v'_i's. This is the reason we delete the b's from $\mathcal{L}_{P,S}$, representing the cancellations using X. Even though we can forget the actual values of u_i, v_i's, we need to be sure that there exist some terms to fill their place: this is the reason for the emptiness tests in the extended epsilon clauses of $\mathcal{C}_{inter}$. In this way we take care of the non-linearity and cancellation in the equation $x + x = 0$ using some kind of intersection-emptiness tests, and the remaining equations are dealt with using the results on ACU automata. This is made precise by Lemmas 5 and 6.

Lemma 5. *If $P(t')$ and $Q(t'')$ are derivable in $\mathcal{C}$ modulo ACU and $t' =_{ACUX} t''$, then for some $t =_{ACUX} t'$, $(P, Q)(t)$ is derivable in $\mathcal{C}_{inter}$ modulo ACU.*

Proof. By induction on the sum of the sizes of the derivations of $P(t')$ and $Q(t'')$. The derivations of $P(t')$ and $Q(t'')$ must have functional supports of the form $P_1(t'_1), ..., P_m(t'_m), I_1(u'_1), I'_1(u''_1), ..., I_n(u'_n), I'_n(u''_n)$ and $Q_1(t''_1), ..., Q_m(t''_m), J_1(v'_1), J''_1(v''_1), ..., J_p(v'_p), J''_p(v''_p)$ respectively such that $t'_i =_{ACUX} t''_i$, $u'_i =_{ACUX} u''_i$ and $v'_i =_{ACUX} v''_i$ respectively. From Lemma 4, $P(a_{P_1,Q_1} + ... + a_{P_m,Q_m} + 2b_{I_1,I'_1} + ... + 2b_{I_n,I'_n})$ and $Q(a_{P_1,Q_1} + ... + a_{P_m,Q_m} + 2b_{J_1,J'_1} + ... + 2b_{J_p,J'_p})$ are derivable in $\mathcal{C}^*_{eq}$ modulo ACU. Let $S = \{b_{I_1,I'_1}, ..., b_{I_n,I'_n}\}$ and $T = \{b_{J_1,J'_1}, ..., b_{J_p,J'_p}\}$. $F_{P,Q,S,T}(a_{P_1,Q_1} + ... + a_{P_m,Q_m})$ is derivable in $\mathcal{C}_{P,Q,S,T}$ modulo ACU. We must have $t'_i = f_i(t'^1_i, ..., t'^{k_i}_i)$, $t''_i = f_i(t''^1_i, ..., t''^{k_i}_i)$, $t'^j_i =_{ACUX} t''^j_i$, the derivation of $P_i(t'_i)$ must use a clause $P_i(f_i(x_1, ..., x_{k_i})) \Leftarrow P^1_i(x_1) \wedge ... \wedge P^{k_i}_i(x_{k_i})$ and the derivation of $Q_i(t''_i)$ must use a clause $Q_i(f_i(x_1, ..., x_{k_i})) \Leftarrow Q^1_i(x_1) \wedge ... \wedge Q^{k_i}_i(x_{k_i})$. By induction hypothesis, we must have $t^j_i =_{ACUX} t'^j_i$ so that $(P^j_i, Q^j_i)(t^j_i)$ is derivable in $\mathcal{C}$ modulo ACU. Let $t_i = f_i(t^1_i, ..., t^{k_i}_i)$. $\widehat{(P_i, Q_i)}(t_i)$ is derivable in $\mathcal{C}$ modulo ACU using the clause $\widehat{(P_i, Q_i)}(f_i(x_1, ..., x_{x_i})) \Leftarrow (P^1_i, Q^1_i)(x^1_i) \wedge ... \wedge (P^{k_i}_i, Q^{k_i}_i)(x^{k_i}_i)$. Let the derivation of $F_{P,Q,S,T}(a_{P_1,Q_1} + ... + a_{P_m,Q_m})$ in $\mathcal{C}_{P,Q,S,T}$ have functional support $R_1(a_{P_1,Q_1}), ..., R_m(a_{P_m,Q_m})$. Then clauses $R_i(x) \Leftarrow \widehat{(P_i, Q_i)}(x)$ are in $\mathcal{C}_{inter}$. Hence $R_1(t_1), ..., R_m(t_m)$ are derivable in $\mathcal{C}_{inter}$ modulo ACU and from Lemma 4, $F_{P,Q,S,T}(t_1 + ... + t_m)$ is derivable in $\mathcal{C}_{inter}$ modulo ACU. Let the required t be $t_1 + ... + t_m$. Also by induction hypothesis we must have $u_j =_{ACUX} u'_j$,$v_k =_{ACUX} v'_k$ such that $(I_j, I'_j)(u_j)$ and $(J_k, J'_k)(v_k)$ are derivable in $\mathcal{C}_{inter}$ modulo ACU for $1 \leq j \leq n$ and $1 \leq k \leq p$. Using the clause $(P, Q)(x) \Leftarrow F_{P,Q,S,T}(x) \wedge \bigwedge_{b_{R,R'} \in S \cup T}(R, R')(x_{R,R'})$ we get a derivation of $(P, Q)(t)$ in $\mathcal{C}_{inter}$ modulo ACU. Also it is clear that $t =_{ACUX} t'$. □

Lemma 6. *For $P, Q \in \mathbb{P}$, if $(P, Q)(t)$ is derivable in $\mathcal{C}_{inter}$ modulo ACU then for some $t' =_{ACUX} t'' =_{ACUX} t$, $P(t')$ and $Q(t'')$ are derivable in $\mathcal{C}$ modulo ACU.*

Proof. There must be some $S, T \subseteq S_2$ such that $F_{P,Q,S,T}(t)$ is derivable in $\mathcal{C}_{inter}$ modulo ACU (with a strictly smaller derivation), and terms $t_{R,R'}$ for each $b_{R,R'}$ in $S \cup T$, such that $(R, R')(t_{R,R'})$ is derivable in $\mathcal{C}_{inter}$ (with a strictly smaller derivation). From induction hypothesis we get terms $t'_{R,R'} =_{ACUX} t''_{R,R'} =_{ACUX} t_{R,R'}$ such that:

$R(t'_{R,R'})$ and $R'(t''_{R,R'})$ are derivable in $\mathcal{C}$ mod ACU, for each $b_{R,R'}$ in $S \cup T$. (*)

From the definition of $\mathcal{C}_{inter}$, the derivation of $F_{P,Q,S,T}(t)$ must have a functional support of the form $\widehat{(R'_1, R''_1)}(t_1), ..., \widehat{(R'_m, R''_m)}(t_m)$ for some $m \geq 0$, and the clause occurring immediately above the derivation of $\widehat{(R'_i, R''_i)}(t_i)$ must be of the form $R_i(x) \Leftarrow \widehat{(R'_i, R''_i)}(x)$. Then the clauses $R_i(a_{R'_i,R''_i})$ are in $\mathcal{C}_{P,Q,S,T}$ $(1 \leq i \leq m)$. From Lemma 4

$F_{P,Q,S,T}(a_{R'_1,R''_1} + ... + a_{R'_m,R''_m})$ must be derivable in $\mathcal{C}_{P,Q,S,T}$. So $a_{R'_1,R''_1} + ... + a_{R'_m,R''_m} \in \mathcal{L}'_{P,S}$. Therefore we must have $b_{I_1,I'_1}, ..., b_{I_n,I'_n} \in S (n \geq 0)$ such that $a_{R'_1,R''_1} + ... + a_{R'_m,R''_m} + 2b_{I_1,I'_1} + ... + 2b_{I_n,I'_n} \in \mathcal{L}_P(\mathcal{C}^*_{eq}/ACU)$.

Since each t_i is functional, there are $f_i, t_i^1, ..., t_i^{k_i}$ such that $t_i = f_i(t_i^1, ..., t_i^{k_i})$ and the derivation of $(\widehat{R'_i, R''_i})(t_i)$ in $\mathcal{C}_{inter}$ uses as last clause $(\widehat{R'_i, R''_i})(f_i(x_1, ..., x_{k_i})) \Leftarrow (R_i'^1, R_i''^1)(x_1) \wedge ... \wedge (R_i'^{k_i}, R_i''^{k_i})(x_{k_i})$ corresponding to clauses $R'_i(f_i(x_1, ..., x_{k_i})) \Leftarrow R_i'^1(x_1) \wedge ... \wedge R_i'^{k_i}(x_{k_i})$ and $R''_i(f_i(x_1, ..., x_{k_i})) \Leftarrow R_i''^1(x_1) \wedge ... \wedge R_i''^{k_i}(x_{k_i})$ of $\mathcal{C}_{free}$. Then $(R_i'^j, R_i''^j)(t_i^j)$ must be derivable in $\mathcal{C}_{inter}$ using derivations strictly smaller than that of $(P, Q)(t)$. Hence by induction hypothesis, we have $t_i'^j =_{ACUX} t_i''^j =_{ACUX} t_i^j$ such that $R_i'^j(t_i'^j), R_i''^j(t_i''^j)$ are derivable in $\mathcal{C}$ modulo ACU. Let $t'_i = f_i(t_i'^1, ..., t_i'^{k_i})$ and $t''_i = f_i(t_i''^1, ..., t_i''^{k_i})$. $R'_i(t'_i)$ is derivable in $\mathcal{C}$ using the clause $R'_i(f_i(x_1, ..., x_{k_i})) \Leftarrow R_i'^1(x_1) \wedge ... \wedge R_i'^{k_i}(x_{k_i})$. Similarly $R''_i(t''_i)$ is derivable in $\mathcal{C}$.

Now the derivation of $P(a_{R'_1,R''_1} + ... + a_{R'_m,R''_m} + 2b_{I_1,I'_1} + ... + 2b_{I_n,I'_n})$ in $\mathcal{C}^*_{eq}$ must have a functional support of the form $R_1^\dagger(a_{R'_1,R''_1}), ..., R_m^\dagger(a_{R'_m,R''_m}), I_1^\dagger(b_{I_1,I'_1}), I_1^\ddagger(b_{I_1,I'_1}), ..., I_n^\dagger(b_{I_n,I'_n}), I_n^\ddagger(b_{I_n,I'_n})$ where $R_i^\dagger \in \{R'_i, R''_i\}$ and $I_i^\dagger, I_i^\ddagger \in \{I_i, I'_i\}$. Since each $b_{I_i,I'_i} \in S$, by (*) $I_i(t'_{I_i,I'_i}), I'_i(t'_{I_i,I'_i})$ are derivable in $\mathcal{C}$ modulo ACU. Recall that $R'_i(t'_i), R''_i(t''_i)$ are derivable in $\mathcal{C}$ modulo ACU. So from Lemma 4 $P(t_1^\dagger + ... + t_m^\dagger + t^\dagger_{I_1,I'_1} + t^\ddagger_{I_1,I'_1} + ... + t^\dagger_{I_n,I'_n} + t^\ddagger_{I_n,I'_n})$ is derivable in $\mathcal{C}$ modulo ACU, where $t_i^\dagger \in \{t'_i, t''_i\}$ and $t^\dagger_{I_i,I'_i}, t^\ddagger_{I_i,I'_i} \in \{t'_{I_i,I'_i}, t''_{I_i,I'_i}\}$. Let the required t' be $t_1^\dagger + ... + t_m^\dagger + t^\dagger_{I_1,I'_1} + t^\ddagger_{I_1,I'_1} + ... + t^\dagger_{I_n,I'_n} + t^\ddagger_{I_n,I'_n})$. Then $t =_{ACUX} t'$ and $P(t')$ is derivable in $\mathcal{C}$. Similarly we can find t'' such that $t =_{ACUX} t''$ and $Q(t'')$ is derivable in $\mathcal{C}$. □

From Lemma 1, $\mathcal{L}_P(\mathcal{C}/ACUX) \cap \mathcal{L}_Q(\mathcal{C}/ACUX) = \mathcal{L}_{(P,Q)}(\mathcal{C}_{inter}/ACUX)$ for $P, Q \in \mathbb{P}$. The extended epsilon clauses can be eliminated using Lemma 2. Hence:

Theorem 1. *One-way $ACUX$ automata are closed under intersection.*

5 Two-Way ACUX Case: Translation to One-Way ACUX

Consider a two-way $ACUX$ automaton $\mathcal{C}$ with predicates from $\mathbb{P}$. To convert it to a automaton, we describe a saturation procedure which adds new epsilon clauses till the push clauses become redundant. The idea is that if any push clause is ever used then the corresponding free functional symbol must have been introduced by some pop clause. But the clauses from $\mathcal{C}_{eq}$ might have been used inbetween to add new terms, which eventually get canceled using X to leave only one functional term. Below, the b's act as abstractions for the terms that are canceled, and a's for the terms which remain.

We introduce new sets of constants $S_1 = \{a_P \mid P \in \mathbb{P}\}$ and $S_2 = \{b_{P,Q} \mid P, Q \in \mathbb{P}\}$. We do not distinguish between $b_{P,Q}$ and $b_{Q,P}$. We define $\mathcal{C}^*_{eq} = \mathcal{C}_{eq} \cup \{P(a_P) \mid P \in \mathbb{P}\} \cup \{P(b_{P,Q}) \mid P, Q \in \mathbb{P}\}$. $\mathcal{L}_P(\mathcal{C}^*_{eq}/ACU)$ is a semilinear set for every P. For $P, Q \in \mathbb{P}$ and $S \subseteq S_2$, define $\mathcal{L}_{P,Q,S,\mathcal{C}}$ to be the set of $t \in \mathcal{L}_P(\mathcal{C}^*_{eq}/ACU)$ such that

- a_Q occurs in t exactly once
- $\forall Q' \neq Q . a_{Q'}$ does not occur in t

– each constant in S occurs in t a positive and even number of times
– no constant from $S_2 \setminus S$ occurs in t.

Clearly $\mathcal{L}_{P,Q,S,\mathcal{C}}$ is also semilinear because it is Presburger-definable. In particular, we can effectively check its emptiness.

If $\mathcal{C}$ has a push clause $R(x_i) \Leftarrow P(f(x_1,...,x_n)) \wedge R_1(x_{i_1}) \wedge ... \wedge R_k(x_{i_k})$, a pop clause $Q(f(x_1,...,x_n)) \Leftarrow Q_1(x_1),...,Q_n(x_n)$, and there is a set $S \subseteq S_2$ such that

– $\mathcal{L}_{P,Q,S,\mathcal{C}} \neq \emptyset$
– $\forall b_{Q',Q''} \in S.\exists t.$ both Q' and Q'' accept t in $\mathcal{C}_{one-way}$ modulo $ACUX$
– $\forall j \in \{1,...,k\}.\exists t.$ both Q_{i_j} and R_j accept t in $\mathcal{C}_{one-way}$ modulo $ACUX$
– $\forall j \in \{1,...,n\} \setminus \{i, i_1,...,i_k\}.\exists t.Q_j$ accepts t in $\mathcal{C}_{one-way}$

then we will write $\mathcal{C} \rhd \mathcal{C} \cup \{R(x_i) \Leftarrow Q_i(x_i)\}$, which we take to constitute one step of our saturation procedure. This can be effectively decided because of the fact that one-way $ACUX$-automata are closed under intersection and hence their intersection emptiness is decidable. The saturation step is harmless:

Lemma 7. *Let $\mathcal{C} \rhd \mathcal{C} \cup \{R(x_i) \Leftarrow Q_i(x_i)\}$ as above. Then any atom derivable in $\mathcal{C} \cup \{R(x_i) \Leftarrow Q_i(x_i)\}$ modulo $ACUX$ is also derivable in $\mathcal{C}$ modulo $ACUX$.*

Proof. It is sufficient to show that for any t_i, if $Q_i(t_i)$ is derivable in $\mathcal{C}$, then $R(t_i)$ is derivable in $\mathcal{C}$. As in the definition above, let t_{i_j} be the term accepted at Q_{i_j} and R_j for $j \in \{1,...,k\}$. Also let t_j be the term accepted at Q_j for $j \in \{1,...,n\} \setminus \{i, i_1,...,i_k\}$. $Q(f(t_1,...,t_n))$ is derivable in $\mathcal{C}$ using the pop clause. As $L_{P,Q,S,\mathcal{C}} \neq \emptyset$, there must be an atom of the form $P(a_Q + 2b_{Q'_1,Q''_1} + ... + 2b_{Q'_p,Q''_p})$ derivable in $\mathcal{C}^*_{eq}$ modulo ACU, with $b_{Q'_i,Q''_i} \in S$. Let u_i be the term accepted at Q'_i and Q''_i in $\mathcal{C}_{one-way}$ modulo $ACUX$. The derivation of $P(a_Q+2b_{Q'_1,Q''_1}+...+2b_{Q'_p,Q''_p})$ must have a functional support of the form $Q(a_Q), Q^\dagger_1(b_{Q'_1,Q''_1}), Q^\ddagger_1(b_{Q'_1,Q''_1}),..., Q^\dagger_p(b_{Q'_p,Q''_p}), Q^\ddagger_p(b_{Q'_p,Q''_p})$ where $Q^\dagger_i, Q^\ddagger_i \in \{Q'_i, Q''_i\}$. From Lemma 4, we get a derivation of $P(f(t_1,...,t_n) + 2u_1 + ... + 2u_p)$ in $\mathcal{C}$ modulo $ACUX$. Thus modulo $ACUX$, we have a derivation of $P(f(t_1, ..., t_n))$. Then we get a derivation of $R(t_i)$ using the push clause. □

The converse is trivially true. Thus $\mathcal{C}$ and $\mathcal{C} \cup \{R(x_i) \Leftarrow Q_i(x_i)\}$ have the same set of derivable atoms modulo $ACUX$.

Given a two-way $ACUX$ automaton $\mathcal{C}$ our saturation procedure consists of (don't care non-deterministically) generating a sequence $\mathcal{C}_0(= \mathcal{C}) \rhd \mathcal{C}_1 \rhd \mathcal{C}_2...$ until no new clauses can be generated. This always terminates because there are only a finite number of epsilon clauses possible. Let the final (saturated) automaton be $\mathcal{D}$. Then we remove clauses (11) from $\mathcal{D}$ to get a one-way automaton $\mathcal{D}_{one-way}$. This step is also harmless:

Lemma 8. *If any atom is derivable in $\mathcal{D}$ modulo $ACUX$, then it is derivable in $\mathcal{D}_{one-way}$ modulo $ACUX$.*

Proof. It is sufficient to show that a derivation in $\mathcal{D}$, which has a push clause at the root and nowhere else, can be converted to a derivation in $\mathcal{D}_{one-way}$. Suppose we have a derivation of $R(t_i)$ from the derivations of $P(f(t_1,...,t_n)), R_1(t_{i_1}),...,R_k(t_{i_k})$ using the push clause $R(x_i) \Leftarrow P(f(x_1,...,x_n)) \wedge R_1(x_{i_1}) \wedge ... \wedge R_k(x_{i_k})$. Also, the latter derivations use only clauses from $\mathcal{D}_{one-way}$. Hence because of Lemma 1, we must have a derivation in $\mathcal{D}_{one-way}$ modulo ACU of an atom of the form $P(f(t'_1, ..., t'_n) + u_1 + v_1 + ... + u_p + v_p)$ with functional support of the form $Q(f(t'_1, ..., t'_n)), I_1(u_1), J_1(v_1), ..., I_k(u_p), J_k(v_p)$ such that $t_i =_{ACUX} t'_i$ and $u_i =_{ACUX} v_i$. By Lemma 4, $P(a_Q + 2b_{I_1,J_1} + ... + 2b_{I_p,J_p})$ is derivable in $\mathcal{D}^*_{eq}$ modulo ACU. Let $S = \{(I_1, J_1), ..., (I_p, J_p)\}$. Clearly $\mathcal{L}_{P,Q,S,\mathcal{D}} \neq \emptyset$. Now the derivation of $Q(f(t'_1, ..., t'_n))$ must be using some clause $Q(f(x_1, ..., x_n)) \Leftarrow Q_1(x_1) \wedge ... \wedge Q_n(x_n)$ and the derivations of $Q_1(t'_1), ..., Q_n(t'_n)$ in $\mathcal{D}_{one-way}$. As $t'_i =_{ACUX} t_i$, we have that $\mathcal{D} \rhd \mathcal{D} \cup \{R(x_i) \Leftarrow Q_i(x_i)\}$. But $\mathcal{D}$ is already saturated, hence $R(x_i) \Leftarrow Q_i(x_i) \in \mathcal{D}$. Also $Q_i(t_i)$ is derivable in $\mathcal{D}_{one-way}$. Hence $R(t_i)$ is derivable in $\mathcal{D}_{one-way}$. $\square$

The converse is trivially true. Combining Lemmas 7 and 8, we get:

Theorem 2. *A two-way $ACUX$ automaton can be effectively converted to a one-way $ACUX$ automaton accepting the same language.*

The results of the $ACUX$ case are easily generalized to the $ACUX_n$ case for every $n \geq 2$ by computing intersections of n-tuples of states instead of pairs of states. Hence:

Theorem 3. *Two-way $ACUX_n$ automata have same expressiveness as one-way $ACUX_n$ automata, and are closed under intersection.*

6 Constant-Only $ACUD$ Automata

We shall see that as the ACU case helped us in dealing with the $ACUX$ case, the $ACUD$ case helps us in dealing with the Abelian groups case.

Let $\mathcal{C}$ be a constant-only $ACUD$ automaton with predicates in $\mathbb{P}$. Except for clauses (9) which introduce '$-$' symbols, $\mathcal{C}$ would have been just an ACU automaton. In the $ACUD$ case, the languages are in fact very similar to semilinear sets. We now make this more precise. Recall that Σ_f is the set of constants in our signature. Define a set of fresh constants $\overline{\Sigma}_f = \{\overline{a} \mid a \in \Sigma_f\}$. Terms built from $\Sigma_f \cup \{+, -, 0\}$ modulo $ACUD$ are of the form $a_1 + ... + a_m - b_1 - ... - b_n$ ($m, n \geq 0$ and $a_i, b_j \in \Sigma_f$) while those built from $\Sigma_f \cup \overline{\Sigma}_f \cup \{+, 0\}$ modulo ACU are of the form $a_1 + ... + a_m + \overline{b_1} + ... + \overline{b_n}$ ($m, n \geq 0$ and $a_i, b_j \in \Sigma_f$). Hence there is a natural 1-1 correspondence between terms (resp. languages) on $\Sigma_f \cup \{+, -, 0\}$ modulo $ACUD$ and terms (resp. languages) on $\Sigma \cup \overline{\Sigma}_f \cup \{+, 0\}$ modulo ACU.

Consider new predicate symbols $\overline{P}$ for every $P \in \mathbb{P}$. Define automaton $\mathcal{C}^\dagger$ to consist of the following clauses:

- for clause (3) in $\mathcal{C}$, the same clause, and $\overline{P}(x) \Leftarrow \overline{P_1}(x)$.
- for clause (6) in $\mathcal{C}$, the same clause, and $\overline{P}(x + y) \Leftarrow \overline{P_1}(x) \wedge \overline{P_2}(y)$
- for clause (7) in $\mathcal{C}$, the same clause, and $\overline{P}(0)$
- for clause (8) in $\mathcal{C}$, the same clause, and $\overline{P}(\overline{a})$.
- for clause (9) in $\mathcal{C}$, the clauses $P(x) \Leftarrow \overline{P_1}(x)$ and $\overline{P}(x) \Leftarrow P_1(x)$

By simple induction on the derivations, we can show:

Lemma 9. *(1) If $P(a_1 + ... + a_m - b_1 ... - b_n)$ is derivable in $\mathcal{C}$ modulo $ACUD$ then $P(a_1 + ... + a_m + \overline{b_1} + ... + \overline{b_n})$ and $\overline{P}(\overline{a_1} + ... + \overline{a_m} + b_1 + ... + b_n)$ are derivable in $\mathcal{C}^\dagger$ modulo ACU.*
(2) If $P(a_1 + ... + a_m + \overline{b_1} + ... + \overline{b_n})$ or $\overline{P}(\overline{a_1} + ... + \overline{a_m} + b_1 + ... + b_n)$ is derivable in $\mathcal{C}^\dagger$ modulo ACU then $P(a_1 + ... + a_m - b_1 - ... - b_n)$ is derivable in $\mathcal{C}$ modulo $ACUD$.

Hence modulo the correspondence of languages discussed above:

Corollary 1. *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.*

7 Cancellative '$-$' Symbol: Abelian Groups Automata

The $ACUM$ equations are remarkably similar to $ACUX$: instead of canceling equal terms, we now cancel terms with opposite signs. In fact the constructions and proofs in this case are exactly the same as in the $ACUX$ case. As the ACU case helped in the $ACUX$ case, similarly the $ACUD$ case helps in the $ACUM$ case. Easy generalizations of Lemma 4 and Definition 1 to the $ACUD$ case are used.

The key new idea is that instead of canceling pairs of b's as in the $ACUX$ case, we cancel a b with a $\overline{b}$, where the $\overline{b}$'s act as abstractions for the negated terms. So we omit the full proofs of Theorems 4 and 5, but still make the constructions explicit.

Theorem 4. *One-way $ACUM$ automata are effectively closed under intersection.*

Proof. Let $\mathcal{C}$ be a one-way $ACUM$ automaton with predicates in $\mathbb{P}$. As in $ACUX$ case we use new predicate symbols (P, Q) and $\widehat{(P, Q)}$ for each $P, Q \in \mathbb{P}$. We will construct automaton $\mathcal{C}_{inter}$ which has states (P, Q) to accept intersection of P and Q. The new sets of constants used are $S_1 = \{a_{P,Q} \mid P, Q \in \mathbb{P}\}$, $S_2 = \{b_{P,Q} \mid P, Q \in \mathbb{P}\}$, $\overline{S_1} = \{\overline{a_{P,Q}} \mid P, Q \in \mathbb{P}\}$ and $\overline{S_2} = \{\overline{b_{P,Q}} \mid P, Q \in \mathbb{P}\}$.

Let $\mathcal{C}^*_{eq} = \mathcal{C}_{eq} \cup \{P(a_{P,Q}), P(b_{P,Q})\}$ for $P, Q \in \mathbb{P}$. From Corollary 1, for every P $\mathcal{L}_P(\mathcal{C}^*_{eq}/ACUD)$ is a semilinear set on the symbols from $S_1 \cup \overline{S_1} \cup S_2 \cup \overline{S_2}$. For each $S \subseteq S_2$, define $\mathcal{L}_{P,S}$ to be the set of those $t \in \mathcal{L}_P(\mathcal{C}^*_{eq}/ACUD)$ such that

- each $b_{P,Q} \in S$ occurs in t at least once
- no $b_{P,Q} \in S_2 \setminus S$ occurs in t
- for each $b_{P,Q} \in S_2$, $b_{P,Q}$ occurs exactly as many times as $\overline{b_{P,Q}}$ in t.

$\mathcal{L}_{P,S}$ is a semilinear set. Let $\mathcal{L}'_{P,S}$ be the language obtained from$\mathcal{L}_{P,S}$ by deleting all the symbols from $S_2 \cup \overline{S_2}$. This is again a semilinear set. For $P, Q \in \mathbb{P}$ and $S, T \subseteq S_2$ $\mathcal{L}'_{P,S} \cap \mathcal{L}'_{Q,T}$ is a semilinear set. By the second part of Corollary 1, we can construct a constant-only $ACUD$ automaton $\mathcal{C}_{P,Q,S,T}$ on signature $S_1 \cup \{+, -, 0\}$ with a final state $F_{P,Q,S,T}$ such that $\mathcal{L}(\mathcal{C}_{P,Q,S,T}/ACUD) = \mathcal{L}'_{P,S} \cap \mathcal{L}'_{Q,T}$. We assume that automata $\mathcal{C}_{P,Q,S,T}$'s are built from mutually disjoint sets of (fresh) states.

The required automaton $\mathcal{C}_{inter}$ has the following clauses:

– for each $P, Q \in \mathbb{P}$ and each $S, T \subseteq S_2$, the clause $(P,Q)(x) \Leftarrow F_{P,Q,S,T}(x) \wedge \bigwedge_{b_{R,R'} \in S \cup T} (R, R')(x_{R,R'})$.
– clauses (3, (6), (7) and (9) (but not (8)) from each $\mathcal{C}_{P,Q,S,T}$.
– for each clause $R(a_{R',R''})$ in some $\mathcal{C}_{P,Q,S,T}$, the clause $R(x) \Leftarrow \widehat{(R', R'')}(x)$.
– for each pair of clauses $P(f(x_1, ..., x_n)) \Leftarrow P_1(x_1) \wedge ... \wedge P_n(x_n)$ and $Q(f(x_1, ..., x_n)) \Leftarrow Q_1(x_1) \wedge ... \wedge Q_n(x_n)$ in $\mathcal{C}_{free}$ the clause $\widehat{(P,Q)}(f(x_1, ..., x_n)) \Leftarrow (P_1, Q_1)(x_1) \wedge ... \wedge (P_n, Q_n)(x_n)$. $\square$

Theorem 5. *Two-way $ACUM$ automata can be effectively converted to one-way $ACUM$ automata.*

Proof. Consider a two-way $ACUM$ automaton $\mathcal{C}$ with predicates in $\mathbb{P}$. As with $ACUX$ we describe the base step of the saturation procedure that adds epsilon clauses. (No new ideas are used.) The new sets of constants used are $S_1 = \{a_P \mid P \in \mathbb{P}\}$, $\overline{S_1} = \{\overline{a_P} \mid P \in \mathbb{P}\}$, $S_2 = \{b_{P,Q} \mid P, Q \in \mathbb{P}\}$, and $\overline{S_2} = \{\overline{b_{P,Q}} \mid P, Q \in \mathbb{P}\}$. We define $\mathcal{C}^*_{eq} = \mathcal{C}_{eq} \cup \{P(a_P) \mid P \in \mathbb{P}\} \cup \{P(b_{P,Q}) \mid P, Q \in \mathbb{P}\}$. From Corollary 1, for every P, $\mathcal{L}_P(\mathcal{C}^*_{eq}/ACUD)$ is a semilinear set on constants from $S_1 \cup S_2 \cup \overline{S_1} \cup \overline{S_2}$. For $P, Q \in \mathbb{P}$ and $S \subseteq S_2$, define $\mathcal{L}_{P,Q,S,\mathcal{C}}$ to be the set of $t \in \mathcal{L}_P(\mathcal{C}^*_{eq}/ACUD)$ such that

– a_Q occurs in t exactly once
– $\forall Q' \neq Q. a_{Q'}$ does not occur in t
– each constant in S occurs in t at least once
– no constant from $S_2 \setminus S$ occurs in t
– for each $b_{P,Q} \in S_2$, $b_{P,Q}$ occurs exactly as many times as $\overline{b_{P,Q}}$ in t.

$\mathcal{L}_{P,Q,S,\mathcal{C}}$ is also semilinear because the above conditions are definable using Presburger formulas. In particular, we can effectively check emptiness of $\mathcal{L}_{P,Q,S,\mathcal{C}}$.

If $\mathcal{C}$ contains a push clause $R(x_i) \Leftarrow P(f(x_1, ..., x_n)) \wedge R_1(x_{i_1}) \wedge ... \wedge R_k(x_{i_k})$, a pop clause $Q(f(x_1, ..., x_n)) \Leftarrow Q_1(x_1), ..., Q_n(x_n)$, and a set $S \subseteq S_2$ such that

– $\mathcal{L}_{P,Q,S,\mathcal{C}} \neq \emptyset$
– $\forall b_{Q',Q''} \in S. \exists t.$ both Q' and Q'' accept t in $\mathcal{C}_{one-way}$ modulo $ACUM$
– $\forall j \in \{1, ..., k\}. \exists t.$ both Q_{i_j} and R_j accept t in $\mathcal{C}_{one-way}$ modulo $ACUM$
– $\forall j \in \{1, ..., n\} \setminus \{i, i_1, ..., i_k\}. \exists t. Q_j$ accepts t in $\mathcal{C}_{one-way}$

then we write $\mathcal{C} \rhd \mathcal{C} \cup \{R(x_i) \Leftarrow Q_i(x_i)\}$, which is one step of our saturation procedure. The rest works as in the $ACUX$ case. $\square$

8 Simpler Cases: General ACU and $ACUD$ Automata

We observe that all the complications in the $ACUX$ and $ACUM$ cases were because of the equations $x + x = 0$ and $x - x = 0$ which cancel terms. Forgetting them (but keeping the distributivity of '$-$' symbol,) we get the ACU and $ACUD$ cases, which formed the basis of our results in the $ACUX$ and $ACUM$ cases. By looking at the proofs, it is easy to see that all the results proved for $ACUX$ and $ACUM$ automata continue to hold for ACU and $ACUD$ automata. In fact they become much simpler: all we need to do is to systematically ignore the parts about the $b, \overline{b}$'s which accounted for cancellations. For lack of space we content ourselves to summarize the results:

Theorem 6. *Two-way ACU (resp. $ACUD$) automata can be effectively converted to one-way ACU (resp. $ACUD$) automata and are effectively closed under intersection.*

9 Idempotence Axiom: $ACUI$ Automata

In the $ACUI$ case also we use techniques similar to the previous cases. We have:

Theorem 7. *One-way $ACUI$ automata are effectively closed under intersection.*

Proof. Let $\mathcal{C}$ be a one-way $ACUI$ automaton with predicates from $\mathbb{P}$. Instead of computing intersections of pairs of states, we will need to compute intersections of all tuples of states. Hence we introduce new predicates S and $\widehat{S}$ for every $\emptyset \neq S \subseteq \mathbb{P}$. A state $S = \{P_1, ..., P_n\}$ represents the intersection of states $P_1, ..., P_n$. $\widehat{S}$ accepts the functional terms among the terms accepted at S.

For each $\emptyset \neq S \subseteq \mathbb{P}$ we introduce a new constant a_S which will be used as an abstraction for the terms to be accepted at $\widehat{S}$. Let $\mathcal{C}^*_{eq} = \mathcal{C}_{eq} \cup \{P(a_S) \mid P \in S\}$. From Lemma 3 $\mathcal{L}_P(\mathcal{C}^*_{eq}/ACU)$ is a semilinear set for every P. Define $\mathcal{L}_P = \{n_1 a_{S_1} + ... + n_k a_{S_k} \mid \text{ some } m_1 a_{S_1} + ... + m_k a_{S_k} \in \mathcal{L}_P(\mathcal{C}^*_{eq}/ACU), 1 \leq n_i \leq m_i\}$. This step accounts for the contractions using the equation I. $\mathcal{L}_P$ is a semilinear set. For every $\emptyset \neq S \subseteq \mathbb{P}$, $\mathcal{L}_S = \bigcap_{P \in S} \mathcal{L}_P$ is a semilinear set. By Lemma 3, we can construct a constant-only ACU 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.

The required automaton $\mathcal{C}_{inter}$ has the following clauses:

- for each $\emptyset \neq S \subseteq \mathbb{P}$, the clause $S(x) \Leftarrow F_S(x)$.
- clauses (3), (6) and (7) (but not (8)) 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 clauses $P^i(f(x_1, ..., x_n)) \Leftarrow P^i_1(x_1) \wedge ... \wedge P^i_n(x_n)$ in $\mathcal{C}$ for $1 \leq i \leq k, k \geq 1$, the clause $\widehat{S}(f(x_1, ..., x_n)) \Leftarrow S_1(x_1) \wedge ... \wedge S_n(x_n)$ where $S = \{P^1, ..., P^k\}$ and $S_j = \{P^1_j, ..., P^k_j\}$. □

While the one-way automata are closed under intersection, unlike in the other theories, we do not know whether the two-way automata have the same expressiveness as the one-way automata. Still we do know that the two-way $ACUI$ automata are powerful enough to encode alternation: the clause $P(x) \Leftarrow P_1(x) \wedge P_2(x)$ can be encoded as $Q_1(f(x)) \Leftarrow P_1(x), Q_2(f(x)) \Leftarrow P_2(x), Q(x + y) \Leftarrow Q_1(x) \wedge Q_2(y), P(x) \Leftarrow Q(f(x))$ for fresh predicates Q_1, Q_2, Q. We have already seen that alternation produces undecidability for ACU, $ACUD$ and $ACUM$ theories. This suggests that this problem is difficult and might require new techniques.

10 Conclusion

We have dealt with 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, $n \geq 2$), and $ACUI$.

For each of these theories, we have shown that the languages accepted by one-way automata are effectively closed under union and intersection, and that emptiness is decidable. Also for all these theories except $ACUI$, the two-way automata can be translated to equivalent one-way automata. Care has been taken to suitably restrict the format of push clauses (look back at the side-conditions of (11)): without them emptiness would be undecidable in the ACU, $ACUD$ and $ACUM$ cases. In particular the corresponding two-way automata would not be reducible to one-way automata. We also saw that alternation leads to undecidable cases modulo these theories. In this sense, these automata have different behavior than classical (i.e. non-equational) automata.

Two questions remain to be answered. First, the equivalence between two-way and one-way $ACUI$ automata is conjectured. Second, the effect of adding clauses (12), which is trivial for the theories $ACUX$ and $ACUM$, is unknown for the others. Preliminary results show that this strictly increases expressiveness modulo ACU or $ACUD$.

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

References

1. T. Colcombet. Rewriting in the partial algebra of typed terms modulo AC. In A. Kucera and R. Mayr, editors, *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. T. Frühwirth, E. Shapiro, M. Y. Vardi, and E. Yardeni. Logic programs as types for logic programs. In *LICS'91*, 1991.
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. D. Lugiez. A good class of tree automata. Application to inductive theorem proving. In *ICALP'98*, pages 409–420. Springer-Verlag LNCS 1443, 1998.
9. D. Lugiez. Counting and equality constraints for multitree automata. In *FOSSACS'03*. Springer-Verlag LNCS, 2003.
10. M. L. Minsky. Recursive unsolvability of Post's problem of "tag" and other topics in the theory of Turing machines. *Annals of Mathematics, Second Series*, 74(3):437–455, 1961.
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. R. J. Parikh. On context-free languages. *Journal of the ACM*, 13(4):570–581, 1966.
15. M. Y. Vardi. Reasoning about the past with two-way automata. In *ICALP'98*, pages 628–641. Springer Verlag LNCS 1443, 1998.