

Alternation in Equational Tree Automata modulo XOR

Kumar Neeraj Verma

Institut für Informatik, TU München, Germany
verma@in.tum.de

Abstract. Equational tree automata accept terms modulo equational theories, and have been used to model algebraic properties of cryptographic primitives in security protocols. A serious limitation is posed by the fact that alternation leads to undecidability in case of theories like ACU and that of Abelian groups, whereas for other theories like XOR, the decidability question has remained open. In this paper, we give a positive answer to this open question by giving effective reductions of alternating general two-way XOR automata to equivalent one-way XOR automata in 3EXPTIME, which also means that they are closed under intersection but not under complementation. We also show that emptiness of these automata, which is needed for deciding secrecy, can be decided directly in 2EXPTIME, without translating them to one-way automata. A key technique we use is the study of Branching Vector Plus-Minimum Systems (BVPMS), which are a variant of VASS (Vector Addition Systems with States), and for which we prove a pumping lemma allowing us to compute their coverability set in EXPTIME.

1 Introduction

Tree automata [7, 4] are a well known tool for verifying cryptographic protocols [14, 8, 3]. Most approaches to verifying cryptographic protocols are based on the assumption of *perfect cryptography* which ignores the algebraic properties of encryption. Such an analysis is often unrealistic. For example an attack [17] was found against Bull's recursive authentication protocol which uses XOR for encryption, although the protocol was shown to be secure [16] assuming perfect cryptography. To deal with such algebraic properties, we have introduced *equational tree automata* [20, 22, 21] which accept terms modulo equational theories. While related, but not identical, notions of automata have independently been introduced by others [15, 13], the distinguishing feature of our approach is the description of automata transitions using Horn clauses, which provide a uniform framework for expressing variants of *general two-wayness* and *alternation* [18] (see also [4], Chapter 7), as well as for dealing with arbitrary equational theories.

Protocol insecurity is NP-complete [2, 1] for several theories including XOR, assuming bounded number of sessions. This may help in detecting attacks which require very small number of sessions. But for *certifying* protocols we need some *safe* abstraction, which does not miss any attacks. A common safe abstraction is to let a bounded number of nonces be used in infinitely many sessions, although security still remains undecidable [3], even with perfect cryptography. With this abstraction, any protocol can easily be modeled using Horn clauses, following e.g. the approach of [5]. The idea is to define an unary predicate I such that $I(m)$ holds exactly for messages m known to the

intruder. Then to obtain clauses of alternating general two-way automata we use some safe abstraction, as in [10], or [9]. The secrecy problem then reduces to the intersection-emptiness problem of these automata: we check that $I(m)$ is false for certain set of m 's.

In our previous work, we have dealt with equational tree automata [22, 21] for the theory ACU (consisting of the axioms $x + (y + z) = (x + y) + z$, $x + y = y + x$ and $x + 0 = x$) of an associative-commutative symbol $+$ with unit 0 , and its variants, since these are the ones which occur most frequently in cryptographic protocols. These variants are obtained by adding certain axioms to the theory ACU. The variants studied include the theory AG of Abelian groups obtained by adding the axiom $x + (-x) = 0$, the theory XOR obtained by adding the axiom $x + x = 0$, and the theory ACUI obtained by adding the axiom $x + x = x$ of idempotence. We have used the theories ACU and AG to model [20] the IKA.1 group key agreement protocol [19] using decidable fragments of our equational tree automata. More efficient approximate verification techniques for the same modeling are used in [10] to obtain a fully automated proof of security of this protocol in the so-called pure eavesdropper model. Unfortunately alternation leads to undecidability in case of theories ACU and AG. Similarly general two-wayness encodes alternation and leads to undecidability.

While alternation and general two-wayness lead to undecidability for theories ACU and AG, the question was left open for the theories XOR and ACUI. Surprisingly, we show in this paper that XOR automata are decidable even with alternation and general two-wayness. We show that these automata can actually be reduced to equivalent one-way XOR automata in 3EXPTIME, hence they are closed under intersection but not under complementation, and their emptiness is decidable. We also show the latter problem to be decidable in 2EXPTIME, without constructing the equivalent one-way automata. Recall also that emptiness of alternating tree automata is EXPTIME-hard already in the non-equational case [4]. Our results imply that secrecy of cryptographic protocols with XOR modeled using alternating general two-way XOR automata is decidable in 2EXPTIME.

While our techniques used to deal with non-alternating equational tree automata relied heavily on semilinear, or Presburger-definable, sets, the techniques required in this paper to deal with the alternating case are remarkably different, based on studying Branching Vector Plus-Minimum Systems (BVPMS), which are a variant of VASS [11], similar to Branching VASS (BVASS) [23]. Unlike VASS and BVASS, BVPMS have a minimum operation but no subtraction. We show an interesting and natural connection between BVPMS and alternating XOR automata. A key result of the paper is a pumping lemma for BVPMS, allowing us to compute their *coverability sets*, which are approximations of the set of reachable configurations, in EXPTIME. In contrast, coverability sets of VASS and BVASS are computed using the Karp-Miller construction [12] which however only gives a non-primitive recursive algorithm. Our techniques also yield similar decidability results for the theory ACUI which was the other open case.

Note that alternation is crucial for precise analysis of protocols. Consider the following example protocol in standard notation, where K_{ab}^i are private keys

$$\begin{array}{l}
 A \rightarrow B : \{Na\}_{K_{ab}^1}, \{Na\}_{K_{ab}^2} \\
 B \rightarrow A : Na \\
 A \rightarrow B : \{Nb\}_{K_{ab}^1}
 \end{array}$$

between A and B . Following [5], and knowing that an intruder knows (m_1, m_2) iff it knows both m_1 and m_2 , we get clauses of the form $I(\text{enc}(Na, K_{ab}^1)), I(\text{enc}(Na, K_{ab}^2))$,

$I(x) \Leftarrow I(\text{enc}(x, K_{ab}^1)) \wedge I(\text{enc}(x, K_{ab}^2)), I(\text{enc}(Nb, K_{ab}^1))$ (enc represents encryption) and also clauses for deductive abilities of the intruder, e.g. $I(\text{pair}(x, y)) \Leftarrow I(x) \wedge I(y), I(y) \Leftarrow I(\text{pair}(x, y))$. These clauses are translatable to alternating general two-way automata clauses. E.g. the third clause above is translated as $P(K_{ab}^1), Q(K_{ab}^2), R(x) \Leftarrow I(\text{enc}(x, y)) \wedge Q(y), I(x) \Leftarrow I(\text{enc}(x, y)) \wedge R(x) \wedge P(y)$ for fresh predicates P, Q, R . Nb is secret, and $I(Nb)$ is not deducible. Note however that the variable x on the left side of the third clause appears twice on the right. Such clauses violate the restrictions imposed to obtain decidability in [22] and can easily encode alternation [22]. We may abstract this as the clause $I(x) \Leftarrow I(\text{enc}(x, K_{ab}^1)) \wedge I(\text{enc}(y, K_{ab}^2))$, (note distinct variables x and y). But now $I(Nb)$ becomes true, producing a false attack. The point we are making here is that disallowing alternation may force us to abstract away too much information from the protocols.

To our knowledge the only other work which presents decidability results for unbounded number of sessions in presence of some equational theory is [5, 6], which presents a decidable class \mathcal{C}^\oplus of Horn clauses with XOR. This is incomparable to ours: they don't allow *+pop clauses* (see Section 2) unless $P = P_1 = P_2$ but allow more general clauses involving other symbols. The clauses required for the example protocol modeled using \mathcal{C}^\oplus in [5, 6] belong to our class. We give a 2EXPTIME algorithm for deciding secrecy whereas the complexity upper bound known for \mathcal{C}^\oplus is non-elementary, as remarked in [6]. In this work we are also interested in the expressiveness and closure properties of our automata, besides the secrecy problem.

The paper is organized as follows. We start in the next section by introducing equational tree automata and examining the structure of derivations in alternating XOR automata. We then introduce BVPMS in Section 3 and show how to decide emptiness of alternating XOR automata by using their connections with BVPMS. In Section 4 we prove a pumping lemma for BVPMS which allows us to compute their coverability sets. This is then used in Section 5 to eliminate alternation from XOR automata. These results are used to treat general two-wayness in Section 6.

2 Equational Tree Automata

Consider a signature which includes the special symbols $+$ and 0 . Symbols other than $+, 0$ are called *free*. *Functional terms* are terms of the form $f(t_1, \dots, t_n)$ where f is free. We use unary predicates to represent states of automata. Read an atom $P(t)$ as ‘term t is accepted at state P ’. An (*equational*) *tree automaton* is a finite set of definite clauses $P(t) \Leftarrow P_1(t_1) \wedge \dots \wedge P_n(t_n)$ (t, t_i 's may contain variables). Read this clause as ‘if t_i is accepted at P_i for $1 \leq i \leq n$ then t is accepted at P ’. $P(t)$ is the *head*, and the remaining part the *tail* of the clause. Given an equational theory \mathcal{E} modulo which an automaton \mathcal{A} is considered, *derivations* of ground atoms are defined using the rules:

$$\boxed{\frac{P_1(t_1\sigma) \dots P_n(t_n\sigma)}{P(t\sigma)} (P(t) \Leftarrow P_1(t_1) \wedge \dots \wedge P_n(t_n)) \in \mathcal{A} \quad \frac{P(s)}{P(t)} (s =_{\mathcal{E}} t)}$$

where σ is a ground substitution, and $=_{\mathcal{E}}$ is the congruence on terms induced by the theory \mathcal{E} . Hence the derivable atoms are the elements of the least Herbrand model modulo \mathcal{E} . If S is a set of states then we also say that $S(t)$ is derivable to mean that $P(t)$ is derivable for all $P \in S$. We write \mathcal{A}/\mathcal{E} to indicate the equational theory modulo which the automaton \mathcal{A} is considered. We define the language $L_P(\mathcal{A}/\mathcal{E}) = \{t \mid$

$P(t)$ is derivable}. If in addition some state P is designated as *final* then the language accepted by the automaton is $L_P(\mathcal{A}/\mathcal{E})$. A state is *empty* if it accepts no term.

We are in particular interested in the following kinds of clauses, called *zero clauses*, *+pop clauses*, *alternation clauses*, *free pop clauses* and *general push clauses* respectively, where x_1, \dots, x_n are mutually distinct in clauses (4) and (5),

$$P(0) \quad (1) \quad P(x + y) \Leftarrow P_1(x) \wedge P_2(y) \quad (2)$$

$$P(x) \Leftarrow P_1(x) \wedge \dots \wedge P_n(x) \quad (n \geq 1)(3)$$

$$P(f(x_1, \dots, x_n)) \Leftarrow P_1(x_1) \wedge \dots \wedge P_n(x_n) \quad (f \text{ is free})(4)$$

$$P(x_i) \Leftarrow Q(f(x_1, \dots, x_n)) \wedge P_1(x_{i_1}) \wedge \dots \wedge P_k(x_{i_k})(5)$$

$$(f \text{ is free}, 1 \leq i, i_1, \dots, i_k \leq n)$$

x, y are distinct in clause (2), and P, Q, P_i 's are states. See [10] for an example modeling of cryptographic protocols using these clauses. *Alternating automata* contain clauses (1-4). *Alternating general two-way automata* contain clauses (1- 5). Alternation clauses (3) in which $n = 1$ are called ϵ clauses. *One-way automata* are alternating automata in which all alternation clauses are ϵ clauses. In the non-equational case, one-way automata are exactly the classical tree automata usually described in the literature: clause (4) is usually written as the rewrite rule $f(P_1, \dots, P_n) \rightarrow P$, and ϵ clauses are written as $P_1 \rightarrow P$. For an automaton \mathcal{A} , we denote by \mathcal{A}_{free} the set of clauses (4) in \mathcal{A} , and by \mathcal{A}_{eq} the set of clauses (1-3) in \mathcal{A} . The (emptiness of) *intersection* of a set S of states is the (emptiness of) the set of terms t such that $S(t)$ is derivable. In presence of alternation clauses, deciding intersection-emptiness of S is the same as deciding emptiness of a fresh state P by adding the clause $P(x) \Leftarrow \bigwedge_{Q \in S} Q(x)$.

XOR derivations. We examine the structure of derivations in alternating XOR automata and relate them to BVPMS. Recall that modulo XOR, any term can be converted to a *normal* form by repeatedly replacing subterms $t + t$ by 0. If $s =_{XOR} t$ then s and t have the same normal form (upto $=_{ACU}$ congruence). As our interest is in the theory XOR, throughout this paper, if $s =_{ACU} t$ then we treat s and t as the same object. This will not cause any confusion. We think of derivations in equational tree automata as trees. At each node we either apply a clause or rewrite using the equational theory. Let \mathcal{A}/\mathcal{E} be an alternating automaton on a finite set of states \mathbb{P} . Then any derivation δ of an atom $P(t)$ in \mathcal{A}/\mathcal{E} is uniquely described as $\mathcal{C}[\delta_1, \dots, \delta_n]$ (the ordering of the δ_i 's being ignored) such that each subtree δ_i uses an application of a free pop clauses at the root node, and the nodes in \mathcal{C} contain only applications of clauses from \mathcal{A}_{eq} and rewritings using \mathcal{E} . If the conclusion of each δ_i is $P_i(t_i)$ then we call the (unordered) list $P_1(t_1), \dots, P_n(t_n)$ as the *functional support* of δ . (Clearly each t_i is functional.) This definition generalizes that of [22] for one-way automata.

Let $\mathcal{Z}_{\mathbb{P}} = \{S \mid \emptyset \neq S \subseteq \mathbb{P}\}$. Introduce a new set of constants: $A = \{a_S \mid S \in \mathcal{Z}_{\mathbb{P}}\}$. We use a_S as abstraction for functional terms accepted at each state in S , in order to analyze derivations. Let $p = |\mathcal{Z}_{\mathbb{P}}| = 2^{|\mathbb{P}|} - 1$. We name the elements of $\mathcal{Z}_{\mathbb{P}}$ as $\mathbb{S}_1, \dots, \mathbb{S}_p$. Modulo ACU, any term on the signature $A \cup \{0, +\}$ is of the form $\sum_{i=1}^p n_i a_{\mathbb{S}_i}$, equivalently p -tuples $(n_1, \dots, n_p) \in \mathbb{N}^p$. In this paper we consider them interchangeably as terms or p -tuples. For $\nu \in \mathbb{N}^p$ the i th component of ν is $\nu[i]$. If $I = \{i_1, \dots, i_k\}$ where $1 \leq i_1 < \dots < i_k \leq p$ then $\nu[I]$ denotes $(\nu[i_1], \dots, \nu[i_k]) \in \mathbb{N}^k$. The tuple (n, \dots, n) is also written as n . For $n \in \mathbb{N}$ define the *characteristic* $n^* \in \{0, 1\}$ as: $n^* = 0$ if n is even, and $n^* = 1$ otherwise. For $\nu \in \mathbb{N}^p$ define $\nu^* = (\nu[1]^*, \dots, \nu[p]^*) \in \{0, 1\}^p$. Define partial order \leq_{xor} (on \mathbb{N} and on \mathbb{N}^p) as:

$x \leq_{xor} y$ iff $x \leq y$ and $x^* = y^*$. For $Z \subseteq \mathbb{Z}_{\mathbb{P}}$, define $\mathcal{X}_{\mathcal{A},Z}$ to consist of atoms modulo ACU which can be deduced by the following rules.

$$\begin{array}{l} \mathbf{1.} \frac{}{P(a_S)} (P \in S \in Z) \quad \mathbf{2.} \frac{P(t+a_S+a_S)}{P(t)} \quad \mathbf{3.} \frac{P_1(t_1) \quad P_2(t_2)}{P(t_1+t_2)} (P(x+y) \Leftarrow P_1(x) \wedge P_2(y) \in \mathcal{A}) \\ \mathbf{4.} \frac{}{P(0)} (P(0) \in \mathcal{A}) \quad \mathbf{5.} \frac{P_1(t) \dots P_n(t)}{P(t)} (P(x) \Leftarrow P_1(x) \wedge \dots \wedge P_n(x) \in \mathcal{A}) \end{array}$$

Rule 2 is another way of saying that if $P(t) \in \mathcal{X}_{\mathcal{A},Z}$ and $t' \leq_{xor} t$ then $P(t') \in \mathcal{X}_{\mathcal{A},Z}$. An atom in $\mathcal{X}_{\mathcal{A},Z}$ summarizes the effect of an arbitrarily large derivation in \mathcal{A}/XOR using clauses of \mathcal{A}_{eq} . The constants a_S represent the effect of applying the clauses of \mathcal{A}_{free} . The sets in Z account for the terms which are canceled using the XOR axiom during the derivation. This is formally stated by Lemmas 1 and 2, and illustrated by Example 1. Let $(X_j)_{1 \leq j \leq k}$ denote an unordered list X_1, \dots, X_k .

Lemma 1. *Let $t_1 + \dots + t_n$ be the normal form of t where each t_i is functional. Then every derivation δ of $P(t)$ in \mathcal{A}/XOR has a functional support of the form $(P_i^j(t_i))_{1 \leq i \leq n, 1 \leq j \leq k_i}, (Q_i^j(u_i))_{1 \leq i \leq m, 1 \leq j \leq l_i}$ with $k_i \geq 1$ for $1 \leq i \leq n$, $m \geq 0$ and $l_i \geq 1$ for $1 \leq i \leq m$, such that, letting $S_i = \{P_i^j \mid 1 \leq j \leq k_i\}$ for $1 \leq i \leq n$ and letting $T_i = \{Q_i^j \mid 1 \leq j \leq l_i\}$ for $1 \leq i \leq m$, for all $U_1 \supseteq S_1, \dots, U_n \supseteq S_n, V_1 \supseteq T_1, \dots, V_m \supseteq T_m, Z \supseteq \{U_1, \dots, U_n, V_1, \dots, V_m\}$ we have $P(a_{U_1} + \dots + a_{U_n}) \in \mathcal{X}_{\mathcal{A},Z}$.*

Lemma 2. *Let $P(a_{S_1} + \dots + a_{S_n}) \in \mathcal{X}_{\mathcal{A},Z}$. Then $S_i \in Z$ for $1 \leq i \leq n$. If there are terms t_1, \dots, t_n such that $S_1(t_1), \dots, S_n(t_n)$ are derivable in \mathcal{A}/XOR , and if for each $S \in Z$ there is some term t_S such that $S(t_S)$ is derivable in \mathcal{A}/XOR , then $P(t_1 + \dots + t_n)$ is derivable in \mathcal{A}/XOR .*

Example 1. Let automaton \mathcal{A} have following clauses:

$$\begin{array}{lll} P_1(g(x)) \Leftarrow P_5(x) & P(x+y) \Leftarrow P_1(x) \wedge P_2(y) & P_5(f(x)) \Leftarrow P_6(x) \\ P_2(x+y) \Leftarrow P_3(x) \wedge P_4(y) & P_3(x) \Leftarrow P_7(x) \wedge P_8(x) & P_9(0) \\ P_4(x+y) \Leftarrow P_6(x) \wedge P_9(y) & P_6(a) & P_7(a) & P_8(a) \end{array}$$

Here is a possible derivation in \mathcal{A}/XOR . Its functional support is $P_1(g(f(a))), P_7(a), P_8(a), P_6(a)$.

Also $P(a_{\{P_1\}}) \in \mathcal{X}_{\mathcal{A},\{\{P_1\},\{P_7,P_8,P_6\}\}}$.

$$\begin{array}{lll} & P_6(a) & P_7(a) \quad P_8(a) \quad P_6(a) \quad P_9(0) \\ & \underline{P_6(a)} & \underline{P_7(a) \quad P_8(a)} \quad \underline{P_6(a)} \quad \underline{P_9(0)} \\ & P_5(f(a)) & P_3(a) \quad P_4(a) \\ & \underline{P_5(f(a))} & \underline{P_3(a)} \quad \underline{P_4(a)} \\ P_1(g(f(a))) & \underline{P_1(g(f(a)))} & \underline{P_2(a+a)} \\ & \underline{P_1(g(f(a)))} & \underline{P_2(a+a)} \\ & P(g(f(a)) + a + a) & \\ & \underline{P(g(f(a)) + a + a)} & \\ & P(g(f(a))) & \end{array}$$

3 Branching Vector Plus-Minimum Systems

We will see that alternating XOR automata are naturally related to Branching Vector Plus-Minimum System (BVPMS). Fix some $1 \leq r \in \mathbb{N}$. We will be dealing with non-negative r -tuples (i.e. elements of \mathbb{N}^r). A BVPMS on some finite set \mathbb{Q} of states is defined to be a finite set of clauses of the following form, called *constant clauses*, *addition clauses* and *minimum clauses* respectively, where P, P_i 's are from \mathbb{Q} , x, y are distinct variables, and x_1, \dots, x_n are mutually distinct variables.

$$P(\nu) \quad (\nu \in \{0, 1\}^r) \quad (6) \quad P(x+y) \Leftarrow P_1(x) \wedge P_2(y) \quad (7)$$

$$P(x_1 \sqcap \dots \sqcap x_n) \Leftarrow P_1(x_1) \wedge \dots \wedge P_n(x_n) (n \geq 1) \quad (8)$$

A *configuration* is of the form $P(\nu)$ for $P \in \mathbb{Q}$ and $\nu \in \mathbb{N}^r$. As for tree automata, we can read it as ‘ ν is accepted at P ’. *Derivations* of configurations in BVPMS are inductively defined in a similar way as derivations of atoms in equational tree automata. We interpret $+$ as componentwise addition of r -tuples, and \sqcap as componentwise minimum. Then by instantiating all the variables in a clause by r -tuples, if the resulting configurations in the tail are derivable then the resulting configuration in the head is said to be derivable. The restriction $\nu \in \{0, 1\}^r$ in clause (6) is for the complexity results. Using clauses (6) and (7), we can clearly express clauses $P'(\nu')$ for arbitrary $\nu' \in \mathbb{N}^r$.

Lemma 3. *The emptiness problem for BVPMS, i.e. the question whether some state accepts no tuple, is decidable in polynomial time.*

From XOR Automata to BVPMS. The minimum clauses allow us to model the alternation clauses of automata in presence of the cancellation axiom of XOR. Let \mathcal{A} be an alternating XOR automaton on a finite set of states \mathbb{P} . Let $\mathcal{Z}_{\mathbb{P}}$ and p be as in Section 2. We use a BVPMS \mathcal{V} on p -tuples to compute the sets $\mathcal{X}_{\mathcal{A}, Z}$. (The parameter r above is instantiated to p). The set of states of \mathcal{V} is $\mathbb{Q} = \mathbb{P} \times \{0, 1\}^p \times 2^{\mathcal{Z}_{\mathbb{P}}}$. The configuration $(P, \nu', Z)(\nu)$ represents the fact that $P(\nu) \in \mathcal{X}_{\mathcal{A}, Z}$ and $\nu' = \nu^*$. The intuition is that Rules 2 and 5 in the definition of sets $\mathcal{X}_{\mathcal{A}, Z}$ can be represented by minimum clauses. However we should not take the minimum of an even number with an odd number. This is controlled by ν' in the state (P, ν', Z) . The clauses of \mathcal{V} are as listed below. Note that by the conventions of Section 2, $a_S \in \{0, 1\}^p$ in item (iii). We use the parameter c_1 to denote the maximum size of any clause in \mathcal{A} , the size of a clause being the number of atoms in it, and c_2 to denote the number of clauses in \mathcal{A} . The corresponding parameters for BVPMS are v_1 and v_2 . Without loss of generality we assume that in clause (3), P_1, \dots, P_n are mutually distinct. We have $|\mathbb{Q}| = 2^{2^{O(|\mathbb{P}|)}}$, $v_1 = O(|\mathbb{P}|)$, and $v_2 = c_2 \cdot 2^{2^{O(|\mathbb{P}|)}}$. \mathcal{V} is computable in time $c_2 \cdot 2^{2^{O(|\mathbb{P}|)}}$.

(i) $(P, 0, Z)(0)$ for each $Z \subseteq \mathcal{Z}_{\mathbb{P}}$ and clause $P(0) \in \mathcal{A}$	(iii) $(P, a_S, Z)(a_S)$ for each $P \in S \in Z \subseteq \mathcal{Z}_{\mathbb{P}}$.
(ii) $(P, (\nu_1 + \nu_2)^*, Z)(x + y) \Leftarrow (P_1, \nu_1, Z)(x) \wedge (P_2, \nu_2, Z)(y)$ for each $\nu_1, \nu_2 \in \{0, 1\}^p$, $Z \subseteq \mathcal{Z}_{\mathbb{P}}$ and clause $P(x + y) \Leftarrow P_1(x) \wedge P_2(y) \in \mathcal{A}$.	(iv) $(P, \nu, Z)(x_1 \sqcap \dots \sqcap x_n) \Leftarrow (P_1, \nu, Z)(x_1) \wedge \dots \wedge (P_n, \nu, Z)(x_n)$ for each $\nu \in \{0, 1\}^p$, $Z \subseteq \mathcal{Z}_{\mathbb{P}}$ and clause $P(x) \Leftarrow P_1(x) \wedge \dots \wedge P_n(x) \in \mathcal{A}$.

Lemma 4. *We have the following results:*

- (i) *If $P(\nu) \in \mathcal{X}_{\mathcal{A}, Z}$ then for some ν' , $(P, \nu^*, Z)(\nu')$ is derivable in \mathcal{V} and $\nu \leq_{xor} \nu'$.*
- (ii) *If $(P, \nu', Z)(\nu)$ is derivable in \mathcal{V} then $P(\nu) \in \mathcal{X}_{\mathcal{A}, Z}$ and $\nu' = \nu^*$.*

This relation between BVPMS and sets $\mathcal{X}_{\mathcal{A}, Z}$, together with the relation between sets $\mathcal{X}_{\mathcal{A}, Z}$ and derivations in \mathcal{A} as stated by Lemmas 1 and 2, are the basis of all results about alternating XOR automata in this paper. First we show how to decide emptiness of the latter. Recall that *propositional Horn clauses* are of the form $X \Leftarrow X_1 \wedge \dots \wedge X_n$ where X, X_i 's are *propositions*. Derivations of propositions using a set of propositional Horn clauses are inductively defined as usual. Let \mathcal{A} and \mathcal{V} be as above. Introduce proposition X_S for $S \in \mathcal{Z}_{\mathbb{P}}$, and define a set H of propositional Horn clause to contain: (i) clause $X_S \Leftarrow X_{S_1} \wedge \dots \wedge X_{S_n}$, where $S = \{P^1, \dots, P^k\}$, $S_i = \{P_i^1, \dots, P_i^k\}$ for $1 \leq i \leq n$, and $P^j(f(x_1, \dots, x_n)) \Leftarrow P_1^j(x_1) \wedge \dots \wedge P_n^j(x_n) \in \mathcal{A}_{free}$ for $1 \leq j \leq k$

(ii) clause $X_S \Leftarrow \bigwedge_{T \in Z} X_T$, where for some $\nu' \in \{0, 1\}^p$, for all $P \in Z$, the state (P, ν', Z) is non-empty in \mathcal{V} .

Then X_S is derivable from H iff S has non-empty intersection. Clauses (i) are standard. Clauses (ii) take care of permutations and cancellations using the XOR axioms.

Theorem 1. *Emptiness of alternating XOR automata is decidable in 2EXPTIME.*

Eliminating alternation requires more work. The *coverability set* of BVPMS is (some finite representation of) the downward closure of the set of derivable configurations, for the following ordering on configurations: $P(\nu) \leq P'(\nu')$ iff $P = P'$ and $\nu \leq \nu'$, where \leq also denotes the component-wise ordering on tuples. Since the sets $\mathcal{X}_{A,Z}$ are downward closed (for an ordering on configurations based on \leq_{xor}), our main interest is in the coverability set of \mathcal{V} . For this we use a pumping lemma.

4 A Pumping Lemma for BVPMS

Fix a BVPMS \mathcal{V} on r -tuples on a finite set of states \mathbb{Q} . We will show that if a certain component in a derivable configuration exceeds $2^{|\mathbb{Q}|}$ then it can become arbitrarily large. We use a pumping argument which iterates certain portions of the derivation. Compared to usual pumping arguments, the main difficulty here is that we need to do simultaneous pumping on several (possibly overlapping) parts of the derivation. For example in clauses (8), in order to *strictly* increase $x_1 \sqcap \dots \sqcap x_n$ in some l th component, we have to ensure that *each* x_i *strictly* increases in the l th component. At the same time we have to take care that the other components do not decrease while pumping. Also note that because of the minimum clauses, the values of configurations need not increase and might even decrease if an observer goes from a node in a derivation towards the root.

An *Add-Min Tree (AMT)* in \mathcal{V} is a finite tree \mathcal{T} , each of whose nodes is labeled with an element of \mathbb{Q} , and each of whose edges is labeled with some $\nu \in \mathbb{N}^r$. Instead of pumping directly on the derivations in \mathcal{V} , we will compute AMTs which pick out those paths on which we will pump. The values from other paths will label the edges. AMTs are what we will pump on. As usual, positions (or nodes) in an AMT are described using strings of positive integers. The empty string λ is the root node. The k children of a node α are $\alpha \cdot 1, \dots, \alpha \cdot k$. The state labeling the node α is denoted $\theta_{\mathcal{T}}(\alpha)$. For all notations we define, we omit the AMTs in subscripts if there is no confusion. If β is a descendant of some node α then the *distance* from α to β , denoted $d_{\mathcal{T}}(\alpha, \beta)$ is the sum of the labels of all the edges on the path from α to β . An AMT \mathcal{T} is *feasible* if for every non-leaf node α with k children, if $(\theta(\alpha \cdot i))(\nu_i)$ is derivable in \mathcal{V} for some ν_i for $1 \leq i \leq k$, then $(\theta(\alpha))(\prod_{1 \leq i \leq k} (\nu_i + d(\alpha, \alpha \cdot i)))$ is derivable in \mathcal{V} . A *valued AMT* is an AMT \mathcal{T} together with a label $\omega_{\mathcal{T}}(\alpha) \in \mathbb{N}^r$ for every node α of \mathcal{T} , such that, if α is a non-leaf node with k children then $\omega(\alpha) = \prod_{1 \leq i \leq k} (\omega(\alpha \cdot i) + d(\alpha, \alpha \cdot i))$, and if α is a leaf node then $(\theta(\alpha))(\omega(\alpha))$ is derivable in \mathcal{V} . From definitions:

Lemma 5. *For a node α in a valued feasible AMT \mathcal{T} , $(\theta(\alpha))(\omega(\alpha))$ is derivable in \mathcal{V} .*

As addition distributes over minimum, we have:

Lemma 6. *In a valued AMT \mathcal{T} , for every node α we have*

$$\omega(\alpha) = \prod_{\beta \text{ is a leaf in the subtree rooted at } \alpha} (\omega(\beta) + d(\alpha, \beta)).$$

For $1 \leq l \leq r$, we say $\nu_1 <_l \nu_2$ to mean that $\nu_1 \leq \nu_2$ and $\nu_1[l] < \nu_2[l]$. Given two (possibly valued) AMTs \mathcal{T}_1 and \mathcal{T}_2 , we say $\mathcal{T}_1 <_l \mathcal{T}_2$ to mean that $\theta_{\mathcal{T}_1}(\lambda) = \theta_{\mathcal{T}_2}(\lambda)$, and for every leaf node α of \mathcal{T}_2 , there is a leaf node β of \mathcal{T}_1 , such that $\theta_{\mathcal{T}_1}(\beta) = \theta_{\mathcal{T}_2}(\alpha)$ and $d_{\mathcal{T}_1}(\lambda, \beta) <_l d_{\mathcal{T}_2}(\lambda, \alpha)$.

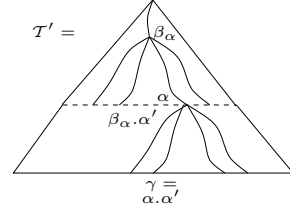
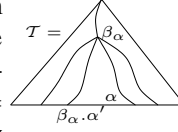
Lemma 7. *Let \mathcal{T}_1 be a valued AMT and \mathcal{T}_2 a feasible AMT such that $\mathcal{T}_1 <_l \mathcal{T}_2$. Then $(\theta_{\mathcal{T}_2}(\lambda))(\nu)$ is derivable in \mathcal{V} for some ν such that $\omega_{\mathcal{T}_1}(\lambda) <_l \nu$.*

Proof. (Sketch:) For each leaf node α of \mathcal{T}_2 let β_α be a leaf node of \mathcal{T}_1 such that $\theta_{\mathcal{T}_1}(\beta_\alpha) = \theta_{\mathcal{T}_2}(\alpha)$ and $d_{\mathcal{T}_1}(\lambda, \beta_\alpha) <_l d_{\mathcal{T}_2}(\lambda, \alpha)$. We transform \mathcal{T}_2 into a valued AMT by defining $\omega_{\mathcal{T}_2}(\alpha)$ for each node α of \mathcal{T}_2 . We do this by induction on the height of α . If α is a leaf, we define $\omega_{\mathcal{T}_2}(\alpha) = \omega_{\mathcal{T}_1}(\beta_\alpha)$. If α is a non-leaf node with k children, we define $\omega_{\mathcal{T}_2}(\alpha) = \prod_{1 \leq i \leq k} (\omega_{\mathcal{T}_2}(\alpha.i) + d_{\mathcal{T}_2}(\alpha, \alpha.i))$. Clearly this makes \mathcal{T}_2 a valued AMT. From Lemma 5, $(\theta_{\mathcal{T}_2}(\lambda))\omega_{\mathcal{T}_2}(\lambda)$ is derivable. We now use Lemma 6 and arithmetic properties of minimum to show that $\omega_{\mathcal{T}_1}(\lambda) <_l \omega_{\mathcal{T}_2}(\lambda)$. \square

We call a node α of an AMT \mathcal{T} an l -major node if every edge out of α has a label ν with $0 <_l \nu$. Otherwise α is an l -minor node. Trivially a leaf node is always l -major. We use l -major nodes for pumping in AMTs:

Lemma 8. *Let \mathcal{T} be a feasible AMT such that every leaf node α has a strict ancestor (call it β_α) which is l -major, such that $\theta_{\mathcal{T}}(\beta_\alpha) = \theta_{\mathcal{T}}(\alpha)$. Then there is a feasible AMT \mathcal{T}' with $\mathcal{T} <_l \mathcal{T}'$.*

Proof. Let AMT \mathcal{T}' be obtained from \mathcal{T} by replacing each leaf node α by the subtree of \mathcal{T} at position β_α . \mathcal{T}' is feasible since \mathcal{T} is feasible and $\theta_{\mathcal{T}}(\beta_\alpha) = \theta_{\mathcal{T}}(\alpha)$. To show that $\mathcal{T} <_l \mathcal{T}'$, pick any leaf node γ of \mathcal{T}' . We have to find a leaf node β of \mathcal{T} such that $\theta_{\mathcal{T}}(\beta) = \theta_{\mathcal{T}'}(\alpha)$



and $d_{\mathcal{T}}(\lambda, \beta) <_l d_{\mathcal{T}'}(\lambda, \alpha)$. By construction, there is a leaf node α of \mathcal{T} and a string α' such that $\gamma = \alpha \cdot \alpha'$, $d_{\mathcal{T}}(\beta_\alpha, \beta_\alpha \cdot \alpha') = d_{\mathcal{T}'}(\alpha, \gamma)$ and $\theta_{\mathcal{T}}(\beta_\alpha \cdot \alpha') = \theta_{\mathcal{T}'}(\gamma)$. We show that the required β is $\beta_\alpha \cdot \alpha'$. As β_α is a strict major ancestor of α we have $0 <_l d_{\mathcal{T}}(\beta_\alpha, \alpha)$. Hence $d_{\mathcal{T}}(\lambda, \beta_\alpha \cdot \alpha') <_l d_{\mathcal{T}}(\lambda, \beta_\alpha \cdot \alpha') + d_{\mathcal{T}}(\beta_\alpha, \alpha) = d_{\mathcal{T}}(\lambda, \beta_\alpha) + d_{\mathcal{T}}(\beta_\alpha, \beta_\alpha \cdot \alpha') + d_{\mathcal{T}}(\beta_\alpha, \alpha) = d_{\mathcal{T}'}(\lambda, \beta_\alpha) + d_{\mathcal{T}'}(\alpha, \gamma) + d_{\mathcal{T}'}(\beta_\alpha, \alpha) = d_{\mathcal{T}'}(\lambda, \gamma)$. \square

Observe that in the above proof we pump on every path of \mathcal{T} . The pumping lemma is now proved below. Given a derivation of a configuration in which the l th component is large enough, we compute a suitable AMT in which every path has large enough number of l -major nodes, allowing us to do pumping.

Lemma 9. *If $P(\nu)$ is derivable in \mathcal{V} and $\nu[l] \geq 2^{|\mathcal{Q}|}$ then $P(\nu')$ is derivable in \mathcal{V} for some $\nu <_l \nu'$.*

Proof. Given some derivation δ of some atom $Q(\mu)$ in \mathcal{V} , we define a valued feasible AMT \mathcal{T}_δ by recursion on δ . \mathcal{T} looks almost the same as δ except that when a node of δ uses an addition clause, then in \mathcal{T}_δ we selectively forget one of the children, and the corresponding tuple is used as an edge label. The construction is such that the root of \mathcal{T}_δ is labeled with Q and μ . The construction is as follows:

(i) If δ is the derivation of $Q(\mu)$ by applying the clause $Q(\mu)$, then \mathcal{T}_δ is a leaf labeled Q and μ .

(ii) If δ is the derivation of $Q(\mu_1 \sqcap \dots \sqcap \mu_k)$, using a minimum clause, from the derivations δ_i of $Q_i(\mu_i)$ for $1 \leq i \leq k$, then \mathcal{T}_δ has the root node λ labeled with Q and $\mu_1 \sqcap \dots \sqcap \mu_k$, has k children $\mathcal{T}_{\delta_1}, \dots, \mathcal{T}_{\delta_k}$, and each edge out of λ is labeled 0.

(iii) If δ is the derivation of $Q(\mu_1 + \mu_2)$ using an addition clause, from the derivations δ_1 and δ_2 of $Q_1(\mu_1)$ and $Q_2(\mu_2)$ respectively, then assume $\mu_1[l] \leq \mu_2[l]$. (The case $\mu_2[l] \leq \mu_1[l]$ is treated similarly.) Then \mathcal{T}_δ has the root node λ labeled with Q and $\mu_1 + \mu_2$, has one child \mathcal{T}_{δ_2} , and the edge out of λ is labeled μ_1 .

Clearly \mathcal{T}_δ is valued and feasible. The first step produces an l -major (leaf) node. The second step creates an l -minor node. The third step creates an l -major node iff $\mu_1[l] > 0$. Thus \mathcal{T}_δ has the property that if β is a child of α then:

(i) If α is l -minor then $\omega(\beta)[l] \geq \omega(\alpha)[l]$. (ii) If α is l -major then $2\omega(\beta)[l] \geq \omega(\alpha)[l]$.

Hence for any $i \geq 1$, if α is the i th l -major node on any path from the root, then $2^{i-1} \cdot \omega(\alpha)[l] \geq \omega(\lambda)[l]$. Now let δ' be a derivation in \mathcal{V} of the atom $P(\nu)$. Let $\mathcal{T} = \mathcal{T}_{\delta'}$. Since $\nu[l] \geq 2^{|\mathbb{Q}|}$, and since by construction $\omega_{\mathcal{T}}(\alpha) \leq 1$ for any leaf node α , hence every maximal path ξ in \mathcal{T} contains at least $|\mathbb{Q}| + 1$ distinct l -major nodes. Hence at least two of them should be labeled with the same state. Let them be at the (distinct) positions α_ξ and $\alpha_\xi \cdot \beta_\xi$. Let \mathcal{T}_1 be the AMT obtained from \mathcal{T} by chopping off the subtree below position $\alpha_\xi \cdot \beta_\xi$ for each maximal path ξ of \mathcal{T} . Since \mathcal{T} is valued and feasible, by Lemma 5, \mathcal{T}_1 is also valued and feasible. For every leaf position γ of \mathcal{T}_1 , there is some maximal path ξ of \mathcal{T} such that $\gamma = \alpha_\xi \cdot \beta_\xi$. Hence every γ has a strict major ancestor labeled with the same state. By Lemma 8 there is a feasible AMT \mathcal{T}_2 such that $\mathcal{T}_1 <_l \mathcal{T}_2$. The result then follows from Lemma 7. \square

Repeated applications of Lemma 9 allows us to show:

Theorem 2. *If $P(\nu)$ is derivable in \mathcal{V} and $\nu[I] \geq 2^{|\mathbb{Q}|}$, where $I \subseteq \{1, \dots, r\}$, then for any number K there is some $\nu' \geq \nu$ such that $P(\nu')$ is derivable in \mathcal{V} and $\nu'[I] \geq K$.*

Computing the coverability set. The pumping lemma allows us to compute the coverability set for BVPMS. Define $\nu^\# \in (\mathbb{N} \cup \{\infty\})^r$ for $\nu \in \mathbb{N}^r$ as: $\nu^\#[i] = \infty$ if $\nu[i] \geq 2^{|\mathbb{Q}|}$ and $\nu^\#[i] = \nu[i]$ otherwise. We extend $+$, \sqcap and \leq by letting $n + \infty = \infty + \infty = \infty$, $n \sqcap \infty = n$, $\infty \sqcap \infty = \infty$, $n \leq \infty$ and $\infty \leq \infty$ for $n \in \mathbb{N}$. They are extended to r -tuples as usual. Define set $C_{\mathcal{V}} = \{P(\nu^\#) \mid P(\nu) \text{ is derivable in } \mathcal{V}\}$.

Lemma 10. *$C_{\mathcal{V}}$ is the coverability set of \mathcal{V} , in the sense that*

(i) *If $P(\nu) \in C_{\mathcal{V}}$ and $\nu \geq \nu' \in \mathbb{N}^r$ then there is some $\nu'' \geq \nu'$ such that $P(\nu'')$ is derivable in \mathcal{V} .*

(ii) *If $P(\nu)$ is derivable in \mathcal{V} then $\nu \leq \nu'$ for some $P(\nu') \in C_{\mathcal{V}}$.*

Theorem 3. *The coverability set of BVPMS can be computed in EXPTIME.*

5 Eliminating Alternation

Let \mathcal{A} be an alternating XOR automaton on a finite set of states \mathbb{P} . We define a corresponding BVPMS \mathcal{V} on set of states \mathbb{Q} as in Section 3. Let $\mathcal{Z}_{\mathbb{P}} = \{\mathbb{S}_1, \dots, \mathbb{S}_p\}$ as

in Section 2. We now define an equivalent one-way XOR automaton \mathcal{B} . We will need clauses of the form $P(x_1 + \dots + x_n) \Leftarrow P_1(x_1) \wedge \dots \wedge P_n(x_n)$ which are translatable in linear time to clauses of one-way automata by using some auxiliary states. We also need *extended ϵ -clauses* of the form $P(x) \Leftarrow P_1(x) \wedge Q_1(y_1) \wedge \dots \wedge Q_n(y_n)$ where x, y_1, \dots, y_n are mutually distinct variables. This can be thought of as the ϵ clause $P(x) \Leftarrow P_1(x)$ together with emptiness tests on states Q_1, \dots, Q_n . As emptiness for one-way equational tree automata is decidable in polynomial time [22, 20], these clauses do not increase the expressiveness of one-way equational tree automata and can be eliminated in polynomial time [22, 20]. The states of \mathcal{B} are of the form $(S, \nu', Z, (\nu_P)_{P \in S})$ and \overline{S} for $S \in \mathcal{Z}_{\mathbb{P}}$, $\nu' \in \{0, 1\}^p$, $Z \subseteq \mathcal{Z}_{\mathbb{P}}$ and tuples $\nu_P \in (\mathbb{N} \cup \{\infty\})^p$ such that $(P, \nu', Z)(\nu_P) \in C_{\mathcal{V}}$ for each $P \in S$. State \overline{S} is supposed to accept the terms accepted at each state in S . The clauses added to \mathcal{B} are as follows. We would have liked \mathcal{B} to contain the clause $\overline{S}(x_1 + \dots + x_n) \Leftarrow \overline{S}_1(x_1) \wedge \dots \wedge \overline{S}_n(x_n) \wedge \bigwedge_{T \in Z} \overline{T}(y_T)$ if $P(a_{S_1} + \dots + a_{S_n}) \in \mathcal{X}_{\mathcal{A}, Z}$ for each $P \in S$. However $\mathcal{X}_{\mathcal{A}, Z}$ may be infinite. Hence we use states $(S, \nu', Z, (\nu_P)_{P \in S})$, and clauses (1a) and (1c), together with the loops in clauses (1b) to achieve this effect. Clauses (2) are standard.

(1) For each state $(S, \nu', Z, (\nu_P)_{P \in S})$ (call it \mathcal{S}) of \mathcal{B} , let $\nu = \prod_{P \in S} \nu_P$. Let $I = \{i \mid \nu[i] = \infty\}$ and $J = \{1, \dots, p\} \setminus I$. We add to \mathcal{B} the clauses

(a) $\mathcal{S}(\sum_{i \in I, \nu[i]=1} x_i + \sum_{i \in J} \sum_{j=1}^{\nu[i]} x_i^j) \Leftarrow \bigwedge_{i \in I, \nu[i]=1} \overline{S}_i(x_i) \wedge \bigwedge_{i \in J} \bigwedge_{j=1}^{\nu[i]} \overline{S}_i(x_i^j)$
for n_i 's such that $n_i \leq_{xor} \nu[i]$,

(b) $\mathcal{S}(x + y + z) \Leftarrow \mathcal{S}(x) \wedge \overline{S}_i(y) \wedge \overline{S}_i(z)$ for each $i \in I$,

(c) the extended ϵ -clause $\overline{S}(x) \Leftarrow \mathcal{S}(x) \wedge \bigwedge_{T \in Z} \overline{T}(y_T)$.

(2) For each free f and clauses $P^i(f(x_1, \dots, x_n)) \Leftarrow P_1^i(x_1) \wedge \dots \wedge P_n^i(x_n) \in \mathcal{A}$ for $1 \leq i \leq k$, we add to \mathcal{B} the clause $\overline{S}(f(x_1, \dots, x_n)) \Leftarrow \overline{S}_1(x_1) \wedge \dots \wedge \overline{S}_n(x_n)$ where $S = \{P^1, \dots, P^k\}$ and for $1 \leq i \leq n$, $S_i = \{P_i^1, \dots, P_i^k\}$.

The purpose of the clauses (1a) and (1b) is to accept at \mathcal{S} a summation of terms accepted at the \overline{S}_i 's. For $i \in I$, the number of summands from \overline{S}_i is an arbitrarily large even or odd number with characteristic $\nu'[i]$. For $i \in J$ the number of summands from \overline{S}_i is some $n_i \leq_{xor} \nu[i]$. We do so because the elements of $C_{\mathcal{V}}$ can be considered as 'limits' of elements of $\mathcal{X}_{\mathcal{A}, Z}$'s. If P_f is the final state of \mathcal{A} then $\{P_f\}$ is the final state of \mathcal{B} . Now since alternating automata are trivially closed under intersection, and one-way XOR automata are not closed under complementation [21]:

Theorem 4. *Alternating XOR automata can be converted to equivalent one-way XOR automata in 3EXPTIME. The class of languages accepted by them is same as that accepted by one-way XOR automata, is closed under intersection, but not closed under complementation.*

6 Eliminating General Two-Wayness

Let \mathcal{A} be an alternating general two-way XOR automaton on a finite set of states \mathbb{P} . Define BVPMS \mathcal{V} as in Section 3. In the absence of alternation clauses, and with restriction $i \notin \{i_1, \dots, i_k\}$ in clauses (5), these automata have been shown to be equivalent to one-way XOR automata in [22]. The case of alternation clauses was open. Also clauses (5) without restriction can encode alternation [22] and the question of its decidability was open. We now show decidability in presence of all these clauses. The

standard ‘saturation’ procedure in the non-equational case ‘short-cuts’ pop and general push clauses to get new alternation clauses, till the general push clauses become redundant. The problematic +-pop clauses are now dealt with using BVPMS. For example, given clauses $R(x) \Leftarrow Q(f(x, y)) \wedge P^1(x) \wedge P^2(y)$, $Q(x + y) \Leftarrow P_1(x) \wedge Q'(y)$, $Q'(x + y) \Leftarrow P_2(x) \wedge P_3(y)$ and $P_1(f(x, y)) \Leftarrow Q_1(x) \wedge Q_2(y)$, we can infer the clause $R(x) \Leftarrow Q_1(x) \wedge P^1(x)$, provided that $\{P^2, Q_2\}$ has non-empty intersection, and $\{P_2, P_3\}$ has non-empty intersection. Formally for any alternating general two-way automaton \mathcal{A}' , let automaton \mathcal{A}'_{alt} consist of clauses (1-4) of \mathcal{A}' . If

- (i) \mathcal{A} contains a general push clause $R(x_l) \Leftarrow Q(f(x_1, \dots, x_n)) \wedge \bigwedge_{i=1}^n \bigwedge_{j=1}^{k_i} P_i^j(x_i)$,
- (ii) (Q, a_S, Z) is non-empty in \mathcal{V} for some S, Z ,
- (iii) $P(f(x_1, \dots, x_n)) \Leftarrow Q_1^P(x_1) \wedge \dots \wedge Q_n^P(x_n) \in \mathcal{A}$ for some Q_i^P 's for each $P \in S$
- (iv) the set $\{P_i^j \mid 1 \leq j \leq k_i\} \cup \{Q_i^P \mid P \in S\}$ has non-empty intersection in \mathcal{A}_{alt}/XOR for $i \in \{1, \dots, n\} \setminus \{l\}$
- (v) each $T \in Z$ has non empty intersection in \mathcal{A}_{alt}/XOR

then consider the clause $C = R(x_l) \Leftarrow \bigwedge_{j=1}^{k_l} P_l^j(x_l) \wedge \bigwedge_{P \in S} Q_l^P(x_l)$. If $C \notin \mathcal{A}$ then we write $\mathcal{A} \triangleright \mathcal{A} \cup \{C\}$, which we take to constitute one step of our saturation procedure. Arbitrarily many applications of clauses (1-3) may occur in between the applications of the free pop clauses and the general push clause. However after cancellations using the XOR axiom, only a functional term should be left. This is checked by condition (ii). Conditions (iv) and (v) can be effectively checked by Theorem 1.

Given any alternating general two-way automaton \mathcal{A} our saturation procedure now consists of (don't care non-deterministically) generating a sequence $(\mathcal{A} =) \mathcal{A}_0 \triangleright \mathcal{A}_1 \triangleright \mathcal{A}_2 \dots$ until no new clauses can be added. This terminates because only finitely many alternation clauses are possible. Let the final saturated automaton be \mathcal{C} . We remove all general push clauses to get alternating automaton \mathcal{C}_{alt} equivalent to \mathcal{A} .

Theorem 5. *Alternating general two-way XOR automata can be converted to equivalent one-way XOR automata in 3EXPTIME. The class of languages accepted by them is same as that accepted by one-way XOR automata, is closed under intersection, but not closed under complementation.*

7 Conclusion

We have given a positive answer to the open question of the decidability of alternating general two-way XOR automata, in contrast to previous negative answers in case of other theories. We have given 3EXPTIME reduction of these automata to one-way XOR automata, thus also settling the expressiveness and closure properties of these automata. We have shown that emptiness of these automata is decidable in 2EXPTIME, meaning that secrecy of protocols modeled using these automata is decidable in 2EXPTIME. Emptiness test for alternating general two-way XOR automata is trivially EXPTIME-hard, but a more precise characterization of its complexity remains to be done. A key technique of independent interest is a pumping lemma for Branching Vector Plus Minimum Systems, allowing us to compute their coverability sets in EXPTIME.

Acknowledgments: I thank Helmut Seidl for many discussions and suggestions, and Jean Goubault-Larrecq as well as the anonymous referees for valuable comments.

References

1. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. Deciding the security of protocols with Diffie-Hellman exponentiation and products in exponents. In *FSTTCS'03*, pages 124–135. Springer-Verlag LNCS 2914, 2003.
2. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP decision procedure for protocol insecurity with XOR. In *LICS'03*, pages 261–270, 2003.
3. H. Comon and V. Cortier. Tree automata with one memory, set constraints and cryptographic protocols. *Theoretical Computer Science*, 2004. To appear.
4. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. <http://www.grappa.univ-lille3.fr/tata>, 1997.
5. H. Comon-Lundh and V. Cortier. New decidability results for fragments of first-order logic and application to cryptographic protocols. In *RTA'03*, pages 148–164. Springer-Verlag LNCS 2706, 2003.
6. V. Cortier. *Vérification Automatique des Protocoles Cryptographiques*. Ph.D. thesis, ENS Cachan, France, 2003.
7. F. Gézeg and M. Steinby. Tree languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, chapter 1, pages 1–68. Springer-Verlag, 1997.
8. J. Goubault-Larrecq. A method for automatic cryptographic protocol verification. In *FMPPTA'00*, pages 977–984. Springer-Verlag LNCS 1800, 2000.
9. J. Goubault-Larrecq. Une fois qu'on n'a pas trouvé de preuve, comment le faire comprendre à un assistant de preuve? In V. Ménessier-Morain, editor, *Actes des 12èmes Journées Francophones des Langages Applicatifs (JFLA'04)*. INRIA, collection didactique, 2004.
10. J. Goubault-Larrecq, M. Roger, and K. N. Verma. Abstraction and resolution modulo AC: How to verify Diffie-Hellman-like protocols automatically. *Journal of Logic and Algebraic Programming*, 2004. To Appear. Available as Research Report LSV-04-7, LSV, ENS Cachan.
11. J. Hopcroft and J. J. Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Theoretical Computer Science*, 8:135–159, 1979.
12. R. M. Karp and R. E. Miller. Parallel program schemata. *J. Computer and System Sciences*, 3(2):147–195, 1969.
13. D. Lugiez. Counting and equality constraints for multitree automata. In *FOSSACS'03*, pages 328–342. Springer-Verlag LNCS 2620, 2003.
14. D. Monniaux. Abstracting cryptographic protocols with tree automata. In *SAS'99*, pages 149–163. Springer-Verlag LNCS 1694, 1999.
15. H. Ohsaki. Beyond regularity: Equational tree automata for associative and commutative theories. In *CSL'01*, pages 539–553. Springer-Verlag LNCS 2142, 2001.
16. L. C. Paulson. Mechanized proofs for a recursive authentication protocol. In *CSFW'97*, pages 84–95. IEEE Computer Society Press, 1997.
17. P. Ryan and S. Schneider. An attack on a recursive authentication protocol: A cautionary tale. *Information Processing Letters*, 65(1):7–10, 1998.
18. G. Slutzki. Alternating tree automata. *Theoretical Computer science*, 41:305–318, 1985.
19. 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.
20. K. N. Verma. *Automates d'arbres bidirectionnels modulo théories équationnelles*. Ph.D. thesis, ENS Cachan, 2003.
21. K. N. Verma. On closure under complementation of equational tree automata for theories extending AC. In *LPAR'03*, pages 183–197. Springer-Verlag LNCS 2850, 2003.
22. 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.
23. K. N. Verma and J. Goubault-Larrecq. Karp-Miller trees for a branching extension of VASS. Research Report LSV-04-3, LSV, ENS Cachan, France, January 2004.