# THÈSE

présentée à l'École Normale Supérieure de Cachan

pour obtenir le grade de

**Docteur de l'École Normale Supérieure de Cachan**

par : Kumar Neeraj VERMA

Spécialité : INFORMATIQUE

# Automates d'arbres bidirectionnels modulo théories équationnelles

Soutenue le 30 septembre 2003 devant un jury composé de :

| | |
|---|---|
| Rémi GILLERON | examinateur |
| Jean GOUBAULT-LARRECQ | directeur de thèse |
| Denis LUGIEZ | rapporteur |
| Dale MILLER | président du jury |
| Michaël RUSINOWITCH | rapporteur |
| Helmut SEIDL | examinateur |

# Remerciements

First of all I am extremely grateful to Jean Goubault-Larrecq for having supervised my thesis and trained me as a researcher. With him I also had my introduction to research during internships at Dyade and INRIA. His interest in and knowledge of wide variety of subjects in computer science and mathematics has always surprised me and inspired me. Also his enthusiasm for speaking on these subjects, as well as various non-academic subjects have helped me learn a lot. I thank him for his patience and availability to help me and to answer my questions.

I thank Dale Miller for being president of my PhD jury. I thank Denis Lugiez for accepting to referee this thesis, as well as for discussions and for giving me new ideas through his work. I thank Michäel Rusinowitch for accepting to referee this thesis. Also thanks to Rémi Gilleron and Helmut Seidl for accepting to take part in my PhD jury.

Thanks to Hubert Comon for numerous discussions, ideas, and his enthusiasm for working on problems. I also acknowledge discussions with and feedback from Stéphane Demri, Alain Finkel, Laurent Fribourg, Jerôme Leroux, Slawek Lasota, Hitoshi Ohsaki, Ralf Treinen, and many others whose names don't appear here.

I thank all the members of LSV for providing a wonderful environment for work, for their jokes, discussions and for supporting me in various ways. Especially, I thank the PhD students and interns for an enjoyable stay at LSV. I also thank the administrative staff, Brigitte Van-Elsen and Catherine Forestier, for their patience with my numerous problems.

Finally I thank my parents and family for always supporting me in every way.

# Automates d'arbres bidirectionnels modulo théories équationnelles


## Résumé

Les automates d'arbres bidirectionnels étendent les automates classiques, en ce sens que les transitions peuvent non seulement construire mais aussi détruire des termes. Récemment des variantes équationnelles des automates d'arbres ont été proposés qui acceptent des termes modulo une théorie équationnelle. Dans cette thèse on étudie des automates d'arbres bidirectionnels modulo des théories équationnelles. On étudie les théories $\mathbb{AC}$ (associativité, commutativité de $+$), $\mathbb{ACU}$ (avec unité $0$), $\mathbb{ACUI}$ (avec idempotence $x + x = x$), $\mathbb{ACUX}$ (avec annulation $x + x = 0$) et $\mathbb{ACUM}$ (groupes abéliens). Ces automates sont importants pour la vérification des protocoles cryptographiques qui utilisent des primitives cryptographiques non parfaites.

On étudie la décidabilité et la clôture par opérations booléennes de ces automates. On commence par des résultats négatifs : le vide est indécidable pour les automates bidirectionnels, ou unidirectionnels alternants, modulo les théories $\mathbb{AC}$, $\mathbb{ACU}$, $\mathbb{ACUM}$.

On montre que les automates unidirectionnels modulo toutes ces théories sont clos par intersection, et que le vide est décidable. La clôture par union est triviale. A l'opposé, alors que les automates unidirectionnels modulo $\mathbb{AC}$, $\mathbb{ACU}$ sont clos par complémentation, ceux modulo $\mathbb{ACUX}$, $\mathbb{ACUM}$ et $\mathbb{ACUI}$ ne le sont pas. Les propriétés de clôture et de décidabilité s'étendent au cas bidirectionnel, pour chaque théorie sauf $\mathbb{ACUI}$, si on restreint le format des clauses push. On montre cela en réduisant les automates bidirectionnels aux automates unidirectionnels. (Le cas $\mathbb{ACUI}$ est ouvert.) Pour traiter certaines clauses ayant le symbole $+$ dans le corps, on développe une extension des systèmes d'addition de vecteurs avec états (VASS), appelés VASS étendus. On montre que la construction des arbres de Karp et Miller pour VASS peut être étendue pour les VASS étendus.

Ces résultats diffèrent nettement du cas des automates non équationnels qui ont toutes les propriétés de clôture et de décidabilité.

# Two-Way Equational Tree Automata


## Abstract

Two-way tree automata extend classical tree automata by allowing transitions that can destruct terms besides constructing them. Recently equational variants of tree automata have been proposed which accept terms modulo an equational theory. In this thesis we study two-way equational tree automata. We study the theories $\mathbb{AC}$ (associativity, commutativity of $+$), $\mathbb{ACU}$ (with unit $0$), $\mathbb{ACUI}$ (with idempotence $x + x = x$), $\mathbb{ACUX}$ (with cancellation $x + x = 0$) and $\mathbb{ACUM}$ (Abelian groups). These automata are useful in verification of cryptographic protocols which use non-perfect cryptographic primitives.

We study the properties of decidability and closure under Boolean operations of these automata. We start with negative results : emptiness is undecidable for alternating one-way automata, as well as for general two-way automata modulo the theories $\mathbb{AC}$, $\mathbb{ACU}$, $\mathbb{ACUM}$.

We show that the one-way automata modulo all these theories are closed under intersection, and emptiness is decidable. Closure under union is trivial. On the other hand, while the one-way automata modulo $\mathbb{AC}$, $\mathbb{ACU}$ are closed under complementation, those modulo $\mathbb{ACUX}$, $\mathbb{ACUM}$ and $\mathbb{ACUI}$ are not. These closure and decidability properties extend to the two-way case, for each theory except $\mathbb{ACUI}$, if we restrict the format of push clauses. We show this by reducing the two-way automata to one-way automata. (The $\mathbb{ACUI}$ case is open.) In order to deal with certain push clauses which have the $+$ symbol in the body, we develop an extension of Vector Addition Systems with States (VASS), called Extended VASS, in which transitions can add configurations. We show that the construction of Karp and Miller trees for VASS can be extended for Extended VASS.

Our results on one-way and two-way equational tree automata are strikingly different from the case of non-equational automata which have all the good closure and decidability properties.

# Table des matières

# Chapitre 1

# Introduction

Les automates d'arbres [CDG+97, GS97] sont un outil important en informatique. Diverses extensions des automates d'arbres ont été proposées en vue d'accroître leur expressivité. Une qui a été considérée très tôt est celle des *automates d'arbres bidirectionnels*, où les transitions peuvent non seulement construire des termes, mais aussi les détruire. Une autre consiste en les automates d'arbres *alternants* [Slu85], où l'on peut accepter non seulement des unions mais aussi des intersections d'ensembles de termes acceptés en certains états. Récemment on a vu des propositions de variantes *équationnelles* des automates d'arbres [Ohs01, Lug03], qui fonctionnent sur des termes modulo une théorie équationnelle. Dans cette thèse, nous étudions des combinaisons de ces aspects, c'est-à-dire des automates équationnels, bidirectionnels, et alternants. Ceci est motivé par des applications en vérification de protocoles cryptographiques.

Nous discutons brièvement dans ce chapitre des protocoles cryptographiques, d'automates d'arbres et extensions, puis nous exposons les contributions et le plan de cette thèse.

**Contributions** Dans cette thèse nous définissons et étudions la notion d'automates d'arbres équationnels unidirectionnels et bidirectionnels. Nous adoptons une approche de description des automates unidirectionnels et bidirectionnels en logique du premier ordre, et nous les étendons au cas des théories équationnelles. Nous nous concentrons sur les théories équationnelles d'associativité-commutativité et ses extensions. Plus précisément, nous traitons des théories $\mathbb{AC}$ (associativité et commutativité de $+$), $\mathbb{ACU}$ ($\mathbb{AC}$ avec unité 0), $\mathbb{ACUX}$ ($\mathbb{ACU}$ plus l'axiome $x + x = 0$, soit la théorie du ou exclusif), $\mathbb{ACUX}_n$ ($\mathbb{ACU}$ plus l'axiome $\underbrace{x + \ldots + x}_{n \text{ times}} = 0$, une généralisation du ou exclusif), $\mathbb{ACUM}$ ($\mathbb{ACU}$ plus l'axiome $x + (-x) = 0$, c'est-à-dire la théorie des groupes abéliens), $\mathbb{ACUD}$ ($\mathbb{ACU}$ plus les axiomes $-(x+y) = (-x)+(-y)$, $-(-x) = x$ et $-0 = 0$) et $\mathbb{ACUI}$ ($\mathbb{ACU}$ plus l'axiome $x+x = x$, c'est-à-dire la théorie de l'idempotence). La théorie $\mathbb{ACUD}$ est celle d'un symbole $-$ "distributif". Elle est impliquée par la théorie $\mathbb{ACUM}$, et est en fait strictement plus faible qu'$\mathbb{ACUM}$.

Pour illustrer l'utilisation de ces automates équationnels bidirectionnels, nous montrons que nos automates d'arbres modulo la théorie $\mathbb{ACU}$ et la théorie des groupes abéliens ($\mathbb{ACUM}$) peuvent être mises à profit pour modéliser des protocoles fondés sur l'exponentiation modulaire, notamment le protocole de Diffie-Hellman en groupe, utilisé par un groupe de participants pour se mettre d'accord sur une clé. On rappelle que nous traitons aussi de la théorie du ou exclusif qui est utilisé fréquemment dans les protocoles cryptographiques.

Le cœur de cette thèse porte sur l´etude des propriétés algorithmiques des automates unidirectionnels et bidirectionnels modulo les théories mentionnées ci-dessus. Spécifiquement, nous étudions la décidabilité de la vacuité de tels automates, et s'il est possible de calculer les intersections et les complémentaires de tels automates sous forme d'automates de la même sorte, et si les automates bidirectionnels peuvent se réduire, et même effectivement, à des automates unidirectionnels reconnaissant les mêmes langages. Pour toutes les classes d'automates que nous étudions, la décidabilité du vide et la clôture par intersection impliquent trivialement la décidabilité de l'appartenance.

Nous commençons par les résultats négatifs. Dans toutes les théories $\mathbb{AC}$, $\mathbb{ACU}$, $\mathbb{ACUM}$, il est déjà indécidable de tester si le langage accepté par un automate unidirectionnel alternant donné est vide. Ceci a comme conséquence que divers formats d'automates bidirectionnels (non alternants) ont aussi un problème du vide indécidable modulo ces théories ; à savoir celles qui sont capables de coder l'alternance. Notons que ceci contraste avec le cas non équationnel, ou l'alternance et la bidirectionnalité sont essentiellement bénignes. En effet, tout ensemble récursivement énumérable est le langage d'un automate alternant modulo $\mathbb{AC}$, $\mathbb{ACU}$, ou $\mathbb{ACUM}$ ; alors que les langages d'automates unidirectionnels et non alternants modulo les mêmes théories ne reconnaissent que les clôtures équationnelles de langages rationnels.

En ce qui concerne les résultats positifs, nous montrons d'abord que le vide du langage accepté par un automate unidirectionnel équationnel est décidable. Ce résultat est vrai pour toute théorie, y compris celles dont on ne traite pas dans cette thèse. En termes de propriétés de clôture, nous montrons que les automates unidirectionnels sont clos par union et intersection dans toutes les théories ci-dessus. Alors que la clôture par union est immédiate, la clôture par intersection se fonde sur des procédures que l'on peut voir comme des constructions de produits améliorées, et utilisant intensément les correspondances entre certaines classes d'automates associatifs-commutatifs et les ensembles semi-linéaires ou définissables dans l'arithmétique de Presburger. Bien que les automates d'arbres unidirectionnels modulo toutes ces théories sont clos par intersection, la situation est complètement différente dans le cas de la complémentation. Nous montrons que, alors que les automates d'arbres équationnels modulo les théories $\mathbb{AC}$, $\mathbb{ACU}$ et $\mathbb{ACUD}$ sont clos par complémentation, ceux modulo les théories $\mathbb{ACUX}$, $\mathbb{ACUX}_n$, $\mathbb{ACUM}$ et $\mathbb{ACUI}$ ne le sont pas. Nous donnons des contre-exemples dans ces cas.

En ce qui concerne les automates bidirectionnels, nous montrons que dans toutes les théories mentionnées ci-dessus sauf $\mathbb{ACUI}$, les automates bidirectionnels peuvent se réduire effectivement à des automates unidirectionnels modulo la même théorie, à condition que certaines précautions soient prises dans la définition des clauses dites "push", de sorte à éviter les cas indécidables mentionnés plus haut. (Le cas $\mathbb{ACUI}$ est ouvert.) En conséquence, les résultats de clôture par opérations booléennes des automates unidirectionnels se généralisent à ces automates bidirectionnels. De plus le problème du vide de ces automates bidirectionnels est décidable.

Les clauses push considérées dans ces automates bidirectionnels ne font apparaître que des symboles de fonction non équationnels (c'est-à-dire ne contenant pas le symbole $+$). Dans la suite nous étudions les automates avec clauses push contenant $+$. Pour les traiter, nous devons d'abord définir une extension des *systèmes d'addition de vecteurs à états* (VASS) [Reu89], que nous appelons *VASS étendus (EVASS)*. Nous montrons que les *arbres de Karp-Miller* définis de sorte à calculer les limites de configurations accessibles des VASS peut se généraliser aux EVASS. Grâce à des traductions des automates $\mathbb{AC}$ en EVASS, nous sommes capables de montrer que les automates que l'on obtient en ajoutant des *clauses $+$-push standard* aux automates $\mathbb{AC}$ unidirectionnels et bidirectionnels se réduisent effectivement aux automates $\mathbb{AC}$ unidirectionnels. Cependant comme la construction de Karp et Miller (même pour les VASS) n'est pas primitive récursive, ceci ne nous donne pas un algorithme exponentiel. À l'opposé nous montrons que dans le cas $\mathbb{ACU}$ (et non $\mathbb{AC}$), il n'est pas besoin de passer par les EVASS, et nous donnons des réductions étonnamment simples des automates obtenus par l'ajout de clauses $+$-push standard aux automates $\mathbb{ACU}$ unidirectionnels et bidirectionnels vers les automates $\mathbb{ACU}$ unidirectionnels. En ce qui concerne les *clauses $+$-push*, qui sont strictement plus expressives que les clauses $+$-push standard, et bien que ces clauses puissent être trivialement éliminées des automates $\mathbb{ACUX}$ et $\mathbb{ACUM}$, le test du vide pour les automates contenant ces clauses modulo $\mathbb{ACU}$ est au moins aussi difficile que le problème de l'accessibilité des VASS ou des réseaux de Petri, qui est un problème bien connu pour être difficile. Nous montrons que le problème du vide de l'intersection pour cette classe d'automates se réduit au problème du vide pour la même classe d'automates, ce qui donne une indication du pouvoir expressif des clauses $+$-push. Ceci suggère que les automates $\mathbb{AC}$ et $\mathbb{ACU}$ avec clauses $+$-push sont difficiles à traiter, et la question de leur décidabilité est aujourd'hui ouverte.

Tree automata [CDG+97, GS97] are an important tool in computer science. Various extensions of tree automata have been proposed with a view to increase their expressiveness. One that has been considered very early is that of *two-way tree automata* where transitions may not only construct terms, but also destruct them. Another one is *alternating* tree automata [Slu85] where we may accept not just unions but also intersections of sets of terms accepted at some states. Recently there have been proposals for *equational* variants of tree automata [Ohs01, Lug03] which work with terms modulo an equational theory. In this thesis, we study combinations of these features, that is, of equational, two-way, and alternating tree automata. This is motivated by applications in verification of cryptographic protocols.

## 1.1 Cryptographic Protocols and Tree Automata

Security of computer systems is an issue of major concern today. In particular study of cryptography and of cryptographic protocols has received considerable attention in recent years. They are being increasingly used today notably because of the growth of electronic commerce, where secrets need to be preserved or authenticity of messages need to be established.

Cryptography refers to encryption and decryption algorithms as well as related algorithmic techniques. Such algorithms allow us e.g. to encrypt a given message with a given key so that the original message can be recovered from the encrypted message only by someone who knows the inverse key.

On the other hand cryptographic protocols are rules for exchanging messages, using cryptographic algorithms as black boxes. It is well known that even provably secure encryption and decryption algorithms are not enough to ensure sufficient level of security. Despite the ingenuity of their designers, most cryptographic protocols are found to be faulty later on. Further, the attacks found against these faulty protocols are of a purely logical nature, i.e. these attacks can be described independently of the actual cryptographic means used, by showing how an intruder can replay old messages, forge new messages, gets keys and use them to decrypt messages, etc. Such attacks can have serious economic consequences. These protocols generally have descriptions short enough to make us believe that an informal analysis of these protocols is enough to be assured of their correctness. However in practice the flaws in these protocols are often some silly errors which escape the attention of their designers. For these reasons formal verification of cryptographic protocols is needed to be assured of their security properties.

Several methods exist for evaluating security of cryptographic protocols, including model checking methods [Low95, Low96, MCJ97, Mea92, Mea96], computer assisted proof development [Bol96, Pau98], logics of belief [Bie90, BAN89], rewriting [GK99], process algebras [AG97], tree automata [Mon99, GL00, CCM01] to name a few. In this thesis, we pursue the idea of using tree automata to model cryptographic protocols. In this approach terms are used to represent messages. Then (an upper approximation of) the set of messages known to an intruder is expressed as the language accepted by a tree automaton. We then check that this automaton does not accept any message that is intended to remain secret, in other words the intruder does not know any message intended to be a secret.

This approach is based on the assumption of *perfect cryptographic primitives*. Representing the encryption of message $m$ by key $k$ as $\{m\}_k$, this assumption means that the messages $\{m\}_k$ and $\{m'\}_{k'}$ cannot be confounded if $m \neq m'$ or $k \neq k'$, $\{\ldots\{\{m\}_{k_1}\}_{k_2}\ldots\}_{k_n} \neq m$, $\{m\}_k \neq (m_1, m_2)$ (where $(m_1, m_2)$ represents the pair of messages $m_1$ and $m_2$), etc. Under this assumption, the set of messages becomes a free term algebra : the term $\texttt{encrypt}(m, k)$ is used to represent the encryption of $m$ with $k$, the term $\texttt{pair}(m_1, m_2)$ is used to represent the pairing of two messages $m_1$ and $m_2$.

## 1.2 Tree Automata and its Extensions

Under these assumptions, there is a natural way to represent the knowledge of an intruder in a protocol using transitions of tree automata [CDG+97, GS97]. In this thesis we use a first order logic approach to representing tree automata. In such a approach, definite clauses of first order logic are used to represent transitions of an automaton. The predicates of logic represent states of automata. An atom of the form $P(t)$ means that term $t$ is accepted at state $P$. For example, the clause

$$P(f(x_1,\ldots,x_n)) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n)$$

can be read as "if terms $x_1,\ldots,x_n$ are accepted at states $P_1,\ldots,P_n$ then term $f(x_1,\ldots,x_n))$ is accepted at state $P$". The clause

$$P(x) \Leftarrow Q(x)$$

can be read as "if $x$ is accepted at $Q$ then $x$ is accepted at $P$". These two forms of clauses correspond to the clauses of the classical tree automata usually described in the literature [CDG+97]. They are usually written as rewrite rules :

$$f(P_1,\ldots,P_n) \to P$$

and

$$Q \to P$$

The automata containing the above two kinds of clauses are called *one-way automata* in this thesis, to distinguish them from *two-way automata* which are their extensions.

Returning to our modeling of cryptographic protocols, we now use state (predicate) $I_C$ to accept the set of messages (terms) known to an intruder in configuration $C$. Then the following clauses represent some deductive abilities of the intruder.

| | |
|---|---|
| $I_C(\texttt{encrypt}(m,k)) \Leftarrow I_C(m) \wedge I_C(k)$ | Intruder can encrypt messages |
| $I_C(\texttt{pair}(x,y)) \Leftarrow I_C(x) \wedge I_C(y)$ | Intruder can form pairs |
| $I_{C_{new}}(x) \Leftarrow I_{C_{old}}(x)$ | Intruder remembers past messages |

For example the first clause says that if the intruder knows message $m$ and key $k$ in configuration $C$ then the intruder knows the encryption of $m$ by $k$. In other words this clause says that the intruder has the power to encrypt messages. We can see that these clauses are clauses of one-way automata.

Having represented a protocol using tree automata, we are then interested in knowing whether a particular set of messages is known to the intruder. Such questions can then be thought of as questions about the emptiness or emptiness of intersections of languages accepted by tree automata.

### 1.2.1 Two-Way Automata

However the clauses of one-way automata described above are too limited to express all the deductive abilities of an intruder. For example we need to state the following properties :

| | |
|---|---|
| $I_C(m) \Leftarrow I_C(\texttt{encrypt}(m,k)) \wedge I_C(k)$ | Intruder can decrypt messages |
| $I_C(x) \Leftarrow I_C(\texttt{pair}(x,y))$ | Intruder can unpair messages |
| $I_C(y) \Leftarrow I_C(\texttt{pair}(x,y))$ | |

The first clause says that if the intruder knows an encrypted message, and he also knows the corresponding key then he knows the original (unencrypted) message (because the intruder can decrypt messages). Clearly these clauses are not in the format of one-way automata clauses. Intuitively, the difference is that these clauses allow us to destruct terms, whereas the clauses of one-way automata allow us to construct terms. For these reasons, we extend one-way automata by adding following form of clauses :

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

where $1 \leq i, i_1, \ldots, i_k \leq n$. These clauses are called *general push clauses* and the extended form of automata are called *general two-way automata*.

Sometimes we also need the following kinds of clauses

$$P(x) \Leftarrow P_1(x) \wedge P_2(x)$$

called *intersection clauses*. Automata containing such clauses are called *alternating automata*. This clause can be read as "if $x$ is accepted at both $P_1$ and $P_2$ then $x$ is accepted at $P$".

### 1.2.2 Equational Tree Automata

Now we come to the most important extension that is treated in this thesis : the extension of tree automata by adding equational theories. These extensions are motivated by the fact that in actual practice, protocols often don't satisfy the perfect cryptographic primitive assumption mentioned above. As an example the original version of the Bull's recursive authentication protocol presented in [Pau97] was formally proved correct using the assumption of perfect cryptographic primitives. However this protocol uses the exclusive-or operation for encryption. By taking into account the properties of the exclusive-or operation like associativity, commutativity, cancellation, an attack against this protocol was found in [RS98]. Also protocols like the group Diffie-Hellman protocol [STW00], use algebraic properties of modular exponentiation [DH76]. For these reasons we need to take into account the algebraic properties of cryptographic primitives in the formal analysis of protocols. These properties are conveniently expressed using a set of equations. Some of the theories that often occur are the theories of an associative and commutative symbol with a unit, the theory of Abelian groups, and the theory of exclusive-or.

This motivates our extensions of (one-way and two-way) tree automata to (one-way and two-way) *equational tree automata*, which work with terms modulo an equational theory. In this thesis we define and study the decidability and closure properties of one-way and two-way equational tree automata for a certain number of theories. Note that for the purposes of cryptographic protocols, the theory of associativity and commutativity occurs most often. Theories like that of Abelian groups or of exclusive-or operator are obtained from the theory of exclusive-or by adding some additional axioms. For this reason, in this thesis, we have chosen to deal with the theories of associativity-commutativity (with or without unit), and theories obtained by certain additional axioms to the theory of associativity-commutativity. In particular we also deal with the theory of exclusive-or and the theory of Abelian groups.

There has recently been considerable work on dealing with equational theories in cryptographic protocols. The theory of exclusive-or has been dealt with in [Cor03, CLC03]. Their approach is very similar to ours in that they study satisfiability of horn clauses in the presence of the theory of exclusive-or. The class of clauses they consider is incomparable to the classes we deal with. Our approach is also more automata theoretic, in that we deal with the closure properties of the automata defined by our clauses, and also our clauses have been chosen specifically to define one-way and two-way tree automata. [CLS03] deals with protocols in the presence of the theories of exclusive-or and Abelian groups, although they use constraint solving instead of Horn Clauses. An equational unification algorithm for protocols involving modular exponentiation is studied in [KNW03]. [CKRT03] presents an NP-decision procedure for deciding insecurity of protocols in the presence of an exclusive-or operator.

## 1.3 Contributions of the Thesis

In this thesis we define and study the notion of one-way and two-way equational tree automata. We take a first-order logic approach to describing one-way and two-way tree automata, and extend them to deal with equational theories. We focus on the equational theories of associativity-commutativity and its extensions. More precisely, we deal with the theories $\mathbb{AC}$ (associativity and commutativity of $+$), $\mathbb{ACU}$ ($\mathbb{AC}$ with unit 0), $\mathbb{ACUX}$ ($\mathbb{ACU}$ with the axiom $x + x = 0$, i.e. the theory of exclusive-or), $\mathbb{ACUX}_n$ ($\mathbb{ACU}$ with the axiom $\underbrace{x + \ldots + x}_{n \text{ times}} = 0$, generalization of exclusive-or), $\mathbb{ACUM}$ ($\mathbb{ACU}$ with the axiom $x + (-x) = 0$, i.e. the theory of Abelian groups), $\mathbb{ACUD}$ ($\mathbb{ACU}$ with the axioms $-(x + y) = (-x) + (-y)$, $-(-x) = x$ and $-0 = 0$) and $\mathbb{ACUI}$ ($\mathbb{ACU}$ with the axioms $x + x = x$, i.e. the theory of idempotence). The theory $\mathbb{ACUD}$ is the theory of a "distributive" $-$ symbol. It is implied by the theory $\mathbb{ACUM}$, and is actually strictly weaker than $\mathbb{ACUM}$.

To illustrate the use of these equational two-way automata, we show that our tree automata modulo the theory $\mathbb{ACU}$ and the theory of Abelian groups ($\mathbb{ACUM}$) can be profitably used to model protocols based on modular exponentiation, e.g. the group Diffie-Hellman protocol, used by a group of participants to agree on a common key. Recall that we also deal with the theory of exclusive-or, which is used frequently in cryptographic protocols.

The core of this thesis is concerned with the study of the algorithmic properties of one-way and two-way tree automata modulo the theories listed above. Specifically, we study the decidability of emptiness of such automata, and whether it is possible to compute intersections and complements of such automata as automata of the same kind, and whether two-way automata can be reduced, and if so effectively, to one-way automata recognizing the same languages. For all classes of automata that we study, decidability of emptiness and closure under intersection trivially implies decidability of membership.

We start with negative results. In all theories $\mathbb{AC}$, $\mathbb{ACU}$, $\mathbb{ACUM}$, it is already undecidable to test whether the language accepted by a given alternating one-way automaton is empty. This has the consequence that various formats of two-way (non-alternating) automata also have an undecidable emptiness problem modulo these theories ; namely the ones that are able to encode alternation. Note that this is in contrast with the non-equational case, where alternation and two-way-ness are essentially harmless. Indeed, every recursively enumerable set is the language of some alternating automaton modulo $\mathbb{AC}$, $\mathbb{ACU}$, or $\mathbb{ACUM}$ ; while languages of one-way, non-alternating automata modulo the same theories only recognize equational closures of regular languages.

As far as the positive results are concerned, we first show that the emptiness of the language accepted by a one-way equational tree automaton is decidable. This result holds for an arbitrary theory, including those not dealt with in this thesis. Coming to closure properties, we show that the one-way automata for all the theories above are closed under union and intersection. While closure under union is immediate, closure under intersection uses procedures that can be thought of as souped-up product constructions, with a lot of help from connections between some classes of associative-commutative automata and semilinear sets or Presburger-definable sets. While the one-way tree automata modulo all these theories are closed under intersection, the situation is strikingly different in the case of complementation. We show that while the one-way tree automata modulo the theories $\mathbb{AC}$, $\mathbb{ACU}$ and $\mathbb{ACUD}$ are closed under complementation, those modulo the theories $\mathbb{ACUX}$, $\mathbb{ACUX}_n$, $\mathbb{ACUM}$ and $\mathbb{ACUI}$ are not closed under complementation. We give counter-examples for the latter cases.

As far as two-way automata are concerned. we show that in all theories mentioned above except $\mathbb{ACUI}$, two-way automata can be effectively reduced to one-way automata modulo the same theory, provided some care is taken in the definition of the so-called push clauses in order to avoid the undecidable cases mentioned above. (The $\mathbb{ACUI}$ case is open.) As a result, the results about closure under Boolean operations of one-way automata also generalize to these two-way automata. Also the emptiness problem of these two-way automata is decidable.

The push clauses considered in these two-way automata involved only non-equational functional symbols (i.e. they didn't contain the symbol $+$). Next we study the automata with push clauses involving $+$. To deal with them we first need to define an extension of the *Vector Addition Systems with States (VASS)* [Reu89] which we call *Extended VASS (EVASS)*. We show that the *Karp-Miller Trees* defined to compute limits of reachable configurations of VASS, can be generalized for the EVASS. By translations from $\mathbb{AC}$ automata to EVASS, we are able to show that the automata obtained by adding *standard $+$-push clauses* to one-way and two-way $\mathbb{AC}$ automata can be effectively reduced to one-way $\mathbb{AC}$ automata. However since the Karp-Miller construction (even for VASS) is not primitive-recursive, this does not give us an exponential algorithm. In contrast we show that in the $\mathbb{ACU}$ (and not $\mathbb{AC}$) case, we do not need to go through EVASS, and we give surprisingly easy reductions of the automata obtained by adding the standard $+$-push clauses to one-way and two-way $\mathbb{ACU}$ automata, to one-way $\mathbb{ACU}$ automata. As far as *$+$-push clauses* are concerned, which are strictly more expressive than standard $+$-push clauses, while these clauses can be trivially eliminated from $\mathbb{ACUX}$ and $\mathbb{ACUM}$ automata, the emptiness test for automata containing these clauses modulo $\mathbb{ACU}$ is more difficult than the reachability problem for VASS or Petri Nets, which are known to be difficult. We also show that the intersection emptiness problem for this class of automata reduces to emptiness problem for the same class of automata, indicating the power of $+$-push clauses. This suggests that $\mathbb{AC}$ and $\mathbb{ACU}$ automata with $+$-push clauses are difficult to deal with, and the decidability question for them is currently open.

## 1.4 Plan of the Thesis

This thesis is divided into four parts.

In the first part we present equational tree automata and applications to cryptographic protocols. We start in Chapter 2 by having a quick look at first order logic, tree automata and equational logic and their interconnections. In particular we see how various classes of tree automata can be considered as logic programs, and how decision problems about automata can be reduced to the satisfiability problem in first order logic. We also see how to deal with logic programs with equality, since we are interested in studying equational variants of tree automata. In Chapter 3 we deal more specifically with one-way and two-way equational tree automata. We look at the various associative-commutative theories we are going to study and also look at some basic properties of these theories used later. We also look at the various kinds of clauses which we use to define various classes of automata. We end this chapter with a discussion of other formalisms similar to our equational tree automata, and we argue why the work presented in this thesis is original compared to already existing results. Since in most of this thesis we will be dealing with the $\mathbb{AC}$ theory and its extensions, it is also natural to ask about the properties of equational tree automata modulo the restrictions of the theories $\mathbb{AC}$, namely the theories $\mathbb{A}$ and $\mathbb{C}$. So we start by dealing with these two cases in Chapter 4. We show that languages accept by $\mathbb{C}$ tree automata are merely regular tree languages. On the other hand $\mathbb{A}$ tree automata can accept context free languages, implying undecidability of intersection emptiness. Then we turn our attention to $\mathbb{AC}$ theories. To motivate our study of two-way equational tree automata for $\mathbb{AC}$-like theories, in Chapter 5, we give an application of our equational tree automata in modeling group Diffie Hellman protocols. In particular it should be observed that we model these protocols using subclasses of our equational tree automata which are later shown to be decidable.

In the second part (Chapter 6), we present the undecidability results. We show that either alternation clauses or general push clauses are sufficient to produce undecidability. This is shown for the theories $\mathbb{ACU}$, $\mathbb{AC}$, $\mathbb{ACUD}$ and $\mathbb{ACUM}$. These results lead us to study suitable restricted versions of general two-way automata, with the goal of obtaining decidability results, in Chapters 10 and 11.

Before that we first deal with one-way equational tree automata in the third part. We start in Chapter 7 by developing some basic tools which shall be very helpful throughout the rest of the thesis. Among others we show that emptiness of one-way equational tree automata is decidable for any theory. We prove some results about the runs of equational tree automata. Most importantly we show that modulo some of our theories, one-way automata on signatures in which all non-equational symbols are constants, accept exactly semilinear sets or Presburger-definable sets, upto some encoding. The results in this Chapter are very crucial, and all the later results in the thesis are based on them.

In Chapter 8 we study the closure under intersection of one-way equational tree automata. We show that modulo each of the $\mathbb{AC}$ theories (namely the theories $\mathbb{AC}$, $\mathbb{ACU}$, $\mathbb{ACUX}$, $\mathbb{ACUX}_n$,$\mathbb{ACUD}$, $\mathbb{ACUM}$, $\mathbb{ACUI}$) the one-way equational tree automata are closed under intersection. These in particular imply decidability of intersection emptiness and membership test. As is the case for the results later in the thesis, these results depend strongly on the results of Chapter 7. Next we deal with closure under complementation of one-way equational tree automata in Chapter 9. We see that the results on complementation are strikingly different from those on complementation. While the one-way tree automata modulo the theories $\mathbb{AC}$, $\mathbb{ACU}$ and $\mathbb{ACUD}$ are closed under complementation, those modulo the theories $\mathbb{ACUX}$, $\mathbb{ACUX}_n$, $\mathbb{ACUM}$ and $\mathbb{ACUI}$ are not closed under complementation (we give counter-examples for the latter theories).

Having dealt with the one-way equational tree automata, we now study the two-way variants of equational tree automata in the fourth part. We already show in Chapter 6 that emptiness of general two-way equational tree automata is undecidable. Hence to obtain decidability results, we need to consider restricted versions of these automata. In Chapter 10 we study the so-called two-way equational tree automata which are obtained by adding *free push clauses* to one-way equational tree automata. Also observe that the clauses used in the modeling of the group Diffie-Hellman protocol protocol described in Chapter 5 belong to this class. We show that for all our $\mathbb{AC}$ theories other than $\mathbb{ACUI}$, these two-way equational tree automata can be effectively reduced to equivalent one-way equational tree automata. (The two-way $\mathbb{ACUI}$ case is left open.) We then study automata containing some additional clauses in Chapter 11. We show $+$-push clauses can be added to $\mathbb{ACUX}$ and $\mathbb{ACUM}$ automata without increasing their expressiveness. On the other hand, these $+$-push clauses turn out to be difficult for $\mathbb{AC}$ and $\mathbb{ACU}$ automata, and decidability question for these cases is left open. For the $\mathbb{AC}$ and $\mathbb{ACU}$ cases, we introduce a further weaker form of $+$-push clauses, called standard $+$-push clauses. We show that standard $+$-push clause do not increase the expressiveness of two-way $\mathbb{AC}$ and $\mathbb{ACU}$ tree automata. While this result is easy for the $\mathbb{ACU}$ case, for the $\mathbb{AC}$ case, we need to take a detour through an extended notion of VASS. For this reason, we also define and study in this chapter Extended VASS (EVASS), and show a generalization of the Karp-Miller tree for VASS to extended VASS.

Chapters 8 and 10 are described in [Ver03b]. Chapter 9 is described in [Ver03a]. Chapter 11 is described in [GLV]. Chapters 6 and 7 are described in [Ver03b] and [GLV]. [GLV] also describes how techniques of refinements of the resolution procedure for satisfiability checking in first order logic [GLM97] can be used to obtain decision procedures for the extensions of two-way $\mathbb{AC}$ tree automata with standard $+$-push clauses. These techniques have not been described in this thesis, however we have shown the results of [GLV] in this thesis using alternative techniques. Readers interested in decision procedures using resolution for two-way equational tree automata may refer to that paper.

# Première partie

# Automates d'arbres équationnels et une application aux protocoles cryptographiques (Equational Tree Automata and Application to Cryptographic Protocols)

# Chapitre 2

# Logique du premier ordre, automates d'arbres et théories équationnelles (First Order Logic, Tree Automata and Equational Theories)

Nous rappelons ici les notions de base de termes du premier ordre, les sémantiques de Tarski et de Herbrand de la logique du premier ordre, et la notion de plus petit modèle de Herbrand d'un ensemble de clauses de Horn. Nous opérons aussi un rappel des notions de base concernant les systèmes de réécriture. Ceci fait, nous passons aux automates d'arbres — les automates d'arbres classiques, c'est-à-dire unidirectionnels, non alternants, non équationnels — et rappelons leur définition en terme de systèmes de réécriture. Nous rappelons que ces automates définissent une famille de langages close par union, intersection et complémentaires et telle que l'appartenance à un tel langage ou la vacuité d'un tel langage est décidable. Il sera intéressant dans la suite de la thèse, ne serait-ce que pour définir la notion même d'automate d'arbres bidirectionnel, de redéfinir les automates d'arbres comme des ensembles de clauses de Horn d'une forme spécifique, ce que nous faisons. Nous profitons de cette nouvelle présentation pour définir non seulement les automates d'arbres bidirectionnels, mais aussi alternants, et nous montrons que l'on peut effectivement réduire ces derniers automates à des automates d'arbres ordinaires acceptant les mêmes langages. Enfin nous définissons les modèles de Herbrand équationnels, ce qui nous servira à définir les automates équationnels dans le chapitre suivant.

## 2.1 Terms

A *signature* $\Sigma$ is a finite set of *function symbols* each having a non-negative *arity* such that $\Sigma$ has at least one zero-ary symbol. The assumption of $\Sigma$ containing at least one zero-ary symbol is usually not made in the definition of a signature, and is made only for certain results. However we make this assumption at the beginning as it doesn't cause any significant loss of generality.

Given a signature $\Sigma$ and a set $X$ of *variables*, the set $T(\Sigma, X)$ of *terms* built over $\Sigma$ and $X$ is defined inductively by the following two rules :

- If $x \in X$ then $x \in T(\Sigma, X)$.
- If $f \in \Sigma$ has arity $n$, and $t_1, \ldots, t_n \in T(\Sigma, X)$ then $f(t_1, \ldots, t_n) \in T(\Sigma, X)$.

The set of *ground terms* is $T(\Sigma) = T(\Sigma, \emptyset)$. Since $\Sigma$ has at least one zero-ary symbol, $T(\Sigma, X)$ and $T(\Sigma)$ are guaranteed to be non-empty.

**Example 1** *Let signature $\Sigma = \{O, S, +, \times\}$. We let $O$ to be zero-ary, $S$ to be unary and $+$ and $\times$ to be binary. Some examples of terms on this signature are $O$, $S(O)$, $S(S(S(O)))$ and $S(\times(+(S(S(O))), O), S(O))$.*

It is also natural to think of these terms as trees (which justifies the name "tree automata" instead of "term automata" ; the "tree automata" are supposed to accept terms on some signature $\Sigma$). For example the term $S(\times(+(S(S(O))), O), S(O))$ of Example 1 can be written as in Figure 2.1. For any set $S$ we denote by $S^*$ the set of all finite sequences of elements of $S$, which are also called the strings on $S$. The empty string is denoted by $\epsilon$. Given strings $x$ and $y$, we write $x \leq_{pref} y$ iff $x \cdot z = y$ for some string $z$. We write $x <_{pref} y$ iff $x \leq_{pref} y$ and $x \neq y$. Hence we can describe the positions of the various nodes in a tree using strings from $\mathbb{N}^*$. The position of the root node is $\epsilon$. If the position of a node is the string $p$, and the node has $k$ children, then the position of the children nodes are $p \cdot 1, \ldots, p \cdot k$. Hence a tree can be thought of as a function form a finite set of positions to $\Sigma$. (Remember however that not all these functions represent terms on $\Sigma$.) If $t$ is a term and $p$ is a position in the term, then the subterm rooted at position $p$ is denoted by $t|_p$. If $s$ is another term then $t[s]_p$ denotes the replacement of the subterm at position $p$ in term $t$ by term $s$.

## 2.2 First Order Logic

A *first order logic* consists of a countably infinite set $X$ of variables, a signature $\Sigma$, and a finite set $\mathbb{P}$ of *predicates* each of which has a fixed arity. The terms of the logic are the elements of $T(\Sigma, X)$. The *atoms* are of the form $P(t_1, ..., t_n)$ where $P$ is a predicate of arity $n$. The *formulas* are of the form $A, \phi_1 \vee \phi_2, \phi_1 \wedge \phi_2, \phi_1 \Rightarrow \phi_2, \phi_1 \Leftrightarrow \phi_2, \neg\phi_1, \forall x \cdot \phi_1, \exists x \cdot \phi_1$ where $A$ is an atom, $x$ is a variable, and $\phi_1, \phi_2$ are formulas. A *Tarskian interpretation* (or simply *interpretation*) $I$ is a tuple $(D, (f_I)_{f \in \Sigma}, (P_I)_{P \in \mathbb{P}})$ such that $D$ is a non-empty set, $f_I : D^n \to D$ for each $f \in \Sigma$ of arity $n$, and $P_I \subseteq D^n$ for each predicate $P$ of arity $n$. $D$ will also be called the *domain* of the interpretation. If there is no confusion then the interpretation will be denoted by $D$, the functions will be denoted by $f_D$ for $f \in \Sigma$, and the predicates will be denoted by $P_D$ for $P \in \mathbb{P}$.

A *$D$-assignment* is a function $\rho : X \to D$. Given an interpretation $I$ on domain $D$, and a $D$ assignment $\rho$, the semantics $[\![t]\!] I\rho \in D$ and $[\![\phi]\!] I\rho \in Bool = \{tt, ff\}$ of term $t$ and formula $\phi$ respectively is recursively defined as follows, where $\vee_{Bool}, \wedge_{Bool}, \Rightarrow_{Bool}, \Leftrightarrow_{Bool}$ are the usual logical operations on the booleans.

- $[\![x]\!] I\rho = \rho(x)$
- $[\![f(t_1, \ldots, t_n)]\!] I\rho = f_I([\![t_1]\!] I\rho, \ldots, [\![t_n]\!] I\rho)$

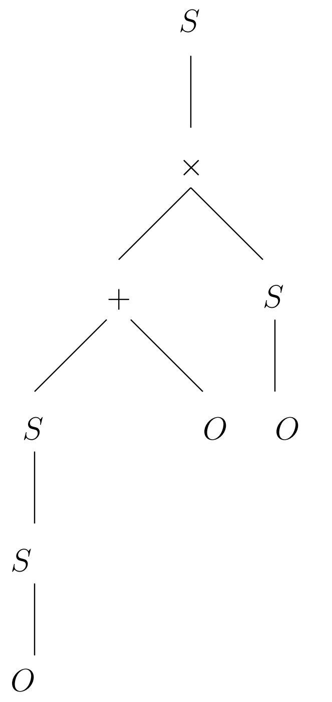


FIG. 2.1 – The term $S(\times(+(S(S(O))), O), S(O))$ as a tree

- $[\![P(t_1, \ldots, t_n)]\!]I\rho = tt$ iff $([\![t_1]\!]I\rho, \ldots, [\![t_n]\!]I\rho) \in P_I$.
- $[\![\neg\phi]\!]I\rho = \neg_{Bool}[\![\phi]\!]I\rho$.
- $[\![\phi_1 \vee \phi_2]\!]I\rho = [\![\phi_1]\!]I\rho \vee_{Bool} [\![\phi_2]\!]I\rho$.
- $[\![\phi_1 \wedge \phi_2]\!]I\rho = [\![\phi_1]\!]I\rho \wedge_{Bool} [\![\phi_2]\!]I\rho$.
- $[\![\phi_1 \Rightarrow \phi_2]\!]I\rho = [\![\phi_1]\!]I\rho \Rightarrow_{Bool} [\![\phi_2]\!]I\rho$.
- $[\![\phi_1 \Leftrightarrow \phi_2]\!]I\rho = [\![\phi_1]\!]I\rho \Leftrightarrow_{Bool} [\![\phi_2]\!]I\rho$.
- $[\![\forall x \cdot \phi]\!]I\rho = tt$ iff $[\![\phi]\!]I(\rho[x \mapsto d]) = tt$ for all $d \in D$.
- $[\![\exists x \cdot \phi]\!]I\rho = tt$ iff $[\![\phi]\!]I(\rho[x \mapsto d]) = tt$ for some $d \in D$.

$I$ is a *model* of $\phi$ iff for every $\rho$, $[\![\phi]\!]I\rho = tt$. If $S$ is a set of formulas then $I$ is a *model* of $S$ iff $I$ is a model of every $\phi \in S$. These are written as $I \models \phi$ and $I \models S$. If $S$ is a formula or a set of formulas, and $S'$ is a formula or a set of formulas, then $S'$ is a *semantic consequence* of $S$, written $S \models S'$, iff for every $I$ such that $I \models S$ we have $I \models S'$. We write $\models S$ iff $\emptyset \models S$. ($\emptyset$ is the empty set.) $S$ is said to be *satisfiable* iff it has some model, otherwise it is said to be unsatisfiable. We use the symbol $\perp$ to denote some formula of the form $\phi \wedge \neg\phi$. (The choice of $\phi$ will be irrelevant for our discussion.) Observe that $[\![\perp]\!]I\rho = ff$ for every $I$ and every $\rho$. Hence $\perp$ represents the "false" formula. (Alternatively we could have chosen to have a special symbol for a "false" formula in our syntax.)

A *Herbrand interpretation* is an interpretation $(D, (f_D)_{f\in\Sigma}, (P_D)_{P\in\mathbb{P}})$ such that $D = T(\Sigma)$ and $f_D(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$ for $f \in \Sigma$ of arity $n$ and $t_1, \ldots, t_n \in T(\Sigma)$. Since the domain of the interpretation as well as the interpretation of function symbols is the same for all Herbrand interpretations, we only need to specify the interpretation of predicates. Hence a Herbrand interpretation can also be thought of as the set of ground atoms $P(t_1, \ldots, t_n)$ such that the tuple $(t_1, \ldots, t_n)$ is in the interpretation of $P$. A *Herbrand model* (of a formula or a set of formulas) is a Herbrand interpretation which is a model.

A *substitution* is a function $\sigma : X \to T(\Sigma, X)$. $\sigma$ is a *ground substitution* iff $\sigma(x)$ is a ground term for every $x \in X$. We assume that the result $t\sigma \in T(\Sigma, X)$ of applying a substitution $\sigma$ on a term $t \in T(\Sigma, X)$ is defined as usual.

The notion of free variables in a formula is defined as usual. A formula is called *quantifier free* if the universal quantifier $\forall$ and the existential quantifier $\exists$ don't occur in it. A *universal formula* is a formula of the form $\forall x_1 \cdot \ldots \forall x_n \cdot \phi$ where $\phi$ is quantifier free and all the free variables of $\phi$ are in the set $\{x_1, \ldots x_n\}$.

**Theorem 1 (Herbrand)** *A set of universal formulas has a model iff it has a Herbrand model.*

Observe from our definitions that if $\forall x_1 \cdot \ldots \forall x_n \cdot \phi$ is a universal formula where $\phi$ is quantifier free then for any interpretation $I \models \phi$ iff $I \models \forall x_1 \cdot \ldots \forall x_n \cdot \phi$. Hence we can also restate Herbrand's Theorem to state that a set of quantifier free formulas has a model iff it has a Herbrand model.

A *definite clause* is a quantifier free formula of the form $A \Leftarrow A_1 \wedge \ldots \wedge A_n$ where $n \geq 0$ and $A, A_1, \ldots A_n$ are atoms. A *logic program* is a set of definite clauses.

Given a logic program $\mathcal{P}$, define a Herbrand interpretation $H_P$ inductively using the following rule :

- If $A \Leftarrow A_1 \wedge \ldots \wedge A_n \in \mathcal{P}$ and for some ground substitution $\sigma$, $A_1\sigma, ..., A_n\sigma \in H_P$ then $A\sigma \in H_P$.

**Lemma 1 (Least Herbrand Model)** *We have the following results :*

*(i)* *$H_P$ is a Herbrand model of $\mathcal{P}$.*

*(ii)* *For every Herbrand model $H$ of $\mathcal{P}$, we have $H_P \subseteq H$. Hence $H_P$ is called the* least Herbrand model *of $\mathcal{P}$.*

*(iii)* *For every ground atom $A$, $A \in H_P$ iff $\mathcal{P} \models A$.*

Define a *query clause* to be a clause of the form $\perp \Leftarrow A_1 \wedge \ldots \wedge A_n$ where $A_1, \ldots, A_n$ are atoms. We have the result

**Lemma 2 (Queries)** *Let $\mathcal{P}$ be a logic program and $\perp \Leftarrow A_1 \wedge \ldots \wedge A_n$ a query clause. Then $\mathcal{P} \cup \{\perp \Leftarrow A_1 \wedge \ldots \wedge A_n\}$ is unsatisfiable iff there is a ground substitution $\sigma$ such that $A_1\sigma, \ldots, A_n\sigma \in H_{\mathcal{P}}$.*

## 2.3 Rewriting Systems and Equational Theories

A *rewriting rule* is a pair of terms $s$ and $t$, written as $s \to t$. A rewriting system is a set of rewriting rules. Given a rewriting system $\mathcal{R}$, we define the one-step reduction relation $\to_{\mathcal{R}}$ by the following rule : for every term $u$, position $p$ in $u$, rewriting rule $s \to t \in \mathcal{R}$, and substitution $\sigma$, $u[s\sigma]_p \to_{\mathcal{R}} u[t\sigma]_p$. The rewriting relation $\to^*_{\mathcal{R}}$ is the reflexive transitive closure of $\to_{\mathcal{R}}$.

An *equation* is a first order logic formula of the form $s \doteq t$ where $\doteq$ is a special binary predicate symbol $\doteq$ which is called the equality symbol. An *equational theory* is a set of equations.

Given an equational theory $\mathbb{E}$ we define the associated rewriting system $\mathcal{R}_{\mathbb{E}}$ to consist of rules $s \to t$ and $t \to s$ for every equation $s \doteq t \in \mathbb{E}$. The one-step reduction relation is defined to be $\to_{\mathbb{E}} = \to_{\mathcal{R}_{\mathbb{E}}}$ and relation $=_{\mathbb{E}}$ is defined to be $\to^*_{\mathcal{R}_{\mathbb{E}}}$. $=_{\mathbb{E}}$ is an equivalence relation on terms. Actually it is a *congruence relation* which is defined as an equivalence relation $\sim$ such that whenever $s_i \sim t_i$ for $1 \leq i \leq n$ and $f \in \Sigma$ is $n$-ary, then $f(s_1, \ldots, s_n) \sim f(t_1, \ldots, t_n)$. For an equivalence relation $\sim$ on set $S$, we use the notation $S/\sim$ to denote the set of equivalence classes of $S$ modulo $\sim$. For $s \in S$, $[s]_\sim$ denotes the equivalence class which contains $s$.

An *equational term rewriting system (ETRS)* $\mathcal{R}/\mathbb{E}$ is a rewriting system $\mathcal{R}$ together with an equational theory $\mathbb{E}$. An equational rewriting system $\mathcal{R}/\mathbb{E}$ defines a binary relation $\rightarrow_{\mathcal{R}/\mathbb{E}}$ on the set $T(\Sigma)/=_{\mathbb{E}}$ of equivalence classes modulo $=_{\mathbb{E}}$ as follows : we say that $[s]_{\mathbb{E}} \rightarrow_{\mathcal{R}/\mathbb{E}} [t]_{\mathbb{E}}$ iff there are terms $s' \in [s]_{\mathbb{E}}$ and $t' \in [t]_{\mathbb{E}}$ such that $s' \rightarrow_{\mathcal{R}} t'$.

**Definition 1** *Let $\rightarrow$ be binary relation on a set $S$.*

*(i)* *$s \in S$ is* normal *iff there is no $s' \in S$ such that $s \rightarrow s'$. $s \in S$ is a* normal form *for $t \in S$ iff $s$ is normal and $t \rightarrow^* s$.*

*(ii)* *$\rightarrow$ is* terminating *iff there is no infinite sequence of the form $s_0 \rightarrow s_1 \rightarrow s_2 \ldots$.*

*(iii)* *$\rightarrow$ is* confluent *iff $\forall s, s_1, s_2 \cdot s \rightarrow^* s_1 \wedge s \rightarrow^* s_2 \Rightarrow \exists s' \cdot s_1 \rightarrow^* s' \wedge s_2 \rightarrow s'$*

*(iv)* *$\rightarrow$ is* locally confluent *iff $\forall s, s_1, s_2 \cdot s \rightarrow s_1 \wedge s \rightarrow s_2 \Rightarrow \exists s' \cdot s_1 \rightarrow^* s' \wedge s_2 \rightarrow s'$*

*(v)* *Let $\sim$ be the reflexive symmetric transitive closure of $\rightarrow$. $\rightarrow$ is said to be* Church-Rosser *iff $\forall s_1, s_2 \cdot s_1 \sim s_2 \Rightarrow \exists s \cdot s_1 \rightarrow^* s \wedge s_2 \rightarrow^* s$.*

**Lemma 3** *A binary relation $\rightarrow$ on a set $S$ is Church-Rosser iff it is confluent.*

**Lemma 4 (Newman's Lemma[New42])** *A terminating binary relation $\rightarrow$ on a set $S$ is confluent iff it is locally confluent.*

## 2.4 Tree Automata

Tree automata [CDG+97, GS97] are extensions of the concept of the automata for strings which accept regular languages of strings. Tree automata work on terms instead of strings. A tree automaton accepts a set of terms which is said to be the language accepted by the automaton.

We first present the 'classical' notion of tree automata, before talking of its extensions. These are the automata that are called "tree automata" usually in the literature. As remarked in [CDG+97], it is convenient to describe classical tree automata as rewriting rules. We assume fixed a signature $\Sigma$.

**Definition 2 (Classical Tree Automata)** *A* classical tree automaton *$\mathcal{A}$ has a finite set $\mathbb{Q}$ of states (disjoint from $\Sigma$), and a finite set of transitions. A transition is a rewriting rule of the form*

$$f(q_1, \ldots, q_n) \rightarrow q$$

*where $f \in \Sigma$ is a $n$-ary symbol, and $q_1, \ldots, q_n, q \in \mathbb{Q}$, or of the form*

$$q \rightarrow q'$$

*where $q, q' \in \mathbb{Q}$.*

Hence the set of transitions is a rewriting system (also denoted by the same $\mathcal{A}$) on the signature $\Sigma \cup \mathbb{Q}$ where the elements of $Q$ are treated as constants. We then say that a term $t$ (on signature $\Sigma$) is accepted at a state $q$ iff $t \rightarrow^*_{\mathcal{A}} q$. In addition we specify one state in $Q$ to be the *final state* of $\mathcal{A}$. Then the set of states accepted at the final state is called the language accepted by the automaton. Unlike in other presentations, in this thesis we always assume our automata to have a unique final state. This does not entail any loss of expressiveness for the classes of automata studied in this thesis.

In the above presentation, the rewriting derivations play the role of the runs of the automata described in other presentations. Intuitively, a transition $f(q_1, \ldots, q_n) \rightarrow q$ means : "if terms $t_1, \ldots, t_n$ are accepted at states $q_1, \ldots, q_n$ respectively, then the term $f(t_1, \ldots, t_n)$ is accepted at state $q$". A transition of the form $q \rightarrow q'$ means : "if term $t$ is accepted at state $q$ then $t$ is accepted at state $q'$".

**Example 2** *Let* $\Sigma = \{O, S, +, \times\}$ *where O is zero-ary, S is unary and* $+, \times$ *are binary. Consider an automaton with* $\mathbb{Q} = \{q_{even}, q_{odd}, q_{all}\}$ *as the set of states and having the following transitions :*

$$\begin{array}{rl}
O & \rightarrow q_{even} \\
S(q_{even}) & \rightarrow q_{odd} \\
S(q_{odd}) & \rightarrow q_{even} \\
+(q_{even}, q_{even}) & \rightarrow q_{even} \\
+(q_{even}, q_{odd}) & \rightarrow q_{odd} \\
+(q_{odd}, q_{even}) & \rightarrow q_{odd} \\
+(q_{odd}, q_{odd}) & \rightarrow q_{even} \\
\times(q_{even}, q_{even}) & \rightarrow q_{even} \\
\times(q_{even}, q_{odd}) & \rightarrow q_{even} \\
\times(q_{odd}, q_{even}) & \rightarrow q_{even} \\
\times(q_{odd}, q_{odd}) & \rightarrow q_{odd} \\
q_{even} & \rightarrow q_{all} \\
q_{odd} & \rightarrow q_{all}
\end{array}$$

*Then thinking of the terms as denoting natural numbers, it can be checked that* $q_{even}$ *and* $q_{odd}$ *accept exactly the even and odd numbers, whereas* $q_{all}$ *accepts all numbers. The following is an example run in the automaton, where the binary operators have been written in infix notation for readability :*

$$\begin{array}{l}
(S(O) + S(S(O))) \times S(S(S(O))) \\
\rightarrow (S(q_{even}) + S(S(O))) \times S(S(S(O))) \\
\rightarrow^{*} (S(q_{even}) + S(S(q_{even}))) \times S(S(S(q_{even}))) \\
\rightarrow^{*} (q_{odd} + S(q_{odd})) \times S(S(q_{odd})) \\
\rightarrow^{*} (q_{odd} + q_{even}) \times S(q_{even}) \\
\rightarrow^{*} q_{odd} \times q_{odd} \\
\rightarrow q_{odd} \\
\rightarrow q_{all}
\end{array}$$

There are several questions that we may ask about a given class of tree automata :
**Decidability of emptiness :** Given an automaton, can we decide whether it accepts at least one term ?
**Decidability of membership :** Is it decidable whether a given term is accepted by a given automaton ?
**Decidability of intersection emptiness :** Given two automata, is it decidable whether they accept a common term ?
**Closure under union :** Is the class of languages accepted by the automata closed under union ?
**Closure under intersection :** Is the class of languages accepted by the automata closed under intersection ?
**Closure under complementation :** Is the class of languages accepted by the automata closed under complementation ?

As an abuse of language, we say the a class of automata is closed under union (resp. intersection, complementation) if the class of languages accepted by them is closed under union (resp. intersection, complementation). Also by "emptiness of automaton" we mean emptiness of the language accepted by the automaton. As far as classical tree automata automata are concerned, the languages accepted by them are referred to as *regular tree languages* (the equivalent notion of the regular languages of strings). It is well known that regular tree languages are closed under all Boolean operations and their emptiness is decidable. This also implies decidability of membership and intersection emptiness problem.

There is a large literature on finite tree automata, see [CDG+97, GS97]. Applications abound in rewriting and automated theorem proving notably : approximations of reachability sets for rewrite systems [Gen98], disunification and inductive reducibility [LM94], unification under constraints [KFK97], ground reducibility [CJ97], automated inductive theorem proving [BJ97], fast tree matching [Li88], automated model building in first-order logic [Pel97], etc. These applications deal with automata on *finite* trees, and this is what we are interested in here. We won't deal with automata on infinite trees [Tho90], which are also fundamental, e.g. in temporal and program logics [EJ88].

### 2.4.1 Tree Automata and First Order Logic

It is natural to consider the transitions of tree automata described above as formulas of first order logic. The idea of the translation can be expressed as the following informal identities :

| | | |
|---|---|---|
| state of automata | $\equiv$ | predicate of first order logic |
| transition of automata | $\equiv$ | definite clause |
| automaton | $\equiv$ | logic program |
| atom $P(t)$ | $\equiv$ | term $t$ is accepted at state $P$ |

The signature $\Sigma$ of the logic is the same as the signature on which the automaton works. The set of predicates of the logic is the set of states in the automaton. The atom $P(t)$ of first order logic is considered to mean that the term $t$ is accepted at the state $P$. Then it is natural to translate the transition

$$f(q_1, \ldots, q_n) \to q$$

of automata as the definite clause

$$q(f(x_1, \ldots, x_n)) \Leftarrow q_1(x_1) \wedge \ldots \wedge q_n(x_n)$$

where the $x_i$'s are distinct variables. The transition

$$q \to q'$$

can be translated as the definite clause

$$q'(x) \Leftarrow q(x)$$

**Example 3** *The transitions of the automaton in Example 2 translated to definite clauses are listed below (in the order of the corresponding transitions in Example 2).*

$$\begin{array}{rl}
q_{even}(O) & \\
q_{odd}(S(x)) & \Leftarrow q_{even}(x) \\
q_{even}(S(x)) & \Leftarrow q_{odd}(x) \\
q_{even}(+(x,y)) & \Leftarrow q_{even}(x) \wedge q_{even}(y) \\
q_{odd}(+(x,y)) & \Leftarrow q_{even}(x) \wedge q_{odd}(y) \\
q_{odd}(+(x,y)) & \Leftarrow q_{odd}(x) \wedge q_{even}(y) \\
q_{even}(+(x,y)) & \Leftarrow q_{odd}(x) \wedge q_{odd}(y) \\
q_{even}(\times(x,y)) & \Leftarrow q_{even}(x) \wedge q_{even}(y) \\
q_{even}(\times(x,y)) & \Leftarrow q_{even}(x) \wedge q_{odd}(y) \\
q_{even}(\times(x,y)) & \Leftarrow q_{odd}(x) \wedge q_{even}(y) \\
q_{odd}(\times(x,y)) & \Leftarrow q_{odd}(x) \wedge q_{odd}(y) \\
q_{all}(x) & \Leftarrow q_{even}(x) \\
q_{all}(x) & \Leftarrow q_{odd}(x)
\end{array}$$

Then this translation has the following nice property :

**Lemma 5 (Automata as Logic Programs)** *Let $\mathcal{P}$ be the set of definite clauses obtained by translation of automaton $\mathcal{A}$. Then for any term $t$ and state $q$ in the automaton $\mathcal{A}$, $t$ is accepted at $q$ iff $\mathcal{P} \models q(t)$.*

In other words, the least Herbrand model of the logic program associated with the automaton contains exactly the atoms $q(t)$ such that $t$ is accepted at $q$ in the automaton. We have the following corollaries :

**Corollary 1 (Automata Queries as Query Clauses)** *Let $\mathcal{P}$ be the set of definite clauses obtained by translation of automaton $\mathcal{A}$. Then :*

1. *The language accepted by the state $q$ in $\mathcal{A}$ is non-empty iff $\mathcal{P} \cup \{\perp \Leftarrow q(x)\}$ is unsatisfiable.*
2. *Term $t$ is accepted at state $q$ iff $\mathcal{P} \cup \{\perp \Leftarrow q(t)\}$ is unsatisfiable.*
3. *Term $t$ is accepted at each of the states $q_1, \ldots, q_n$ iff $\mathcal{P} \cup \{\perp \Leftarrow q_1(t) \wedge \ldots \wedge q_n(t)\}$ is unsatisfiable.*
4. *There is some term accepted at each of the states $q_1, \ldots, q_n$ iff $\mathcal{P} \cup \{\perp \Leftarrow q_1(x) \wedge \ldots \wedge q_n(x)\}$ is unsatisfiable.*

We have purposely chosen to view automata as logic programs in this thesis, because the formalism of first order logic makes it much more convenient to describe the extensions of the classical tree automata that we study in this thesis. Firstly by allowing newer forms of definite clauses other than the two seen above, we have a very expressive formalism for describing various extended notions to tree automata. Secondly the formalism of definite clauses makes it possible to define the equational variants of tree automata in an elegant way, as we will see later. Thirdly the above corollary allows us to reduce various decision problems about automata to the satisfiability problem in first order logic. For the latter problem, there are many powerful techniques known, notably various refinements of resolution, which allow us to get decision procedures for various fragments of first order logic.

### 2.4.2 General Two-Way Tree Automata

Now we describe an extension of the above notion of classical tree automata, which we call *general two-way tree automata*. Having seen how classical tree automata can be viewed as a set of definite clauses, it is now easy to describe this new class of automata by adding the following new kind of definite clause

$$q(x_i) \Leftarrow p(f(x_1, \ldots, x_n)) \wedge q_1(x_{i_1}) \wedge \ldots \wedge q_k(x_{i_k})$$

where $1 \leq i, i_1, \ldots, i_k \leq n$ and $k \geq 0$. The variables $x_1, \ldots, x_n$ are pairwise distinct, but the indexes $i_j$'s need not be pairwise distinct. These clauses are called *general push clauses*.

In comparison the previous two forms of clauses

$$q(f(x_1, \ldots, x_n)) \Leftarrow q_1(x_1) \wedge \ldots \wedge q_n(x_n)$$

and

$$q'(x) \Leftarrow q(x)$$

are called *pop clauses* and *epsilon clauses* respectively. We define *general two-way automata* to contain pop clauses, epsilon clauses and general push clauses. The classical tree automata seen above contain pop clause and epsilon clauses, and they are called *one-way automata* to distinguish them from two-way automata.

The general push clause above can be read as "if term $f(t_1, \ldots, t_n)$ is accepted at state $p$ an subterms $t_{i_1}, \ldots, t_{i_k}$ are accepted at states $q_1, \ldots, q_k$ then the subterm $t_i$ is accepted at state $q_i$". Hence intuitively this clause provides us a mechanism of destructing terms, as against the pop clauses which provide us a mechanism of constructing terms.

Sometimes we also need *intersection clauses* :

$$q(x) \Leftarrow q_1(x) \wedge q_2(x)$$

Such clauses can be read as "if term $t$ is accepted at both states $q_1$ and $q_2$ then it is also accepted at state $q$". This clause can easily be seen to encode the following more general form of intersection clauses, by using some new auxiliary states :

$$q(x) \Leftarrow q_1(x) \wedge \ldots \wedge q_n(x)$$

where $n \geq 2$. *Alternating* (one-way or general two-way) automata are the automata obtained by adding intersection clauses to (one-way or two-way) automata.

Alternating and two-way variants have been well studied [CDG$^+$97, FSVY91, Slu85]. Their correspondence with set constraints is described in [CDG$^+$97]. Beware that the names of "pop" and "push" clauses are interchanged in [CDG$^+$97]. [CDG$^+$97] states that :

**Lemma 6 ([CDG$^+$97])** *An automaton containing pop clauses, intersection clauses and clauses of the form $q(x) \Leftarrow q'(t)$ where $t$ is linear, can be effectively converted to one-way tree automata accepting the same language.*

This result actually allows us to conclude that an automaton containing pop clauses, epsilon clauses, intersection clauses and general push clauses can be effectively converted to an equivalent one-way tree automaton. However since we have been unable to find a proof of this result in full generality in print, we outline here a simple translation based on Lemma 6.

**Theorem 2** *An automaton containing pop clauses, epsilon clauses, intersection clauses and general push clauses can be effectively converted to a one-way tree automaton accepting the same language.*

**Proof:** Let $\mathcal{A}$ be an automaton containing pop clauses, epsilon clauses, intersection clauses and general push clauses. Without loss of generality we can assume that in the general push clauses

$$q(x_i) \Leftarrow p(f(x_1, \ldots, x_n)) \wedge q_1(x_{i_1}) \wedge \ldots \wedge q_k(x_{i_k})$$

every $x_j$ occurs in the set $\{x_{i_1}, \ldots, x_{i_k}\}$. If not we can replace this clause by the clause

$$q(x_i) \Leftarrow p(f(x_1, \ldots, x_n)) \wedge q_1(x_{i_1}) \wedge \ldots \wedge q_k(x_{i_k}) \wedge q_{all}(x_j)$$

where $q_{all}$ is a new state which accepts all terms. Such a translation preserves the language accepted by the automaton. Hence we can now assume that all the general push clauses in $\mathcal{A}$ are of the form

$$q(x_i) \Leftarrow p(f(x_1, \ldots, x_n)) \wedge B_1(x_1) \wedge \ldots \wedge B_n(x_n)$$

where for $1 \leq i \leq n$, $B_i$ is a non-empty set of states, and given any set $B = \{p_1, \ldots, p_m\}$ we write $B(x)$ as an abbreviation for the conjunction $p_1(x) \wedge \ldots \wedge p_m(x)$. Now we define a new automaton $\mathcal{B}$ as follows. Let $\mathbb{Q}$ be the set of states in $\mathcal{A}$. $\mathcal{B}$ has all the states of $\mathbb{Q}$, as well as states $\widehat{B}$ for each $\emptyset \neq B \subseteq \mathbb{Q}$, and states $q_{C,1}$ and $q_{C,2}$ for each general push clause $C$ in $\mathcal{A}$. $\mathcal{B}$ has all the pop clauses, epsilon clauses and intersection clauses of $\mathcal{A}$. For each $\emptyset \neq B = \{p_1, \ldots, p_m\} \subseteq \mathbb{Q}$, $\mathcal{B}$ has the clause

$$\widehat{B}(x) \Leftarrow B(x)$$

Intuitively, the state $\widehat{B}$ is intended to accept the intersection of the languages accepted at states $p_1, \ldots, p_m$. Besides, for each general push clause

$$C = q(x_i) \Leftarrow p(f(x_1, \ldots, x_n)) \wedge B_1(x_1) \wedge \ldots \wedge B_n(x_n)$$

in $\mathcal{A}$, $\mathcal{B}$ has the clauses

$$\begin{aligned} q_{C,1}(f(x_1, \ldots, x_n)) &\Leftarrow \widehat{B_1}(x_1) \wedge \ldots \wedge \widehat{B_n}(x_n) \\ q_{C,2}(x) &\Leftarrow q_{C,1}(x) \wedge p(x) \\ q(x_i) &\Leftarrow q_{C,2}(f(x_1, \ldots, x_n)) \end{aligned}$$

$\mathcal{B}$ does not have any other clause. Then we can see that for every state $q \in \mathbb{Q}$, $q$ accepts the same language in $\mathcal{A}$ as in $\mathcal{B}$. In particular, letting the final state of $\mathcal{B}$ to be the same as the final state of $\mathcal{A}$, $\mathcal{A}$ and $\mathcal{B}$ accept the same language. Note that $\mathcal{B}$ only has pop clauses, epsilon clauses, intersection clauses and general push clauses of the restricted form $q_1(x_i) \Leftarrow q_2(f(x_1, \ldots, x_n))$ where $1 \leq i \leq n$ and $x_1, \ldots, x_n$ are mutually distinct variables. The required result then follows from Lemma 6. □

General two-way tree automata are sometimes more convenient to work with because of the fact that they allow transition rules to destruct terms. In particular we will see their utility in modeling of cryptographic protocols in Chapter 5.

Finally we remark that the qualifier "general" have been used above because later when we study equational variants of these automata, we will need to consider some restricted subclasses of these automata which will be more relevant for our purposes.

## 2.5 Logic with Equality

We now consider how to deal with equality in first order logic. There are several more or less equivalent ways to deal with equality in first order logic. Here we have chosen to deal with equality using axioms of equality. Given a signature $\Sigma$ and a set $\mathbb{P}$ of predicate symbols, we define the *theory of equality* $Eq_{\Sigma,\mathbb{P}}$ to consist of the formulas

– $x \doteq x$ (Reflexivity)
– $x \doteq y \Rightarrow y \doteq x$ (Symmetry)
– $x \doteq y \wedge y \doteq z \Rightarrow x \doteq z$ (Transitivity)
– $\bigwedge_{1 \leq i \leq n} x_i \doteq y_i \Rightarrow f(x_1, \ldots, x_n) \doteq f(y_1, \ldots, y_n)$ for every $f \in \Sigma$ of arity $n$ (Congruence axiom for function symbols)
– $\bigwedge_{1 \leq i \leq n} x_i \doteq y_i \wedge P(x_1, \ldots, x_n) \Rightarrow P(y_1, \ldots, y_n)$ for every $P \in \mathbb{P}$, $P \neq \doteq$ of arity $n$ (Congruence axiom for predicate symbols)

We simply write $Eq$ when the signature and the set of predicates involved is clear from context. An *equational interpretation* is an interpretation $I$ such that $I \models Eq$. Given an equational theory $\mathbb{E}$, an $\mathbb{E}$-*interpretation* is an equational interpretation such that $I \models \mathbb{E}$. We say that $I$ is an $\mathbb{E}$-*model* of $S$, written $I \models_{\mathbb{E}} S$, iff $I$ is an $\mathbb{E}$-interpretation such that $I \models S$. $S'$ is a *semantic* $\mathbb{E}$*-consequence* of $S$, written $S \models_{\mathbb{E}} S'$, iff every $\mathbb{E}$-model of $S$ is an $\mathbb{E}$-model of $S'$.

Since our principal motivation is to study equational variants of tree automata, we need to deal with logic programs in the presence of equality. Also observe that for our purposes, the predicates of logic represent states of automata, we only need to consider logic programs in which only unary predicates occur. In particular the equality symbol does not occur in these logic programs. Consider a logic program $\mathcal{P}$ in which the equality symbol $\doteq$ does not occur, and let $\mathbb{E}$ be an equational theory. Note that the formulas in $\mathbb{E}$ and $Eq$ are all definite clauses. Hence $\mathcal{P} \cup \mathbb{E} \cup Eq$ can be considered as a logic program (in which the equality symbol occurs). Hence the logic program $\mathcal{P} \cup \mathbb{E} \cup Eq$ has a least Herbrand model $H_{\mathcal{P} \cup \mathbb{E} \cup Eq}$. (For the purposes of defining this least Herbrand model, the $\doteq$ symbol is treated just like any other predicate.)

Now we inductively define a Herbrand interpretation $H_{\mathcal{P},\mathbb{E}}$ using the following two rules :

– If $A \Leftarrow A_1 \wedge \ldots A_n \in \mathcal{P}$ and for some ground substitution $\sigma$, $A_1\sigma, ..., A_n\sigma \in H_{\mathcal{P},\mathbb{E}}$ then $A\sigma \in H_{\mathcal{P},\mathbb{E}}$.

– If $P(s_1, \ldots, s_n) \in H_{\mathcal{P},\mathbb{E}}$ and $s_i =_{\mathbb{E}} t_i$ for $1 \leq i \leq n$ then $P(t_1, \ldots, t_n) \in H_{\mathcal{P},\mathbb{E}}$.

Observe that by definition $\mathcal{P} \cup \mathbb{E} \cup Eq \models S$ iff $\mathcal{P} \models_{\mathbb{E}} S$.

**Lemma 7 (Equational Herbrand Models)** *We have the following results :*

*(i) for* $s, t \in T(\Sigma)$, $s =_{\mathbb{E}} t$ *iff* $\models_{\mathbb{E}} s \doteq t$ *iff* $\mathcal{P} \models_{\mathbb{E}} s \doteq t$.

*(ii)* $H_{\mathcal{P} \cup \mathbb{E} \cup Eq} = H_{\mathcal{P},\mathbb{E}} \cup \{s \doteq t \mid s, t \in T(\Sigma), s =_{\mathbb{E}} t\}$.

For this reason, we can think of $H_{\mathcal{P},\mathbb{E}}$ as the least Herbrand model of $\mathcal{P}$ modulo the equational theory $\mathbb{E}$.

# Chapitre 3

# Automates d'arbres bidirectionnels équationnels (Two-Way Equational Tree Automata)

Nous introduisons la notion d'automate d'arbres (resp. alternant, bidirectionnel) *équationnel* simplement comme un automate d'arbres (resp. alternant, bidirectionnel), mais cette fois-ci interprété dans une théorie équationnelle donnée. Nous nous basons pour cela sur la notion de modèle de Herbrand équationnel du dernier chapitre. Une typologie des différents formats de clauses dont nous avons besoin pour décrire ces automates est établie : les automates d'arbres sont des ensembles de clauses *pop*, *epsilon*, *intersection* (dans le cas d'automates alternants), *push libres* et $+$*-push générales* (dans le cas d'automates bidirectionnels).

Nous examinons au passage plus en détail le cas des théories équationnelles $\mathbb{AC}$, $\mathbb{ACU}$, $\mathbb{ACUI}$, $\mathbb{ACUX}$, $\mathbb{ACUX}_n$, $\mathbb{ACUM}$, $\mathbb{ACUD}$, et analysons en particulier la forme des termes modulo ces théories. Cette analyse sera fondamentale dans les algorithmes de réduction présentés dans le reste de cette thèse.

Nous terminons ce chapitre par une comparaison détaillée entre notre approche et des approches similaires proposées dans la littérature, et principalement les automates modulo $\mathbb{AC}$ de Lugiez et les E-automates d'Ohsaki.

Having studied general two-way tree automata in the form of logic programs, and having seen how to deal with logic programs in the presence of an equational theory, we are now well equipped to define equational variants of general two-way tree automata.

## 3.1 Equational Tree Automata as Logic Programs Modulo Equational Theories

Fix a signature $\Sigma$ of function symbols, each coming with a fixed arity, and let $\mathbb{E}$ be an equational theory, inducing the relation $=_{\mathbb{E}}$ on the terms built from $\Sigma$, as defined in Chapter 1.
Recall that a definite clause (on unary predicates) is an implication of the form :

$$P(t) \Leftarrow P_1(t_1) \wedge ... \wedge P_n(t_n) \tag{3.1}$$

where $P, P_1, ..., P_n$ are unary predicates and $t, t_1, ..., t_n$ are terms built from $\Sigma$ and variables. We consider only definite clauses with unary predicates because that is all that we need for defining our automata. We call $P(t)$ the *head* of the clause, and the list of atoms $P_1(t_1), \ldots, P_n(t_n)$ to be the *body* or *tail* of the clause. Given a finite set $\mathcal{A}$ of such definite clauses we define *derivations* of ground atoms using the following two rules :

$$\frac{P_1(t_1\sigma)...P_n(t_n\sigma)}{P(t\sigma)}\,(AUTO)$$

(if $\sigma$ is a ground substitution and $P(t) \Leftarrow P_1(t_1) \wedge ... \wedge P_n(t_n) \in \mathcal{A}$)

$$\frac{P(s)}{P(t)}\,(\mathbb{E})$$

(if $s =_{\mathbb{E}} t$)
Rule $(AUTO)$ allows us to derive a new atom from other atoms using an automata clause. Sometimes we use the notation

$$\frac{P_1(t_1\sigma)...P_n(t_n\sigma)}{P(t\sigma)}\,(P(t) \Leftarrow P_1(t_1) \wedge ... \wedge P_n(t_n))$$

to make explicit the clause used. Rule $(\mathbb{E})$ allows us to derive a new atom from an old one using equational rewriting. Thus a derivation is a tree-like structure, which should not be confused with the trees which are the terms built from $\Sigma$. If there is a derivation with atom $P(t)$ as conclusion then we say that $P(t)$ is derivable in $\mathcal{A}/\mathbb{E}$. Recall that from the discussion in Chapter 1, the set of derivable atoms is exactly the least Herbrand model of $\mathcal{A}$ modulo the theory $\mathbb{E}$. Hence we have the result :

**Lemma 8** *For any set of definite clauses $\mathcal{A}$, equational theory $\mathbb{E}$ and ground atom $P(t)$, $P(t)$ is derivable in $\mathcal{A}/\mathbb{E}$ iff $\mathcal{A} \models_{\mathbb{E}} P(t)$ iff $P(t) \in H_{\mathcal{A},\mathbb{E}}$.*

The connection of definite clauses with automata is as described in Chapter 1 for the non-equational case : predicates are states, finite sets of definite clauses are automata, and an atom $P(t)$ is derivable in $\mathcal{A}/\mathbb{E}$ , iff the term $t$ is accepted at state $P$ in the automaton $\mathcal{A}$. The derivations of atoms are the equivalent of the runs in classical tree automata. Of course as in Chapter 1 we will be interested in automata in which the format of the definite clauses is sufficiently restricted so as to be able to prove any good properties.

The language $\mathcal{L}_P(\mathcal{A}/\mathbb{E})$ is the set of terms $t$ such that $P(t)$ is derivable in $\mathcal{A}/\mathbb{E}$. When $\mathbb{E}$ is the empty theory, we call it $\mathcal{L}_P(\mathcal{A})$. If in addition some state $P_f$ is specified as being *final* then the language accepted by $\mathcal{A}$ is $\mathcal{L}(\mathcal{A}/\mathbb{E}) = \mathcal{L}_{P_f}(\mathcal{A}/\mathbb{E})$. Also, given a language $\mathcal{L}$ and an equational theory $\mathbb{E}$, the $\mathbb{E}$-*closure* of $\mathcal{L}$ is defined as $\mathbb{E}(\mathcal{L}) = \{t \mid \exists s \in \mathcal{L} \cdot t =_{\mathbb{E}} s\}$. A set $\mathcal{L}$ of terms is said to be $\mathbb{E}$-*closed* iff $\mathbb{E}(\mathcal{L}) = \mathcal{L}$. Observe that we have chosen to define languages accepted by equational tree automata as sets of terms. Another equivalent approach is to let them accept equivalence classes of terms modulo an equational theory.

The $(AUTO)$ and $(\mathbb{E})$ rules can be combined to get the following derived rule :

$$\frac{P_1(s_1)\ldots P_n(s_n)}{P(s)}\,(AUTO/\mathbb{E})$$

if some $P(t) \Leftarrow P_1(t_1) \wedge \ldots \wedge P_n(t_n) \in \mathcal{A}$ and $\sigma$ is a ground substitution such that $s =_{\mathbb{E}} t\sigma$ and for $1 \leq i \leq n$, $s_i =_{\mathbb{E}} t_i\sigma)$

To make explicit the clause used the above step will also be written as

$$\frac{P_1(s_1)\ldots P_n(s_n)}{P(s)}\,(P(t) \Leftarrow P_1(t_1) \wedge \ldots \wedge P_n(t_n)/\mathbb{E})$$

Adding this new rule to the rules $(AUTO)$ and $(\mathbb{E})$ does not allow us to derive any new atoms. Also it is easy to see that this new derived rule subsumes the previous two clauses. That is :

**Observation 1** *If $P(t)$ is derivable in $\mathcal{A}/\mathbb{E}$ then it has a derivation which uses only rules of the form $P(t) \Leftarrow P_1(t_1) \wedge \ldots \wedge P_n(t_n)/\mathbb{E}$ for clauses $P(t) \Leftarrow P_1(t_1) \wedge \ldots \wedge P_n(t_n) \in \mathcal{A}$.*

We also introduce some notation. We will use the notation

$$\begin{array}{c}\vdots\\ P(t)\end{array}$$

to denote some derivation of $P(t)$. If we want to give this derivation a name $\pi$ then we use the notation

$$\frac{\pi}{P(t)}$$

A *subderivation* $\pi'$ of a derivation $\pi$ is precisely a subtree of $\pi$. We allow $\pi'$ to be the same as $\pi$. (The notion of subtree is well defined since as we remarked above, our derivations are trees.) Similarly the notion of derivation $\pi_1$ being a *child* of derivation $\pi_2$ (or $\pi_2$ being the *parent* of $\pi_1$) is well defined.

We write

$$\pi = \frac{\dfrac{\pi_1}{P_1(t_1)} \quad \ldots \quad \dfrac{\pi_n}{P_n(t_n)}}{P(t)}\,(S/\mathbb{E})$$

to mean that $\pi$ is a derivation of $P(t)$, that the derivations $\pi_1, \ldots, \pi_n$ of $P_1(t_1), \ldots, P_n(t_n)$ respectively are the children of $\pi$ and that the rule used at the root node of $\pi$ is $C/\mathbb{E}$ for some $C \in S$.

$C$ is also called the *last clause used* by $\pi$. Similar remarks hold when instead of writing $(S/\mathbb{E})$ in the above diagram, we write just $(S)$ or $(C)$ for some clause $C$.

We write

$$\pi = \frac{\dfrac{\pi_1}{P_1(t_1)} \quad \ldots \quad \dfrac{\pi_n}{P_n(t_n)}}{P(t)} \; (S/\mathbb{E})$$

to mean that $\pi$ is a derivation of $P(t)$, that the derivations $\pi_1, \ldots, \pi_n$ of $P_1(t_1), \ldots, P_n(t_n)$ respectively are subderivations of $\pi$ and that outside the subderivations $\pi_1, \ldots, \pi_n$, the only clauses used (probably none) are of the form $C/\mathbb{E}$ for some $C \in S$. Similar remarks hold when in the diagram above we write just $(S)$ instead of $(S/\mathbb{E})$. In both diagrams above, we may also write some subderivation using the nameless notation, i.e. we may write

$$\begin{array}{c} \vdots \\ P_i(t_i) \end{array}$$

instead of

$$\frac{\pi_i}{P_i(t_i)}$$

## 3.2 Associative-Commutative Theories

Having seen what equational tree automata are for an arbitrary equational theory, in this section we have a look at some of the specific equational theories that we are going to deal with in this thesis. We are primarily interested in the theory of associativity and commutativity, and their variants obtained by additional axioms. Since we are interested in associative-commutative theories, we assume that $\Sigma$ always contains a binary symbol $+$. The axioms of associativity and commutativity are

($\mathbb{A}$) $x + (y + z) = (x + y) + z$ (associativity)

($\mathbb{C}$) $x + y = y + x$ (commutativity)

Sometimes we also include a zero-ary symbol $0$ with the axiom

($\mathbb{U}$) $x + 0 = x$ (unit)

which says that $0$ is a unit of $+$.

We are also interested in the equational theories obtained by adding one of the following axioms to the theory of associativity and commutativity.

($\mathbb{X}$) $x + x = 0$ (axiom for exclusive-or)

($\mathbb{X}_n$) $\underbrace{x + \ldots + x}_{n \text{ times}} = 0$ (axiom for generalized xor) ($n \geq 2$)

($\mathbb{D}$) $-(x + y) = (-x) + (-y)$, $-(-x) = x$ and $-0 = 0$ (axioms for distributive but not cancellative minus symbol)

($\mathbb{M}$) $x + (-x) = 0$ (axiom for cancellative minus symbol)

($\mathbb{I}$) $x + x = x$ (idempotence)

where if the axioms $\mathbb{M}$ or $\mathbb{D}$ are considered then we assume that $\Sigma$ additionally contains a unary symbol $-$.

We name a theory by the names of its axioms ; e.g., $\mathbb{ACU}$ is the theory of an associative-commutative symbol with unit, $\mathbb{ACUX}$ is the theory of exclusive-or, $\mathbb{ACUM}$ is the theory of Abelian groups. Note that the distributivity axioms $\mathbb{D}$ are implied by $\mathbb{ACUM}$, so $\mathbb{ACUD}$ is a (strictly) weaker theory than

$\mathbb{ACUM}$. The $\mathbb{D}$ axioms should not be confused with the distributivity axioms for a binary symbol $\times$ over $+$ which are considered sometimes in the context of $\mathbb{AC}$ theories. The theories dealt with in this thesis are $\mathbb{AC}$, $\mathbb{ACU}$, $\mathbb{ACUX}$, $\mathbb{ACUX}_n$, $\mathbb{ACUD}$, $\mathbb{ACUM}$ and $\mathbb{ACUI}$. Observe that this includes the theory of associativity-commutativity (with unit), the theory of exclusive-or and the theory of Abelian groups, which are of particular importance from the point of view of verification of cryptographic protocols.

The symbols $+, -, 0$ are called *equational symbols* while the symbols in $\Sigma_f = \Sigma \setminus \{+, -, 0\}$, are called *free symbols*. 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*. To avoid confusion we assume that the symbols $-$ and $0$ are present in the signature only when we are working with an equational theory in which these symbols occur. For example, when working with $\mathbb{ACUX}$ automata, we assume that $- \notin \Sigma$ since the symbol $-$ does not occur in any of the axioms $\mathbb{A}$, $\mathbb{C}$, $\mathbb{U}$ and $\mathbb{X}$. When working with the theory $\mathbb{AC}$, we assume that $0, - \notin \Sigma$.

Assuming that the signature $+ \in \Sigma$, we use the notation $t_1 + t_2 + \ldots + t_n$ for $n \geq 1$ to denote the term $((\ldots(t_1 + t_2) + \ldots) + t_n) \in T(\Sigma, X)$. (For $n = 1$ it is just the term $t_1$.) The choice of parentheses above is arbitrary, and any other choice would lead to an equivalent term modulo $\mathbb{AC}$. This notation would be used only in a context where the choice of parentheses does not matter. Assuming that $0 \in \Sigma$, the notation $t_1 + \ldots + t_n$ for $n = 0$ denotes the term $0$. If $0 \notin \Sigma$ then this notation is undefined. The notation $\sum_{i=1}^{n} t_i$ is also used to denote the term $t_1 + \ldots + t_n$. The notation $nt$ denotes the term $\underbrace{t + \ldots + t}_{n \text{ times}}$. Observe that we have already used such notation for describing the equation $\mathbb{X}_n$ in the list of equations above.

### 3.2.1 Properties of Terms Modulo $\mathbb{AC}$ Theories

We now discuss some basic properties of terms modulo the above theories. We are interested in some canonical forms of terms modulo various theories, as well as conditions on when two terms are equivalent wrt a certain theory.

**Observation 2 ($\mathbb{AC}$ Terms)** *Let* $+ \in \Sigma$.

*(i) For any $t \in T(\Sigma)$ we have terms $t_1, \ldots, t_n \in T(\Sigma)$ for some $n \geq 1$ such that $t =_{\mathbb{AC}} t_1 + \ldots + t_n$ and the symbol $+$ does not occur at the head of any $t_i$. In particular if $-, 0 \notin \Sigma$ then each $t_i$ is functional.*

*(ii) If $s_1 + \ldots + s_n =_{\mathbb{AC}} t_1 + \ldots + t_m$ and no $s_i$ or $t_j$ has the symbol $+$ at the root, then we have $n = m$ and for some permutation $\sigma$ of $\{1, \ldots, n\}$ we have $s_i =_{\mathbb{AC}} t_{\sigma(i)}$ for $1 \leq i \leq n$.*

**Observation 3 ($\mathbb{ACU}$ Terms)** *Let* $+, 0 \in \Sigma$.

*(i) For any $t \in T(\Sigma)$ we have terms $t_1, \ldots, t_n \in T(\Sigma)$ for some $n \geq 0$ such $t =_{\mathbb{ACU}} t_1 + \ldots + t_n$ and the symbols $+$ and $0$ do not occur at the head of any $t_i$. In particular if $- \notin \Sigma$ then each $t_i$ is functional.*

*(ii) If $s_1 + \ldots + s_n =_{\mathbb{ACU}} t_1 + \ldots + t_m$ and no $s_i$ or $t_j$ has the symbols $+$ or $0$ at the root, then we have $n = m$ and for some permutation $\sigma$ of $\{1, \ldots, n\}$ we have $s_i =_{\mathbb{ACU}} t_{\sigma(i)}$.*

Observations 2 and 3 are an alternative approach to defining the so called 'flattened' representation of a term modulo $\mathbb{AC}$ (or $\mathbb{ACU}$). In that approach, terms of the form $t_1 + \ldots + t_n$ are written as $f(t_1, \ldots, t_n)$ where $f$ has variable arity and the order of the children of a node labeled with $f$ is irrelevant.

We use the notation $s - t$ to denote the term $s + (-t)$.

For any $p$, we define the partial order $\leq$ on $\mathbb{N}^p$ as : $\nu_1 \leq \nu_2$ iff $\nu_1(i) \leq \nu_2(i)$ for each $1 \leq i \leq p$. We write $\nu_1 < \nu_2$ iff $\nu_1 \leq \nu_2$ and $\nu_1 \neq \nu_2$.

Let $\mathcal{D}$ denote the rewriting system $\{-(x+y) \to (-x)+(-y), -(-x) \to x, -0 \to 0, x+0 \to x\}$. Then we have the following result :

**Observation 4 ($\mathbb{ACUD}$ Terms)** *Let* $+, -, 0 \in \Sigma$.

- *(i)* $\to_{\mathcal{D}/\mathbb{AC}}$ *is terminating.*
- *(ii)* $\to_{\mathcal{D}/\mathbb{AC}}$ *is locally confluent. Hence from (i) and Lemma 4,* $\to_{\mathcal{D}/\mathbb{AC}}$ *is confluent.*
- *(iii) If* $[t]_{\mathbb{AC}}$ *is normal wrt* $\to_{\mathcal{D}/\mathbb{AC}}$ *then we have terms* $s_1, \ldots, s_m, t_1, \ldots, t_n$ *for some* $m, n \geq 0$ *such that each* $s_i$ *and each* $t_j$ *is functional and we have* $t =_{\mathbb{AC}} s_1 + \ldots + s_m - t_1 - \ldots - t_n$, *and if* $t'$ *is any strict subterm of some* $s_i$ *or* $t_j$ *then* $[t']_{\mathbb{AC}}$ *is normal.*
- *(iv) If* $s =_{\mathbb{ACUD}} t$ *then we have some functional terms* $s_1, \ldots, s_m, t_1, \ldots, t_n$ *for some* $m, n \geq 0$ *such that* $s =_{\mathbb{ACUD}} t =_{\mathbb{ACUD}} s_1 + \ldots + s_m - t_1 - \ldots - t_n$, *and* $[s_1 + \ldots + s_m - t_1 - \ldots - t_n]_{\mathbb{AC}}$ *is a normal form of both* $[s]_{\mathbb{AC}}$ *and* $[t]_{\mathbb{AC}}$.

**Proof:** We define a measure $\nu$ on terms as $\nu(t) = (\nu_1(t), \nu_2(t))$ where

- $\nu_1(t)$ is numbers of pairs $(p, q)$ of positions in $t$ such that $p <_{pref} q$, the symbol $-$ occurs at $p$ and for all positions $p <_{pref} r \leq_{pref} q$, one of the symbols $+, -$ of $0$ occurs at $r$.
- $\nu_2(t)$ is the number of occurrences of the symbol $0$ in $t$.

We first observe that if $s =_{\mathbb{AC}} t$ then $\nu(s) = \nu(t)$. Secondly if $s \to_{\mathcal{D}} t$ then $\nu(t) < \nu(s)$. Hence if $s \to_{\mathcal{D}/\mathbb{AC}}$ then $\nu(t) < \nu(s)$. Since $<$ is well founded on $\mathbb{N}^2$, this implies property (i). Property (ii) is proved by case analysis on all rewriting steps possible from a term. Property (iii) is proved using Observation 3 and case analysis on all possible structures of the term $t$.

To prove property (iv) define relation $\sim$ on equivalence classes modulo $\mathbb{AC}$ as $[s]_{\mathbb{AC}} \sim [t]_{\mathbb{AC}}$ iff $s =_{\mathbb{ACUD}} t$. Then $\sim$ is the reflexive transitive closure of $\to_{\mathcal{D}/\mathbb{AC}}$. Now assume that $s =_{\mathbb{ACUD}} t$. Then we have $[s]_{\mathbb{AC}} \sim [t]_{\mathbb{AC}}$. From Lemma 3 we have some $t'$ such that $[s]_{\mathbb{AC}} \to^*_{\mathcal{D}/\mathbb{AC}} [t']_{\mathbb{AC}}$ and $[t]_{\mathbb{AC}} \to^*_{\mathcal{D}/\mathbb{AC}} [t']_{\mathbb{AC}}$. We have $[s]_{\mathbb{AC}} \sim [t']_{\mathbb{AC}} \sim [t]_{\mathbb{AC}}$, i.e. $s =_{\mathbb{ACUD}} t' =_{\mathbb{ACUD}} t$. Using property (i) let $[t'']_{\mathbb{AC}}$ be in normal form such that $[t']_{\mathbb{AC}} \to^*_{\mathcal{D}/\mathbb{AC}} [t'']_{\mathbb{AC}}$. We have $t' =_{\mathbb{ACUD}} t''$. Using property (iii) we have functional terms $s_1, \ldots, s_m, t_1, \ldots, t_n$ for some $m, n \geq 0$ such that $t''x =_{\mathbb{ACUD}} t =_{\mathbb{ACUD}} s_1 + \ldots + s_m - t_1 - \ldots - t_n$. □

Let $\mathcal{X}$ denote the rewriting system $\{x + x \to 0, x + 0 \to x\}$.

**Observation 5 ($\mathbb{ACUX}$ Terms)** *Let* $+, 0 \in \Sigma$ *and* $- \notin \Sigma$.

- *(i)* $\to_{\mathcal{X}/\mathbb{AC}}$ *is terminating.*
- *(ii)* $\to_{\mathcal{X}/\mathbb{AC}}$ *is locally confluent. Hence from (i) and Lemma 4,* $\to_{\mathcal{X}/\mathbb{AC}}$ *is confluent.*
- *(iii) If* $[s]_{\mathbb{AC}} \to^*_{\mathcal{X}/\mathbb{AC}} [t]_{\mathbb{AC}}$ *then for some functional terms* $s_1, \ldots, s_n, t_1, \ldots, t_n, u_1, v_1, \ldots, u_p, v_p$ $(n, p \geq 0)$ *we have* $s =_{\mathbb{ACU}} s_1 + \ldots + s_n + u_1 + v_1 + \ldots + u_p + v_p$, $t =_{\mathbb{ACU}} t_1 + \ldots + t_n$, $s_i =_{\mathbb{ACUX}} t_i$ *for* $1 \leq i \leq n$ *and* $u_i =_{\mathbb{ACUX}} v_i$ *for* $1 \leq i \leq p$.
- *(iv) If* $s =_{\mathbb{ACUX}} t$ *then we have some functional terms* $s_1, \ldots, s_n, t_1, \ldots, t_n, u_1, v_1, \ldots, u_p, v_p, u'_1, v'_1, \ldots, u'_q, v'_q$ $(n, p, q \geq 0)$ *such that* $s = \mathbb{ACU} s_1 + \ldots + s_n + u_1 + v_1 + \ldots + u_p + v_p$, $t =_{\mathbb{ACU}} t_1 + \ldots + t_n + u'_1 + v'_1 + \ldots + u'_q + v'_q$, $s_i =_{\mathbb{ACUX}} t_i$ *for* $1 \leq i \leq n$, $u_i =_{\mathbb{ACUX}} v_i$ *for* $1 \leq i \leq p$ *and* $u'_i =_{\mathbb{ACUX}} v'_i$ *for* $1 \leq i \leq q$.

**Proof:** Property (i) follows from the fact that if $s =_{\mathbb{AC}} t$ then $s$ and $t$ have the same size, while if $s \rightarrow_{\mathcal{X}} t$ then the size of $t$ is strictly lesser than the size of $s$. Property (ii) is shown by a case analysis on all rewriting steps possible from a term $t$. Property (iii) is proved by induction on the length of rewrite derivations.

To prove property (iv) define relation $\sim$ on equivalence classes modulo $\mathbb{AC}$ as $[s]_{\mathbb{AC}} \sim [t]_{\mathbb{AC}}$ iff $s =_{\mathbb{ACUX}} t$. Then $\sim$ is the reflexive transitive closure of $\rightarrow_{\mathcal{X}/\mathbb{AC}}$. Now assume that $s =_{\mathbb{ACUX}} t$. Then we have $[s]_{\mathbb{AC}} \sim [t]_{\mathbb{AC}}$. From Lemma 3 we have some $t'$ such that $[s]_{\mathbb{AC}} \rightarrow^*_{\mathcal{X}/\mathbb{AC}} [t']_{\mathbb{AC}}$ and $[t]_{\mathbb{AC}} \rightarrow^*_{\mathcal{X}/\mathbb{AC}} [t']_{\mathbb{AC}}$. We have $[s]_{\mathbb{AC}} \sim [t']_{\mathbb{AC}} \sim [t]_{\mathbb{AC}}$, i.e. $s =_{\mathbb{ACUX}} t' =_{\mathbb{ACUX}} t$. Using property (i) let $[t'']_{\mathbb{AC}}$ be normal form of $[t']_{\mathbb{AC}}$. We have $t' =_{\mathbb{ACUX}} t''$. Let $t'' =_{\mathbb{ACU}} t'_1 + \ldots + t'_n$ for some $n \geq 0$ where each $t'_i$ is functional. Since $[s]_{\mathbb{AC}} \rightarrow^*_{\mathcal{X}/\mathbb{AC}} [t'']_{\mathbb{AC}}$, using property (iii) we have functional terms $s_1, \ldots, s_n, u_1, v_1, \ldots, u_p, v_p$ for some $p \geq 0$ such that $s =_{\mathbb{ACU}} s_1 + \ldots + s_n + u_1 + v_1 + \ldots + u_p + v_p$, $s_i =_{\mathbb{ACUX}} t'_i$ for $1 \leq i \leq n$ and $u_i =_{\mathbb{ACUX}} v_i$ for $1 \leq i \leq p$. Similarly we have functional terms $t_1, \ldots, t_n, u'_1, v'_1, \ldots, u'_q, v'_q$ for some $q \geq 0$ such that $t =_{\mathbb{ACU}} t_1 + \ldots + t_n + u'_1 + v'_1 + \ldots + u'_q + v'_q$, $t_i =_{\mathbb{ACUX}} t'_i$ for $1 \leq i \leq n$ and $u'_i =_{\mathbb{ACUX}} v'_i$ for $1 \leq i \leq q$. Hence we also have $s_i =_{\mathbb{ACUX}} t_i$ for $1 \leq i \leq n$. $\square$

This generalizes for the theory $\mathbb{ACUX}_n$ for any $n \geq 2$. For this we define the rewriting system $\mathcal{X}_n = \{\underbrace{x + \ldots + x}_{n \text{ times}} \rightarrow 0, x + 0 \rightarrow x\}$. We have

**Observation 6 ($\mathbb{ACUX}_n$ Terms)** *Let $+, 0 \in \Sigma$, $- \notin \Sigma$ and $n \geq 2$.*

*(i)* $\rightarrow_{\mathcal{X}_n/\mathbb{AC}}$ *is terminating.*

*(ii)* $\rightarrow_{\mathcal{X}_n/\mathbb{AC}}$ *is locally confluent. Hence from (i) and Lemma 4,* $\rightarrow_{\mathcal{X}_n/\mathbb{AC}}$ *is confluent.*

*(iii) If* $[s]_{\mathbb{AC}} \rightarrow^*_{\mathcal{X}_n/\mathbb{AC}} [t]_{\mathbb{AC}}$ *then for some functional terms* $s_1, \ldots, s_m, t_1, \ldots, t_m, u^1_1, \ldots, u^n_1, \ldots, u^1_p, \ldots, u^n_p$ $(m, p \geq 0)$ *we have* $s =_{\mathbb{ACU}} s_1 + \ldots + s_m + u^1_1 + \ldots + u^n_1 + \ldots + u^1_p + \ldots + u^n_p$, $t =_{\mathbb{ACU}} t_1 + \ldots + t_m$, $s_i =_{\mathbb{ACUX}} t_i$ *for* $1 \leq i \leq m$ *and* $u^j_i =_{\mathbb{ACUX}} u^k_i$ *for* $1 \leq i \leq p, 1 \leq j, k \leq n$.

*(iv) If* $s =_{\mathbb{ACUX}_n} t$ *then we have some functional terms* $s_1, \ldots, s_m, t_1, \ldots, t_m, u^1_1, \ldots, u^1_1, \ldots, u^1_p, \ldots, u^n_p, u'^1_1, \ldots, u'^n_1, \ldots, u'^1_q, \ldots, u'^n_q$ $(m, p, q \geq 0)$ *such that* $s =_{\mathbb{ACU}} s_1 + \ldots + s_m + u^1_1 + \ldots + u^n_1 + \ldots + u^1_p + \ldots + u^n_p$, $t =_{\mathbb{ACU}} t_1 + \ldots + t_m + u'^1_1 + \ldots + u'^n_1 + \ldots + u'^1_q + \ldots + u'^n_q$, $s_i =_{\mathbb{ACUX}_n} t_i$ *for* $1 \leq i \leq m$, $u^j_i =_{\mathbb{ACUX}_n} u^k_i$ *for* $1 \leq i \leq p, 1 \leq j, k \leq n$ *and* $u'^j_i =_{\mathbb{ACUX}_n} u'^k_i$ *for* $1 \leq i \leq q, 1 \leq j, k \leq n$.

Let $\mathcal{M}$ denote the rewriting system $\{x + (-x) \rightarrow 0, x + 0 \rightarrow x\}$.

**Observation 7 ($\mathbb{ACUM}$ Terms)** *Let $+, 0, - \in \Sigma$.*

*(i)* $\rightarrow_{\mathcal{M}/\mathbb{AC}}$ *is terminating.*

*(ii)* $\rightarrow_{\mathcal{M}/\mathbb{AC}}$ *is locally confluent. Hence from (i) and Lemma 4,* $\rightarrow_{\mathcal{M}/\mathbb{AC}}$ *is confluent.*

*(iii) If* $[s]_{\mathbb{AC}} \rightarrow^*_{\mathcal{X}/\mathbb{AC}} [t]_{\mathbb{AC}}$ *then for some functional terms* $s_1, \ldots, s_n, s'_1, \ldots, s'_m, t_1, \ldots, t_n, t'_1, \ldots, t'_m, u_1, v_1, \ldots, u_p, v_p$ $(n, m, p \geq 0)$ *we have* $s =_{\mathbb{ACUD}} s_1 + \ldots + s_n - s'_1 - \ldots - s'_m + u_1 - v_1 + \ldots + u_p - v_p$, $t =_{\mathbb{ACUD}} t_1 + \ldots + t_n - t'_1 - \ldots - t'_m$, $s_i =_{\mathbb{ACUM}} t_i$ *for* $1 \leq i \leq n$, $s'_i =_{\mathbb{ACUM}} t'_i$ *for* $1 \leq i \leq m$ *and* $u_i =_{\mathbb{ACUM}} v_i$ *for* $1 \leq i \leq p$.

*(iv) If* $s =_{\mathbb{ACUM}} t$ *then we have some functional terms* $s_1, \ldots, s_n, s'_1, \ldots, s'_m, t_1, \ldots, t_n, t'_1, \ldots, t'_m, u_1, v_1, \ldots, u_p, v_p, u'_1, v'_1, \ldots, u'_q, v'_q$ $(n, m, p, q \geq 0)$ *such that* $s =_{\mathbb{ACUD}} s_1 + \ldots + s_n - s'_1 - \ldots - s'_m + u_1 - v_1 + \ldots + u_p - v_p$, $t =_{\mathbb{ACUD}} t_1 + \ldots + t_n - t'_1 - \ldots - t'_m + u'_1 - v'_1 + \ldots + u'_q - v'_q$, $s_i =_{\mathbb{ACUM}} t_i$ *for* $1 \leq i \leq n$, $u_i =_{\mathbb{ACUM}} v_i$ *for* $1 \leq i \leq p$ *and* $u'_i =_{\mathbb{ACUM}} v'_i$ *for* $1 \leq i \leq q$.

**Proof:** Similar to Observation 5.

Let $\mathcal{I}$ denote the rewriting system $\{x + x \rightarrow x, x + 0 \rightarrow x\}$.

**Observation 8 ($\mathbb{ACUI}$ Terms)** *Let* $+, 0 \in \Sigma$ *and* $- \notin \Sigma$.

*(i)* $\rightarrow_{\mathcal{I}/\mathbb{AC}}$ *is terminating.*

*(ii)* $\rightarrow_{\mathcal{I}/\mathbb{AC}}$ *is locally confluent. Hence from (i) and Lemma 4,* $\rightarrow_{\mathcal{I}/\mathbb{AC}}$ *is confluent.*

*(iii) If* $[s]_{\mathbb{AC}} \rightarrow^*_{\mathcal{I}/\mathbb{AC}} [t]_{\mathbb{AC}}$ *then for some functional terms* $s_1^1, \ldots, s_1^{p_1}, \ldots, s_n^1, \ldots, s_n^{p_n}$ *(*$n \geq 0$ *and* $p_i \geq 1$ *for* $1 \leq i \leq n$*) we have* $s =_{\mathbb{ACU}} s_1^1 + \ldots + s_1^{p_1} + \ldots + s_n^1 + \ldots + s_n^{p_n}$, $t =_{\mathbb{ACU}} s_1^1 + \ldots + s_n^1$ *and* $s_i^j =_{\mathbb{ACUI}} s_i^k$ *for* $1 \leq i \leq n, 1 \leq j, k \leq p_i$.

*(iv) If* $s =_{\mathbb{ACUI}} t$ *then we have some functional terms* $s_1^1, \ldots, s_1^{p_1}, \ldots, s_n^1, \ldots, s_n^{p_n}, t_1^1, \ldots, t_1^{q_1}, \ldots, t_n^1, \ldots, t_n^{q_n}$ *(*$n \geq 0$ *and* $p_i, q_i \geq 1$ *for* $1 \leq i \leq n$*) such that* $s =_{\mathbb{ACU}} s_1^1 + \ldots + s_1^{p_1} + \ldots + s_n^1 + \ldots + s_n^{p_n}$, $t =_{\mathbb{ACU}} t_1^1 + \ldots + t_1^{q_1} + \ldots + t_n^1 + \ldots + t_n^{q_n}$, $s_i^j =_{\mathbb{ACUI}} t_i^k$ *for* $1 \leq i \leq n, 1 \leq j \leq p_i, 1 \leq k \leq q_i$.

**Proof:** Similar to Observation 5. □

## 3.3 Tree Automata Clauses

Let us now look at the special forms of definite clauses (3.1) that we need for our automata.

### 3.3.1 Clauses of One-Way Equational Tree Automata

We have already met the following two kinds of clauses in Chapter 1 :

$$P(f(x_1, ..., x_n)) \Leftarrow P_1(x_1) \wedge ... \wedge P_n(x_n) \quad \textit{pop clause} \tag{3.2}$$
$$P(x) \Leftarrow P_1(x) \quad \textit{epsilon clause} \tag{3.3}$$

In clause 3.2, the variables $x_1, ..., x_n$ are distinct. We define *one-way equational tree automata* to consist of pop clauses and epsilon clauses (of course, we also specify the specific equational theory that we want the automata to work with).

Depending on whether $f$ is a free symbol, a constant symbol or one of the equational symbols, the pop clauses 3.2 can be classified into the following types :

$$P(f(x_1, ..., x_n)) \Leftarrow P_1(x_1) \wedge ... \wedge P_n(x_n), f \text{ being free} \quad \textit{free pop clause} \tag{3.4}$$
$$P(a), a \text{ being a constant} \quad \textit{base clause} \tag{3.5}$$
$$P(x + y) \Leftarrow P_1(x) \wedge P_2(y) \quad \textit{+-pop clause} \tag{3.6}$$
$$P(-x) \Leftarrow Q(x) \quad \textit{minus clause} \tag{3.7}$$
$$P(0) \quad \textit{zero clause} \tag{3.8}$$

Note that clauses 3.5 are a special case of clauses 3.4.

### 3.3.2 Clauses of Alternating Automata

Next we introduce intersection clauses :

$$P(x) \Leftarrow P_1(x) \wedge P_2(x) \quad \textit{intersection clause} \tag{3.9}$$

*An alternating automaton* is a set of intersection clauses and one-way automata clauses. The above clause can be read as "if term $t$ is accepted at each of the states $P_1$ and $P_2$ then $t$ is accepted at state $P$".

### 3.3.3 Clauses of Two-Way Automata

*General two-way equational tree automata* are sets of clauses of one-way equational tree automata as well as of clauses of the form

$$P(x_i) \Leftarrow Q(f(x_1, ..., x_n)) \wedge P_1(x_{i_1}) \wedge ... \wedge P_k(x_{i_k}), \quad \textit{general push clause} \tag{3.10}$$
$$1 \leq i, i_1, ..., i_k \leq n$$

where the variables $x_1, ..., x_n$ are distinct.

Depending upon whether the symbol $f$ in clause (3.10) is free or the symbol $+$, it will be called a general free push clause or a general $+$-push clause :

$$P(x_i) \Leftarrow Q(f(x_1, ..., x_n)) \wedge P_1(x_{i_1}) \wedge ... \wedge P_k(x_{i_k}), \quad \textit{general free push clause} \tag{3.11}$$
$$f \text{ being free }, 1 \leq i, i_1, ..., i_k \leq n$$
$$P(x_i) \Leftarrow Q(x_1 + x_2) \wedge P_1(x_{i_1}) \wedge ... \wedge P_k(x_{i_k}), \quad \textit{general +-push clause} \tag{3.12}$$
$$1 \leq i, i_1, ..., i_k \leq 2$$

As remarked before, the use of the term 'general' above is intentional : we will see that in the equational case, the general push clauses are problematic and easily lead to undecidability. This leads us to study some restricted forms of general push clauses to keep decidability. We study the following three forms of clauses :

$$P(x_i) \Leftarrow Q(f(x_1, ..., x_n)) \wedge P_1(x_{i_1}) \wedge ... \wedge P_k(x_{i_k}), \quad \textit{free push clause} \tag{3.13}$$
$$f \text{ being free}, 1 \leq i_1, ..., i_k \leq n, i \in \{1, ..., n\} \setminus \{i_1, ..., i_k\}$$
$$P(x) \Leftarrow P_1(x + y) \wedge P_2(y) \quad \textit{+-push clause} \tag{3.14}$$
$$P(x) \Leftarrow Q(x + y) \quad \textit{standard +-push clause} \tag{3.15}$$

Note that we don't need to study push clauses involving the $-$ symbol, i.e. clauses of the form $P(x) \Leftarrow P(-x)$. This is because this clause is equivalent to clause 3.7 if the equational theory that we are dealing with contains either the axioms $\mathbb{ACUM}$, or the axiom $\mathbb{D}$. This condition is satisfied by all the equational theories involving the $-$ symbol that we deal with. *A two-way equational tree automaton* (to be distinguished from general two-way equational tree automaton) is a set of free push clauses and one-way equational tree automata clauses. We define *constant-only* automata to be those one-way automata for which the all free symbols in the signature are constants (so that all free pop clauses are base clauses).

Observe that general two-way clauses are more expressive than intersection clauses. An intersection clause $P(x) \Leftarrow P_1(x) \wedge P_2(x)$ can be translated to the clauses $Q(f(x)) \Leftarrow P_1(x)$ and $P(x) \Leftarrow Q(f(x)) \wedge P_2(x)$ for a free unary $f$ and fresh predicate $Q$. Hence we have the following result (we have restricted ourselves to the theory $\mathbb{ACU}$ ; this is also we are going to use) :

**Observation 9** *If automaton $\mathcal{A}$ consists only of intersection clauses and clauses of general two-way automata, then we can find a general two-way automaton $\mathcal{A}'$ such that for every state $P$ in $\mathcal{A}$, $\mathcal{L}_P(\mathcal{A}'/\mathbb{ACU}) = \mathcal{L}_P(\mathcal{A}/\mathbb{ACU})$.*

However this translation required a free unary symbol. We also sometimes need to restrict our signature such that all free symbols are constants. In such a case we can translate the intersection clause $P(x) \Leftarrow P_1(x) \wedge P_2(x)$ as the general $+$-push clause $P(x) \Leftarrow Q(x+y) \wedge Q_a(y) \wedge P_2(x)$ and the clauses $P_a(a)$ and $Q(x+y) \Leftarrow P_1(x) \wedge P_a(y)$ where $a$ is some constant in the signature, and $Q_a$ is a fresh predicate. As a result :

**Observation 10** *If automaton $\mathcal{A}$ consists only of intersection clauses and clauses of general two-way automata and all free symbols in the signature are constants, then we can find a general two-way automaton $\mathcal{A}'$ on the same signature such that for every state $P$ in $\mathcal{A}$, $\mathcal{L}_P(\mathcal{A}'/\mathbb{ACU}) = \mathcal{L}_P(\mathcal{A}/\mathbb{ACU})$.*

Given a general two-way automaton $\mathcal{A}$, we define $\mathcal{A}_{eq}$ (the equational part of $\mathcal{A}$) to be the set of $+$-pop clauses, minus clauses, zero clauses and general $+$-push clauses in $\mathcal{A}$. The remaining part is called $\mathcal{A}_{free}$ (the free or non-equational part of $\mathcal{A}$). Hence $\mathcal{A}_{free}$ contains the free pop clauses (and base clauses) and general free push clauses of $\mathcal{A}$. We also define $\mathcal{A}_{one-way}$ (the one-way part of $\mathcal{A}$) to be the set of all pop clauses and epsilon clauses in $\mathcal{A}$.

Finally we will also sometimes use clauses of other form which can easily be expanded into equivalent automata clauses. For example the clause

$$P(f(0) + a)$$

can be expanded to the clauses

$$P_0(0) \quad P_a(a) \quad P_f(f(x)) \Leftarrow P_0(x) \quad P(x+y) \Leftarrow P_f(x) \wedge P_a(y)$$

where $P_0, P_a, P_f$ are fresh states. In general any clause of the form $P(t)$ for some ground term $t$ can be expanded to automata clauses. Clause

$$P(x+t) \Leftarrow Q(x)$$

can be translated to

$$P(x+y) \Leftarrow Q(x) \wedge Q_t(t) \qquad Q_t(t)$$

where $Q_t$ is a fresh state. Clause

$$P(f(x+y)) \Leftarrow Q(g(x)) \wedge R(y)$$

can be translated to

$$Q_g(x) \Leftarrow Q(g(x)) \quad Q_+(x+y) \Leftarrow Q_g(x) \wedge R(y) \quad P(f(x)) \Leftarrow Q_+(x)$$

where $Q_g, Q_+$ are fresh predicates. For ground terms $s$ and $t$, the clause

$$P(f(x+s)) \Leftarrow Q(g(x+t))$$

can be translated to

$$\begin{array}{ccc} Q_g(x) \Leftarrow Q(g(x)) & Q_t(t) & Q_+(x) \Leftarrow Q_g(x+y) \wedge Q_t(t) \\ Q_s(s) & P(f(x+y)) \Leftarrow Q_+(x) \wedge Q_s(y) & \end{array} \; c$$

where $Q_g, Q_t, Q_+, Q_s$ are fresh states.

## 3.4 Other Formalisms of Equational Tree Automata

We end this chapter with a discussion on some other formalisms similar to our equational tree automata. Tree automata modulo $\mathbb{AC}$ have been considered several times [Lug98, Lug03, DZL03, Ohs01, OT02], though not all these notions coincide. The automata of [Lug98] have additional sort restrictions, but are also extended with a rich constraints language. Recent work by Lugiez [Lug03] extends it to include equality and counting constraints, providing a rich framework that includes most known proposals for *one-way* $\mathbb{AC}$ tree automata with decidable emptiness problems. This framework is however incomparable to ours : while Lugiez's automata accommodate equality tests naturally, we shall see that our two-way automata cannot avoid undecidability in the presence of equality constraints. These automata, also called *multitree automata* can also be thought of as extensions of our one-way $\mathbb{AC}$ (or $\mathbb{ACU}$) automata with a rich set of constraints. The multitree automata also enjoy the good properties of classical tree automata : they are closed under Boolean operations and emptiness is decidable. [DZL03] presents *sheaves automata* which are a tailored version of multitree automata and are aimed at manipulation of XML Schemas.

These extensions of automata are motivated more by adding constraints to tree automata, instead of considering equational tree automata for arbitrary equational theories. It happens that one-way $\mathbb{ACU}$ automata can easily be thought of as tree automata with such a constraints language.

On the other hand Ohsaki et. al [Ohs01, OT02, OST03] consider a larger framework of $\mathbb{E}$ tree automata, where $\mathbb{E}$ is an equational theory. The $\mathbb{E}$ tree automata of Ohsaki are essentially rewriting systems, together with an equational theory $\mathbb{E}$. We have already seen in Chapter 2 how one-way tree automata can be thought of as rewriting systems. Whereas we have chosen to consider tree automata as logic programs, so that equational tree automata are logic programs modulo an equational tree theory, Ohsaki adopts viewpoint of the tree automata as rewriting systems so that his equational tree automata are rewriting systems modulo equational theories. The *regular tree automata* of Ohsaki consist of rewriting rules of the form

$$f(q_1, \ldots, q_n) \to q$$

(The clauses $q \to q'$ can also be added without increasing expressiveness.) In the non-equational case, we have seen that these are exactly the clauses of one-way tree automata. Hence the regular $\mathbb{E}$ automata of Ohsaki can be thought of as the counterpart of our one-way equational tree automata. However this analogy does not go too far in the equational case. If we restrict ourselves to linear theories (like $\mathbb{A}$, $\mathbb{C}$, $\mathbb{AC}$, $\mathbb{ACU}$), then Ohsaki's regular $\mathbb{E}$ tree automata coincide with our one-way $\mathbb{E}$ tree automata. This follows from the results that firstly, if $\mathbb{E}$ is linear Ohsaki's regular $\mathbb{E}$ tree automata accept exactly the $\mathbb{E}$-closure of the corresponding non-equational tree automata, and secondly as we show in Lemma 19 (Chapter 7), our one-way $\mathbb{E}$ tree automata accept exactly the $\mathbb{E}$-closure of the corresponding non-equational tree automata. This in particular means that for linear $\mathbb{E}$, emptiness of regular $\mathbb{E}$ tree automata is decidable.

However this correspondence between the two notions of equational tree automata does not hold for non-linear theories. While the Lemma 19 holds for non-linear theories also, the corresponding result for the regular tree automata of Ohsaki does *not* hold in general for non-linear theories (like $\mathbb{ACUX}$, $\mathbb{ACUM}$, $\mathbb{ACUI}$.) Here is a counter-example :

**Example 4** *In Ohsaki's equational tree automata, with a single transition rule $q_1 + q_1 \to q$ and with the theory $\mathbb{ACUX}$, we have $0 =_{\mathbb{ACUX}} q_1 + q_1 \to q$, meaning $0$ is accepted at $q$. In our case, the corresponding one-way automaton has $q(x + y) \Leftarrow q_1(x) \wedge q_2(y)$ as the only clause. Modulo the theory $\mathbb{ACUX}$ (or modulo any other theory) no term is accepted at any of the two states. If we denote Ohsaki's automata by $\mathcal{A}_{Ohsaki}$ and our automata by $\mathcal{A}_{our}$, and we let $q$ be the final state in both automata, then we have*

$$\mathcal{L}(\mathcal{A}_{Ohsaki}/\mathbb{ACUX}) = \mathbb{ACUX}(\{0\}) \neq \mathbb{ACUX}(\mathcal{L}(\mathcal{A}_{Ohsaki}/\emptyset)) = \emptyset$$

*whereas*

$$\mathcal{L}(\mathcal{A}_{our}/\mathbb{ACUX}) = \mathbb{ACUX}(\mathcal{L}(\mathcal{A}_{our}/\emptyset)) = \emptyset$$

*where the empty theory is denoted as $\emptyset$.*

Besides Ohsaki also defines "$\mathbb{E}$ tree automata" (not to be confused with our usage of the same expression) which extend regular $\mathbb{E}$ automata by rewriting rules of the form

$$f(p_1, \ldots, q_n) \to f(q_1, \ldots, q_n)$$

where $f$ is a functional symbol and $p_i, q_i$'s are states. As far as the $\mathbb{AC}$ case is concerned, it is remarked in [Ohs01] that Ohsaki's (non-regular) $\mathbb{AC}$ tree automata are strictly more expressive than regular $\mathbb{AC}$ tree automata. This means that Ohsaki's non-regular $\mathbb{AC}$ tree automata are strictly more expressive than our one-way $\mathbb{AC}$ tree automata. On the other hand our general two way $\mathbb{AC}$ tree automata have been shown to recognize all recursively enumerable sets in Chapter 6, whereas emptiness of Ohsaki's non-regular $\mathbb{AC}$ tree automata is decidable. This means that our general two-way $\mathbb{AC}$ tree automata are strictly more expressive than Ohsaki's $\mathbb{AC}$ tree automata.

For arbitrary $\mathbb{E}$ we do not know the relation between our automata and Ohsaki's automata, and the two notions appear rather dissimilar. One of the key differences is that while we extend the traditional correspondence between tree automata and logic programs from the non-equational case to the equational case, Ohaski's automata do not do so, as they mix equality on terms with equality on states, as illustrated by the above example. We have made this choice keeping in mind our goal of modeling cryptographic protocols, where the states (predicates) represent the set of messages known to an intruder, and hence they cannot be mixed with the terms which represent messages.

Note that while all the papers cited above deal with one-way equational tree automata (although some of them are incomparable with our two-way automata) our work seems to be the first one to deal with equational variants of two-way tree automata. As we will see in this thesis, unlike in the non-equational case where all classes of alternating two-way automata have the same expressiveness as one-way tree automata, the various subclasses two-way equational tree automata exhibit a wide range of behavior from undecidability and encoding of all recursively enumerable sets to translation into one-way equational tree automata. As far as the one-way case is concerned, most of the previous work has been concerned with the theory $\mathbb{AC}$ ([OST03] deals with $\mathbb{A}$ tree automata), the various extensions of the $\mathbb{AC}$ theory like $\mathbb{ACUX}$, $\mathbb{ACUM}$, $\mathbb{ACUI}$ that we consider in this thesis have not been considered before in the context of equational tree automata. For these reasons, this work can be regarded as the first general treatment of one-way and two-way equational tree automata for $\mathbb{AC}$-like theories.

While we consider tree automata as logic programs and extend them by adding equational theories, there has also been much work on *equational logic programs* [Han94] as such, which are logic programs in which the special equality predicate also appears, so that the equational theory can be coded in the logic program itself. However our equational tree automata differ in that the equality symbol does not occur in the logic programs (in fact we only consider logic programs with unary predicates), and we separately consider an equational theory in which the equality predicate occurs. Moreover we are not aware of any work on finding decision procedures or closure properties for subclasses of equational logic programs, especially for the ones corresponding to our automata.

Finally we have found that closure under Boolean operations has also been shown for the *multiset automata* of Colcombet [Col02], which correspond to the subclass of our one-way $ACU$ automata in which all symbols other than $+, 0$ are unary, and were introduced for studying process rewrite systems.

To avoid confusion we clarify that the "two-way automata" of [Var98] are a different notion from ours. For example in the string case their two-wayness refers to the fact that they have a fixed input on a tape and the transitions allow the read head on the tape to move in either direction. In our automata we don't have any concept of a fixed input on which the automata can work. During the runs of our automata, the tree can grow by addition of function symbols at the top, or shorten by removal of function symbols from the top. This is the two-wayness that our terminology refers to. In particular no obvious notion of "equational" extensions of the two-way automata of [Var98] appears to us.

## Chapitre 4

# Automates d’arbres modulo les théories $\mathbb{A}$ et $\mathbb{C}$ (Tree Automata Modulo Theories $\mathbb{A}$ and $\mathbb{C}$)

Bien que le sujet principal de cette thèse soit les automates d’arbres modulo les théories $\mathbb{AC}$ et ses extensions, il est aussi intéressant de savoir ce qui se passe dans les cas des restrictions de la théorie $\mathbb{AC}$, à savoir les théories $\mathbb{A}$ et $\mathbb{C}$. Dans ce chapitre nous étudions les automates d’arbres modulo ces deux théories. Essentiellement nous montrons qu’aucun de ces deux cas n’est intéressant, mais pour des raisons exactement contraires : alors que les automates d’arbres modulo $\mathbb{C}$ sont facilement traduits (en temps linéaire) en des automates d’arbres non-équationnels équivalents, tous les problèmes intéressants sont indécidables pour les automates d’arbres modulo $\mathbb{A}$. Il nous restera à traiter la théorie $\mathbb{AC}$ et ses extensions, ce que nous ferons dans le reste de la thèse.

Although the main subject of this thesis is tree automata modulo the theories $\mathbb{AC}$ and its extensions, it is also interesting to know what happens in the cases of the restrictions of the theory $\mathbb{AC}$, namely the theories $\mathbb{A}$ and $\mathbb{C}$. In this chapter we study the tree automata modulo these two theories. We essentially show that neither of the two cases is interesting, although for exactly opposite reasons : while the $\mathbb{C}$ tree automata are easily translated (in linear time) to equivalent non-equational tree automata, all the interesting problems are undecidable for $\mathbb{A}$ tree automata. This leaves us to deal with the theories $\mathbb{AC}$ and their extensions in the rest of the thesis.

## 4.1 $\mathbb{C}$ Tree Automata

In this section we show that $\mathbb{C}$ tree automata are reducible in linear time to non-equational tree automata accepting the same language. Note that this statement makes sense since we have chosen to consider equational tree automata as accepting $\mathbb{E}$-closed sets of terms instead of as sets of equivalence classes of terms modulo the equational theory.

Let $\mathcal{A}$ be a general two-way tree automata. Define the general two-way tree automata $\mathcal{B}$ to consist of all clauses of $\mathcal{A}$ as well as new clauses $P(x+y) \Leftarrow P_2(x) \wedge P_1(y)$ for every clause $P(x+y) \Leftarrow P_1(x) \wedge P_2(y)$ occurring in $\mathcal{A}$. We show that $\mathcal{A}/\mathbb{C}$ is equivalent to $\mathcal{B}/\emptyset$. Recall that $\emptyset$ denotes the empty theory. One part is easy :

**Lemma 9** *If $P(t)$ is derivable in $\mathcal{B}/\emptyset$ then $P(t)$ is derivable in $\mathcal{A}/\mathbb{C}$.*

**Proof:** We do induction on the size of the derivation $\pi$ of $P(t)$ in $\mathcal{B}/\emptyset$.

**Case 1.** Suppose the last clause used is some definite clause (pop clause, epsilon clause or general push clause) of $\mathcal{A}$, and $\pi$ is of the form :

$$\frac{\dfrac{\pi_1}{P_1(s_1\sigma)} \;\; \ldots \;\; \dfrac{\pi_n}{P_n(s_n\sigma)}}{P(s\sigma)} \, (P(s) \Leftarrow P_1(s_1) \wedge \ldots \wedge P_n(s_n))$$

where $s\sigma = t$. Then by applying induction hypothesis on $\pi_1, \ldots, \pi_n$, we get derivations $\pi'_1, \ldots, \pi'_n$ of $P_1(t_1\sigma), \ldots, P_n(t_n\sigma)$ in $\mathcal{A}/\mathbb{C}$ respectively. Then we get the following derivation of $P(t)$ in $\mathcal{A}/\mathbb{C}$ :

$$\frac{\dfrac{\pi'_1}{P_1(s_1\sigma)} \;\; \ldots \;\; \dfrac{\pi'_n}{P_n(s_n\sigma)}}{P(s\sigma)} \, (P(s) \Leftarrow P_1(s_1) \wedge \ldots \wedge P_n(s_n))$$

**Case 2.** Suppose $t = t_2 + t_1$, $P(x+y) \Leftarrow P_1(x) \wedge P_2(y) \in \mathcal{A}$ and $\pi$ is of the form

$$\frac{\dfrac{\pi_1}{P_2(t_2)} \quad \dfrac{\pi_2}{P_1(t_1)}}{P(t_2+t_1)} \, (P(x+y) \Leftarrow P_2(x) \wedge P_1(y))$$

By applying induction hypothesis on $\pi_1$ and $\pi_2$ we get derivations $\pi'_1$ and $\pi'_2$ of $P_2(t_2)$ and $P_1(t_1)$ respectively in $\mathcal{A}/\mathbb{C}$. Then we get the following derivation of $P(t)$ in $\mathcal{A}/\mathbb{C}$.

$$\cfrac{\dfrac{\dfrac{\pi'_2}{P_1(t_1)} \quad \dfrac{\pi'_1}{P_2(t_2)}}{P(t_1+t_2)} \, (P(x+y) \Leftarrow P_1(x) \wedge P_2(y))}{P(t_2+t_1)} \, (\mathbb{C})$$

$\square$

For the second part, we show the following statement :

**Lemma 10** *If $P(s)$ is derivable in $\mathcal{A}/\mathbb{C}$ and $s =_{\mathbb{C}} t$ then $P(t)$ is derivable in $\mathcal{B}/\emptyset$.*

**Proof:** We do induction on the size of the derivation $\pi$ of $P(s)$ in $\mathcal{A}/\mathbb{C}$.

**Case 1.** Suppose $s = f(s_1, ..., s_n)$ where $f$ is free, and $\pi$ is of the form

$$\dfrac{\dfrac{\pi_1}{P_1(s_1)} \;...\; \dfrac{\pi_n}{P_n(t_n)}}{P(f(s_1, ..., s_n))}\,(P(f(x_1, ..., x_n)) \Leftarrow P_1(x_1) \wedge ... \wedge P_n(x_n))$$

If $s =_{\mathbb{C}} t$ then there are terms $t_1, ..., t_n$ such that $t = f(t_1, ..., t_n)$ and $s_i =_{\mathbb{C}} t_i$ for $1 \leq i \leq n$. By applying induction hypothesis on $\pi_1, ..., \pi_n$, we get derivations $\pi'_1, ..., \pi'_n$ of $P_1(t_1), ..., P_n(t_n)$ respectively in $\mathcal{B}/\emptyset$. Then we have the following derivation of $P(t)$ in $\mathcal{B}/\emptyset$ :

$$\dfrac{\dfrac{\pi'_1}{P_1(t_1)} \;...\; \dfrac{\pi'_n}{P_n(t_n)}}{P(f(t_1, ..., t_n))}\,(P(f(x_1, ..., x_n)) \Leftarrow P_1(x_1) \wedge ... \wedge P_n(x_n))$$

**Case 2.** Suppose $s = s_1 + s_2$, and $\pi$ is of the form

$$\dfrac{\dfrac{\pi_1}{P_1(s_1)} \quad \dfrac{\pi_2}{P_2(s_2)}}{P(s_1 + s_2)}\,(P(x + y) \Leftarrow P_1(x) \wedge P_2(y))$$

If $s =_{\mathbb{C}} t$ then we must have terms $t_1$ and $t_2$ such that $s_1 =_{\mathbb{C}} t_1$ and $s_2 =_{\mathbb{C}} t_2$ and $t = t_l + t_{3-l}$ for some $l \in \{1, 2\}$. By applying induction hypothesis on $\pi_1$ and $\pi_2$ we get derivations $\pi'_1$ and $\pi'_2$ of $P_1(t_1)$ and $P_2(t_2)$ respectively. By definition of $\mathcal{B}$, the clause $P(x + y) \Leftarrow P_l(x) \wedge P_{3-l}(y)$ is in $\mathcal{B}$. Hence we have the following derivation of $P(t)$ in $\mathcal{B}/\emptyset$ :

$$\dfrac{\dfrac{\pi'_1}{P_1(t_1)} \quad \dfrac{\pi'_2}{P_2(t_2)}}{P_l(t_l + t_{3-l})}\,(P(x + y) \Leftarrow P_l(x) \wedge P_{3-l}(y))$$

**Case 3.** Suppose $s = s_i$ and $\pi$ is of the form

$$\dfrac{\dfrac{\delta}{Q(f(s_1, ..., s_n))} \quad \dfrac{\pi_1}{P_1(s_{i_1})} \;...\; \dfrac{\pi_k}{P_k(s_{i_k})} \quad \dfrac{\delta_1}{Q_1(s_i)} \;...\; \dfrac{\delta_p}{Q_p(s_i)}}{P(s_i)}\,(C)$$

where $C = P(x_i) \Leftarrow Q(f(x_1, ..., x_n)) \wedge P_1(x_{i_1}) \wedge ... \wedge P_k(x_{i_k}) \wedge Q_1(x_i) \wedge ... \wedge Q_p(s_i)$ and $i \notin \{i_1, ..., i_k\}$. If $s_i =_{\mathbb{C}} t$ then $f(s_1, ..., s_n) =_{\mathbb{C}} f(s_1, ..., s_{i-1}, t, s_{i+1}, ..., s_n)$. Then by applying induction hypothesis on $\delta, \delta_1, ..., \delta_p$, we get derivations $\delta', \delta'_1, ..., \delta'_p$ of $Q(f(s_1, ..., s_{i-1}, t, s_{i+1}, ..., s_n)), Q_1(t), ..., Q_p(t)$ respectively in $\mathcal{B}/\emptyset$. Then we have the following derivation of $P(t)$ in $\mathcal{B}/\emptyset$ :

$$\dfrac{\dfrac{\delta}{Q(f(s_1, ..., s_{i-1}, t, s_{i+1}, ..., s_n))} \quad \dfrac{\pi_1}{P_1(s_{i_1})} \;...\; \dfrac{\pi_k}{P_k(s_{i_k})} \quad \dfrac{\delta_1}{Q_1(t)} \;...\; \dfrac{\delta_p}{Q_p(t)}}{P(s_i)}\,(C)$$

**Case 4.** Suppose $\pi$ is of the form

$$\cfrac{\cfrac{\pi_1}{P(s')}}{P(s)}\,(\mathbb{C})$$

If $s =_{\mathbb{C}} t$ then $s' =_{\mathbb{C}} t$ and by applying induction hypothesis on $\pi_1$, $P(t)$ is derivable in $\mathcal{B}/\emptyset$ :

$$\cfrac{\cfrac{\pi_1}{P(s')}}{P(t)}\,(\mathbb{C})$$

□

From Lemmas 9 and 10 it follows that $\mathcal{A}$ and $\mathcal{A}'$ accept the same language if we let the final states of $\mathcal{A}$ and $\mathcal{A}'$ to be the same. Also observe that the only new clauses in $\mathcal{A}'$ are $+$-pop clauses. This in particular implies that if $\mathcal{A}'$ is a one-way automaton then $\mathcal{B}$ is also a one-way automaton. Similarly if $\mathcal{A}$ is a two-way automaton or general two-way automaton or constant-only automaton then $\mathcal{B}$ is also a two-way automaton or general two-way automaton or constant-only automaton respectively. We have similar conclusions for other classes of automata considered in the thesis. Finally we observe that $\mathcal{A}'$ is computable in time linear in the number of clauses in $\mathcal{A}$ since we only need to add a $+$-pop clause for every $+$-pop clause in $\mathcal{A}$. We summarize our results in the following theorem :

**Theorem 3** *For every general two-way $\mathbb{C}$ tree automaton $\mathcal{A}$, we can compute in linear time a general two-way automaton $\mathcal{B}$ such that $\mathcal{L}(\mathcal{B}/\emptyset) = \mathcal{L}(\mathcal{A}/\mathbb{C})$. Also the only clauses in $\mathcal{B}$ which are not in $\mathcal{A}$ are $+$-pop clauses.*

This means that studying extensions of tree automata by adding the theory $\mathbb{C}$ is not an interesting exercise.

## 4.2 $\mathbb{A}$ Tree Automata

While we saw in the previous section that $\mathbb{C}$ tree automata accept only regular tree languages and have good decidability properties, the case of $\mathbb{A}$ tree automata is completely different. We show that emptiness of intersection of languages accepted by $\mathbb{A}$ tree automata is undecidable, even when the only symbols in the signature othen that $+$ are constants. This is done by showing that these automata can accept *context free languages* for which the intersection emptiness problem is known to be undecidable [HU79]. First we recall some notions on context-free languages.

**Definition 3 (Context Free Languages)** *A* context free grammar *is a quadruple $(V, T, P, S)$ where $V$ is a finite set of* non-terminals *, $T$ is a finite set of* terminals*, $P$ is a finite set of* production rules *of the form $A \to \alpha$, where $A$ is a nonterminal and $\alpha \in (V \cup T)^*$, and $S \in V$ is the* start symbol*.*

Derivations *in $G$ are defined as follows : we write $\alpha A\gamma \Longrightarrow_G \alpha\beta\gamma$ iff $A \to \beta \in P$ and $\alpha, \gamma \in (V \cup T)^*$. The* language generated *by $G$, denoted $L(G)$, is the set $\{w \mid w \text{ is in } T^* \text{ and } S \Longrightarrow_G^* w\}$.* Context-free languages *are defined to be the languages that can be generated by context-free grammars.*

Now we show a correspondence between constant-only $\mathbb{A}$ tree automata and context free grammars. Let the signature $\Sigma$ contain only constants besides the symbol $+$. As in the $\mathbb{AC}$ case, we use the notation $a_1+\ldots+a_n$ to denote a term of the form $(\ldots((a_1+a_2)+a_2)+\ldots)$. For any term $t \in T(\Sigma)$, $t =_{\mathbb{A}} a_1 + \ldots + a_n$ for some constants $a_1, \ldots, a_n \in \Sigma, n \geq 1$. This representation is also unique, unlike in the $\mathbb{AC}$ case where the order of the summands did not matter. Let $T$ be the set of constants of $\Sigma$. Hence the function $str : T(\Sigma) \rightarrow T^*$ which maps every term $t$ to the string $a_1 \ldots a_n \in T^*$ where $t =_{\mathbb{AC}} a_1 + \ldots + a_n$ is well defined. We have $str(t_1) = str(t_2)$ iff $t_1 =_{\mathbb{A}} t_2$. The range of $str$ is every string in $T^*$ except the empty string $\epsilon$. Hence there is a one-one correspondence between $\mathbb{A}$-closed tree languages built on $\Sigma$, and the subsets of $T^*$ which do not contain the empty string, with the correspondence being defined by $str$.

Given a constant only automaton $\mathcal{A}$ on signature $\Sigma$, we define a context-free grammar $gram(\mathcal{A})$ as follows. The non-terminals are the predicates of $\mathcal{A}$. $T$ is the set of terminals. The start symbol $S$ is the final state of $\mathcal{A}$. We have production rules in $gram(\mathcal{A})$ corresponding to clauses of $\mathcal{A}$ as indicated in the table below.

| | |
|---|---|
| $P(x+y) \Leftarrow P_1(x) \wedge P_2(y)$ | $P \rightarrow P_1P_2$ |
| $P(a)$ | $P \rightarrow a$ |
| $P(x) \Leftarrow P_1(x)$ | $P \rightarrow P_1$ |

Similarly given a context free grammar $G = (V, T, P, S)$ such that $P$ contains only production rules of the form $P \rightarrow P_1P_2$, $P \rightarrow a$ and $P \rightarrow P_1$, we have a constant-only automaton $gram^{-1}(G)$ (which is uniquely defined upto renaming of variables in clauses) on signature $T \cup \{+\}$, such that its set of predicates is $V$ and the final state is $S$.

First of all we can show the following easy result by induction on the derivations in a context free grammar,

**Lemma 11** *Let $G = (V, T, P, S)$ be a context free grammar. For $x_1, \ldots, x_n \in V \cup T$ and $\alpha \in (V \cup T)^*$, $x_1 \ldots x_n \Longrightarrow_G^* \alpha$ iff for some $\alpha_1, \ldots, \alpha_n \in (V \cup T)^*$ we have $x_i \Longrightarrow_G^* \alpha_i$ for $1 \leq i \leq n$.*

we can now show

**Lemma 12** *$P(t)$ is derivable in $\mathcal{A}/\mathbb{A}$ iff $P \Longrightarrow_{gram(\mathcal{A})}^* str(t)$. Hence $str(\mathcal{L}(\mathcal{A}/\mathbb{A})) = L(gram(\mathcal{A}))$.*

**Proof:** To show the "only if" part, we do induction on the length of the derivation of $str(t)$ in $gram(a)$. If $t = a$ and the derivation is simply $P \Longrightarrow a$ then we use the clause $P(a)$ to get the required result. If the derivation is of the form $P \Longrightarrow_{gram(\mathcal{A})} P_1 \Longrightarrow_{gram(\mathcal{A})}^* str(t)$ then by induction hypothesis $P_1(t)$ is derivable in $\mathcal{A}/\mathbb{A}$. We then use the clause $P(x) \Leftarrow P_1(x)$ to get a derivation of $P(t)$. If the derivation is of the form $P \Longrightarrow_{gram(\mathcal{A})} P_1P_2 \Longrightarrow_{gram(\mathcal{A})}^* str(t)$ then by Lemma 11 we have some $t_1, t_2$ such that $t =_{\mathbb{A}} t_1 + t_2$ and we have $P_1 \Longrightarrow_{gram(\mathcal{A})}^* str(t_1)$ and $P_2 \Longrightarrow_{gram(\mathcal{A})}^* str(t_2)$. By induction hypothesis, $P_1(t_1)$ and $P_2(t_2)$ are derivable in $\mathcal{A}/\mathbb{A}$. Then we use the clause $P(x_1 + x_2) \Leftarrow P_1(x) \wedge P_2(y)$ to get a derivation of $P(t)$.

To show the "if" part, we do induction on the size of the derivation of $P(t)$. If $P(a)$ is derivable using the same clause, then we gave $P \Longrightarrow_{gram(\mathcal{A})} a$. If $P(t)$ is derivable using $P(x) \Leftarrow P_1(x)$ as the last clause, then by induction hypothesis, $P_1 \Longrightarrow_{gram(\mathcal{A})}^* str(t)$, hence we have $P \Longrightarrow_{gram(\mathcal{A})} P_1 \Longrightarrow_{gram(\mathcal{A})}^* str(t)$. If $P(t_1 + t_2)$ is derivable using $P(x+y) \Leftarrow P_1(x) \wedge P_2(y)$ as the last clause, then by induction hypothesis we have $P_1 \Longrightarrow_{gram(\mathcal{A})}^* str(t_1)$ and $P_2 \Longrightarrow_{gram(\mathcal{A})}^* str(t_2)$. Hence we have $P \Longrightarrow_{gram(\mathcal{A})} P_1P_2 \Longrightarrow_{gram(\mathcal{A})}^* str(t)$. □

We recall the following result on context free languages :

**Lemma 13 ([HU79])** *Any context-free language not containing $\epsilon$ is generated by a context free grammar in which all productions are of the form $A \rightarrow BC$ or $A \rightarrow a$.*

Grammars of the above form are said to be in *Chomsky normal form (CNF)*. In fact given any context-free grammar $G$ we can effectively obtain a grammar $G'$ in CNF such that $L(G') = L(G) \setminus \{\epsilon\}$. We have the following characterization of the languages accepted by constant-only $\mathbb{A}$-automata :

**Theorem 4** *Constant only $\mathbb{A}$ automata on signature $\Sigma$ accept exactly the languages $\mathcal{L}$ such that $str(\mathcal{L})$ is a context-free language on symbols from $\Sigma \setminus \{+\}$ and not containing $\epsilon$.*

Since context free languages are not closed under intersection, nor under complementation [HU79], it follows that the class of context free languages not containing $\epsilon$ are also not closed under intersection nor under complementation. Hence :

**Corollary 2** *Constant-only $\mathbb{A}$ automata are closed under union, but are not closed under intersection, nor under complementation.*

Finally we show the undecidability of intersection emptiness of the languages accepted by these automata.

**Theorem 5** *Given two constant-only $\mathbb{A}$ automata $\mathcal{A}_1$ and $\mathcal{A}_2$, it is undecidable whether $\mathcal{L}(\mathcal{A}_1/\mathbb{A}) \cap \mathcal{L}(\mathcal{A}_2/\mathbb{A}) = \emptyset$.*

**Proof:** Given two context-free grammars $G_1$ and $G_2$ it is undecidable [HU79] $L(G_1) \cap L(G_2) = \emptyset$. We reduce it to the problem of deciding whether $\mathcal{L}(\mathcal{A}_1/\mathbb{A}) \cap \mathcal{L}(\mathcal{A}_2/\mathbb{A}) = \emptyset$ for two constant-only automata $\mathcal{A}_1$ and $\mathcal{A}_2$. By Lemma 13 the problem remains undecidable when $G_1$ and $G_2$ are restricted to be in $CNF$, since it is decidable whether the string $\epsilon$ belongs to the language generated by a context-free grammar. As a result, we have automata $gram^{-1}(G_1)$ and $gram^{-1}(G_2)$ such that by Lemma 12, $str(\mathcal{L}(gram^{-1}(G_1)/\mathbb{A})) = L(G_1)$ and $str(\mathcal{L}(gram^{-1}(G_2)/\mathbb{A}) = L(G_2)$. Hence $L(G_1) \cap L(G_2) = \emptyset$ iff $str(\mathcal{L}(gram^{-1}(G_1)/\mathbb{A})) \cap str(\mathcal{L}(gram^{-1}(G_2)/\mathbb{A})) = \emptyset$ iff $\mathcal{L}(gram^{-1}(G_1)/\mathbb{A}) \cap \mathcal{L}(gram^{-1}(G_2)/\mathbb{A}) = \emptyset$. This completes the reduction. □

## 4.3 Conclusion

We have dealt with the equational tree automata modulo the theories $\mathbb{A}$ and $\mathbb{C}$. We showed that $\mathbb{C}$ tree automata can be converted in linear time to non-equational tree automata. On the other hand, $\mathbb{A}$ tree automata, even in the constant-only case, accept context free languages, implying undecidability intersection emptiness, as well as the fact that constant-only $\mathbb{A}$ tree automata are closed under union, but are not closed under intersection nor under complementation.

# Chapitre 5

# Application des automates d'arbres équationnels (Application of Equational Tree Automata)

Il est naturel de se demander à quoi servent les automates bidirectionnels (possiblement alternants) modulo $\mathbb{E}$. Les automates d'arbres modulo $\mathbb{AC}$ ont été récemment utilisés en vérification de schémas XML (voir par exemple [Lug03, DZL03]). Dans cette thèse nous n'avons pas regardé en detail les applications aux schémas XML, et les lecteurs intéresses peuvent consulter [Lug03, DZL03]. Notre motivation principale en ce qui concerne l'introduction de ces variantes équationnelles des automates bidirectionnels était de pouvoir modéliser des protocoles cryptographiques qui utilisent des primitives cryptographiques non parfaites. Dans de telles applications, la bidirectionnalité est indispensable. Nous l'illustrons par une étude de cas dans le domaine de la vérification de protocoles cryptographiques. L'exemple que nous avons choisi est celui du protocole de Diffie-Hellman en groupe, utilisé par un groupe de participants pour se mettre d'accord sur une clé commune, non divulguée hors du groupe, et telle qu'aucun sous-groupe strict du groupe n'a pu former de coalition pour créer la clé et l'imposer au reste du groupe.

It is natural to ask what uses two-way $\mathbb{E}$-tree automata (possibly alternating) may have. $\mathbb{AC}$-tree automata have enjoyed quite some interest recently, insofar as checking XML Schemas for example seems to require $\mathbb{AC}$-tree automata techniques (see, e.g., [Lug03, DZL03]). In this thesis we have not focused on applications in XML Schemas, and we refer the interested reader to [Lug03, DZL03]. Our primary motivation for introducing these equational variants of two-way tree automata was to be able model cryptographic protocols which use imperfect cryptographic primitives. For these applications, two-wayness is indispensable. We illustrate this with a case study in the field of cryptographic protocol verification.

## 5.1 Group Diffie-Hellman Protocols

This example is a so-called group key agreement scheme. To keep the exposition short, we only mention salient features exhibiting the role of two-way $\mathbb{AC}$-tree automata. We feel that this example, where we encode the Diffie-Hellman primitive (modular exponentiation) with the help of an $\mathbb{AC}$ symbol to represent multiplication of exponents, is more faithful to actual implementations than previous models, e.g., [KFK97]. Arguably, the theory of Abelian groups would be suitable to the task, too ; see below.

Consider the initial key agreement protocol IKA.1 [STW00] (formerly known as GDH.2), used to create an initial group key in the CLIQUES protocol suite. The goal is for a group of agents $\mathcal{M}_1$, ..., $\mathcal{M}_k$ to obtain a common key that they can use for further communication. This key should be unavailable to external eavesdroppers. (An *eavesdropper* is an intruder who may only listen to communication channels, but not forge messages, remove messages or redirect channels.) Moreover, no agent should be able to decide of the value of the key for the others ; in general, no proper subset of the agents should be able to collude to create the common key.

We study the IKA.1 protocol, in particular because it has attracted some attention recently. IKA.1 is based on a Diffie-Hellman scheme [DH76], which works as follows. We take the standpoint of a Dolev-Yao-like model [DY83], where terms—here, modulo $\mathbb{ACU}$—are used to denote messages.

First, assume that there is a free unary function symbol $e$, which will be used as a way of encapsulating some secret—a kind of cryptographic hash function. Assume $+$ is some $\mathbb{ACU}$ symbol, with unit 0. Then the signature $\Sigma$ consists of $e$, $+$, $0$. We require that every participant of the protocols to come, whether honest or dishonest, can compute $e(M)$ from any message $M$, and $e(M + M')$ from $e(M)$ and $M'$, but not more. Then for two agents $\mathcal{M}_1$ and $\mathcal{M}_2$ to get a common key, $\mathcal{M}_1$ chooses some secret $M_1$, sends $\mathcal{M}_2$ the message $e(M_1)$ ; then $\mathcal{M}_2$ chooses some other secret $M_2$, sends $\mathcal{M}_1$ the message $e(M_2)$ ; finally both can compute the common secret $e(M_1 + M_2)$ : $\mathcal{M}_1$ can compute it from $M_1$ and $e(M_2)$, $\mathcal{M}_2$ can compute it from $e(M_1)$ and $M_2$. Note that $+$ has to be commutative for this to succeed.

This can be implemented using modular exponentiation [DH76], by letting $e(M)$ be coded as $\alpha^M \bmod N$ for well-chosen numbers $\alpha$ and $N$, $+$ as multiplication, and $0$ as the identity element for of multiplication : to get $e(M+M') = \alpha^{M.M'} \bmod N$ from $e(M) = \alpha^M \bmod N$, just raise the latter to $M'$ mod $N$. Modern proposals implement $e(M)$ as $g^M$, where $g$ is a generator of some group, not necessarily $(\mathbb{Z}/N\mathbb{Z})^*$, typically groups generated by elliptic curves or hyperelliptic curves mod $N$.

To describe the IKA.1 protocol [STW00], we shall not just use the function symbols $e$, $+$, and $0$, but also a binary function $cons$ and a constant $nil$ to represent lists. We shall also use additional constants, to be introduced later. We abbreviate $cons(M_1, cons(M_2, \ldots, cons(M_n, nil) \ldots))$ as $M_1; M_2; \ldots; M_n$. For simplicity, assume we have 3 members in the group, $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$.

First, IKA.1 starts with an so-called *upflow* phase : $\mathcal{M}_1$ sends $\mathcal{M}_2$ the message $e(N_1)$, where $N_1$ is a fresh nonce ; $N_1$ is modeled, as usual [Mon99], as a new constant. Then $\mathcal{M}_2$ sends $e(N_2); e(N_1); e(N_1 + N_2)$ to $\mathcal{M}_3$, where $N_2$ is another fresh nonce (modeled as another new constant $N_2 \neq N_1$). This is possible due to our assumptions that anybody can build $e(M)$ from $M$ and $e(M + M')$ from $e(M)$ and $M'$.

Once this is done, $\mathcal{M}_3$ starts the *downflow* phase, and broadcasts $e(N_2 + N_3); e(N_1 + N_3)$, from which all members can compute the group key $e(N_1 + N_2 + N_3)$. ($N_3$ is a third fresh nonce created by $\mathcal{M}_3$.)

All possible interleaved executions of the protocol can be described using Horn clauses modulo $\mathbb{ACU}$, and we claim that the resulting set of clauses is a two-way $\mathbb{ACU}$-automaton. Let us write selected clauses from this set.

To model communication, introduce predicates $ch_C$ for each configuration $C$ that is reachable in an interleaved run of $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$. The formula $ch_C(M)$ is meant to hold if and only if $M$ is a possible message present on some communication channel when in configuration $C$. We also create distinct predicate $k_C$ such that $k_C(M)$ is meant to hold when $M$ is deducible, in a sense inspired by Dolev and Yao [DY83], from all messages present on the communication channels at or before configuration $C$.

As such, for every reachable configuration $C$, we generate the clauses :

| | | |
|---|---|---|
| $k_C(e(0))$ | Intruder knows $e(0)$ | (5.1) |
| $k_C(e(x + y)) \Leftarrow k_C(e(x)) \wedge k_C(y)$ | Intruder can exponentiate | (5.2) |
| $k_C(nil)$ | Intruder knows the empty list | (5.3) |
| $k_C(cons(x, y)) \Leftarrow k_C(x), k_C(y)$ | Intruder can build lists | (5.4) |
| $k_C(x) \Leftarrow k_C(cons(x, y))$ | Intruder can read heads | (5.5) |
| $k_C(y) \Leftarrow k_C(cons(x, y))$ | Intruder can read tails | (5.6) |

We shall define several different intrusion models. This will interfere notably on the clauses we shall generate to define $ch_C$. Whatever the model, we shall generate the clauses

| | | |
|---|---|---|
| $k_C(x) \Leftarrow ch_C(x)$ | Intruder spies on every channel | (5.7) |
| $k_C(x) \Leftarrow k_{C'}(x)$ | Intruder remembers past messages | (5.8) |

for any reachable configuration $C$, and where $C'$ is the predecessor of $C$ (if any) in (5.8).

The most benign model from a security viewpoint is the *pure eavesdropper model*, where the intruder does not interfere with communication, except for remembering all messages exchanged by the honest agents $\mathcal{M}_i$. If $\mathcal{M}_i$ sends message $M$ under some conditions $B$, then we generate the clause $ch_C(M) \Leftarrow B$. If $\mathcal{M}_i$ waits for some message $M$ to be read before proceeding to so some action described by a clause $A \Leftarrow B$, then we generate $A \Leftarrow B, ch_C(M)$. (This will be described in more detail below.) In the pure eavesdropper model, this will be all.

In the *copycat model*, the intruder is also able to replay old messages, and to divert communication channels. We generate the additional clause

$$ch_C(x) \Leftarrow ch_{C'}(x) \tag{5.9}$$

for any reachable configuration $C$ with predecessor $C'$. This specifies that every message put on some channel remains on this channel, and can be read at any time in the future, and replayed as many times as we wish.

In the *Dolev-Yao model*, so named because it is closest to that of [DY83], instead of (5.9), we generate the clause

$$ch_C(x) \Leftarrow k_C(x) \quad \text{Intruder has complete control over channels} \tag{5.10}$$

In other words, anything read from any channel is directly forged by the intruder from past messages. As far as security is concerned, this is the model giving the most abilities to the intruder. Any attack in all previous models can be played in this model.

The final ingredient in our description of the protocol is the specification of the contents of the communication channels in the initial configuration $C_0$ : we assume that some predicate $ch_{C_0}$ has been defined by some two-way $\mathbb{ACU}$-tree automaton. We further restrict the pure eavesdropper model to be such that the channel is empty in configuration $C_0$ : just don't produce any clause defining $ch_{C_0}$.

Starting from $C_0$, execution may proceed by letting $\mathcal{M}_1$ send its upflow message to $\mathcal{M}_2$, letting the whole system progress to some new configuration $C_1$ (and $C_0$ is its predecessor) :

$$ch_{C_1}(e(N_1)) \qquad \text{Intruder gets } \mathcal{M}_1\text{'s message} \tag{5.11}$$

Note that $ch_{C_1}$ holds of exactly one message in the pure eavesdropper model. This will be an invariant of our description : for any reachable configuration $C$, there will be at most one ground term $M$ such that $ch_C(M)$ holds—the *contents* of the channel. In the copycat and Dolev-Yao models, $ch_C(M)$ may be true for several messages $M$, because of clauses (5.9), resp. (5.10).

Let us write what happens when the next action is $\mathcal{M}_2$ sending its own message to $\mathcal{M}_3$ :

$$ch_{C_2}(e(N_2); e(x); e(x + N_2)) \Leftarrow ch_{C_1}(e(x)) \tag{5.12}$$

Clause (5.12) means that $\mathcal{M}_2$ reads the message from $\mathcal{M}_1$ first ; this should be $e(N_1)$, but $\mathcal{M}_2$ can only check that it is $e(x)$ for some $x$ ; also, the only way it can get $e(x)$ is by querying the channel through $ch_{C_1}$. Then $\mathcal{M}_2$ should build $e(N_2); e(N_1); e(N_1+N_2)$. Since the variable $x$ should contain $N_1$, actual implementations build $e(N_2); e(x); e(x+N_2)$, and send it to the intruder in the new configuration $C_2$.

The downflow message from $\mathcal{M}_3$ gives rise to the clause :

$$ch_{C_3}(e(x + N_3); e(y + N_3)) \Leftarrow ch_{C_2}(e(x); e(y); e(z)) \tag{5.13}$$

Now the secrecy requirement on, say, $\mathcal{M}_1$'s view of the group key is that

$$\bot \Leftarrow ch_{C_3}(cons(e(x), z')) \wedge k_{C_3}(e(x + N_1)) \tag{5.14}$$

Indeed, $\mathcal{M}_1$'s view of the group key is $e(x + N_1)$, where the message broadcasted by $\mathcal{M}_3$ is $e(x); y$, i.e., where $ch_{C_3}(e(x); y)$ holds (reminder : if this message is not forged, then $x = N_2 + N_3$). Clause (5.14) states that this view $e(x + N_1)$ is not known to the intruder in configuration $C_3$ (and therefore neither in $C_1$ or $C_2$.)

There are many other possible interleavings, whose description we leave to the reader. As we have said, our purpose here is not to actually verify this protocol, but to illustrate the application of two-way $\mathbb{ACU}$-tree automata on a concrete example. All clauses but a few are automata clauses. E.g., clauses (5.3) (5.4) are pop clauses, clauses (5.5) and (5.6) are free push clauses. Clauses (5.7), (5.8), (5.9), and (5.10) are $\epsilon$-clauses. From the abbreviations discussed in Chapter 3, clauses (5.1), (5.2), (5.11), can be expanded into free push clauses, free pop clauses, epsilon clauses and zero clauses.

Clause (5.14) can be expanded to

$$R(z) \Leftarrow ch_{C_3}(cons(z, z')) \qquad R'(x) \Leftarrow R(e(x)) \qquad R''(x + N_1) \Leftarrow R'(x)$$
$$R'''(e(z)) \Leftarrow R''(z) \qquad \bot \Leftarrow R'''(z) \wedge k_{C_3}(z)$$

where $R$, $R'$, $R''$, $R'''$ are fresh and the last clause is a query clause which tests the intersection of languages defined by $R'''$, $ch_{C_1}, \ldots, ch_{C_3}$.

Clause (5.12) is more problematic : The clauses

$$valx(x) \Leftarrow ch_{C_1}(e(x)) \qquad ch_{C_2}(x_1; x_2; x_3) \Leftarrow P_1(x_1) \wedge P_2(x_2) \wedge P_3(x_3)$$
$$P_1(e(N_2)) \qquad P_2(e(x)) \Leftarrow valx(x) \qquad P_3(e(x + N_2)) \Leftarrow valx(x)$$

are equivalent to (5.12) in the Dolev-Yao model, in the copycat model (where sending lists of elements or sending each element separately has the same net effect) and in the pure eavesdropper model (where any $ch_C$ recognizes at most one value anyway).

By a similar argument, clause (5.13) can be expanded into the clauses

$$P_1(x) \Leftarrow ch_{C_2}(cons(x, y)) \qquad P_{23}(y) \Leftarrow ch_{C_2}(cons(x, y))$$
$$P_2(x) \Leftarrow P_{23}(cons(x, y)) \qquad ch_{C_3}(e(x + N_3); e(y + N_3)) \Leftarrow P_1(e(x)) \wedge P_2(e(y))$$

The resulting set of clauses defines a two-way $\mathbb{ACU}$-automaton (recall that two-way $\mathbb{ACU}$ automata extend one-way $\mathbb{ACU}$ automata by adding free push clauses) together with query clauses of the form $\perp \Leftarrow P_1(x) \wedge P_2(x)$. As remarked in Corollary 1 (which can be easily generalized to the equational case) the purpose of these clauses is to check whether some common term is accepted by both $P_1$ and $P_2$. We show in Chapter 10 that these two-way $\mathbb{ACU}$ automata can be converted to equivalent one-way automata. The one-way $\mathbb{ACU}$ automata are shown to be closed under intersection in Chapter 8. Also we have already seen in Chapter 3 that emptiness of one-way equational tree automata is decidable. Together these results mean that the above queries can be decided for our modeling of the IKA.1 protocol.

Recall our remark at the beginning of this section that two-wayness is indispensable as far as applications in cryptographic protocols are concerned. This can be seen from the clauses used in this modeling. For example, clauses (5.5) and (5.6) are push clauses. Clause (5.2) when expanded requires push clauses. The expansions described above of clauses (5.12), (5.13) and (5.14) also require push clauses.

The curious reader might want to know that the secrecy property fails, i.e., above set of clauses (i.e. together with the query clauses) is unsatisfiable. Alternatively the intersection of states $R'''$, $ch_{C_1}$, $\ldots$, $ch_{C_3}$ is not empty, i.e., there is an attack, in both the Dolev-Yao and the copycat models [MD02]. Note that IKA.1 was indeed designed so as to be resistant only to pure eavesdroppers.

While the above modeling is done using $\mathbb{ACU}$ automata, it is more realistic to also also include the fact that $+$ is a group law, requiring the use of two-way $\mathbb{ACUM}$-tree automata. For example the A-GDH.2-MA protocol in [PQ01], which is used to add a new member to a group, uses keys $K$ as well as their inverses $K^{-1}$. Here $K^{-1}$ is the multiplicative inverse of $K$, so that we have the property $\alpha^{yxx^{-1}} = \alpha^y$. In our modeling we can accommodate such messages by also having $-$ symbol in the signature, and representing message $K^{-1}$ by the term $-K$. We then need to work modulo the Abelian groups theory $\mathbb{ACUM}$ to model cancellation of keys in exponents.

# Deuxième partie

# Résultats d’indécidabilité (Undecidability Results)

# Chapitre 6

# Résultats d'indécidabilité (Undecidability Results)

Dans ce chapitre nous exhibons plusieurs problèmes indécidables de la théorie des automates équationnels. En particulier nous montrons que les automates alternants modulo $\mathbb{ACU}$, ainsi que les automates bidirectionnels généraux modulo $\mathbb{ACU}$ (ceux contenant des clauses push dites générales) ont un test du vide indécidable. C'est la raison pour laquelle nous avons introduit des versions restreintes des clauses push en chapitre 3 ; c'est sur ces dernières versions que nous obtiendrons des résultats de décidabilité au chapitre 10. Ces automates avec clauses push restreintes redeviennent indécidables si on leur ajoute des contraintes d'égalité entre frères, comme nous le verrons dans ce chapitre. Ces résultats d'indécidabilité sont vrais même pour d'autres théories comme $\mathbb{AC}$, $\mathbb{ACUD}$ et $\mathbb{ACUM}$. Pour quelques théories comme $\mathbb{ACUX}$, $\mathbb{ACUI}$ la question de la décidibalité est ouverte. Notons que ces résultats de décidabilité contrastent avec le cas non équationnel où l'alternance est essentiellement bénigne.

In this chapter we show several undecidable problems concerning equational tree automata. In particular we show that alternating $\mathbb{ACU}$ automata, as well as general two-way $\mathbb{ACU}$ automata (those with general push clauses) have an undecidable emptiness test. This is the reason we have introduced restricted versions of push clauses in Chapter 3 to obtain decidability results. These restricted versions of automata, called two-way automata, for which the decidability results are studied in Chapter 10, become undecidable if we add equality constraints between brothers, as we will see in this chapter. These undecidability results continue to hold for other equational theories like $\mathbb{AC}$, $\mathbb{ACUD}$ and $\mathbb{ACUM}$. For some theories like $\mathbb{ACUX}$, $\mathbb{ACUI}$, the decidability question is open. Note that these undecidability results are in contrast to the non-equational case where alternation is essentially harmless.

A two-counter machine [Min61] has a finite number of states, two registers $R_1$ and $R_2$, each of which can store a non-negative integer, and a finite control which makes the machine change state while incrementing or decrementing one of the registers or while checking whether one of the registers stores the value $0$.

Formally, a *two-counter machine* $M$ is a finite labeled transition system with an initial state $q_0$, a final (acceptance) state $q_f$, and transitions $q \xrightarrow{a} q'$ where $a$ may be Inc $R_i$, Dec $R_i$ or Zero $R_i$, $i \in \{1, 2\}$. A *configuration* of the machine $M$ is a triple $(q, m, n)$ where $q$ is a state, $m, n \in \mathbb{N}$ are the values of $R_1$ and $R_2$ respectively.

Inc $R_i$ increments $R_i$, Dec $R_i$ checks whether $R_i$ is $\geq 1$, and if so decrements $R_i$, and Zero $R_i$ checks whether $R_i = 0$. Accordingly, the allowed moves of $M$ corresponding to the various transitions are described by the relation $\vdash_M$ as follows :

$$
\begin{array}{lrl}
q \xrightarrow{a} q', a = \text{Inc } R_1 : & (q, m, n) & \vdash_M (q', m+1, n) \\
q \xrightarrow{a} q', a = \text{Inc } R_2 : & (q, m, n) & \vdash_M (q', m, n+1) \\
q \xrightarrow{a} q', a = \text{Dec } R_1 : & (q, m+1, n) & \vdash_M (q', m, n) \\
q \xrightarrow{a} q', a = \text{Dec } R_2 : & (q, m, n+1) & \vdash_M (q', m, n) \\
q \xrightarrow{a} q', a = \text{Zero } R_1 : & (q, 0, n) & \vdash_M (q', 0, n) \\
q \xrightarrow{a} q', a = \text{Zero } R_2 : & (q, n, 0) & \vdash_M (q', n, 0)
\end{array}
$$

We denote by $\vdash_M^*$ the reflexive transitive closure of $\vdash_M$.

The following theorem states that every recursively enumerable set of integers is accepted by some two-counter machine.

**Theorem 6 ([Min61])** *For any recursively enumerable set $E$ of integers there is a two-counter machine $M$ with initial state $q_0$ and final state $q_f$ such that $E = \{n \in \mathbb{N} \mid \exists m', n' \in \mathbb{N} \cdot (q_0, 0, n) \vdash_M^* (q_f, m', n')\}$.*

## 6.1 $\mathbb{ACU}$ Case

First we deal with the undecidable problems in the $\mathbb{ACU}$ case. We show that alternating general two-way $\mathbb{ACU}$ automata have undecidable emptiness-test even with a signature in which all free symbols are constants. We discuss various cases according to the kinds of clauses allowed.

### 6.1.1 With General $+$-Push Clauses

In case we allow general $+$-push clauses, we can produce undecidability using only two constants. Assume a signature $\Sigma = \{+, 0, a_1, a_2\}$ where $a_1$ and $a_2$ are constants. To simulate two-counter machines using $\mathbb{ACU}$ automata, we encode configurations $(q, m, n)$ of a two-counter machine by the atom $q(\overline{m, n})$ where $\overline{m, n} = ma_1 + na_2$ for $m, n \geq 0$. Observe that for every term $t$ we have some $m, n \geq 0$ such that $t =_{\mathbb{ACU}} \overline{m, n}$.

**Lemma 14** *Given any two-counter machine $M$ with initial state $q_0$ and final state $q_f$, we can compute an $\mathbb{ACU}$ automaton $\mathcal{A}$ containing at most epsilon clauses (3.3), base clauses (3.5), $+$-pop clauses (3.6), zero clauses (3.8) and general $+$-push clauses (3.12) such that $\mathcal{L}(\mathcal{A}/\mathbb{ACU}) = \mathbb{ACU}(\{\overline{m, n} \mid \exists m', n' \in \mathbb{N} \cdot (q_0, m, n) \vdash_M^* (q_f, m', n')\})$.*

**Proof:** We simulate two-counter machines with the direction of computation reversed.

Because of Observation 10 we can also use intersection clauses in our automaton $\mathcal{A}$, knowing that they can be eliminated to get a general two-way automaton. Also, as in the remarks at the end of Section 3.3.3, we can use clauses like $P(x + a) \Leftarrow P_1(x)$ and $P(x) \Leftarrow P_1(x + a)$ knowing that they can be translated using $+$-pop clauses and general $+$-push clauses.

The automaton $\mathcal{A}$ has a state $q$ for each state $q$ in $M$, as well as states $zero_1, zero_2$ and $state$. First we add the clauses

$$\begin{aligned} zero_1(0) & \\ zero_1(x + a_2) &\Leftarrow zero_1(x) \\ zero_2(0) & \\ zero_2(x + a_1) &\Leftarrow zero_2(x) \\ state(x + y) &\Leftarrow zero_1(x) \wedge zero_2(y) \end{aligned}$$

to $\mathcal{A}$. We intend $zero_1$ to accept all possible values of the registers $R_1$ and $R_2$ such that $R_1 = 0$ (and $R_2$ have any value in $\mathbb{N}$). Similarly for $zero_2$. We intend $state$ to accept all possible valid values of the registers $R_1$ and $R_2$. Since we don't add any other clause in $\mathcal{A}$ which contains any of these predicates in the head, we have :

$$\begin{aligned} \mathcal{L}_{zero_1}(\mathcal{A}/\mathbb{ACU}) &= \mathbb{ACU}(\{\overline{0, n} \mid n \in \mathbb{N}\}) \\ \mathcal{L}_{zero_2}(\mathcal{A}/\mathbb{ACU}) &= \mathbb{ACU}(\{\overline{m, 0} \mid m \in \mathbb{N}\}) \\ \mathcal{L}_{state}(\mathcal{A}/\mathbb{ACU}) &= \mathbb{ACU}(\{\overline{m, n} \mid m, n \in \mathbb{N}\}) \end{aligned}$$

Then we add the following clauses to $\mathcal{A}$ corresponding to the transitions of $M$ :

$$\begin{array}{lrl} q \xrightarrow{a} q',\ a = \text{Inc } R_i : & q(x) & \Leftarrow q'(x + a_i) \\ q \xrightarrow{a} q',\ a = \text{Dec } R_i : & q(x + a_i) & \Leftarrow q'(x) \\ q \xrightarrow{a} q',\ a = \text{Zero } R_i : & q(x) & \Leftarrow q'(x) \wedge zero_i(x) \end{array}$$

Finally we add the clause

$$q_f(x) \Leftarrow state(x)$$

corresponding to the acceptance condition of $M$.

We need the following two intermediate results to finish our proof :

**Claim 1** *If $(q, m, n) \vdash_M^* (q_f, m', n')$ then $q(\overline{m, n})$ is derivable in $\mathcal{A}/\mathbb{ACU}$.*

**Proof:** We do induction on the number of moves required by $M$ to go from a configuration $(q, m, n)$ to a configuration $(q_f, m', n')$. We have the following cases :

(i) The number of moves is zero, i.e. $(q', m, n) = (q_f, m', n')$. Since $state(\overline{m', n'})$ is derivable in $\mathcal{A}/\mathbb{ACU}$ hence $q_f(\overline{m', n'})$ is derivable in $\mathcal{A}/\mathbb{ACU}$ using the clause $q_f(x) \Leftarrow state(x)$.

(ii) $M$ has a transition $q \xrightarrow{a} q'$, $a =$ Inc $R_1$ and we have $(q, m, n) \vdash_M (q', m+1, n) \vdash_M^* (q_f, m', n')$. By induction hypothesis $q'(\overline{m+1, n})$ is derivable in $\mathcal{A}/\mathbb{ACU}$. Hence $q(\overline{m, n})$ is derivable in $\mathcal{A}/\mathbb{ACU}$ using the clause $q(x) \Leftarrow q'(x + a)$.

(iii) $M$ has a transition $q \xrightarrow{a} q'$, $a =$ Dec $R_1$ and we have $(q, m+1, n) \vdash_M (q', m, n) \vdash_M^* (q_f, m', n')$. By induction hypothesis $q'(\overline{m, n})$ is derivable in $\mathcal{A}/\mathbb{ACU}$. Hence $q(\overline{m+1, n})$ is derivable in $\mathcal{A}/\mathbb{ACU}$ using the clause $q(x + a_1) \Leftarrow q'(x)$.

(iv) $M$ has a transition $q \xrightarrow{a} q'$, $a =$ Zero $R_1$ and we have $(q, 0, n) \vdash_M (q', 0, n) \vdash_M^* (q', m', n')$. By induction hypothesis we have a derivation of $q'(\overline{0, n})$ in $\mathcal{A}/\mathbb{ACU}$. Also $zero_1(\overline{0, n})$ is derivable in $\mathcal{A}/\mathbb{ACU}$. Hence using the clause $q(x) \Leftarrow q'(x) \wedge zero_1(x)$, $q(\overline{0, n})$ is derivable in $\mathcal{A}/\mathbb{ACU}$.

(v) The three cases involving counter $R_2$ instead of $R_1$ are dealt with in a similar way as the corresponding cases (ii-iv).

$\square$

**Claim 2** *If $q$ is a state of $M$ and $q(\overline{m, n})$ is derivable in $\mathcal{A}/\mathbb{ACU}$ then $(q, m, n) \vdash_M^* (q_f, m', n')$ for some $m', n' \in \mathbb{N}$.*

**Proof:** We do induction on the structure of the derivation of $q(\overline{m, n})$ in $\mathcal{A}/\mathbb{ACU}$. From Observation 1 we can assume that the derivation uses only rules of the form $C/\mathbb{ACU}$ for clauses $C$, and the only atoms appearing as conclusions of subderivations are of the form $q'(\overline{m'', n''})$ (where $q'$ is not necessarily a state of $M$ but could also be one of the new states introduces in $\mathcal{A}$). This can be ensured by replacing atoms $q'(t)$ by atoms $\overline{m'', n''}$ such that $\overline{m'', n''} =_{\mathbb{ACU}} t$. The result is again a well formed derivation.

(i) We have the derivation

$$\frac{\begin{array}{c}\vdots\\ state(\overline{m, n})\end{array}}{q_f(\overline{m, n})}\ (q_f(x) \Leftarrow state(x)/\mathbb{ACU})$$

in $\mathcal{A}/\mathbb{ACU}$. Clearly we have $(q_f, m, n) \vdash_M^* (q_f, m, n)$.

(ii) We have the derivation

$$\frac{\begin{array}{c}\vdots\\ q'(\overline{m+1, n})\end{array}}{q(\overline{m, n})}\ (q(x) \Leftarrow q'(x + a_1)/\mathbb{ACU})$$

in $\mathcal{A}/\mathbb{ACU}$ and the transition $q \xrightarrow{a} q'$, $a =$ Inc $R_1$ is in $M$. By induction hypothesis $(q', m+1, n) \vdash_M^* (q_f, m', n')$ for some $m', n' \in \mathbb{N}$. Also using the above transition $(q, m, n) \vdash_M (q', m+1, n)$. Hence $(q, m, n) \vdash_M^* (q_f, m', n')$.

(iii) We have the derivation

$$\frac{\begin{array}{c}\vdots\\ q'(\overline{m, n})\end{array}}{q(\overline{m+1, n})}\ (q(x + a_1) \Leftarrow q'(x)/\mathbb{ACU})$$

in $\mathcal{A}/\mathbb{ACU}$ and the transition $q \xrightarrow{a} q'$, $a =$ Dec $R_1$ is in $M$. By induction hypothesis $(q', m, n) \vdash_M^* (q_f, m', n')$ for some $m', n' \in \mathbb{N}$. Also using the above transition $(q, m+1, n) \vdash_M (q', m, n)$. Hence $(q, m+1, n) \vdash_M^* (q_f, m', n')$.

(iv) We have the derivation

$$\frac{\begin{matrix}\vdots & & \vdots \\ q'(\overline{0,n}) & & zero_1(\overline{0,n})\end{matrix}}{q(\overline{0,n})}\ (q(x) \Leftarrow q'(x) \wedge zero_i(x)/\mathbb{ACU})$$

in $\mathcal{A}/\mathbb{ACU}$ and the transition $q \xrightarrow{a} q'$, $a =$ Zero $R_1$ is in $M$. By induction hypothesis $(q', 0, n) \vdash_M^* (q_f, m', n')$ for some $m', n' \in \mathbb{N}$. Also using the above transition $(q, 0, n) \vdash_M (q', 0, n)$. Hence $(q, 0, n) \vdash_M^* (q_f, m', n')$.

(v) The three cases involving register $R_2$ instead of $R_1$ are dealt with in a similar way as the corresponding cases (ii-iv).

□

We now return to the proof of Lemma 14. We name $q_0$ as the final state of $\mathcal{A}$. From Claims 1 and 2 we have $\mathcal{L}(\mathcal{A}/\mathbb{ACU}) = \mathbb{ACU}(\{\overline{m,n} \mid \exists m', n' \in \mathbb{N} \cdot (q_0, m, n) \vdash_M^* q_f(m', n')\})$. □

This allows us to prove our first undecidability result :

**Theorem 7** *Every recursively enumerable set $E$ of integers is effectively representable as the language $\mathcal{L}(\mathcal{A}/\mathbb{ACU})$ for some automaton $\mathcal{A}$ containing only epsilon clauses (3.3), base clauses (3.5), $+$-pop clauses (3.6), zero clauses (3.8) and general $+$-push clauses (3.12). In particular emptiness of $\mathbb{ACU}$-automata containing such clauses is undecidable.*

Note that the representation of $E$ as $\mathcal{L}(\mathcal{A}/\mathbb{ACU})$ has to be modulo some encoding, as $E$ contains integers, whereas $\mathcal{L}(\mathcal{A}/\mathbb{ACU})$ contains terms. The encoding is specified in the proof.

**Proof:** Let $E$ be a recursively enumerable set of integers. From Theorem 6 there is a two-counter machine $M$ with initial state $q_0$ and final state $q_f$ such that

$$E = \{n \in \mathbb{N} \mid \exists m', n' \in \mathbb{N} \cdot (q_0, 0, n) \vdash_M^* (q_f, m', n')\}.$$

By Lemma 14 we have an $\mathbb{ACU}$ automaton $\mathcal{A}$ containing at most epsilon clauses (3.3), base clauses (3.5), $+$-pop clauses (3.6) zero clauses (3.8) and general $+$-push clauses (3.12) such that

$$\mathcal{L}(\mathcal{A}/\mathbb{ACU}) = \mathbb{ACU}(\{\overline{m,n} \mid \exists m', n' \in \mathbb{N} \cdot (q_0, m, n) \vdash_M^* q_f(m', n')\}).$$

Let $q$ be the final state of $\mathcal{A}$. Define automaton $\mathcal{B}$ to contain all the states and clauses of $\mathcal{A}$. In addition $\mathcal{B}$ contains fresh states $q_1$ and $q_2$ as well as the clauses

$$\begin{array}{l} q_1(0) \\ q_1(x + a_2) \Leftarrow q_1\ (x) \\ q_2(x) \Leftarrow q(x) \wedge q_1(x) \end{array}$$

We name $q_2$ as the final state of $\mathcal{B}$. We have

$$\mathcal{L}_{q_1}(\mathcal{B}/\mathbb{ACU}) = \mathbb{ACU}(\{\overline{0,n} \mid n \in \mathbb{N}\})$$

Hence

$$\begin{aligned}
\mathcal{L}(\mathcal{B}/\mathbb{ACU}) &= \mathcal{L}_{q_2}(\mathcal{B}/\mathbb{ACU}) \\
&= \mathcal{L}_q(\mathcal{B}/\mathbb{ACU}) \cap \mathcal{L}_{q_1}(\mathcal{B}/\mathbb{ACU}) \\
&= \mathcal{L}_q(\mathcal{A}/\mathbb{ACU}) \cap \mathcal{L}_{q_1}(\mathcal{B}/\mathbb{ACU}) \\
&= \mathcal{L}(\mathcal{A}/\mathbb{ACU}) \cap \mathcal{L}_{q_1}(\mathcal{B}/\mathbb{ACU}) \\
&= \mathbb{ACU}(\{\overline{0,n} \mid \exists m', n' \in \mathbb{N} \cdot (q_0, 0, n) \vdash_M^* (q_f, m', n')\}) \\
&= \mathbb{ACU}(\{na_2 \mid n \in E\})
\end{aligned}$$

Hence $E$ is representable as $\mathcal{L}(\mathcal{B}/\mathbb{ACU})$ modulo the encoding specified by the above equality.

Also $E$ is empty iff $\mathcal{L}(\mathcal{B}/\mathbb{ACU})$ is empty. As emptiness of recursively enumerable sets is undecidable [HU79], emptiness of $\mathbb{ACU}$-automata containing clauses (3.3), (3.5), (3.6), (3.8) and (3.12) is undecidable. □

### 6.1.2 With Intersection Clauses

We saw that emptiness is undecidable for $\mathbb{ACU}$-automata when general $+$-push clauses are allowed. As the general $+$-push clauses are powerful enough to encode intersection clauses, intersection clauses were also used in the undecidability proof. In this section we show that we can show undecidability with only intersection clauses, and without using the general $+$-push clauses. To do this we require four constants compared to two constants used in the previous case. Hence we assume a signature $\Sigma = \{+, 0, a_1, b_1, a_2, b_2\}$. In the previous case, we used the push clauses to translate the increment transitions (which corresponded to subtracting a constant from the concerned term). However in the absence of push clauses this is not possible. To solve this problem we encode configurations $(q, m, n)$ of the counter machine by atoms $q((m+x)a_1+xb_1+(n+y)a_2+yb_2)$ where $x, y \in \mathbb{N}$. Then incrementing $m$ in the configuration corresponds to adding $a_1$ while decrementing $m$ corresponds to adding $b_1$ and checking that the resulting atom represents a valid configuration. Checking whether $m = 0$ corresponds to checking whether the number of $a_1$'s is equal to the number of $b_1$'s. All these operations can be done using base clauses, $+$-pop clauses and intersection clauses. Hence we have the following result : (In passing, this encoding is similar to [ISD+02].)

**Lemma 15** *Let $M$ be a two-counter automaton with initial state $q_0$ and final state $q_f$. Then we can compute an $\mathbb{ACU}$-automaton $\mathcal{A}'$ containing at most epsilon clauses (3.3), base clauses (3.5), $+$-pop clauses (3.6), zero clauses (3.8) and intersection clauses (3.9) such that*

1. *If $t \in \mathcal{L}(\mathcal{A}'/\mathbb{ACU})$ then $t =_{\mathbb{ACU}} (m+x)a_1 + xb_1 + (n+y)a_2 + yb_2$ for some $m, n, x, y \in \mathbb{N}$.*
2. *For all $m, n \in \mathbb{N}$, $(m+x)a_1 + xb_1 + (n+y)a_2 + yb_2 \in \mathcal{L}(\mathcal{A}'/\mathbb{ACU})$ for some $x, y \in \mathbb{N}$ iff for some $m', n' \in \mathbb{N}$, $(q_0, m, n) \vdash_M^* (q_f, m', n')$.*

**Proof:** Let $\mathcal{A}$ be the automaton as described in the proof of Lemma 14 (i.e. $\mathcal{A}$ not only satisfies the property stated by Lemma 14 but also has exactly the same clauses as defined in the proof of the Lemma.) We will define automaton $\mathcal{A}'$ to simulate $\mathcal{A}$. As in the proof of Lemma 14 we will freely use clauses of the form $P(x+t) \Leftarrow P_1(x)$ in $\mathcal{A}'$, where $t$ is some ground term. Such clauses can be eliminated easily using pop clauses.

Observe from the proof of Lemma 14 that $\mathcal{A}$ only contains clauses of the form

$$\begin{array}{rll}
q(0) & & \\
q(x) & \Leftarrow q_1(x) & \\
q(x) & \Leftarrow q_1(x) \wedge q_2(x) & \\
q(a_i) & & i \in \{1,2\} \\
q(x+a_i) & \Leftarrow q_1(x) & i \in \{1,2\} \\
q(x) & \Leftarrow q_1(x+a_i) & i \in \{1,2\}
\end{array}$$

for some states $q, q_1, q_2$ in $\mathcal{A}$. ($q, q_1, q_2$ are not necessarily states of $\mathcal{A}$, they could also be the auxiliary states introduced in Lemma 14.) Also note that the above format of clauses are exactly are exactly the ones which appeared in the proof of Lemma 14, without their expansion using pop and general push clauses of general two-way equational tree automata.

Let $\mathcal{A}'$ contain all states of $\mathcal{A}$, as well as fresh states $zero$, $pos_1$ and $pos_2$. We first add the clauses

$$\begin{array}{rl} zero(0) & \\ zero(x + a_1 + b_1) & \Leftarrow zero(x) \\ zero(x + a_2 + b_2) & \Leftarrow zero(x) \\ pos_1(x + a_1) & \Leftarrow zero(x) \\ pos_1(x + a_1) & \Leftarrow pos_1(x) \\ pos_1(x + a_2) & \Leftarrow pos_1(x) \\ pos_2(x + a_2) & \Leftarrow zero(x) \\ pos_2(x + a_1) & \Leftarrow pos_2(x) \\ pos_2(x + a_2) & \Leftarrow pos_2(x) \end{array}$$

to $\mathcal{A}'/\mathbb{ACU}$. Intuitively in state $zero$ both $R_1$ and $R_2$ are zero. In $pos_1$, $R_1 > 0$ and $R_2 \geq 0$. In state $pos_2$, $R_1 \geq 0$ and $R_2 > 0$. We are not going any other clause which has any of the predicates $zero, pos_1, pos_2$ in the head. Hence given a term $t = m_1a_1 + m_2b_1 + n_1a_2 + n_2b_2$,

$t \in \mathcal{L}_{zero}(\mathcal{A}'/\mathbb{ACU})$ iff $m_1 = m_2$ and $n_1 = n_2$
$t \in \mathcal{L}_{pos_1}(\mathcal{A}'/\mathbb{ACU})$ iff $m_1 > m_2$ and $n_1 \geq n_2$
$t \in \mathcal{L}_{pos_2}(\mathcal{A}'/\mathbb{ACU})$ iff $m_1 \geq m_2$ and $n_1 > n_2$

Then we add clauses to $\mathcal{A}'$ corresponding to clauses of $\mathcal{A}$ as follows :

| clauses in $\mathcal{A}$ | clauses in $\mathcal{A}'$ |
|---|---|
| $q(0)$ | $q(x) \Leftarrow zero(x)$ |
| $q(x) \Leftarrow q_1(x)$ | $q(x) \Leftarrow q_1(x)$ |
| $q(x) \Leftarrow q_1(x) \wedge q_2(x)$ | $q(x) \Leftarrow q_1(x) \wedge q_2(x)$ |
| $q(a_i)$ | $q(x + a_i) \Leftarrow zero(x)$ |
| $q(x + a_i) \Leftarrow q_1(x)$ | $q(x + a_i) \Leftarrow q_1(x)$ |
| $q(x) \Leftarrow q_1(x + a_i)$ | $q(x + b_i) \Leftarrow q_1(x) \wedge pos_i(x)$ |

The fact that $\mathcal{A}'$ simulates $\mathcal{A}$ is stated by Lemmas 16 and 17.

**Lemma 16** *For any state $q$ in $\mathcal{A}$ and any $m, n \in \mathbb{N}$, if $ma_1 + na_2 \in \mathcal{L}_q(\mathcal{A}/\mathbb{ACU})$ then $\exists N \in \mathbb{N}$, for all $x, y \geq N$, $(m + x)a_1 + xb_1 + (n + y)a_2 + yb_2 \in \mathcal{L}_q(\mathcal{A}'/\mathbb{ACU})$.*

In the statement of the above Lemma we consider all $x, y \geq N$ for some $N \in \mathbb{N}$ instead of considering all $x, y \in \mathbb{N}$. The reason for this becomes clear in the case (vi) of the proof below.

**Proof:** We do induction on the size of the derivation of $q(ma_1 + na_2)$ in $\mathcal{A}/\mathbb{ACU}$. We have the following cases :

(i) $q(0)$ is derived in $\mathcal{A}/\mathbb{ACU}$ using the clause $q(0)$. We know that for any $x, y \geq 0$, $xa_1 + xb_1 + ya_2 + yb_2 \in \mathcal{L}_{zero}(\mathcal{A}'/\mathbb{ACU})$. Hence using the clause $q(x) \Leftarrow zero(x)$, the atom $q(xa_1 + xb_1 + ya_2 + yb_2)$ is derivable in $\mathcal{A}'/\mathbb{ACU}$ for all $x, y \geq 0$.

(ii) $q(ma_1 + na_2)$ is derived in $\mathcal{A}/\mathbb{ACU}$ using $q(x) \Leftarrow q_1(x)$ as the last clause. So $q_1(ma_1 + na_2)$ is derivable in $\mathcal{A}/\mathbb{ACU}$. By induction hypothesis, $\exists N \in \mathbb{N}$, for any $x, y \geq N$, $q_1((m + x)a_1 + xb_1 + (n + y)a_2 + yb_2)$ is derivable in $\mathcal{A}'/\mathbb{ACU}$. Then we use the clause $q(x) \Leftarrow q_1(x)$ to get derivations of $q((m + x)a_1 + xb_1 + (n + y)a_2 + yb_2)$ in $\mathcal{A}'/\mathbb{ACU}$ for all $x, y \geq N$.

(iii) $q(ma_1+na_2)$ is derived in $\mathcal{A}/\mathbb{ACU}$ using $q(x) \Leftarrow q_1(x) \wedge q_2(x)$ as the last clause. So $q_1(ma_1+ na_2)$ and $q_2(ma_1 + na_2)$ are derivable in $\mathcal{A}/\mathbb{ACU}$. By induction hypothesis, $\exists N \in \mathbb{N}$, for any $x, y \geq N$, $q_1((m+x)a_1+xb_1+(n+y)a_2+yb_2)$ and $q_2((m+x)a_1+xb_1+(n+y)a_2+yb_2)$ are derivable in $\mathcal{A}'/\mathbb{ACU}$. Then we use the clause $q(x) \Leftarrow q_1(x) \wedge q_2(x)$ to get derivations of $q((m + x)a_1 + xb_1 + (n + y)a_2 + yb_2)$ in $\mathcal{A}'/\mathbb{ACU}$ for all $x, y \geq N$.

(iv) $q(a_1)$ is derived in $\mathcal{A}/\mathbb{ACU}$ using the clause $q(a_i)$. For any $x, y \in \mathbb{N}$, $zero(xa_1 + xb_1 + ya_2 + yb_2)$ is derivable in $\mathcal{A}'/\mathbb{ACU}$. Hence using the clause $q(x + a_1) \Leftarrow zero(x)$, $q((x + 1)a_1 + xb_1 + ya_2 + yb_2)$ is derivable in $\mathcal{A}'/\mathbb{ACU}$ for all $x, y \geq 0$.

(v) $q((m + 1)a_1 + na_2)$ is derived in $\mathcal{A}/\mathbb{ACU}$ using $q(x + a_1) \Leftarrow q_1(x)$ as the last clause. So $q_1(ma_1 + na_2)$ is derivable in $\mathcal{A}/\mathbb{ACU}$. By induction hypothesis, $\exists N \in \mathbb{N}$, for any $x, y \geq N$, $q_1((m + x)a_1 + xb_1 + (n + y)a_2 + yb_2)$ is derivable in $\mathcal{A}'/\mathbb{ACU}$. Then we use the clause $q(x+a_1) \Leftarrow q_1(x)$ to get derivations of $q((m+1+x)a_1+xb_1+(n+y)a_2+yb_2)$ in $\mathcal{A}'/\mathbb{ACU}$ for all $x, y \geq N$.

(vi) $q(ma_1 + na_2)$ is derived in $\mathcal{A}/\mathbb{ACU}$ using $q(x) \Leftarrow q_1(x + a_1)$ as the last clause. So $q_1((m + 1)a_1 + na_2)$ is derivable in $\mathcal{A}/\mathbb{ACU}$. By induction hypothesis, $\exists N \in \mathbb{N}$, for any $x, y \in \mathbb{N}$, $q_1((m + 1 + x)a_1 + xb_1 + (n + y)a_2 + yb_2)$ is derivable in $\mathcal{A}'/\mathbb{ACU}$. Then we use the clause $q(x + b_1) \Leftarrow q_1(x)$ to get derivations of $q((m + 1 + x)a_1 + (x + 1)b_1 + (n + y)a_2 + yb_2)$ in $\mathcal{A}'/\mathbb{ACU}$ for all $x, y \geq N$. It follows that $q((m + x)a_1 + xb_1 + (n + y)a_2 + yb_2)$ is derivable in $\mathcal{A}'/\mathbb{ACU}$ for all $x, y \geq N + 1$.

(vi) The three cases involving clauses $q(a_2)$, $q(x + a_2) \Leftarrow q_1(x)$ and $q(x) \Leftarrow q_1(x + a_2)$ are dealt with in a similar way as the corresponding cases (iv-vi).

□

**Lemma 17** *For any state $q$ in $\mathcal{A}$ and for any $m_1, m_2, n_1, n_2 \in \mathbb{N}$, if $m_1a_1 + m_2b_1 + n_1a_2 + n_2b_2 \in \mathcal{L}_q(\mathcal{A}'/\mathbb{ACU})$ then $m_1 \geq m_2$, $n_1 \geq n_2$ and $(m_1 - m_2)a_1 + (n_1 - n_2)a_2 \in \mathcal{L}_q(\mathcal{A}/\mathbb{ACU})$.*

**Proof:** We do induction on the size of the derivation of $q(m_1a_1 + m_2b_1 + n_1a_2 + n_2b_2)$ in $\mathcal{A}'/\mathbb{ACU}$. We have the following cases :

(i) $q(m_1a_1 + m_2b_1 + n_1a_2 + n_2b_2)$ is derived is $\mathcal{A}'/\mathbb{ACU}$ using $q(x) \Leftarrow zero(x)$ as the last clause. As $zero(m_1a_1 + m_2b_1 + n_1a_2 + n_2b_2)$ is derivable in $\mathcal{A}'/\mathbb{ACU}$, $m_1 = m_2$ and $n_1 = n_2$. Also $q(0)$ is derivable in $\mathcal{A}/\mathbb{ACU}$ using the clause $q(0)$.

(ii) $q(m_1a_1 + m_2b_1 + n_1a_2 + n_2b_2)$ is derived is $\mathcal{A}'/\mathbb{ACU}$ using $q(x) \Leftarrow q_1(x)$ as the last clause. So $q_1(m_1a_1 + m_2b_1 + n_1a_2 + n_2b_2)$ is derivable in $\mathcal{A}'/\mathbb{ACU}$. By induction hypothesis $m_1 \geq m_2$, $n_1 \geq n_2$ and $q_1((m_1 - m_2)a_1 + (n_1 - n_2)a_2)$ is derivable in $\mathcal{A}/\mathbb{ACU}$. Hence using the clause $q(x) \Leftarrow q_1(x)$, $q((m_1 - m_2)a_1 + (n_1 - n_2)a_2)$ is derivable is $\mathcal{A}/\mathbb{ACU}$.

(iii) $q(m_1a_1 + m_2b_1 + n_1a_2 + n_2b_2)$ is derived is $\mathcal{A}'/\mathbb{ACU}$ using $q(x) \Leftarrow q_1(x) \wedge q_2(x)$ as the last clause. So $q_1(m_1a_1 + m_2b_1 + n_1a_2 + n_2b_2)$ and $q_2(m_1a_1 + m_2b_1 + n_1a_2 + n_2b_2)$ are derivable in $\mathcal{A}'/\mathbb{ACU}$. By induction hypothesis $m_1 \geq m_2$, $n_1 \geq n_2$ and $q_1((m_1 - m_2)a_1 + (n_1 - n_2)a_2)$ and $q_2((m_1 - m_2)a_1 + (n_1 - n_2)a_2)$ are derivable in $\mathcal{A}/\mathbb{ACU}$. Hence using the clause $q(x) \Leftarrow q_1(x) \wedge q_2(x)$, $q((m_1 - m_2)a_1 + (n_1 - n_2)a_2)$ is derivable is $\mathcal{A}/\mathbb{ACU}$.

(iv) $q((m_1+1)a_1+m_2b_1+n_1a_2+n_2b_2)$ is derived is $\mathcal{A}'/\mathbb{ACU}$ using $q(x+a_1) \Leftarrow zero(x)$ as the last clause. Since $zero(m_1a_1+m_2b_1+n_1a_2+n_2b_2)$ is derivable in $\mathcal{A}'/\mathbb{ACU}$, $m_1=m_2$ and $n_1=n_2$. Also $q(a_i)$ is derivable in $\mathcal{A}/\mathbb{ACU}$ using the clause $q(a_i)$.

(v) $q((m_1+1)a_1+m_2b_1+n_1a_2+n_2b_2)$ is derived is $\mathcal{A}'/\mathbb{ACU}$ using $q(x+a_1) \Leftarrow q_1(x)$ as the last clause. So $q_1(m_1a_1+m_2b_1+n_1a_2+n_2b_2)$ is derivable in $\mathcal{A}'/\mathbb{ACU}$. By induction hypothesis $m_1 \geq m_2, n_1 \geq n_2$ and $q_1((m_1-m_2)a_1+(n_1-n_2)a_2)$ is derivable in $\mathcal{A}/\mathbb{ACU}$. Hence using the clause $q(x+a_1) \Leftarrow q_1(x)$, $q((m_1-m_2+1)a_1(n_1-n_2)a_2)$ is derivable is $\mathcal{A}/\mathbb{ACU}$.

(vi) $q(m_1a_1+(m_2+1)b_1+n_1a_2+n_2b_2)$ is derived is $\mathcal{A}'/\mathbb{ACU}$ using $q(x+b_1) \Leftarrow q_1(x) \wedge pos_1(x)$ as the last clause. So $q_1(m_1a_1+m_2b_1+n_1a_2+n_2b_2)$ is derivable in $\mathcal{A}'/\mathbb{ACU}$. By induction hypothesis $m_1 \geq m_2, n_1 \geq n_2$ and $q_1((m_1-m_2)a_1+(n_1-n_2)a_2)$ is derivable in $\mathcal{A}/\mathbb{ACU}$. Also $pos_1(m_1a_1+m_2b_1+n_1a_2+n_2b_2)$ is derivable in $\mathcal{A}'/\mathbb{ACU}$. Hence $m_1 > m_2$, i.e. $m_1-m_2 \geq 1$. Hence using the clause $q(x) \Leftarrow q_1(x+a_1)$, $q((m_1-m_2-1)a_1+(n_1-n_2)a_2)$ is derived is $\mathcal{A}/\mathbb{ACU}$.

(vii) The three cases involving clauses $q(x+a_2) \Leftarrow zero(x)$, $q(x+a_2) \Leftarrow q_1(x)$ and $q(x+b_2) \Leftarrow q_1(x) \wedge pos_2(x)$ are dealt with in a similar way as the corresponding cases (iv-vi).

□

Also recall the property of automaton $\mathcal{A}$ as stated in Lemma 14 :

$$\mathcal{L}(\mathcal{A}/\mathbb{ACU}) = \mathbb{ACU}(\{\overline{m,n} \mid \exists m', n' \in \mathbb{N} \cdot (q_0, m, n) \vdash_M^* (q_f, m', n')\}) \tag{6.1}$$

We name $q_0$ (which is the final state of $\mathcal{A}$) as the final state of $\mathcal{A}'$.

Now we prove the required two items in the statement of the Lemma 15 :

1. Let $t \in \mathcal{L}(\mathcal{A}'/\mathbb{ACU})$. By definition of our signature, $t =_{\mathbb{ACU}} m_1a_1+m_2b_1+n_1a_2+n_2b_2$ for some $m_1, m_2, n_1, n_2 \in \mathbb{N}$. From Lemma 17 it follows that $m_1 \geq m_2$ and $n_1 \geq n_2$.
2. To show the 'only if' part, assume $m, n, x, y \in \mathbb{N}$ such that $(m+x)a_1+xb_1+(n+y)a_2+yb_2 \in \mathcal{L}(\mathcal{A}'/\mathbb{ACU})$. Since $\mathcal{A}$ and $\mathcal{A}'$ have the same final state, from Lemma 17, $ma_1+na_2 \in \mathcal{L}(\mathcal{A}/\mathbb{ACU})$. From (6.1), $(q_0, m, n) \vdash_M^* (q_f, m', n')$ for some $m', n' \in \mathbb{N}$.
   To show the 'if' part assume $m, n, m', n' \in \mathbb{N}$ such that $(q_0, m, n) \vdash_M^* (q_f, m', n')$. From (6.1), $ma_1+na_2 \in \mathcal{L}(\mathcal{A}/\mathbb{ACU})$. Since $\mathcal{A}$ and $\mathcal{A}'$ have the same final state, from Lemma 16 it follows that for some $x, y \in \mathbb{N}$ $(m+x)a_1+xb_1+(n+y)a_2+yb_2 \in \mathcal{L}(\mathcal{A}'/\mathbb{ACU})$. □

Now we are ready to show the undecidability result for the alternation case :

**Theorem 8** *Every recursively enumerable set $E$ of integers is effectively representable as the language $\mathcal{L}(\mathcal{A}/\mathbb{ACU})$ for some automata $\mathcal{A}$ containing only epsilon clauses (3.3), base clauses (3.5), $+$-pop clauses (3.6), zero clauses (3.8) and intersection clauses (3.9). In particular emptiness of $\mathbb{ACU}$-automata containing such clauses is undecidable.*

As in the case of general $+$-push clauses, the actual representation is modulo some encoding made precise in the proof of the theorem.

**Proof:** Let $E$ be a recursively enumerable set of integers. From Theorem 6 there is a two-counter machine $M$ with initial state $q_0$ and final state $q_f$ such that

$$E = \{n \in \mathbb{N} \mid \exists m', n' \in \mathbb{N} \cdot (q_0, 0, n) \vdash_M^* (q_f, m', n')\}.$$

From Lemma 15 we can compute an $\mathbb{ACU}$-automaton $\mathcal{A}'$ containing at most clauses (3.5), (3.3), (3.6) and (3.9) such that

1. If $t \in \mathcal{L}(\mathcal{A}'/\mathbb{ACU})$ then $t =_{\mathbb{ACU}} (m+x)a_1 + xb_1 + (n+y)a_2 + yb_2$ for some $m, n, x, y \in \mathbb{N}$.
2. For all $m, n \in \mathbb{N}$, $(m+x)a_1 + xb_1 + (n+y)a_2 + yb_2 \in \mathcal{L}(\mathcal{A}'/\mathbb{ACU})$ for some $x, y \in \mathbb{N}$ iff for some $m', n' \in \mathbb{N}$, $(q_0, m, n) \vdash_M^* (q_f, m', n')$.

Let $q$ be the final state of $\mathcal{A}'$. Define automaton $\mathcal{B}'$ to consist of all the states and all the clauses of $\mathcal{A}'$. In addition $\mathcal{B}'$ contains fresh states $q_1$ and $q_2$ as well as the clauses

$$\begin{aligned} q_1(0) & \\ q_1(x + a_1 + b_1) &\Leftarrow q_1(x) \\ q_1(x + a_2 + b_2) &\Leftarrow q_1(x) \\ q_1(x + b_2) &\Leftarrow q_1(x) \\ q_2(x) &\Leftarrow q(x) \wedge q_1(x) \end{aligned}$$

We name $q_2$ as the final state of $\mathcal{B}'$. We have

$$\mathcal{L}_{q_1}(\mathcal{B}'/\mathbb{ACU}) = \mathbb{ACU}(\{xa_1 + xb_1 + (n+y)a_2 + yb_2 \mid x, n, y \in \mathbb{N}\})$$

Hence we have the following properties of $\mathcal{B}'$ :

1. If $t \in \mathcal{L}(\mathcal{B}'/\mathbb{ACU})$ then $t =_{\mathbb{ACU}} xa_1 + xb_1 + (n+y)a_2 + yb_2$ for some $n, x, y \in \mathbb{N}$.
2. For all $n \in \mathbb{N}$, $xa_1 + xb_1 + (n+y)a_2 + yb_2 \in \mathcal{L}(\mathcal{A}'/\mathbb{ACU})$ for some $x, y \in \mathbb{N}$ iff $n \in E$.

which state the precise sense in which $\mathcal{L}(\mathcal{B}'/\mathbb{ACU})$ represents $E$.

It follows that $E$ is empty iff $\mathcal{L}(\mathcal{B}'/\mathbb{ACU})$ is empty, implying the undecidability of $\mathbb{ACU}$-automata containing clauses (3.5), (3.3), (3.6) and (3.9). $\square$

### 6.1.3 Equality Constraints and Two-Way Automata

We have seen in Observation 9 that general two-way push clauses are more expressive than intersection clauses. We now show that two-way automata, which are restricted versions of general two-way tree automata, together with equality constraints in the automata [CDG+97], are able to express alternation.

Equality constraints are expressed in our automata using clauses of the form

$$P(f(x, y, x)) \Leftarrow P_1(x) \wedge P_2(y) \wedge P_3(x)$$

Note that the first and the third arguments of $f$ are the same variable. (In the above example $f$ is assumed to be ternary. In general $f$ can be of arbitrary arity, and any number of arguments of $f$ can be the same variable.) The above clause can be read as "if the same term $x$ is accepted at both $P_1$ and $P_3$, and term $y$ is accepted at $P_2$, then term $f(x, y, x)$ is accepted at $P$.

An intersection clause

$$P(x) \Leftarrow P_1(x) \wedge P_2(x)$$

can then be expressed as the pair of clauses

$$\begin{aligned} P(x) &\Leftarrow Q(f(x, y)) \\ Q(f(x, x)) &\Leftarrow P_1(x) \wedge P_2(x) \end{aligned}$$

where $f$ is free binary symbol and $Q$ is a fresh state. The first of these two clauses is a free push clause, which are allowed in two-way automata. The second clause is a clause with equality constraint as described above.

Hence we have the result

**Lemma 18** *Adding equality constraints to two-way $\mathbb{ACU}$ automata makes their emptiness undecidable.*

On the other hand, we will see in Chapter 10 that two-way automata have good decidability and closure properties for most of the theories we deal with.

For this reason, we disallow equality constraints in our automata in order to have decidability results. This contrasts with the multitree automata of [Lug03] which can be thought of as extensions of one-way $\mathbb{AC}$ automata, and which accommodate equality constraints without losing decidability (see also the discussion in Section 3.4.

## 6.2 $\mathbb{AC}$ Case

Both the undecidability results for the $\mathbb{ACU}$ case in the previous case can also be shown for the $\mathbb{AC}$ case. The same ideas are used as in the $\mathbb{ACU}$ case. The main difference is that now our signature does not contain a unit for the $+$ symbol. Hence atoms of the form $q(0)$ which represented configurations $(q, 0, 0)$ of two-counter machines in the $\mathbb{ACU}$ case are not allowed. To overcome this problem we now encode configurations $(q, m, n)$ by atoms $q((m+1)a_1 + (n+1)a_2)$ instead of $q(ma_1 + na_2)$ as in the $\mathbb{ACU}$ case. This removes the need of the 0 symbol. This however poses a new problem, that when a constant is removed from an atom using a general $+$-push clause, we can get atoms of the form $q(ma_1)$ or $q(na_2)$ which do not represent any valid configuration. This problem is easily solved by using intersection clauses to filter out the 'bad' terms and keep only the 'good' ones. Instead of repeating all the proofs which are essentially the same as for the corresponding $\mathbb{ACU}$ cases, we merely state the main undecidability results :

**Theorem 9** *Fix a signature $\Sigma = \{+, 0, a_1, a_2\}$ where $a_1$ and $a_2$ are constants. Then every recursively enumerable set $E$ of integers is effectively representable as the language $\mathcal{L}(\mathcal{A}/\mathbb{AC})$ for some automata $\mathcal{A}$ containing only epsilon clauses (3.3), base clauses (3.5), $+$-pop clauses (3.6) and general $+$-push clauses (3.12). In particular emptiness of $\mathbb{AC}$-automata containing such clauses is undecidable.*

**Theorem 10** *Fix a signature $\Sigma = \{+, 0, a_1, b_1, a_2, b_2\}$ where $a_1, b_1, a_2$ and $b_2$ are constants. Then every recursively enumerable set $E$ of integers is effectively representable as the language $\mathcal{L}(\mathcal{A}/\mathbb{AC})$ for some automata $\mathcal{A}$ containing only epsilon clauses (3.3), base clauses (3.5), $+$-pop clauses (3.6) and intersection clauses (3.9). In particular emptiness of $\mathbb{AC}$-automata containing such clauses is undecidable.*

## 6.3 $\mathbb{ACUD}$ and $\mathbb{ACUM}$ Cases

The undecidability results for the $\mathbb{ACU}$ and $\mathbb{AC}$ cases can also be adapted to the $\mathbb{ACUD}$ and $\mathbb{ACUM}$ cases. Instead of repeating the proofs which are similar to the previous ones, we merely give the main results together with the ideas.

Recall that $\mathbb{ACUD}$ is the theory of a distributive (but not cancellative) $-$ symbol. Its axioms are those of $\mathbb{ACU}$, together with the axioms $-(x + y) = -x + (-y)$, $-(-x) = x$ and $-0 = 0$.

In order to adapt the undecidability proofs to the $\mathbb{ACUD}$ case, first observe that if the $-$ symbol does not occur in an automaton then the language accepted by the automaton modulo $\mathbb{ACUD}$ is same as the language accepted by the automaton modulo $\mathbb{ACU}$. This statement however needs to be formulated more carefully, as we do now. (To be formal, this holds only if we consider only the 'normal' terms from which the $-$ symbol has been removed. For example, if some state $P$ accepts term $t$ when the automaton is considered modulo $\mathbb{ACU}$, then modulo $\mathbb{ACUD}$, the automaton also accepts 'unnecessary' terms like $-(-t), -(-(-(-t))), t+(-0)$ etc.) This allows us to conclude the following two results similar to those in the theories $\mathbb{AC}$ and $\mathbb{ACU}$.

**Theorem 11** *Fix a signature $\Sigma = \{+, 0, -, a_1, a_2\}$ where $a_1$ and $a_2$ are constants. Then every recursively enumerable set $E$ of integers is effectively representable as the language $\mathcal{L}(\mathcal{A}/\mathbb{ACUD})$ for some automata $\mathcal{A}$ containing only epsilon clauses (3.3), base clauses (3.5), $+$-pop clauses (3.6), minus clauses (3.7), zero clauses (3.8) and general $+$-push clauses (3.12). In particular emptiness of $\mathbb{ACUD}$-automata containing such clauses is undecidable.*

**Theorem 12** *Fix a signature $\Sigma = \{+, 0, -, a_1, b_1, a_2, b_2\}$ where $a_1, b_1, a_2$ and $b_2$ are constants. Then every recursively enumerable set $E$ of integers is effectively representable as the language $\mathcal{L}(\mathcal{A}/\mathbb{ACUD})$ for some automata $\mathcal{A}$ containing only epsilon clauses (3.3), base clauses (3.5), $+$-pop clauses (3.6), minus clauses (3.7), zero clauses (3.8) and intersection clauses (3.9). In particular emptiness of $\mathbb{ACUD}$-automata containing such clauses is undecidable.*

The minus clauses (3.7) in the above results are not necessary, but we have included them only because they are part of all classes of $\mathbb{ACUD}$ automata that we study.

Now we come to the theory $\mathbb{ACUM}$ which is the equational theory of Abelian groups and consists of the axioms of $\mathbb{ACU}$ together with the axiom $x+(-x)=0$. In this case we first observe that we can do away with the general $+$-push clauses (3.12). This is because the general $+$-push clauses used in the previous undecidability proofs are all of the special form $q(x) \Leftarrow q_1(x+a)$ for some states $q, q_1$ and some constant $a$. In the $\mathbb{ACUM}$ case, we can replace this clause by the clause $q(x-a) \Leftarrow q_1(x)$. Hence we can show undecidability using only intersection clauses but no general $+$-push clauses, and using only two constants. Secondly the translation requires some care compared to the previous theories, because in this case, our clauses allow the states to accept terms like $-a_1, 2a_1 - 5a_2, ...$ which were not present in the $\mathbb{AC}$ or $\mathbb{ACU}$ case, and which do not represent any valid register values of the two-counter machine. However it is easy to filter out these unnecessary terms using intersection clauses. This allows us to conclude :

**Theorem 13** *Fix a signature $\Sigma = \{+, 0, -, a_1, a_2\}$ where $a_1$ and $a_2$ are constants. Then every recursively enumerable set $E$ of integers is effectively representable as the language $\mathcal{L}(\mathcal{A}/\mathbb{ACUM})$ for some automata $\mathcal{A}$ containing only epsilon clauses (3.3), base (3.5), $+$-pop clauses (3.6), minus clauses (3.7), zero clauses (3.8) and intersection clauses (3.9). In particular emptiness of $\mathbb{ACUM}$-automata containing such clauses is undecidable.*

## 6.4 Conclusion

We have shown that the presence of alternation or of general push clauses makes the emptiness problem of equational tree automata undecidable. For this we do not even arbitrary free function symbols, and a very small number of constants, besides the equational symbols $+, -, 0$ suffice. These undecidability proofs are based on encodings of two-counter automata.

These undecidability results have been shown for the theories $\mathbb{AC}$, $\mathbb{ACU}$, $\mathbb{ACUD}$ and $\mathbb{ACUM}$. The encoding of two-counter automata is based on the fact that these theories allow us to do counting, by using terms of the form $ma_1 + na_2$ to represent register values $(m, n)$. On the other hand, these methods seem difficult to generalize to the theories $\mathbb{ACUX}$, $\mathbb{ACUX}_n$ and $\mathbb{ACUI}$. This is because the non-linear equations in the latter theories make it difficult to do counting. For example in the case of the theory $\mathbb{ACUX}$ we have $a =_{\mathbb{ACUX}} 3a =_{\mathbb{ACUX}} 5a \ldots$. Similarly we have $a =_{\mathbb{ACUI}} 2a =_{\mathbb{ACUI}} 3a \ldots$. In fact if all the free function symbols in the signature are constants, then the sets $T(\Sigma)/ =_{\mathbb{ACUX}}$ and $T(\Sigma)/ =_{\mathbb{ACUI}}$ are both finite. Hence the previous encodings of configurations of two-counter automata are clearly impossible. The decidability question of $\mathbb{ACUX}$, $\mathbb{ACUX}_n$ and $\mathbb{ACUI}$ automata in the presence of intersection clauses or general $+$-push clauses is currently open.

## Troisième partie

# Automates d’arbres équationnels unidirectionnels (One-Way Equational Tree Automata)

# Chapitre 7

# Les outils de base (The Basic Toolkit)

Dans ce chapitre nous développons les outils de base qui nous aideront à étudier les automates d'arbres équationnels dans le reste de cette thèse. D'abord nous montrons que la vacuité des automates d'arbres équationnels unidirectionnels est décidable pour toute théorie, même celles qui ne sont pas étudiées dans cette thèse. Deuxièmement nous étudions les propriétés des dérivations dans les automates d'arbres équationnels modulo $\mathbb{ACU}$ et quelques autres théories. Ces résultats nous donnent des outils pour manipuler des dérivations utilisant les parties équationnelles des automates. Troisièmement nous étudions les automates "constant-only" (à savoir les automates unidirectionnels dans lesquels chaque symbole libre de la signature est une constante) pour certaines théories et nous montrons qu'ils acceptent les ensembles semilinéaires. Les résultats de ce chapitre sont très importants en ce qu'ils seront utilisés dans tout le reste de cette thèse.

In this chapter we develop some basic tools which will help us in studying equational tree automata in the rest of this thesis. Firstly we show that emptiness of one-way equational tree automata is decidable for an arbitrary theory including those not studied in this thesis. Secondly we study some properties of derivations in equational tree automata in $\mathbb{ACU}$ and some other theories. These results give us some tools to manipulate derivations involving the equational parts of automata. Thirdly, we study constant-only automata (recall that these are one-way automata in which all free symbols in the signature are constants) for certain theories and prove that they accept semilinear sets. The results in this chapter are very crucial in that they will be used throughout the rest of this thesis.

## 7.1 Emptiness Test for One-Way Equational Tree Automata

First we deal with emptiness of one-way equational tree automata. The following result is an easy consequence of definition of equational tree automata :

**Lemma 19** *For any one-way automaton $\mathcal{A}$ and equational theory $\mathbb{E}$, $\mathcal{L}_P(\mathcal{A}/\mathbb{E}) = \mathbb{E}(\mathcal{L}_P(\mathcal{A}))$.*

**Proof:** First we show that $\mathbb{E}(\mathcal{L}_P(\mathcal{A})) \subseteq \mathcal{L}_P(\mathcal{A}/\mathbb{E})$. Let $t \in \mathbb{E}(\mathcal{L}_P(\mathcal{A}))$. Then there is some $s =_\mathbb{E} t$ such that $s \in \mathcal{L}_P(\mathcal{A})$. Then $s \in \mathcal{L}_P(\mathcal{A}/\mathbb{E})$. Then because of Rule ($\mathbb{E}$), $t \in \mathcal{L}_P(\mathcal{A}/\mathbb{E})$.

Next by induction on the size of the derivation $\pi$ of $P(t)$ in $\mathcal{A}/\mathbb{E}$, we show that if $P(t)$ is derivable in $\mathcal{A}/\mathbb{E}$ then there is some $s =_\mathbb{E} t$ such that $P(s)$ is derivable in $\mathcal{A}/\emptyset$.

**Case 1.** There is some $t' =_\mathbb{E} t$ such that $\pi$ is of the form

$$\cfrac{\cfrac{\pi'}{P(t')}}{P(t)}\,(\mathbb{E})$$

Then applying by induction hypothesis on $\pi'$ we have some $s =_\mathbb{E} t'$ such that $P(s)$ is derivable in $\mathcal{A}/\emptyset$. Then we also have $s =_\mathbb{E} t$.

**Case 2.** $t = f(t_1 + ... + t_n)$ and $\pi$ is of the form

$$\cfrac{\cfrac{\pi_1}{P_1(t_1)} \;\; ... \;\; \cfrac{\pi_n}{P_n(t_n)}}{P(f(t_1, ..., t_n))}\,(P(f(x_1, ..., x_n)) \Leftarrow P_1(x_1), ..., P_n(x_n))$$

For each $i$, by applying induction hypothesis on $\pi_i$ we get terms $s_i =_\mathbb{E} t_i$ and derivations $\pi'_i$ of $P_i(s_i)$ in $\mathcal{A}/\mathbb{E}$. Let $s = f(s_1, ..., s_n)$. Then $s =_\mathbb{E} t$ and $P(s)$ has the following derivation in $\mathcal{A}/\emptyset$ :

$$\cfrac{\cfrac{\pi'_1}{P_1(s_1)} \;\; ... \;\; \cfrac{\pi'_n}{P_n(s_n)}}{P(f(s_1, ..., s_n))}\,(P(f(x_1, ..., x_n)) \Leftarrow P_1(x_1), ..., P_n(x_n))$$

$\square$

As a consequence :

**Theorem 14** *For any equational theory $\mathbb{E}$, emptiness of one-way $\mathbb{E}$ tree automata is decidable.*

**Proof:** From Lemma 19, for any one-way automata $\mathcal{A}$ and predicate $P$, $\mathcal{L}_P(\mathcal{A}/\mathbb{E}) = \emptyset$ iff $\mathcal{L}_P(\mathcal{A}) = \emptyset$. Hence the emptiness of one-way $\mathbb{E}$ tree automata is equivalent to the emptiness of one-way (non-equational) tree automata, and the latter is known to be decidable. □

Observe that the arguments in the above two proofs are rather simple, although our experience shows that these results usually comes as a surprise to most people. We have in fact also shown that emptiness of one-way equational tree automata is decidable is polynomial time for any equational theory, since the emptiness of one-way non-equational tree automata is decidable in polynomial time, in fact in linear time, as remarked in [CDG$^+$97].

The above two results also hold for the "regular $\mathbb{E}$ tree automata" of [Ohs01] when $\mathbb{E}$ is linear. However as we have seen in Example 4, it does not hold in the case of non-linear theories in general. (Recall that the regular $\mathbb{E}$ tree automata of Ohsaki are the equivalent of our one-way $\mathbb{E}$ tree automata.)

Lemma 19 does not hold for alternating (or two-way) equational tree automata in general. We have the following alternating automaton as a counter-example.

**Example 5** *Consider the automaton* $\mathcal{A} = \{P_1(a), P_2(b), P_3(x+y) \Leftarrow P_1(x) \wedge P_2(y), P_4(x+y) \Leftarrow P_2(x) \wedge P_1(y), P(x) \Leftarrow P_3(x) \wedge P_4(x)\}$ *modulo commutativity (i.e. the theory* $\mathbb{C}$*). We see that* $\mathcal{L}_P(\mathcal{A}) = \emptyset$ *whereas* $\mathcal{L}_P(\mathcal{A}/\mathbb{C}) = \{a+b, b+a\}$*. In particular* $\mathcal{L}_P(\mathcal{A}/\mathbb{C}) \neq \mathbb{C}(\mathcal{L}_P(\mathcal{A}))$*.*

Theorem 14 also does not hold in general for alternating or general two-way automata. We have seen subclasses of automata with undecidable emptiness problem in Chapter 6.

We will sometimes need *extended epsilon clauses* :

$$P(x) \Leftarrow Q(x) \wedge P_1(x_1) \wedge ... \wedge P_n(x_n) \tag{7.1}$$

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 $P_1, ..., P_n$. These clauses don't increase the expressiveness of one-way equational tree automata :

**Lemma 20** *For any equational theory* $\mathbb{E}$ *and automaton* $\mathcal{A}$ *which contains clauses of one-way automaton, as well as extended epsilon clauses (7.1), we can compute a one-way automaton* $\mathcal{B}$ *such that for each* $P$, $\mathcal{L}_P(\mathcal{A}/\mathbb{E}) = \mathcal{L}_P(\mathcal{B}/\mathbb{E})$*.*

**Proof:** Let $\mathcal{A}_{ext}$ be the set of clauses of $\mathcal{A}$, which are of the form 7.1 in which $n \geq 1$. Then $\mathcal{A} \setminus \mathcal{A}_{ext}$ is a one-way automaton. We prove our result using induction on the number of clauses in $\mathcal{A}_{ext}$.

In case there is some clause $C = Q(x) \Leftarrow R(x) \wedge Q_1(x_1) \wedge ... \wedge Q_n(x_n)$ in $\mathcal{A}_{ext}$ such that for each $1 \leq i \leq n$, $\mathcal{L}_{Q_i}(\mathcal{A} \setminus \mathcal{A}_{ext}) \neq \emptyset$, then let $\mathcal{A}' = (\mathcal{A} \setminus \{C\}) \cup \{Q(x) \Leftarrow R(x)\}$. Since $Q(x) \Leftarrow R(x) \models Q(x) \Leftarrow R(x) \wedge Q_1(x_1) \wedge ... \wedge Q_n(x_n)$, we have for every predicate $P$, $\mathcal{L}_P(\mathcal{A}'/\mathbb{E}) \subseteq \mathcal{L}_P(\mathcal{A}/\mathbb{E})$. To show the converse, given any derivation $\pi$ in $\mathcal{A}'/\mathbb{E}$, we can assume by Observation 1 that the only rules used in the derivation are of the form $(C/\mathbb{E})$ for $C \in \mathcal{A}'$. Then we replace all subderivations of the form

$$\frac{\begin{array}{c}\vdots\\ R(s)\end{array}}{Q(t)}\,(Q(x) \Leftarrow R(x)/\mathbb{E})$$

(where clearly $s =_{\mathbb{E}} t$) by derivations of the form

$$\dfrac{\begin{matrix}\vdots & \vdots & & \vdots \\ R(s) & Q_1(t_1) & \dots & Q_n(t_n)\end{matrix}}{Q(t)}\ (Q(x) \Leftarrow R(x) \wedge Q_1(x_1) \wedge ... \wedge Q_n(x_n)/\mathbb{E})$$

where $t_1, \dots, t_n$ are some terms accepted at $\mathcal{A}$ (not $\mathcal{A}'$) modulo $\mathbb{E}$. Such terms exist because for each $1 \leq i \leq n$, $\mathcal{L}_{Q_i}(\mathcal{A} \setminus \mathcal{A}_{ext}) \neq \emptyset$. By this construction, we get a derivation in $\mathcal{A}/\mathbb{E}$ which has the same conclusion as $\pi$. Hence $\mathcal{L}_P(\mathcal{A}/\mathbb{E}) \subseteq \mathcal{L}_P(\mathcal{A}'/\mathbb{E})$ for every $P$. We conclude that $\mathcal{L}_P(\mathcal{A}'/\mathbb{E}) = \mathcal{L}_P(\mathcal{A}/\mathbb{E})$ for every $P$.

Also the corresponding set $\mathcal{A}'_{ext}$ has one clause less than $\mathcal{A}_{ext}$. Hence by induction hypothesis we get an automaton $\mathcal{B}$ such that for every $P$, $\mathcal{L}_P(\mathcal{B}/\mathbb{E}) = \mathcal{L}_P(\mathcal{A}'/\mathbb{E})$. Hence $\mathcal{B}$ is the required automaton.

On the other hand, suppose there is no clause $C$ in $\mathcal{A}_{ext}$ satisfying the above requirement. Then no clause from $\mathcal{A}_{ext}$ can be used in any derivation. Otherwise we would have some minimal derivation using this clause. This derivation would have the form

$$\dfrac{\begin{matrix}\vdots & \vdots & & \vdots \\ R(s) & Q_1(t_1) & \dots & Q_n(t_n)\end{matrix}}{Q(t)}\ (Q(x) \Leftarrow R(x) \wedge Q_1(x_1) \wedge ... \wedge Q_n(x_n)/\mathbb{E})$$

then we have $\mathcal{L}_{Q_i}(\mathcal{A} \setminus \mathcal{A}_{ext}) \neq \emptyset$ for each $1 \leq i \leq n$ which is a contradiction. Hence $\mathcal{A} \setminus \mathcal{A}_{ext}$ is the required automaton. $\square$

## 7.2 Functional Supports and Reuse of Derivations

Next we come to the next set of basic results of our toolkit. We show some results about how certain parts of derivations can be reused to get other derivations. These parts are the ones that involve the clauses from the equational part of the automata. We deal with the theories $\mathbb{ACU}$, $\mathbb{AC}$ and $\mathbb{ACUD}$.

### 7.2.1 $\mathbb{ACU}$ Derivations

First of all we deal with the $\mathbb{ACU}$ case :

**Lemma 21** *Let $S$ be a set of epsilon clauses (3.3), $+$-pop clauses (3.6) and zero clauses (3.8). Let $\pi$ be a derivation of an atom $P(t)$ of the form*

$$\pi = \dfrac{\dfrac{\pi_1}{P_1(t_1)} \quad \dots \quad \dfrac{\pi_n}{P_n(t_n)}}{P(t)}\ (S/\mathbb{ACU})$$

*Then we have :*

1. $t =_{\mathbb{ACU}} t_1 + \dots + t_n$
2. *If there are derivations $\pi'_1, \dots, \pi'_n$ of atoms $P_1(s_1), \dots, P_n(s_n)$ respectively then there is a derivation $\pi'$ of $P(s_1 + \dots + s_n)$ of the form*

$$\pi' = \dfrac{\dfrac{\pi'_1}{P_1(s_1)} \quad \dots \quad \dfrac{\pi'_n}{P_n(s_n)}}{P(s_1 + \dots + s_n)}\ (S/\mathbb{ACU})$$

**Proof:** We do induction on the size of $\pi$. We have the following cases :

(i) $n = 0$ and we have

$$\pi = \frac{}{P(0)}\,(P(0)/\mathbb{ACU})$$

where the clause $P(0) \in S$. The required results hold trivially.

(ii) $n = 1$ and

$$\pi = \pi_1$$

Clearly $t = t_1$ and $P = P_1$. This case is also trivial.

(iii)

$$\pi = \frac{\dfrac{\dfrac{\pi_1}{P_1(t_1)} \quad \ldots \quad \dfrac{\pi_n}{P_n(t_n)}}{P'(t')}\,(S/\mathbb{ACU})}{P(t)}\,(P(x) \Leftarrow P'(x)/\mathbb{ACU})$$

where the clause $P(x) \Leftarrow P'(x) \in S$. Clearly we have $t =_{\mathbb{ACU}} t'$. By induction hypothesis we have $t' =_{\mathbb{ACU}} t_1 + \ldots + t_n$ and hence $t' =_{\mathbb{ACU}} t_1 + \ldots + t_n$. If there are derivations $\pi'_1, \ldots, \pi'_n$ of atoms $P_1(s_1), \ldots, P_n(s_n)$ respectively then by induction hypothesis we have a derivation

$$\pi'' = \frac{\dfrac{\pi'_1}{P_1(s_1)} \quad \ldots \quad \dfrac{\pi'_n}{P_n(s_n)}}{P'(s_1 + \ldots + s_n)}\,(S/\mathbb{ACU})$$

The required derivation $\pi'$ of $P(s_1 + \ldots + s_n)$ is

$$\pi' = \frac{\dfrac{\pi''}{P'(s_1 + \ldots + s_n)}}{P(s_1 + \ldots + s_n)}\,(P(x) \Leftarrow P'(x)/\mathbb{ACU})$$

(iv)

$$\pi = \frac{\dfrac{\dfrac{\pi_1}{P_1(t_1)} \quad \ldots \quad \dfrac{\pi_n}{P_m(t_m)}}{P'_1(t'_1)}\,(S/\mathbb{ACU}) \qquad \dfrac{\dfrac{\pi_{m+1}}{P_{m+1}(t_{m+1})} \quad \ldots \quad \dfrac{\pi_n}{P_n(t_n)}}{P'_2(t'_2)}\,(S/\mathbb{ACU})}{P(t)}\,(P(x+y) \Leftarrow P'_1(x) \wedge P'_2(y)/\mathbb{ACU})$$

where the clause $P(x + y) \Leftarrow P'_1(x) \wedge P'_2(y) \in S$. Clearly we have $t =_{\mathbb{ACU}} t'_1 + t'_2$. By induction hypothesis we have $t'_1 =_{\mathbb{ACU}} t_1 + \ldots + t_m$ and $t'_2 =_{\mathbb{ACU}} t_{m+1} + \ldots + t_n$. Hence we have $t =_{\mathbb{ACU}} t'_1 + t'_2 =_{\mathbb{ACU}} t_1 + \ldots + t_n$. If there are derivations $\pi'_1, \ldots, \pi'_n$ of atoms $P_1(s_1), \ldots, P_n(s_n)$ respectively then by induction hypothesis we have derivations

$$\pi''_1 = \frac{\dfrac{\pi'_1}{P_1(s_1)} \quad \ldots \quad \dfrac{\pi'_m}{P_n(s_m)}}{P'_1(s_1 + \ldots + s_m)}\,(S/\mathbb{ACU})$$

and

$$\pi_2'' = \dfrac{\dfrac{\pi'_{m+1}}{P_{m+1}(s_{m+1})} \quad \dots \quad \dfrac{\pi'_n}{P_n(s_n)}}{P'(s_{m+1}+\dots+s_n)}\ (S/\mathbb{ACU})$$

The required derivation $\pi'$ of $P(s_1+\dots+s_n)$ is

$$\pi' = \dfrac{\dfrac{\pi_1''}{P_1'(s_1+\dots+s_m)} \quad \dfrac{\pi_2''}{P_2'(s_{m+1}+\dots+s_n)}}{P(s_1+\dots+s_n)}\ (P(x+y) \Leftarrow P_1'(x) \wedge P_2'(y)/\mathbb{ACU})$$

□

Note the form of the derivation

$$\pi = \dfrac{\dfrac{\pi_1}{P_1(t_1)} \quad \dots \quad \dfrac{\pi_n}{P_n(t_n)}}{P(t)}\ (S/\mathbb{ACU})$$

in the statement of Lemma 21. In this thesis we will frequently need to write the derivations in automata in this kind of format. First observe that any derivation $\pi$ of an atom $P(t)$ in a general two-way $\mathbb{ACU}$ automaton can trivially be written in the above format by letting $n = 1$ and $\pi_1 = \pi$. However if the automaton is a one-way $\mathbb{ACU}$ automaton then we can be more particular and restrict the $\pi_i$'s to use a free pop clause at the root. For this first we have the following observation :

**Observation 11** *Fix a signature $\Sigma$. Let $S_1$ and $S_2$ be two disjoint sets of clauses and $\mathbb{E}$ be an equational theory. Let $\pi$ be a derivation modulo $\mathbb{E}$ using the clauses of $S_1 \cup S_2$. (i.e. the rules used in $\pi$ are of the form $C/\mathbb{E}$ for $C \in S_1 \cup S_2$.) Then $\pi$ is of the form*

$$\pi = \dfrac{\dfrac{\begin{matrix}\vdots\\ P_1^1(t_1^1)\end{matrix} \ \dots \ \begin{matrix}\vdots\\ P_1^{k_1}(t_1^{k_1})\end{matrix}}{P_1(t_1)}\ (S_1/\mathbb{E}) \quad \dots \quad \dfrac{\begin{matrix}\vdots\\ P_n^1(t_n^1)\end{matrix} \ \dots \ \begin{matrix}\vdots\\ P_n^{k_n}(t_n^{k_n})\end{matrix}}{P_n(t_n)}\ (S_n/\mathbb{E})}{P(t)}\ (S_2/\mathbb{E})$$

**Proof:** The derivations leading to the conclusions $P_i(t_i)$'s are the maximal subderivations of $\pi$ which use at the root a rule of the form $C/\mathbb{E}$ for $C \in S_1$. (There may be no such subderivations, in which case we would have $n = 0$.) □

Intuitively the above observation allows us to label the clauses in an automaton using different colors, and then a derivation can be viewed as consisting of different layers, with each layer colored by a different color indicating the kind of clauses used in that layer. For example, if $\mathcal{A}$ is a one-way $\mathbb{ACU}$-automaton then we can assume that any derivation $\pi$ in the $\mathcal{A}/\mathbb{ACU}$ is of the form

$$\pi = \dfrac{\dfrac{\begin{matrix}\vdots\\ P_1^1(t_1^1)\end{matrix} \ \dots \ \begin{matrix}\vdots\\ P_1^{k_1}(t_1^{k_1})\end{matrix}}{P_1(f_1(t_1^1,\dots,t_1^{k_1}))}\ (\mathcal{A}_{free}/\mathbb{ACU}) \quad \dots \quad \dfrac{\begin{matrix}\vdots\\ P_n^1(t_n^1)\end{matrix} \ \dots \ \begin{matrix}\vdots\\ P_n^{k_n}(t_n^{k_n})\end{matrix}}{P_n(f_n(t_n^1,\dots,t_n^{k_n}))}\ (\mathcal{A}_{free}/\mathbb{ACU})}{P(t)}\ (\mathcal{A}_{eq}/\mathbb{ACU})$$

This result follows from Observation 11 by letting $S_1 = \mathcal{A}_{free}, S_2 = \mathcal{A}_{eq}$ and $\mathbb{E} = \mathbb{ACU}$. Note that from Lemma 21 $t =_{\mathbb{ACU}} f_1(t_1^1, \ldots, t_1^{k_1}) + \ldots + f_n(t_n^1, \ldots, t_n^{k_n})$. Also note that $\mathcal{A}_{free}$ is the set of free pop clauses in $\mathcal{A}$ since $\mathcal{A}$ is a one-way automaton. Hence in the above derivation, the subderivation leading to the conclusion $P_i(f_i(t_i^1, \ldots, t_i^{k_i}))$ uses a free pop clause at the root.

This discussion leads to the following definition regarding $\mathbb{ACU}$-derivations.

**Definition 4 ($\mathbb{ACU}$-Functional Support)** *Let $\pi$ be a derivation of an atom $P(t)$ in a one-way automaton $\mathcal{A}$ modulo $\mathbb{ACU}$. If*

$$\pi = \dfrac{\dfrac{\pi_1}{P_1(t_1)} \quad \ldots \quad \dfrac{\pi_n}{P_n(t_n)}}{P(t)}\ (\mathcal{A}_{eq}/\mathbb{ACU})$$

*and if each $\pi_i$ uses a free pop clause as the last clause, then we say that the (unordered) list of atoms $P_1(t_1), \ldots, P_n(t_n)$ is the* $\mathbb{ACU}$-functional support *of* $\pi$.

Note that the functional support of any derivation in a one-way automaton modulo $\mathbb{ACU}$ is uniquely defined (upto $\mathbb{ACU}$ equivalence on terms) because of the preceding discussion. Also it is clear that each $t_i$ is a functional term and from Lemma 21 we have that $t =_{\mathbb{ACU}} t_1 + \ldots + t_n$. We call this concept $\mathbb{ACU}$-functional support to emphasize that the theory under discussion is $\mathbb{ACU}$. Later on we will also need to define the notion support wrt other theories.

**Example 6** *Consider an $\mathbb{ACU}$ automaton $\mathcal{A}$ on signature $\{+, 0, a, f, g\}$, where $a$ is a constant and $f$ and $g$ are unary, having the following clauses :*

$$\begin{aligned}
C_1 &= P_1(0) \\
C_2 &= P_1(a) \\
C_3 &= P_1(f(x)) \Leftarrow P_1(x) \\
C_4 &= P_3(x) \Leftarrow P_1(x) \\
C_5 &= P_3(x+y) \Leftarrow P_3(x) \wedge P_1(y) \\
C_6 &= P_1(g(x)) \Leftarrow P_3(x)
\end{aligned}$$

*The following is a derivation in $\mathcal{A}/\mathbb{ACU}$ :*

$$\pi = \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P_1(a)}(C_2/\mathbb{ACU})}{P_3(a)}(C_4/\mathbb{ACU}) \quad \dfrac{\dfrac{}{P_1(a)}(C_2/\mathbb{ACU})}{P_1(f(a))}(C_3/\mathbb{ACU})}{P_3(a+f(a))}(C_5/\mathbb{ACU})}{P_1(g(a+f(a)))}(C_6/\mathbb{ACU})}{P_3(g(a+f(a)))}(C_4/\mathbb{ACU}) \quad \dfrac{}{P_1(0)}(C_1/\mathbb{ACU})}{P_3(g(a+f(a)))}(C_5/\mathbb{ACU}) \quad \dfrac{}{P_1(a)}(C_2/\mathbb{ACU})}{P_3(g(a+f(a))+a)}(C_5/\mathbb{ACU})$$

*This derivation can be denoted as*

$$\pi = \cfrac{\cfrac{\pi_1}{P_1(g(a+f(a)))} \quad \cfrac{}{P_1(a)}\,(C_2/\mathbb{ACU})}{P_3(g(a+f(a))+a)}\,(\mathcal{A}_{eq}/\mathbb{ACU})$$

*Observe that $\pi_1$ uses a free pop clause ($C_3$) as the last clause. Hence the functional support of $\pi$ is $P_1(g(a+f(a))), P_1(a)$.*

*On the other hand even though $\pi$ can also be written in the form*

$$\pi = \cfrac{\cfrac{\pi_2}{P_3(g(a+f(a)))} \quad \cfrac{}{P_1(a)}\,(C_2/\mathbb{ACU})}{P_3(g(a+f(a))+a)}\,(\mathcal{A}_{eq}/\mathbb{ACU})$$

*however $P_3(g(a+f(a))), P_1(a)$ is not the functional support of $\pi$. The derivation $\pi_2$ above does not use a free pop clauses as the last clause, rather it uses the epsilon clause $C_4$.*

### 7.2.2 $\mathbb{AC}$ Derivations

Observe that the presence of the unit symbol is not crucial for the discussion in the previous Section regarding $\mathbb{ACU}$ derivations. These results can easily be generalized for the $\mathbb{AC}$ case. We merely state the result without repeating the proof :

**Lemma 22** *Let $S$ be a set of epsilon clauses (3.3) and $+$-pop clauses (3.6). Let $\pi$ be a derivation of an atom $P(t)$ of the form*

$$\pi = \cfrac{\cfrac{\pi_1}{P_1(t_1)} \quad \ldots \quad \cfrac{\pi_n}{P_n(t_n)}}{P(t)}\,(S/\mathbb{AC})$$

*Then we have :*

1. *$t =_{\mathbb{AC}} t_1 + \ldots + t_n$*
2. *If there are derivations $\pi'_1, \ldots, \pi'_n$ of atoms $P_1(s_1), \ldots, P_n(s_n)$ respectively then there is a derivation $\pi'$ of $P(s_1 + \ldots + s_n)$ of the form*

$$\pi' = \cfrac{\cfrac{\pi'_1}{P_1(s_1)} \quad \ldots \quad \cfrac{\pi'_n}{P_n(s_n)}}{P(s_1+\ldots+s_n)}\,(S/\mathbb{AC})$$

Similarly the notion of functional support can be defined also for the $\mathbb{AC}$ case :

**Definition 5 ($\mathbb{AC}$-Functional Support)** *Let $\pi$ be a derivation of an atom $P(t)$ in a one-way automaton $\mathcal{A}$ modulo $\mathbb{AC}$. If*

$$\pi = \cfrac{\cfrac{\pi_1}{P_1(t_1)} \quad \ldots \quad \cfrac{\pi_n}{P_n(t_n)}}{P(t)}\,(\mathcal{A}_{eq}/\mathbb{AC})$$

*and if each $\pi_i$ uses a free pop clause as the last clause, then we say that the (unordered) list of atoms $P_1(t_1), \ldots, P_n(t_n)$ is the* $\mathbb{AC}$-functional support *of $\pi$.*

Again the functional support of any derivation in a one-way automaton modulo $\mathbb{AC}$ is uniquely defined (upto $\mathbb{AC}$ equivalence on terms) Also it is clear that each $t_i$ is a functional term and from Lemma 21 we have that $t =_{\mathbb{AC}} t_1 + \ldots + t_n$.

### 7.2.3 $\mathbb{ACUD}$ derivations

We now give the generalization of Lemma 21 to the $\mathbb{ACUD}$ case.

**Lemma 23** *Let $S$ be a set of epsilon clauses (3.3), $+$-pop clauses (3.6), zero clauses (3.8) and minus clauses (3.7). Let $\pi$ be a derivation of an atom $P(t)$ of the form*

$$\dfrac{\dfrac{\pi_1}{P_1(t_1)} \quad \dots \quad \dfrac{\pi_n}{P_n(t_n)}}{P(t)} \;(S/\mathbb{ACUD})$$

*Then we have :*

*(i)* $t =_{\mathbb{ACUD}} \pm_1 t_1 \pm_2 \dots \pm_n t_n$ *for some* $\pm_1, \dots, \pm_n \in \{+, -\}$. *($+t_1$ means $t_1$.)*

*(ii) If there are derivations $\pi'_1, ..., \pi'_n$ of atoms $P_1(s_1), ..., P_n(s_n)$ respectively, then there is a derivation $\pi'$ of $P(\pm_1 s_1 \pm_2 ... \pm_n s_n)$ (the $\pm_i$'s here are the same as in (i)) of the form*

$$\dfrac{\dfrac{\pi'_1}{P_1(s_1)} \quad \dots \quad \dfrac{\pi'_n}{P_n(s_n)}}{P(\pm_1 s_1 \pm_2 ... \pm_n s_n)} \;(S/\mathbb{ACUD})$$

**Proof:** The proof is similar to the proof of the corresponding result for the $\mathbb{ACU}$ case (Lemma 21). We do induction on the size of $\pi$. We have the following cases :

(i) $n = 0$ and we have

$$\pi = \dfrac{}{P(0)} \;(P(0)/\mathbb{ACUD})$$

where the clause $P(0) \in S$. The required results hold trivially.

(ii) $n = 1$ and

$$\pi = \pi_1$$

Clearly $t = t_1$ and $P = P_1$. This case is also trivial.

(iii)

$$\pi = \dfrac{\dfrac{\dfrac{\pi_1}{P_1(t_1)} \quad \dots \quad \dfrac{\pi_n}{P_n(t_n)}}{P'(t')} \;(S/\mathbb{ACUD})}{P(t)} \;(P(x) \Leftarrow P'(x)/\mathbb{ACUD})$$

where the clause $P(x) \Leftarrow P'(x) \in S$. Clearly we have $t =_{\mathbb{ACUD}} t'$. By induction hypothesis we have $t' =_{\mathbb{ACUD}} \pm_1 t_1 \pm_2 \dots \pm_n t_n$ and hence $t' =_{\mathbb{ACUD}} \pm_1 t_1 \pm_2 \dots \pm_n t_n$. If there are derivations $\pi'_1, \dots, \pi'_n$ of atoms $P_1(s_1), \dots, P_n(s_n)$ respectively then by induction hypothesis we have a derivation

$$\pi'' = \dfrac{\dfrac{\pi'_1}{P_1(s_1)} \quad \dots \quad \dfrac{\pi'_n}{P_n(s_n)}}{P'(\pm_1 s_1 \pm_2 \dots \pm_n s_n)} \;(S/\mathbb{ACUD})$$

The required derivation $\pi'$ of $P(\pm_1 s_1 \pm_2 \dots \pm_n s_n)$ is

$$\pi' = \cfrac{\cfrac{\pi''}{P'(\pm_1 s_1 \pm_2 \ldots \pm_n s_n)}}{P(\pm_1 s_1 \pm_2 \ldots \pm_n s_n)}\,(P(x) \Leftarrow P'(x)/\mathbb{ACUD})$$

(iv)

$$\pi = \cfrac{\cfrac{\cfrac{\pi_1}{P_1(t_1)} \quad \ldots \quad \cfrac{\pi_n}{P_n(t_n)}}{P'(t')}\,(S/\mathbb{ACUD})}{P(t)}\,(P(-x) \Leftarrow P'(x)/\mathbb{ACUD})$$

where the clause $P(-x) \Leftarrow P'(x) \in S$. Clearly we have $t =_{\mathbb{ACUD}} -t'$. By induction hypothesis we have $t' =_{\mathbb{ACUD}} \pm_1 t_1 \pm_2 \ldots \pm_n t_n$ and hence $t' =_{\mathbb{ACUD}} \mp_1 t_1 \mp_2 \ldots \mp_n t_n$ (where $\mp_i = \{+,-\} \setminus \{\pm_i\}$). If there are derivations $\pi'_1, \ldots, \pi'_n$ of atoms $P_1(s_1), \ldots, P_n(s_n)$ respectively then by induction hypothesis we have a derivation

$$\pi'' = \cfrac{\cfrac{\pi'_1}{P_1(s_1)} \quad \ldots \quad \cfrac{\pi'_n}{P_n(s_n)}}{P'(\pm_1 s_1 \pm_2 \ldots \pm_n s_n)}\,(S/\mathbb{ACUD})$$

The required derivation $\pi'$ of $P(\mp_1 s_1 \mp_2 \ldots \mp_n s_n)$ is

$$\pi' = \cfrac{\cfrac{\pi''}{P'(\pm_1 s_1 \pm_2 \ldots \pm_n s_n)}}{P(\mp_1 s_1 \mp_2 \ldots \mp_n s_n)}\,(P(-x) \Leftarrow P'(x)/\mathbb{ACUD})$$

(v)

$$\pi = \cfrac{\cfrac{\cfrac{\pi_1}{P_1(t_1)} \quad \ldots \quad \cfrac{\pi_n}{P_m(t_m)}}{P'_1(t'_1)}\,(S/\mathbb{ACUD}) \quad \cfrac{\cfrac{\pi_{m+1}}{P_{m+1}(t_{m+1})} \quad \ldots \quad \cfrac{\pi_n}{P_n(t_n)}}{P'_2(t'_2)}\,(S/\mathbb{ACUD})}{P(t)}\,(P(x+y) \Leftarrow P'_1(x) \wedge P'_2(y)/\mathbb{ACUD})$$

where the clause $P(x+y) \Leftarrow P'_1(x) \wedge P'_2(y) \in S$. Clearly we have $t =_{\mathbb{ACUD}} t'_1 + t'_2$. By induction hypothesis we have $t'_1 =_{\mathbb{ACUD}} \pm_1 t_1 \pm_2 \ldots \pm_m t_m$ and $t'_2 =_{\mathbb{ACUD}} \pm_{m+1} t_{m+1} \pm_{m+2} \ldots \pm_n t_n$. Hence we have $t =_{\mathbb{ACUD}} t'_1 + t'_2 =_{\mathbb{ACU}} \pm_1 t_1 \pm_2 \ldots \pm_n t_n$. If there are derivations $\pi'_1, \ldots, \pi'_n$ of atoms $P_1(s_1), \ldots, P_n(s_n)$ respectively then by induction hypothesis we have derivations

$$\pi''_1 = \cfrac{\cfrac{\pi'_1}{P_1(s_1)} \quad \ldots \quad \cfrac{\pi'_m}{P_n(s_m)}}{P'_1(\pm_1 s_1 \pm_2 \ldots \pm_m s_m)}\,(S/\mathbb{ACUD})$$

and

$$\pi''_2 = \cfrac{\cfrac{\pi'_{m+1}}{P_{m+1}(s_{m+1})} \quad \ldots \quad \cfrac{\pi'_n}{P_n(s_n)}}{P'(\pm_{m+1} s_{m+1} \pm_{m+2} \ldots \pm_n s_n)}\,(S/\mathbb{ACU})$$

The required derivation $\pi'$ of $P(\pm_1 s_1 \pm_2 \ldots \pm_n s_n)$ is

$$\pi' = \cfrac{\cfrac{\pi_1''}{P_1'(\pm_1 s_1 \pm_2 \ldots \pm_m s_m)} \quad \cfrac{\pi_2''}{P_2'(\pm_{m+1} s_{m+1} \pm_{m+2} \ldots \pm_n s_n)}}{P(\pm_1 s_1 \pm_2 \ldots \pm_n s_n)} \, (C/\mathbb{ACUD})$$

where $C = P(x+y) \Leftarrow P_1'(x) \wedge P_2'(y)$.

□

Next we are going to define the equivalent of $\mathbb{ACU}$-functional support for the theory $\mathbb{ACUD}$. Similar to the $\mathbb{ACU}$ theory, we first note that that if $\mathcal{A}$ is a one-way $\mathbb{ACUD}$-automaton then we can assume that any derivation $\pi$ in the $\mathcal{A}/\mathbb{ACUD}$ is of the form

$$\pi = \cfrac{\cfrac{\begin{matrix}\vdots & & \vdots \\ P_1^1(t_1^1) & \ldots & P_1^{k_1}(t_1^{k_1})\end{matrix}}{P_1(f_1(t_1^1,\ldots,t_1^{k_1}))} (\mathcal{A}_{free}/\mathbb{ACUD}) \quad \ldots \quad \cfrac{\begin{matrix}\vdots & & \vdots \\ P_n^1(t_n^1) & \ldots & P_n^{k_n}(t_n^{k_n})\end{matrix}}{P_n(f_n(t_n^1,\ldots,t_n^{k_n}))} (\mathcal{A}_{free}/\mathbb{ACUD})}{P(t)} \, (\mathcal{A}_{eq}/\mathbb{ACUD})$$

This result follows from Observation 11 by letting $S_1 = \mathcal{A}_{free}, S_2 = \mathcal{A}_{eq}$ and $\mathbb{E} = \mathbb{ACUD}$. Note that the set $\mathcal{A}_{eq}$ in this case also contains clauses of the form $P(-x) \Leftarrow P_1(x)$ which were not present in the $\mathbb{ACU}$ case. Also from Lemma 23 we have $t =_{\mathbb{ACUD}} \pm_1 f_1(t_1^1,\ldots,t_1^{k_1}) \pm_2 \ldots \pm_n f_n(t_n^1,\ldots,t_n^{k_n})$.

**Definition 6 ($\mathbb{ACUD}$-functional support)** *Let $\pi$ be a derivation of an atom $P(t)$ in a one-way automaton $\mathcal{A}$ modulo $\mathbb{ACUD}$. If*

$$\pi = \cfrac{\cfrac{\pi_1}{P_1(t_1)} \quad \ldots \quad \cfrac{\pi_n}{P_n(t_n)}}{P(t)} \, (\mathcal{A}_{eq}/\mathbb{ACUD})$$

*and if each $\pi_i$ uses a free pop clause as the last clause, then we say that the (unordered) list of atoms $P_1(t_1),\ldots,P_n(t_n)$ is the* $\mathbb{ACUD}$-functional support *of $\pi$.*

As in the $\mathbb{ACU}$ case, the $\mathbb{ACUD}$-functional support of any derivation in a one-way automaton modulo $\mathbb{ACUD}$ is uniquely defined (upto $\mathbb{ACUD}$ equivalence on terms). Also it is clear that each $t_i$ is a functional term and from Lemma 23 we have that $t =_{\mathbb{ACUD}} \pm_1 t_1 \pm_2 \ldots \pm_n t_n$.

## 7.3 Constant-Only Automata

Now we come to the third ingredient of our toolkit. The results in this Section are in fact the most important results in the thesis in the sense that all further results make abundant use of the results in this Section. We study the constant-only $\mathbb{ACU}$, $\mathbb{AC}$ and $\mathbb{ACUD}$ automata. In particular we show that the languages accepted by these automata are exactly semilinear sets of Presburger-definable sets, upto some encoding.

### 7.3.1 $\mathbb{ACU}$ Automata

In this section, we recall some important results on constant-only $\mathbb{ACU}$ automata which will be useful throughout this thesis. We fix a signature $\Sigma = \{+, 0, a_1, \ldots, a_p\}$ where $a_1, \ldots, a_p$ are mutually distinct constants. Every term $t \in T(\Sigma)$ can be written in the form $t =_{\mathbb{ACU}} \sum_{i=1}^{p} n_i a_i$ with $n_i \in \mathbb{N}$. This term can be identified with the $p$-tuple $(n_1, \ldots, n_p) \in \mathbb{N}^p$. Hence the $\mathbb{ACU}$-automata on $\Sigma$ can be viewed as acceptors of subsets of $\mathbb{N}^p$.

First of all we look at some other acceptors of such sets.

**Definition 7 (Semilinear)** *A* linear set *is a set of the form* $\{\nu + n_1\nu_1 + \ldots + n_k\nu_k \mid n_1, \ldots, n_k \in \mathbb{N}\}$ *for some* $\nu, \nu_1, \ldots, \nu_k \in \mathbb{N}^p$. *A* semi-linear set *is a finite union of linear sets.*

**Definition 8 (Presburger Arithmetic)** *By* Presburger arithmetic *we mean the first order logic in which the function symbols are* $0, 1, +$ *and the predicate symbols are* $<, =$. *The domain of interpretation is fixed to be* $\mathbb{N}$ *and the interpretations of the symbols* $0, 1, +, <$ *and* $=$ *are also fixed to be the usual functions and predicates on* $\mathbb{N}$.

Thus to define the semantics of any formula in Presburger arithmetic, we only need to give an assignment which maps variables to natural numbers. Hence a Presburger formula can be considered as defining a set of tuples of natural numbers. If $\phi$ is a Presburger formula and $x_1, \ldots, x_n$ are distinct variables such that all the free variables of $\phi$ are in the set $\{x_1, \ldots, x_n\}$, then the set defined by $\phi$, written as $[\![\phi]\!]$ is the set of all tuples $\nu \in \mathbb{N}^n$ such that the assignment which maps every $x_i$ to $\nu(i)$, makes $\phi$ true. Then we have the following result

**Lemma 24 ([GS66])** *The semi-linear sets are exactly the sets definable in Presburger arithmetic.*

In particular the semilinear sets are closed under union, intersection, complementation and projection, since these operations correspond to the logical operation $\vee, \wedge, \neg$ and existential quantification $\exists$ respectively.

Now we look at the relationship between semilinear sets and constant-only $\mathbb{ACU}$ automata. For this we first need a translation of our automata to context free grammars, similar to the one defined for constant-only $\mathbb{A}$ tree automata in Chapter 4. However now our signature also contains the symbol $0$ which was not present in the discussion on $\mathbb{A}$ tree automata. Hence we need to extend the $str$ function defined earlier. Observe that for any term $t \in T(\Sigma)$, we have $t =_{\mathbb{AU}} a_{i_1} + \ldots + a_{i_n}$ for some $1 \leq i_1, \ldots, i_n \leq p, n \geq 0$. As in the $\mathbb{A}$ case, this representation is unique. The difference from the $\mathbb{A}$ case is that we allow $n$ to be $0$, to be able to represent the term $0$. Let $T$ be the set of constants of $\Sigma$. The function $str : T(\Sigma) \to T^*$ is defined as $str(t) = a_{i_1} \ldots a_{i_n}$ where $t =_{\mathbb{AU}} a_{i_1} + \ldots + a_{i_n}$. In particular we have $str(0) = \epsilon$. The range of $str$ is now the whole of $T^*$, including the empty string. Hence we have a one-one correspondence between $\mathbb{AU}$-closed tree languages built on $\Sigma$, and the subsets of $T^*$, with the correspondence being defined by $str$.

Now we define a translation from constant-only $\mathbb{AU}$ tree automata to context free grammars. This translation is defined by the function $gram$ which is an extension of the $gram$ function defined in Chapter 4 for the $\mathbb{A}$ case. Now we need to take care of the new kinds of clauses $P(0)$ which were not present in the $\mathbb{A}$ case. Given a constant only automaton $\mathcal{A}$ on signature $\Sigma$, we define a context-free grammar $gram(\mathcal{A})$ as follows. The non-terminals are the predicates of $\mathcal{A}$. $T$ is the set of terminals. The start symbol $S$ is the final state of $\mathcal{A}$. We have production rules in $gram(\mathcal{A})$ corresponding to clauses of $\mathcal{A}$ as indicated in the table below.

| | |
|---|---|
| $P(x+y) \Leftarrow P_1(x) \wedge P_2(y)$ | $P \rightarrow P_1P_2$ |
| $P(a)$ | $P \rightarrow a$ |
| $P(x) \Leftarrow P_1(x)$ | $P \rightarrow P_1$ |
| $P(0)$ | $P \rightarrow \epsilon$ |

Similarly given a context free grammar $G = (V, T, P, S)$ such that $P$ contains only production rules of the form $P \rightarrow P_1P_2$, $P \rightarrow a$, $P \rightarrow P_1$ and $P \rightarrow \epsilon$, we have a constant-only automaton $gram^{-1}(G)$ (which is uniquely defined upto renaming of variables in clauses) on signature $T \cup \{+, 0\}$, such that its set of predicates is $V$ and the final state is $S$.

Then we have the following result :

**Lemma 25** *$P(t)$ is derivable in $\mathcal{A}/\mathbb{AU}$ iff $P \Longrightarrow^*_{gram(\mathcal{A})} str(t)$. Hence $str(\mathcal{L}(\mathcal{A}/\mathbb{AU})) = L(gram(\mathcal{A}))$.*

**Lemma 26** *Constant only $\mathbb{AU}$ tree automata on signature $\Sigma$ accept exactly the languages $\mathcal{L}$ such that $str(\mathcal{L})$ is a context-free language on symbols from $\Sigma \setminus \{+, 0\}$.*

**Proof:** From Lemma 13, every context free language not containing $\epsilon$ is generated by a context free grammar with rules of the form $A \rightarrow BC$ and $A \rightarrow a$. Hence a context free language $L$ containing $\epsilon$ is generated by the context free grammar obtained by adding the rule $S \rightarrow \epsilon$ to the context free grammar which generates $L \setminus \{\epsilon\}$. Here $S$ is the start symbol. (Recall that $L \setminus \{\epsilon\}$ is context free since $L$ is context free.)

Hence every context free language $L$ is generated by a grammar $G$ containing only production rules of the form $A \rightarrow BC$, $A \rightarrow a$ and $A \rightarrow \epsilon$. From Lemma 25 we have $L = str(\mathcal{L}(gram^{-1}(G)/\mathbb{AU}))$.

Conversely, if $\mathcal{A}$ is a constant-only $\mathbb{AU}$ automata, then $gram(\mathcal{A})$ is a context free grammar. Hence $L(gram(\mathcal{A}))$ is a context free language. Also from Lemma 25 we have $str(\mathcal{L}(\mathcal{A}/\mathbb{AU})) = L(gram(\mathcal{A}))$. $\Box$

This shows the equivalence between constant-only $\mathbb{AU}$ automata and context free languages. By adding the axiom $\mathbb{C}$ to the theory $\mathbb{AU}$ we get the theory $\mathbb{ACU}$. We now show that constant-only $\mathbb{ACU}$ automata are equivalent to semilinear sets. For this we use the connection between context free languages and semilinear sets given by Parikh's Theorem. First we define the *commutative image* $comm(x)$ of a string $x \in (\Sigma \setminus \{+, 0\})^*$ to be the tuple $\nu \in \mathbb{N}^p$ such that for $1 \leq i \leq p$, $\nu(i)$ is the number of times the symbol $a_i$ occurs in the string $x$. Given a language $L \subseteq (\Sigma \setminus \{+, 0\})^*$, the set $comm(L) \subseteq \mathbb{N}^p$ is defined as usual, and is called the commutative image of $L$. Parikh's Theorem states that

**Theorem 15 (Parikh [Par66])** *Semilinear sets are exactly the commutative images of context free languages.*

Let us also emphasize that we have effective procedures for converting context free languages to semilinear sets and back, which is necessary for decidability proofs in the thesis.

If $t \in T(\Sigma)$ and $t =_{\mathbb{ACU}} \sum_{i=1}^{p} n_i a_i$ then we have $comm(str(t)) = (n_1, \ldots, n_p)$. Hence we have a one-one correspondence between the terms on $\Sigma$ modulo $\mathbb{ACU}$, and the tuples in $\mathbb{N}^p$. This also gives a one-one correspondence between $\mathbb{ACU}$-closed languages on signature $\Sigma$ and the subsets of $\mathbb{N}^p$. From Lemmas 26 and Theorem 15 and modulo this correspondence of languages, we have :

**Theorem 16** *Constant-only $\mathbb{ACU}$ automata accept exactly semilinear sets.*

$\mathbb{AC}$ **Automata**

While the results in this section have till this point been restricted to the theory $\mathbb{ACU}$, we can also prove analogous results for the $\mathbb{AC}$ theory. Consider constant only $\mathbb{AC}$ automata on a signature $\Sigma = \{+, a_1, \dots, a_p\}$. Since the $\mathbb{U}$ equation is absent from the theory, we use the $str$ function defined in Chapter 4 for terms modulo $\mathbb{A}$. Again, if $t =_{\mathbb{AC}} \sum_{i=1}^{p} n_i a_i$ then $comm(str(t)) = (n_1, \dots, n_p)$. Clearly $comm(str(t)) \neq (0, , \dots, 0)$. Hence in the $\mathbb{AC}$ case, we have one-one correspondence between $\mathbb{AC}$ closed languages and subsets of $\mathbb{N}^p$ which don't contain the zero tuple. Also recall from Chapter 4 that constant-only $\mathbb{A}$ automata accept exactly the context free languages which don't contain $\epsilon$. From Theorem 15, we deduce that the commutative images of such context free languages are exactly the semilinear sets which don't contain the zero tuple, equivalently they are exactly of the form $S \setminus \{(0, \dots, 0)\}$ where $L$ is a semilinear set. We conclude that

**Theorem 17** *Constant-only $\mathbb{AC}$ automata accept exactly the sets of the from $S \setminus \{(0, \dots, 0)\}$ where $S$ is a semilinear set.*

In particular this class of languages are closed under intersection and complementation.

### 7.3.2 $\mathbb{ACUD}$ Automata

Let $\Sigma$ be a signature containing only constants, together with the symbols $+, 0$ and $-$. Let $\mathcal{A}$ be a constant-only $\mathbb{ACUD}$ automaton on signature $\Sigma$ with predicates in $\mathbb{P}$. Except for clauses (3.7) which introduce '$-$' symbols, $\mathcal{A}$ would have been just an $\mathbb{ACU}$ automaton. In the $\mathbb{ACUD}$ case, the languages are in fact very similar to semilinear sets. We now make this more precise. Recall that $\Sigma_f = \Sigma - \{+, 0, -\}$ is the set of constants in $\Sigma$. Let $\overline{\Sigma}_f = \{\overline{a} \mid a \in \Sigma_f\}$ be a set of fresh constants. Terms built from $\Sigma_f \cup \{+, -, 0\}$ modulo $\mathbb{ACUD}$ are of the form $a_1 + \dots + a_m - b_1 - \dots - b_n$ $(m, n \geq 0)$ while those built from $\Sigma_f \cup \overline{\Sigma}_f \cup \{+, 0\}$ modulo $\mathbb{ACU}$ are of the form $a_1 + \dots + a_m + \overline{b_1} + \dots + \overline{b_n}$ $(m, n \geq 0)$. Hence there is a natural one-one correspondence between terms built on $\Sigma_f \cup \{+, -, 0\}$ modulo $\mathbb{ACUD}$ and terms built on $\Sigma \cup \overline{\Sigma}_f \cup \{+, 0\}$ modulo $\mathbb{ACU}$. As a result there is a natural one-one correspondence between $\mathbb{ACUD}$-closed languages on signature $\Sigma_f \cup \{+, -, 0\}$ and $\mathbb{ACU}$-closed languages on signature $\Sigma \cup \overline{\Sigma}_f \cup \{+, 0\}$. We now show that $\mathcal{A}$ is equivalent to a constant-only $\mathbb{ACU}$ automaton modulo this correspondence of languages.

We introduce new predicate symbols $\overline{P}$ for every $P \in \mathbb{P}$. Define automaton $\mathcal{B}$ to consist of clauses corresponding to clauses of $\mathcal{A}$ as defined in the following table :

| clause of $\mathcal{A}$ | clauses added to $\mathcal{B}$ |
|---|---|
| $P(x) \Leftarrow P_1(x)$ | $P(x) \Leftarrow P_1(x)$<br>$\overline{P}(x) \Leftarrow \overline{P_1}(x)$ |
| $P(x+y) \Leftarrow P_1(x) \wedge P_2(y)$ | $P(x+y) \Leftarrow P_1(x) \wedge P_2(y)$<br>$\overline{P}(x+y) \Leftarrow \overline{P_1}(x) \wedge \overline{P_2}(y)$ |
| $P(0)$ | $P(0)$<br>$\overline{P}(0)$ |
| $P(a)$ | $P(a)$<br>$\overline{P}(\overline{a})$ |
| $P(-x) \Leftarrow P_1(x)$ | $P(x) \Leftarrow \overline{P_1}(x)$<br>$\overline{P}(x) \Leftarrow P_1(x)$ |

Note that the $-$ symbol does not occur in $\mathcal{B}$. It is built on the signature $\Sigma_f \cup \overline{\Sigma}_f \cup \{+, 0\}$, and the set of predicates $\{P, \overline{P} \mid P \in \mathbb{P}\}$. The correspondence between $\mathcal{A}$ and $\mathcal{B}$ is stated by the following two lemmas, both of which are proved by easy inductions on derivations.

**Lemma 27** *If $P(a_1+...+a_m-b_1...-b_n)$ is derivable in $\mathcal{A}/\mathbb{ACUD}$ 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 $\mathcal{B}/\mathbb{ACU}$.*

**Proof:** The derivation $\pi$ of $P(a_1 + ... + a_m - b_1... - b_n)$ can be assumed to use only rules of the form $C/\mathbb{ACUD}$ where the clause $C \in \mathcal{A}$. We do induction on the size of the derivation $\pi$. We have the following cases :

(i)

$$\pi = \frac{}{P(0)}\,(P(0)/\mathbb{ACUD})$$

where the clause $P(0) \in \mathcal{A}$. Then the clauses $P(0), \overline{P}(0) \in \mathcal{B}$. Hence we have the derivations

$$\frac{}{P(0)}\,(P(0)/\mathbb{ACU})$$

and

$$\frac{}{\overline{P}(0)}\,(\overline{P}(0)/\mathbb{ACUD})$$

in $\mathcal{B}/\mathbb{ACU}$.

(ii)

$$\pi = \frac{}{P(a)}\,(P(a)/\mathbb{ACUD})$$

where the clause $P(a) \in \mathcal{A}$. Then the clauses $P(a), \overline{P}(a) \in \mathcal{B}$. Hence we have the derivations

$$\frac{}{P(a)}\,(P(a)/\mathbb{ACU})$$

and

$$\frac{}{\overline{P}(\overline{a})}\,(\overline{P}(\overline{a})/\mathbb{ACUD})$$

in $\mathcal{B}/\mathbb{ACU}$.

(iii)

$$\pi = \frac{\begin{array}{c}\vdots\\ P_1(a_1+\ldots+a_m-b_1-\ldots-b_n)\end{array}}{P(a_1+\ldots+a_m-b_1-\ldots-b_n)}\,(P(x) \Leftarrow P_1(x)/\mathbb{ACUD})$$

where the clause $P(x) \Leftarrow P_1(x) \in \mathcal{A}$. Then the clauses $P(x) \Leftarrow P_1(x), \overline{P}(x) \Leftarrow \overline{P_1}(x) \in \mathcal{A}$. By induction hypothesis the atoms $P_1(a_1+\ldots+a_m+\overline{b_1}+\ldots+\overline{b_n})$ and $\overline{P_1}(\overline{a_1}+\ldots+\overline{a_m}+b_1+\ldots+b_n)$ are derivable in $\mathcal{B}/\mathbb{ACU}$. Hence we have the derivations

$$\frac{\begin{array}{c}\vdots\\ P_1(a_1+\ldots+a_m+\overline{b_1}+\ldots+\overline{b_n})\end{array}}{P(a_1+\ldots+a_m+\overline{b_1}+\ldots+\overline{b_n})}\,(P(x) \Leftarrow P_1(x)/\mathbb{ACU})$$

and

$$\frac{\begin{array}{c}\vdots\\ \overline{P_1}(\overline{a_1}+\ldots+\overline{a_m}+b_1+\ldots+b_n)\end{array}}{\overline{P}(\overline{a_1}+\ldots+\overline{a_m}+b_1+\ldots+b_n)}\,(\overline{P}(x)\Leftarrow\overline{P_1}(x)/\mathbb{ACU})$$

in $\mathcal{B}/\mathbb{ACU}$.

(iv)

$$\pi=\frac{\begin{array}{c}\vdots\\ P_1(a_1+\ldots+a_m-b_1-\ldots-b_n)\end{array}}{P(-a_1-\ldots-a_m+b_1+\ldots+b_n)}\,(P(-x)\Leftarrow P_1(x)/\mathbb{ACUD})$$

where the clause $P(-x)\Leftarrow P_1(x)\in\mathcal{A}$. Hence the clauses $P(x)\Leftarrow\overline{P_1}(x),\overline{P}(x)\Leftarrow P_1(x)\in\mathcal{B}$. By induction hypothesis the atoms $P_1(a_1+\ldots+a_m+\overline{b_1}+\ldots+\overline{b_n})$ and $\overline{P_1}(\overline{a_1}+\ldots+\overline{a_m}+b_1+\ldots+b_n)$ are derivable in $\mathcal{B}$. Hence we have the derivations

$$\frac{\begin{array}{c}\vdots\\ \overline{P_1}(\overline{a_1}+\ldots+\overline{a_m}+b_1+\ldots+b_n)\end{array}}{P(\overline{a_1}+\ldots+\overline{a_m}+b_1+\ldots+b_n)}\,(P(x)\Leftarrow\overline{P_1}(x)/\mathbb{ACU})$$

and

$$\frac{\begin{array}{c}\vdots\\ P_1(a_1+\ldots+a_m+\overline{b_1}+\ldots+\overline{b_n})\end{array}}{\overline{P}(a_1+\ldots+a_m+\overline{b_1}+\ldots+\overline{b_n})}\,(\overline{P}(x)\Leftarrow P_1(x)/\mathbb{ACU})$$

in $\mathcal{B}/\mathbb{ACU}$.

(v)

$$\pi=\frac{\begin{array}{c}\vdots\\ P_1(\sum_i a_i-\sum_k c_k)\end{array}\quad\begin{array}{c}\vdots\\ P_2(\sum_j b_j-\sum_l d_l)\end{array}}{P(\sum_i a_i+\sum_j b_j-\sum_k c_k-\sum_l d_l)}\,(P(x+y)\Leftarrow P_1(x)\wedge P_2(y)/\mathbb{ACUD})$$

where the clause $P(x+y)\Leftarrow P_1(x)\wedge P_2(y)\in\mathcal{A}$. Hence the clauses $P(x+y)\Leftarrow P_1(x)\wedge P_2(y),\overline{P}(x+y)\Leftarrow\overline{P_1}(x)\wedge\overline{P_2}(y)\in\mathcal{B}$. By induction hypothesis, the atoms $P_1(\sum_i a_i+\sum_k\overline{c_k})$, $\overline{P_1}(\sum_i\overline{a_i}+\sum_k c_k)$, $P_2(\sum_j b_j+\sum_l\overline{d_l})$ and $\overline{P_2}(\sum_j\overline{b_j}+\sum_l d_l)$ are derivable in $\mathcal{B}$. Hence we have the derivations

$$\frac{\begin{array}{c}\vdots\\ P_1(\sum_i a_i+\sum_k\overline{c_k})\end{array}\quad\begin{array}{c}\vdots\\ P_2(\sum_j b_j+\sum_l\overline{d_l})\end{array}}{P(\sum_i a_i+\sum_j b_j+\sum_k\overline{c_k}+\sum_l\overline{d_l}}\,(P(x+y)\Leftarrow P_1(x)\wedge P_2(y)/\mathbb{ACU})$$

and

$$\frac{\begin{array}{cc}\vdots & \vdots \\ \overline{P_1}(\sum_i \overline{a_i} + \sum_k c_k) & \overline{P_2}(\sum_j \overline{b_j} + \sum_l d_l)\end{array}}{\overline{P}(\sum_i \overline{a_i} + \sum_j \overline{b_j} + \sum_k c_k + \sum_l d_l)} (\overline{P}(x+y) \Leftarrow \overline{P_1}(x) \wedge \overline{P_2}(y)/\mathbb{ACU})$$

in $\mathcal{B}/\mathbb{ACU}$.

$\square$

**Lemma 28** *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 $\mathcal{B}/\mathbb{ACU}$ then $P(a_1 + ... + a_m - b_1 - ... - b_n)$ is derivable in $\mathcal{A}/\mathbb{ACUD}$.*

**Proof:** We assume that the derivation $\pi$ of $P(a_1 + ... + a_m + \overline{b_1} + ... + \overline{b_n})$ or $\overline{P}(\overline{a_1} + ... + \overline{a_m} + b_1 + ... + b_n)$ in $\mathcal{B}/\mathbb{ACU}$ uses only rules of the form $C/\mathbb{ACU}$ where the clause $C \in \mathcal{B}$. We do induction on the size of $\pi$. We have the following cases :

(i)

$$\pi = \frac{}{P(0)} (P(0)/\mathbb{ACU})$$

where the clause $P(0) \in \mathcal{B}$. Then the clause $P(0) \in \mathcal{A}$, and we have the derivation

$$\frac{}{P(0)} (P(0)/\mathbb{ACUD})$$

in $\mathcal{A}/\mathbb{ACUD}$.

(ii)

$$\pi = \frac{}{\overline{P}(0)} (\overline{P}(0)/\mathbb{ACU})$$

where the clause $\overline{P}(0) \in \mathcal{B}$. Then the clause $P(0) \in \mathcal{A}$, and we have the derivation

$$\frac{}{P(0)} (P(0)/\mathbb{ACUD})$$

in $\mathcal{A}/\mathbb{ACUD}$.

(iii)

$$\pi = \frac{}{P(a)} (P(a)/\mathbb{ACU})$$

where the clause $P(a) \in \mathcal{B}$. Then the clause $P(a) \in \mathcal{A}$, and we have the derivation

$$\frac{}{P(a)} (P(a)/\mathbb{ACUD})$$

in $\mathcal{A}/\mathbb{ACUD}$.

(iv)

$$\pi = \frac{}{\overline{P}(\overline{a})}\,(\overline{P}(\overline{a})/\mathbb{ACU})$$

where the clause $\overline{P}(\overline{a}) \in \mathcal{B}$. Then the clause $P(a) \in \mathcal{A}$, and we have the derivation

$$\frac{}{P(a)}\,(P(a)/\mathbb{ACUD})$$

in $\mathcal{A}/\mathbb{ACUD}$.

(v)

$$\pi = \frac{\begin{array}{c}\vdots\\ P_1(a_1 + ... + a_m + \overline{b_1} + ... + \overline{b_n})\end{array}}{P(a_1 + ... + a_m + \overline{b_1} + ... + \overline{b_n})}\,(P(x) \Leftarrow P_1(x)/\mathbb{ACUD})$$

where the clause $P(x) \Leftarrow P_1(x) \in \mathcal{B}$. Then the clause $P(x) \Leftarrow P_1(x) \in \mathcal{A}$. By induction hypothesis, the atom $P_1(a_1 + ... + a_m - b_1 - ... - b_n)$ is derivable in $\mathcal{A}/\mathbb{ACUD}$ and hence we have the derivation

$$\frac{\begin{array}{c}\vdots\\ P_1(a_1 + ... + a_m - b_1 - ... - b_n)\end{array}}{P(a_1 + ... + a_m - b_1 - ... - b_n)}\,(P(x) \Leftarrow P_1(x)/\mathbb{ACU})$$

in $\mathcal{A}/\mathbb{ACU}$.

(vi)

$$\pi = \frac{\begin{array}{c}\vdots\\ P_1(\overline{a_1} + ... + \overline{a_m} + b_1 + ... + b_n)\end{array}}{P(\overline{a_1} + ... + \overline{a_m} + b_1 + ... + b_n)}\,(\overline{P}(x) \Leftarrow \overline{P_1}(x)/\mathbb{ACUD})$$

where the clause $\overline{P}(x) \Leftarrow \overline{P}_1(x) \in \mathcal{B}$. Then the clause $P(x) \Leftarrow P_1(x) \in \mathcal{A}$. By induction hypothesis, the atom $P_1(a_1 + ... + a_m - b_1 - ... - b_n)$ is derivable in $\mathcal{A}/\mathbb{ACUD}$ and hence we have the derivation

$$\frac{\begin{array}{c}\vdots\\ P_1(a_1 + ... + a_m - b_1 - ... - b_n)\end{array}}{P(a_1 + ... + a_m - b_1 - ... - b_n)}\,(P(x) \Leftarrow P_1(x)/\mathbb{ACU})$$

in $\mathcal{A}/\mathbb{ACU}$.

(vii)

$$\pi = \frac{\begin{array}{c}\vdots\\ P_1(\overline{a_1} + ... + \overline{a_m} + b_1 + ... + b_n)\end{array}}{P(\overline{a_1} + ... + \overline{a_m} + b_1 + ... + b_n)}\,(P(x) \Leftarrow \overline{P_1}(x)/\mathbb{ACUD})$$

where the clause $P(x) \Leftarrow \overline{P}_1(x) \in \mathcal{B}$. Then the clause $P(-x) \Leftarrow P_1(x) \in \mathcal{A}$. By induction hypothesis, the atom $P_1(a_1 + ... + a_m - b_1 - ... - b_n)$ is derivable in $\mathcal{A}/\mathbb{ACUD}$ and hence we have the derivation

$$\dfrac{\begin{array}{c}\vdots\\ P_1(a_1+...+a_m-b_1-...-b_n)\end{array}}{P(-a_1-\ldots-a_m+b_1+\ldots+b_n)}\,(P(-x)\Leftarrow P_1(x)/\mathbb{ACU})$$

in $\mathcal{A}/\mathbb{ACU}$.

(viii)

$$\pi=\dfrac{\begin{array}{c}\vdots\\ P_1(a_1+...+a_m+\overline{b_1}+...+\overline{b_n})\end{array}}{\overline{P}(a_1+...+a_m+\overline{b_1}+...+\overline{b_n})}\,(\overline{P}(x)\Leftarrow P_1(x)/\mathbb{ACUD})$$

where the clause $\overline{P}(x)\Leftarrow P_1(x)\in\mathcal{B}$. Then the clause $P(-x)\Leftarrow P_1(x)\in\mathcal{A}$. By induction hypothesis, the atom $P_1(a_1+...+a_m-b_1-...-b_n)$ is derivable in $\mathcal{A}/\mathbb{ACUD}$ and hence we have the derivation

$$\dfrac{\begin{array}{c}\vdots\\ P_1(a_1+...+a_m-b_1-...-b_n)\end{array}}{P(-a_1-\ldots-a_m+b_1+\ldots+b_n)}\,(P(-x)\Leftarrow P_1(x)/\mathbb{ACU})$$

in $\mathcal{A}/\mathbb{ACU}$.

(ix)

$$\pi=\dfrac{\begin{array}{c}\vdots\\ \sum_i a_i+\sum_k\overline{c_k}\end{array}\quad\begin{array}{c}\vdots\\ \sum_j b_j+\sum_l\overline{d_l}\end{array}}{P(\sum_i a_i+\sum_j b_j+\sum_k\overline{c_k}+\sum_l\overline{d_l})}\,(P(x+y)\Leftarrow P_1(x)\wedge P_2(y)/\mathbb{ACU})$$

where the clause $P(x+y)\Leftarrow P_1(x)\wedge P_2(y)\in\mathcal{B}$. Then the clause $P(x+y)\Leftarrow P_1(x)\wedge P_2(y)\in\mathcal{A}$. By induction hypothesis, the atoms $P_1(\sum_i a_i-\sum_k c_k)$ and $P_2(\sum_j b_j-\sum_l d_l)$ are derivable in $\mathcal{A}/\mathbb{ACUD}$ and hence we have the derivation

$$\dfrac{\begin{array}{c}\vdots\\ P_1(\sum_i a_i-\sum_k c_k)\end{array}\quad\begin{array}{c}\vdots\\ P_2(\sum_j b_j-\sum_l d_l)\end{array}}{P(\sum_i a_i+\sum_j b_j-\sum_k c_k-\sum_l d_l)}\,(P(x+y)\Leftarrow P_1(x)\wedge P_2(y)/\mathbb{ACUD})$$

in $\mathcal{A}/\mathbb{ACUD}$.

(x)

$$\pi=\dfrac{\begin{array}{c}\vdots\\ \sum_i\overline{a_i}+\sum_k c_k\end{array}\quad\begin{array}{c}\vdots\\ \sum_j\overline{b_j}+\sum_l d_l\end{array}}{\overline{P}(\sum_i\overline{a_i}+\sum_j\overline{b_j}+\sum_k c_k+\sum_l d_l)}\,(\overline{P}(x+y)\Leftarrow\overline{P_1}(x)\wedge\overline{P_2}(y)/\mathbb{ACU})$$

where the clause $\overline{P}(x+y) \Leftarrow \overline{P_1}(x) \wedge \overline{P_2}(y) \in \mathcal{B}$. Then the clause $P(x+y) \Leftarrow P_1(x) \wedge P_2(y) \in \mathcal{A}$. By induction hypothesis, the atoms $P_1(\sum_i a_i - \sum_k c_k)$ and $P_2(\sum_j b_j - \sum_l d_l)$ are derivable in $\mathcal{A}/\mathbb{ACUD}$ and hence we have the derivation

$$\frac{\begin{array}{cc} \vdots & \vdots \\ P_1(\sum_i a_i - \sum_k c_k) & P_2(\sum_j b_j - \sum_l d_l) \end{array}}{P(\sum_i a_i + \sum_j b_j - \sum_k c_k - \sum_l d_l)}\,(P(x+y) \Leftarrow P_1(x) \wedge P_2(y)/\mathbb{ACUD})$$

in $\mathcal{A}/\mathbb{ACUD}$.

$\square$

Now if the final state of $\mathcal{A}$ is $P$ then we name the same $P$ as the final state of $\mathcal{B}$. Then from Lemmas 27 and 28 and modulo the correspondence of languages described above we have $\mathcal{L}(\mathcal{A}/\mathbb{ACUD}) = \mathcal{L}(\mathcal{B}/\mathbb{ACU})$. Hence modulo this correspondences of languages we conclude that

**Theorem 18** *The language accepted by a constant-only $\mathbb{ACUD}$ automata with constants from $\Sigma_f$ 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 $\mathbb{ACUD}$ automaton with constants from $\Sigma_f$.*

## 7.4 Conclusion

We showed that one-way equational tree automata accept the equational closures of the languages accepted by the corresponding non-equational tree automata. This implies that emptiness of one-way $\mathbb{E}$ tree automata is decidable for any $\mathbb{E}$. We then showed how certain parts of our derivations in equational tree automata can be reused to get new derivations. Finally we discussed an important class of our automata, called the constant-only automata, for which we showed characterizations using semilinear sets of Presburger-definable sets. Together the results in this Section will be used very often throughout the rest of this thesis.

# Chapitre 8

# Intersection des automates équationnels unidirectionnels (Intersection of One-Way Equational Tree Automata)

Dans ce chapitre nous étudions la clôture par intersection des automates unidirectionnels modulo nos théories équationnelles. Nous montrons que les automates unidirectionnels, modulo toutes les théories associatives et commutatives que nous considérons ($\mathbb{AC}$, $\mathbb{ACU}$, $\mathbb{ACUX}$, $\mathbb{ACUX}_n$, $\mathbb{ACUD}$, $\mathbb{ACUM}$, $\mathbb{ACUI}$), sont clos par intersection. En ce sens, ces automates ont un comportement similaire aux automates d'arbres non équationnels. Mais ceci ne se généralise pas à toute théorie équationnelle. En particulier nous avons vu au chapitre 4 que les automates unidirectionnels modulo $\mathbb{A}$ ne sont pas clos par intersection, une conséquence du fait que la classe des langages algébriques n'est pas close par intersection. Nos résultats de clôture impliquent aussi la décidabilité du vide de l'intersection, qui est la question importante en vérification de protocoles cryptographiques. Notons aussi que la décidabilité de l'appartenance est impliquée par ces resultats.

In this chapter we study the closure under intersection of one-way automata modulo the various equational theories. We show that the one-way automata modulo all the associative commutative theories that we consider ($\mathbb{AC}$, $\mathbb{ACU}$, $\mathbb{ACUX}$, $\mathbb{ACUX}_n$, $\mathbb{ACUD}$, $\mathbb{ACUM}$, $\mathbb{ACUI}$) are closed under intersection. In this respect these automata have similar behavior than non-equational tree automata. This result however does not generalize to all equational theories. In particular we have seen in Chapter 4 that one-way $\mathbb{A}$ tree automata are not closed under intersection. Our closure results also imply decidability of intersection-emptiness, which is the relevant question in verification of cryptographic protocols. Also note that decidability of membership is implied by these results.

To deal with intersection, our starting idea is the product construction for classical tree automata. In this approach given two automata $\mathcal{A}_1$ and $\mathcal{A}_2$, we construct an automaton $\mathcal{A}$ which has states $(P, Q)$ where $P$ is a state in $\mathcal{A}_1$ and $Q$ is a state in $\mathcal{A}_2$. We intend $(P, Q)$ to accept the intersection of the languages accepted by $P$ and $Q$. Ignoring epsilon clauses for the moment, it then suffices to add clauses

$$(P, Q)(f(x_1, \ldots, x_n)) \Leftarrow (P_1, Q_1)(x_1) \wedge \ldots \wedge (P_n, Q_n)(x_n)$$

to $\mathcal{A}$ corresponding to every pair of clauses

$$\begin{array}{c} P(f(x_1, \ldots, x_n)) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n) \in \mathcal{A}_1 \\ Q(f(x_1, \ldots, x_n)) \Leftarrow Q_1(x_1) \wedge \ldots \wedge Q_n(x_n) \in \mathcal{A}_2 \end{array}$$

Clearly this is not sufficient in the equational case, as shown in the following example :

**Example 7** *Consider automata*

$$\begin{array}{c} \mathcal{A}_1 = \{P_1(a), P_2(b), P(x+y) \Leftarrow P_1(x) \wedge P_2(y) \\ \mathcal{A}_2 = \{Q_1(b), Q_2(a), Q(x+y) \Leftarrow Q_1(x) \wedge Q_2(y) \end{array}$$

*The new automaton $\mathcal{A}$ obtained using the above procedure is*

$$\mathcal{A} = \{(P, Q)(x+y) \Leftarrow (P_1, Q_1)(x) \wedge (P_2, Q_2)(y)$$

*This means that $(P, Q)$ does not accept any term. However if we are working modulo the theory $\mathbb{C}$ of commutativity, then both the terms $a + b$ and $b + a$ are accepted at each of the states $P$ and $Q$.*

## 8.1 One-Way $\mathbb{ACU}$ Automata

We show in this section that one-way $\mathbb{ACU}$ automata are effectively closed under intersection. Fix a signature $\Sigma$ containing at least the symbols $+$ and $0$. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be one-way equational tree automata on signature $\Sigma$. We will define an automaton $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}/\mathbb{ACU}) = \mathcal{L}(\mathcal{A}_1/\mathbb{ACU}) \cap \mathcal{L}(\mathcal{A}_2/\mathbb{ACU})$. Let the set of states in $\mathcal{A}_1$ and $\mathcal{A}_2$ be $\mathbb{P}$ and $\mathbb{Q}$ respectively.

We introduce new predicate symbols $(P, Q)$ and $\widehat{(P, Q)}$ for $(P, Q) \in \mathbb{P} \times \mathbb{Q}$. Predicate $(P, Q)$ is expected to accept the intersection of the languages accepted by $P$ and $Q$. $\widehat{(P, Q)}$ is expected to accept the functional terms among the terms accepted at $(P, Q)$. We introduce new constants $a_{P,Q}$ for each $(P, Q) \in \mathbb{P} \times \mathbb{Q}$. In the construction below, $a_{P,Q}$ acts as abstraction for terms accepted at $\widehat{P, Q}$. Define automaton $\mathcal{B}_1 = \mathcal{A}_{1eq} \cup \{P(a_{P,Q}) \mid (P, Q) \in \mathbb{P} \times \mathbb{Q}\}$. Define automaton $\mathcal{B}_2 = \mathcal{A}_{2eq} \cup \{Q(a_{P,Q}) \mid (P, Q) \in \mathbb{P} \times \mathbb{Q}\}$. $\mathcal{B}_i$ intuitively represents the set of possible derivations using the clauses of $\mathcal{A}_{ieq}$, with the constants being used as abstractions for functional terms. For $(P, Q) \in$

$\mathbb{P} \times \mathbb{Q}$, $\mathcal{L}_P(\mathcal{B}_1/\mathbb{ACU}), \mathcal{L}_Q(\mathcal{B}_2/\mathbb{ACU})$ are semilinear sets, hence $\mathcal{L}_P(\mathcal{B}_1/\mathbb{ACU}) \cap \mathcal{L}_Q(\mathcal{B}_2/\mathbb{ACU})$ is a semilinear set. Hence we can define constant-only automaton $\mathcal{A}_{P,Q}$ with a final state $F_{P,Q}$ such that $\mathcal{L}_{F_{P,Q}}(\mathcal{A}_{P,Q}/\mathbb{ACU}) = \mathcal{L}_P(\mathcal{B}_1/\mathbb{ACU}) \cap \mathcal{L}_Q(\mathcal{B}_2/\mathbb{ACU})$. We assume that the automata $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_{P,Q}$'s are all built from mutually disjoint sets of states.

The required automaton $\mathcal{A}$ contains

1. clause $(P,Q)(x) \Leftarrow F_{P,Q}(x)$ for $(P,Q) \in \mathbb{P} \times \mathbb{Q}$.
2. clauses from $\mathcal{A}_{P,Q_{eq}}$ for $(P,Q) \in \mathbb{P} \times \mathbb{Q}$.
3. clause $R(x) \Leftarrow (P',Q')(x)$ for each base clause $R(a_{P',Q'})$ in $\mathcal{A}_{P,Q}$ for $(P,Q) \in \mathbb{P} \times \mathbb{Q}$.
4. clause

$$\widehat{(P,Q)}(f(x_1,\ldots,x_n)) \Leftarrow (P_1,Q_1)(x_1) \wedge \ldots \wedge (P_n,Q_n)(x_n)$$

for clauses

$$P(f(x_1,\ldots,x_n)) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n) \in \mathcal{A}_{1free}$$
$$Q(f(x_1,\ldots,x_n)) \Leftarrow Q_1(x_1) \wedge \ldots \wedge Q_n(x_n) \in \mathcal{A}_{2free}$$

Clearly $\mathcal{A}$ is a one-way automaton. That it represents the intersection of $\mathcal{A}_1$ and $\mathcal{A}_2$ is stated by Lemmas 29 and 30 respectively.

**Lemma 29** *If $P(t)$ and $Q(t)$ are derivable in $\mathcal{A}_1/\mathbb{ACU}$ and $\mathcal{A}_2/\mathbb{ACU}$ respectively, then $(P,Q)(t)$ is derivable in $\mathcal{A}/\mathbb{ACU}$.*

**Proof:** We do induction of the sum of the sizes of the derivations $\pi_1$ and $\pi_2$ of $P(t)$ and $Q(t)$ respectively. $\pi_1$ must have functional support of the form $P_1(t_1),\ldots,P_n(t_n)$ and $\pi_2$ must have a functional support of the form $Q_1(t_1),\ldots,Q_n(t_n)$ such that $t = t_1 + \ldots + t_n$ and we have

$$\pi_1 = \cfrac{\begin{matrix}\vdots & & \vdots \\ P_1(t_1) & \ldots & P_n(t_n)\end{matrix}}{P(t_1+\ldots+t_n)}\ (\mathcal{A}_{1eq}/\mathbb{ACU}) \tag{8.1}$$

$$\pi_2 = \cfrac{\begin{matrix}\vdots & & \vdots \\ Q_1(t_1) & \ldots & Q_n(t_n)\end{matrix}}{Q(t_1+\ldots+t_n)}\ (\mathcal{A}_{2eq}/\mathbb{ACU}) \tag{8.2}$$

By definition of $\mathcal{B}_1$, the atoms $P_1(a_{P_1,Q_1}),\ldots,P_n(a_{P_n,Q_n})$ are derivable in $\mathcal{B}_1/\mathbb{ACU}$. Since $\mathcal{A}_{1eq} \subseteq \mathcal{B}_1$, hence from (8.1) and using Lemma 21, $P(a_{P_1,Q_1}+\ldots+a_{P_n,Q_n})$ is derivable in $\mathcal{B}_1/\mathbb{ACU}$. Similarly $Q(a_{P_1,Q_1}+\ldots+a_{P_n,Q_n})$ is derivable in $\mathcal{B}_2/\mathbb{ACU}$ respectively. Hence $F_{P,Q}(a_{P_1,Q_1}+\ldots+a_{P_n,Q_n})$ is derivable in $\mathcal{A}_{P,Q}/\mathbb{ACU}$. This derivation $\pi_3$ must have a functional support of the form $R_1(a_{P_1,Q_1}),\ldots,R_n(a_{P_n,Q_n})$ and we must have

$$\pi_3 = \cfrac{\overline{R_1(a_{P_1,Q_1})} \quad \ldots \quad \overline{R_n(a_{P_n,Q_n})}}{F_{P,Q}(a_{P_1,Q_1}+\ldots+a_{P_n,Q_n})}\ (\mathcal{A}_{P,Q_{eq}}/\mathbb{ACU}) \tag{8.3}$$

For each $1 \leq i \leq n$, since $t_i$ is functional we must have $t_i = f_i(t_i^1,\ldots,t_i^{k_i})$ for some free $f_i$ of arity $k_i$. As $P_i(t_i)$ is in the functional support of $\pi_1$, we must have a clause $P_i(f_i(x_1,\ldots,x_{k_i}) \Leftarrow$

$P_i^1(x_1) \wedge \ldots \wedge P_i^{k_i}(x_{k_i})$ in $\mathcal{A}_1$ such that for $1 \leq j \leq k_i$, $P_i^j(t_i^j)$ is derivable in $\mathcal{A}_1/\mathbb{ACU}$ with derivation strictly smaller than $\pi_1$. Similarly we have a clause $Q_i(f_i(x_1, \ldots, x_{k_i}) \Leftarrow Q_i^1(x_1) \wedge \ldots \wedge Q_i^{k_i}(x_{k_i})$ in $\mathcal{A}_2$ such that for $1 \leq j \leq k_i$, $Q_i^j(t_i^j)$ is derivable in $\mathcal{A}_2/\mathbb{ACU}$ with derivations strictly smaller than $\pi_2$. By induction hypothesis we have derivation of $(P_i^j, Q_i^j)(t_i^j)$ in $\mathcal{A}/\mathbb{ACU}$. Also the clause $(\widehat{P_i, Q_i})(f_i(x_1, \ldots, x_{k_i})) \Leftarrow (P_i^1, Q_i^1)(x_1) \wedge \ldots \wedge (P^{k_i}, Q^{k_i})(x_{k_i}) \in \mathcal{A}$. Hence the atom $(\widehat{P_i, Q_i})(t_i)$ is derivable in $\mathcal{A}/\mathbb{ACU}$. Since the clause $R_i(a_{P_i,Q_i})$ is in $\mathcal{A}_{P,Q}$, hence the clause $R_i(x) \Leftarrow (P_i, Q_i)(x)$ is in $\mathcal{A}$. Hence $R_i(t_i)$ is derivable in $\mathcal{A}/\mathbb{ACU}$. Also $\mathcal{A}_{P,Q_{eq}} \subseteq \mathcal{A}$. Hence using (8.3) and Lemma 21, we get a derivation of $F_{P,Q}(t_1 + \ldots + t_n)$ in $\mathcal{A}/\mathbb{ACU}$. Finally we use the clause $(P, Q)(x) \Leftarrow F_{P,Q}(x)$ to get a derivation of $(P, Q)(t)$ in $\mathcal{A}/\mathbb{ACU}$. □

**Lemma 30** *For $(P, Q) \in \mathbb{P} \times \mathbb{Q}$, if $(P, Q)(t)$ is derivable in $\mathcal{A}/\mathbb{ACU}$ then $P(t)$ and $Q(t)$ are derivable in $\mathcal{A}_1/\mathbb{ACU}$ and $\mathcal{A}_2/\mathbb{ACU}$ respectively.*

**Proof:** We do induction on the size of the derivation $\pi$ of $(P, Q)(t)$. Since $(P, Q)(x) \Leftarrow F_{P,Q}(x)$ is the only clause which has the predicate $(P, Q)$ in the head, $\pi$ uses it as the last clause, and $F_{P,Q}(t)$ is derivable in $\mathcal{A}/\mathbb{ACU}$. Again by examining the clauses in $\mathcal{A}$, we conclude that the derivation $\pi_1$ of $F_{P,Q}(t)$ must have functional support of the form $(\widehat{P_1, Q_1})(t_1), \ldots, (\widehat{P_n, Q_n})(t_n)$ such that $t =_{\mathbb{ACU}} t_1 + \ldots + t_n$ and $\pi_1$ must be of the form

$$\dfrac{\dfrac{\begin{matrix}\vdots\\(\widehat{P_1, Q_1})(t_1)\end{matrix}}{R_1(t_1)}\,(R_1(x) \Leftarrow (\widehat{P_1, Q_1})(x)) \quad \ldots \quad \dfrac{\begin{matrix}\vdots\\(\widehat{P_n, Q_n})(t_n)\end{matrix}}{R_n(t_n)}\,(R_n(x) \Leftarrow (\widehat{P_n, Q_n})(x))}{F_{P,Q}(t_1 + \ldots + t_n)}\,(\mathcal{A}_{P,Q_{eq}}/\mathbb{ACU}) \tag{8.4}$$

Hence the clause $R_i(a_{P_i,Q_i}) \in \mathcal{A}_{P,Q}$ for $1 \leq i \leq n$. Hence $R_i(a_{P_i,Q_i})$ is derivable in $\mathcal{A}_{P,Q}/\mathbb{ACU}$. Also $\mathcal{A}_{P,Q_{eq}} \subseteq \mathcal{A}$. Hence using (8.4) and Lemma 21, $F_{P,Q}(a_{P_1,Q_1} + \ldots + a_{P_n,Q_n})$ is derivable in $\mathcal{A}_{P,Q}/\mathbb{ACU}$.

Now for $1 \leq i \leq n$, since $t_i$ is functional, $t_i = f_i(t_i^1, \ldots, t_i^{k_i})$ for some free $f_i$ of arity $k_i$ and some terms $t_i^1, \ldots, t_i^{k_i}$. As $\widehat{P_i, Q_i}(t_i)$ is in the functional support of $\pi_1$, there is a clause $\widehat{P_i, Q_i}(f_i(x_1, \ldots, x_{k_i})) \Leftarrow (P_i^1, Q_i^1)(x_1) \wedge \ldots \wedge (P_i^{k_i}, Q_i^{k_i})(x_{k_i})$ in $\mathcal{A}$ corresponding to clauses $P_i(f_i(x_1, \ldots, x_{k_i})) \Leftarrow P_i^1(x_1) \wedge \ldots \wedge P_i^{k_i}(x_{k_i})$ and $Q_i(f_i(x_1, \ldots, x_{k_i})) \Leftarrow Q_i^1(x_1) \wedge \ldots \wedge Q_i^{k_i}(x_{k_i})$ in $\mathcal{A}_{1free}$ and $\mathcal{A}_{2free}$ respectively, and for $1 \leq j \leq k_i$, $(P_i^j, Q_i^j)(t_i^j)$ is derivable in $\mathcal{A}/\mathbb{ACU}$. By induction hypothesis we get derivations of $P_i^j(t_i^j)$ and $Q_i^j(t_i^j)$ in $\mathcal{A}_1/\mathbb{ACU}$ and $\mathcal{A}_2/\mathbb{ACU}$ respectively. Hence using the above clauses from $\mathcal{A}_{1free}$ and $\mathcal{A}_{2free}$ respectively, $P_i(t_i)$ and $Q_i(t_i)$ are derivable in $\mathcal{A}_1/\mathbb{ACU}$ and $\mathcal{A}_2/\mathbb{ACU}$ respectively.

Since $F_{P,Q}(a_{P_1,Q_1} + \ldots + a_{P_n,Q_n})$ is derivable in $\mathcal{A}_{P,Q}/\mathbb{ACU}$ so $P(a_{P_1,Q_1} + \ldots + a_{P_n,Q_n})$ must be derivable in $\mathcal{B}_1/\mathbb{ACU}$. This derivation (call it $\pi_2$) must have a functional support $P_1(a_{P_1,Q_1}), \ldots, P_n(a_{P_n,Q_n})$ and we must have

$$\pi_2 = \dfrac{\overline{P_1(a_{P_1,Q_1})} \quad \ldots \quad \overline{P_n(a_{P_n,Q_n})}}{P(a_{P_1,Q_1} + \ldots + a_{P_n,Q_n})}\,(\mathcal{B}_{1eq}/\mathbb{ACU}) \tag{8.5}$$

We know that $P_1(t_1), \ldots, P_n(t_n)$ are derivable in $\mathcal{A}_1/\mathbb{ACU}$. Also by definition $\mathcal{B}_{1eq} = \mathcal{A}_{1eq}$. Hence from (8.5) and Lemma 21 $P(t_1 + \ldots + t_n)$ is derivable in $\mathcal{A}_1/\mathbb{ACU}$. Similarly $Q(t_1 + \ldots + t_n)$ is derivable in $\mathcal{A}_2/\mathbb{ACU}$. □

Let $P$ and $Q$ be the final states of $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively. We name $(P, Q)$ as the final state of $\mathcal{A}$. From Lemmas 29 and 30, $\mathcal{L}_{(P,Q)}(\mathcal{A}/\mathbb{ACU}) = \mathcal{L}_P(\mathcal{A}_1/\mathbb{ACU}) \cap \mathcal{L}(\mathcal{A}_2/\mathbb{ACU})$ implying that $\mathcal{L}(\mathcal{A}/\mathbb{ACU}) = \mathcal{L}(\mathcal{A}_1/\mathbb{ACU}) \cap \mathcal{L}(\mathcal{A}_2/\mathbb{ACU})$. This allows us to conclude that

**Theorem 19** *One-way $\mathbb{ACU}$-tree automata are effectively closed under intersection.*

### 8.1.1 One-Way $\mathbb{AC}$ Automata

Observe that the presence of the 0 symbol is not at all crucial for the above construction and proofs for $\mathbb{ACU}$ automata. It is easy to modify the above proof to work for the $\mathbb{AC}$ case. Theorem 17 gives us a correspondence between constant-only $\mathbb{AC}$ automata and semilinear sets not containing the 0 tuple. Without repeating the whole proof we merely state the result :

**Theorem 20** *One-way $\mathbb{AC}$-tree automata are effectively closed under intersection.*

## 8.2 One-Way $\mathbb{ACUX}$ Automata

Next we show closure under intersection of one-way $\mathbb{ACUX}$-automata. Fix a signature $\Sigma$ containing at least the symbols $+$ and $0$. Consider a one-way $\mathbb{ACUX}$ automaton $\mathcal{A}$ 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 $\mathcal{A}_{inter}$ 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 $(P, Q)$.

Define automaton $\mathcal{B} = \mathcal{A}_{eq} \cup \{P(a_{P,Q}), P(b_{P,Q}) \mid P, Q \in \mathbb{P}\}$. From Theorem 16 $\mathcal{L}_P(\mathcal{B}/\mathbb{ACU})$ 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 $t \in \mathcal{L}_P(\mathcal{B}/\mathbb{ACU})$ 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 Theorem 16, we can construct a constant-only automaton $\mathcal{A}_{P,Q,S,T}$ with final state $F_{P,Q,S,T}$ such that $\mathcal{L}(\mathcal{A}_{P,Q,S,T}/\mathbb{ACU}) = \mathcal{L}'_{P,S} \cap \mathcal{L}'_{Q,T}$. We assume that automata $\mathcal{A}_{P,Q,S,T}$'s are built from mutually disjoint sets of (fresh) states.

The required automaton $\mathcal{A}_{inter}$ 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 of $\mathcal{A}_{P,Q,S,T_{eq}}$ for each $P, Q \in \mathbb{P}$ and $S, T \subseteq S_2$.
- for each base clause $R(a_{R',R''})$ in some $\mathcal{A}_{P,Q,S,T}$, the clause $R(x) \Leftarrow \widehat{(R', R'')}(x)$.
- for each pair of clauses

$$P(f(x_1, \ldots, x_n)) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n)$$
$$Q(f(x_1, \ldots, x_n)) \Leftarrow Q_1(x_1) \wedge \ldots \wedge Q_n(x_n)$$

  in $\mathcal{A}_{free}$, the clause

$$\widehat{(P, Q)}(f(x_1, \ldots, x_n)) \Leftarrow (P_1, Q_1)(x_1) \wedge \ldots \wedge (P_n, Q_n)(x_n)$$

The idea in defining $\mathcal{B}$ and $\mathcal{A}_{P,Q,S,T}$ is to compute all possible derivations using clauses of the equational part. The $a, b$'s act as abstractions for the functional terms. If $t =_{\mathbb{ACUX}} t'$ then we must have $t =_{\mathbb{ACU}} t_1+...+t_m+u_1+v_1+...+u_n+v_n$, $t' = t'_1+...+t'_m+u'_1+v'_1+...+u'_p+v'_p$, $t_i, t'_i, u_i, u'_i, v_i, v'_i$ being functional, such that $t_i =_{\mathbb{ACUX}} t'_i, u_i =_{\mathbb{ACUX}} v_i, u'_i =_{\mathbb{ACUX}} v'_i$. 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 $\mathbb{X}$. 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 $\mathcal{A}_{inter}$. 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 $\mathbb{ACU}$ automata. This is made precise by Lemmas 31 and 32.

**Lemma 31** *If $P(t')$ and $Q(t'')$ are derivable in $\mathcal{A}/\mathbb{ACU}$ and $t' =_{\mathbb{ACUX}} t''$, then for some $t =_{\mathbb{ACUX}} t'$, $(P,Q)(t)$ is derivable in $\mathcal{A}_{inter}/\mathbb{ACU}$.*

**Proof:** We do induction on the sum of the sizes of the derivations $\pi_1$ and $\pi_2$ of $P(t')$ and $Q(t'')$ respectively. Since $t' =_{\mathbb{ACUX}} t''$ we must have

$$t' =_{\mathbb{ACU}} t'_1 + \ldots + t'_m + (u'_1 + u''_1) + \ldots + (u'_n + u''_n) \tag{8.6}$$
$$t'' =_{\mathbb{ACU}} t''_1 + \ldots + t''_m + (v'_1 + v''_1) + \ldots + (v'_p + v''_p) \tag{8.7}$$

such that $m, n, p \geq 0$, $t'_i, t''_i$ are functional and $t'_i =_{\mathbb{ACUX}} t''_i$ for $i \leq i \leq m$, $u'_i, u''_i$ are functional and $u'_i =_{\mathbb{ACUX}} u''_i$ for $1 \leq i \leq n$, and $v'_i, v''_i$ are functional and $v'_i =_{\mathbb{ACUX}} v''_i$ for $1 \leq i \leq p$. $\pi_1$ and $\pi_2$ must have functional supports of the form $P_1(t'_1), \ldots, P_m(t'_m), I_1(u'_1), I'_1(u''_1), \ldots, I_n(u'_n), I'_n(u''_n)$ and $Q_1(t''_1), \ldots, Q_m(t''_m), J_1(v'_1), J'_1(v''_1), \ldots, J_p(v'_p), J'_p(v''_p)$ respectively such that

$$\pi_1 = \frac{\begin{matrix}\vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots \\ P_1(t'_1) & \ldots & P_m(t'_m) & I_1(u'_1) & I'_1(u''_1) & \ldots & I_n(u'_n) & I'_n(u''_n)\end{matrix}}{P(t'_1 + \ldots + t'_m + (u'_1 + u''_1) + \ldots + (u'_n + u''_n))}\ (\mathcal{A}_{eq}/\mathbb{ACU}) \tag{8.8}$$

$$\pi_2 = \frac{\begin{matrix}\vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots \\ Q_1(t''_1) & \ldots & Q_m(t''_m) & J_1(v'_1) & J'_1(v''_1) & \ldots & J_n(v'_n) & J'_n(v''_n)\end{matrix}}{P(t''_1 + \ldots + t''_m + (v'_1 + v''_1) + \ldots + (v'_p + v''_p))}\ (\mathcal{A}_{eq}/\mathbb{ACU}) \tag{8.9}$$

By definition of $\mathcal{B}$, the atoms $P_1(a_{P_1,Q_1}), \ldots, P_m(a_{P_m,Q_m}), I_1(b_{I_1,I'_1}), I'_1(b_{I_1,I'_1}), \ldots, I_n(b_{I_n,I'_n}), I'_n(b_{I_n,I'_n})$ are derivable in $\mathcal{B}/\mathbb{ACU}$. Also $\mathcal{A}_{eq} \subseteq \mathcal{B}$. Hence from (8.8) and Lemma 21, $P(a_{P_1,Q_1}+\ldots+a_{P_m,Q_m}+2b_{I_1,I'_1}+\ldots+2b_{I_n,I'_n})$ is derivable in $\mathcal{B}/\mathbb{ACU}$. Similarly $Q(a_{P_1,Q_1}+\ldots+a_{P_m,Q_m}+2b_{J_1,J'_1}+\ldots+2b_{J_p,J'_p})$ is derivable in $\mathcal{B}/\mathbb{ACU}$. Let $S = \{b_{I_1,I'_1}, \ldots, b_{I_n,I'_n}\}$ and $T = \{b_{J_1,J'_1}, \ldots, b_{J_p,J'_p}\}$. We have $a_{P_1,Q_1} + \ldots + a_{P_m,Q_m} + 2b_{I_1,I'_1} + \ldots + 2b_{I_n,I'_n} \in \mathcal{L}_{P,S}$ and $a_{P_1,Q_1} + \ldots + a_{P_m,Q_m} \in \mathcal{L}'_{P,S}$. Similarly $a_{P_1,Q_1} + \ldots + a_{P_m,Q_m} \in \mathcal{L}'_{Q,T}$. Hence $F_{P,Q,S,T}(a_{P_1,Q_1} + \ldots + a_{P_m,Q_m})$ is derivable in $\mathcal{A}_{P,Q,S,T}/\mathbb{ACU}$. This derivation $\pi_3$ must have functional support of the form $R_1(a_{P_1,Q_1}), \ldots, R_m(a_{P_m,Q_m})$ and we must have

$$\pi_3 = \frac{\overline{R_1(a_{P_1,Q_1})} \quad \ldots \quad \overline{R_m(a_{P_m,Q_m})}}{F_{P,Q,S,T}(a_{P_1,Q_1} + \ldots + a_{P_m,Q_m})}\ (\mathcal{A}_{P,Q_{eq}}/\mathbb{ACU}) \tag{8.10}$$

Since $P_i(t_i')$ and $Q_i(t_i'')$ are in the functional supports of $\pi_1$ and $\pi_2$ respectively, we have $t_i' = f_i(t_i'^1, \ldots, t_i'^{k_i})$ and $t_i'' = f_i(t_i''^1, \ldots, t_i''^{k_i})$ for some free $f_i$ of arity $k_i$ such that $t_i'^j =_{\mathbb{ACUX}} t_i''^j$, the derivation of $P_i(t_i')$ uses $P_i(f_i(x_1, \ldots, x_{k_i})) \Leftarrow P_i^1(x_1) \wedge \ldots \wedge P_i^{k_i}(x_{k_i})$ as the last clause, the derivation of $Q_i(t_i'')$ uses $Q_i(f_i(x_1, \ldots, x_{k_i})) \Leftarrow Q_i^1(x_1) \wedge \ldots \wedge Q_i^{k_i}(x_{k_i})$ as the last clause, and for $1 \leq j \leq k_i$, the atoms $P_i^j(t_i'^j)$ and $Q_i^j(t_i''^j)$ are derivable in $\mathcal{A}/\mathbb{ACU}$. By induction hypothesis, we have $t_i^j =_{\mathbb{ACUX}} t_i'^j$ so that $(P_i^j, Q_i^j)(t_i^j)$ is derivable in $\mathcal{A}/\mathbb{ACU}$. Let $t_i = f_i(t_i^1, \ldots, t_i^{k_i})$. $\widehat{(P_i, Q_i)}(t_i)$ is derivable in $\mathcal{A}/\mathbb{ACU}$ using the clause $\widehat{(P_i, Q_i)}(f_i(x_1, \ldots, x_{x_i})) \Leftarrow (P_i^1, Q_i^1)(x_i^1) \wedge \ldots \wedge (P_i^{k_i}, Q_i^{k_i})(x_i^{k_i})$. Since the clause $R_i(a_{P_i,Q_i}) \in \mathcal{A}P, Q$ hence the clause $R_i(x) \Leftarrow \widehat{(P_i, Q_i)}(x) \in \mathcal{A}$. Hence $R_1(t_1), \ldots, R_m(t_m)$ are derivable in $\mathcal{A}_{inter}/\mathbb{ACU}$. Also $\mathcal{A}_{P,Q_{eq}} \subseteq \mathcal{A}_{inter}$. Hence from (8.10) and Lemma 21, $F_{P,Q,S,T}(t_1 + \ldots + t_m)$ is derivable in $\mathcal{A}_{inter}/\mathbb{ACU}$. Let the required $t$ be $t_1 + \ldots + t_m$. Also by induction hypothesis we must have $u_j =_{\mathbb{ACUX}} u_j'$,$v_k =_{\mathbb{ACUX}} v_k'$ such that $(I_j, I_j')(u_j)$ and $(J_k, J_k')(v_k)$ are derivable in $\mathcal{A}_{inter}/\mathbb{ACU}$ for $1 \leq j \leq n$ and $1 \leq k \leq p$. Then we have the following derivation of $(P, Q)(t)$ in $\mathcal{A}_{inter}/\mathbb{ACU}$

$$\frac{F_{P,Q,S,T}(t) \quad (I_1, I_1')(u_1) \ldots (I_n, I_n')(u_n) \quad (J_1, J_1')(v_1) \ldots (J_p, J_p')(v_p)}{(P, Q)(t)} \, (C/\mathbb{ACU})$$

where $C$ is the clause $(P, Q)(x) \Leftarrow F_{P,Q,S,T}(x) \wedge \bigwedge_{b_{R,R'} \in S \cup T} (R, R')(x_{R,R'})$. Also it is clear that $t =_{\mathbb{ACUX}} t'$. $\square$

**Lemma 32** *For $P, Q \in \mathbb{P}$, if $(P, Q)(t)$ is derivable in $\mathcal{A}_{inter}/\mathbb{ACU}$ then for some $t' =_{\mathbb{ACUX}} t'' =_{\mathbb{ACUX}} t$, $P(t')$ and $Q(t'')$ are derivable in $\mathcal{A}/\mathbb{ACU}$.*

**Proof:** We do induction on the size of the derivation $\pi$ of $(P, Q)(t)$ in $\mathcal{A}_{inter}/\mathbb{ACU}$. From examination of the clauses in $\mathcal{A}_{inter}$, $\pi$ must use a clause

$$C = (P, Q)(x) \Leftarrow F_{P,Q,S,T}(x) \wedge \bigwedge_{b_{R,R'} \in S \cup T} (R, R')(x_{R,R'})$$

as the last clause, for some $S, T \subseteq S_2$. Hence $F_{P,Q,S,T}(t)$ is derivable in $\mathcal{A}_{inter}/\mathbb{ACU}$ with a derivation $\pi_1$ which is strictly smaller than $\pi$. Also we have terms $t_{R,R'}$ for each $b_{R,R'}$ in $S \cup T$, such that $(R, R')(t_{R,R'})$ is derivable in $\mathcal{A}_{inter}/\mathbb{ACU}$ with a derivation which is strictly smaller than $\pi$. From induction hypothesis we get terms $t'_{R,R'} =_{\mathbb{ACUX}} t''_{R,R'} =_{\mathbb{ACUX}} t_{R,R'}$ such that :

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

Again from examination of the clauses in $\mathcal{A}_{inter}$, $\pi_1$ must have a functional support of the form $\widehat{(P_1, Q_1)}(t_1), \ldots, \widehat{(P_m, Q_m)}(t_m)$ for some $m \geq 0$ such that $t =_{\mathbb{ACU}} t_1 + \ldots + t_m$ and $\pi_1$ is of the form

$$\frac{\dfrac{\dfrac{\vdots}{\widehat{(P_1, Q_1)}(t_1)}}{R_1(t_1)} \, (R_1(x) \Leftarrow \widehat{(P_1, Q_1)}(x)) \quad \ldots \quad \dfrac{\dfrac{\vdots}{\widehat{(P_m, Q_m)}(t_m)}}{R_m(t_m)} \, (R_m(x) \Leftarrow \widehat{(P_m, Q_m)}(x))}{F_{P,Q,S,T}(t_1 + \ldots + t_m)} \, (\mathcal{A}_{P,Q,S,T_{eq}}/\mathbb{ACU}) \quad (8.11)$$

Hence the clause $R_i(a_{P_i,Q_i}) \in \mathcal{A}_{P,Q,S,T}$ for $1 \leq i \leq m$. Hence $R_i(a_{P_i,Q_i})$ is derivable in $\mathcal{A}_{P,Q,S,T}/\mathbb{ACU}$. Also $\mathcal{A}_{P,Q,S,T_{eq}} \subseteq \mathcal{A}_{inter}$. Hence using (8.11) and Lemma 21 $F_{P,Q,S,T}(a_{P_1,Q_1} + \ldots + a_{P_m,Q_m})$ is derivable in $\mathcal{A}_{P,Q,S,T}/\mathbb{ACU}$. So $a_{P_1,Q_1} + \ldots + a_{P_m,Q_m} \in \mathcal{L}'_{P,S}$. Therefore we must have $b_{I_1,I'_1}, \ldots, b_{I_n,I'_n} \in S (n \geq 0)$ such that $a_{P_1,Q_1} + \ldots + a_{P_m,Q_m} + 2b_{I_1,I'_1} + \ldots + 2b_{I_n,I'_n} \in \mathcal{L}_P(\mathcal{B}/\mathbb{ACU})$.

For $1 \leq i \leq m$, since $t_i$ is functional, hence there is a free $f_i$ of arity $k_i$, and terms $t_i^1, \ldots, t_i^{k_i}$ such that $t_i = f_i(t_i^1, \ldots, t_i^{k_i})$. Since $\widehat{(P_i, Q_i)}(t_i)$ is in the functional support of $\pi_1$, it uses as last clause a clause of the form $\widehat{(P_i, Q_i)}(f_i(x_1, \ldots, x_{k_i})) \Leftarrow (P_i^1, Q_i^1)(x_1) \wedge \ldots \wedge (P_i^{k_i}, Q_i^{k_i})(x_{k_i})$ corresponding to clauses $P_i(f_i(x_1, \ldots, x_{k_i})) \Leftarrow P_i^1(x_1) \wedge \ldots \wedge P_i^{k_i}(x_{k_i})$ and $Q_i(f_i(x_1, \ldots, x_{k_i})) \Leftarrow Q_i^1(x_1) \wedge \ldots \wedge Q_i^{k_i}(x_{k_i})$ of $\mathcal{A}_{free}$ so that $(P_i^j, Q_i^j)(t_i^j)$ is derivable in $\mathcal{A}_{inter}/\mathbb{ACU}$ using a derivation strictly smaller than $\pi_1$. By induction hypothesis, we have $t_i'^j =_{\mathbb{ACUX}} t_i''^j =_{\mathbb{ACUX}} t_i^j$ such that $P_i^j(t_i'^j)$ and $Q_i^j(t_i''^j)$ are derivable in $\mathcal{A}/\mathbb{ACU}$. Let $t'_i = f_i(t_i'^1, \ldots, t_i'^{k_i})$ and $t''_i = f_i(t_i''^1, \ldots, t_i''^{k_i})$. Then $P_i(t'_i)$ is derivable in $\mathcal{A}/\mathbb{ACU}$ using the clause $P_i(f_i(x_1, \ldots, x_{k_i})) \Leftarrow P_i^1(x_1) \wedge \ldots \wedge P_i^{k_i}(x_{k_i})$. Similarly $Q_i(t''_i)$ is derivable in $\mathcal{A}/\mathbb{ACU}$.

Now the derivation $\pi_2$ of $P(a_{P_1,Q_1} + \ldots + a_{P_m,Q_m} + 2b_{I_1,I'_1} + \ldots + 2b_{I_n,I'_n})$ in $\mathcal{B}/\mathbb{ACU}$ must have a functional support of the form $P_1^\dagger(a_{P_1,Q_1}), \ldots, P_m^\dagger(a_{P_m,Q_m}), I_1^\dagger(b_{I_1,I'_1}), I_1^\ddagger(b_{I_1,I'_1}), \ldots, I_n^\dagger(b_{I_n,I'_n}), I_n^\ddagger(b_{I_n,I'_n})$ where $P_i^\dagger \in \{P_i, Q_i\}$ and $I_i^\dagger, I_i^\ddagger \in \{I_i, I'_i\}$. We have

$$\pi_2 = \frac{\overline{P_1^\dagger(a_{P_1,Q_1})} \ \ldots \ \overline{P_m^\dagger(a_{P_m,Q_m})} \ \ \overline{I_1^\dagger(b_{I_1,I'_1})} \ \ \overline{I_1^\ddagger(b_{I_1,I'_1})} \ \ldots \ \overline{I_n^\dagger(b_{I_n,I'_n})} \ \ \overline{I_n^\ddagger(b_{I_n,I'_n})}}{P(a_{P_1,Q_1} + \ldots + a_{P_m,Q_m} + 2b_{I_1,I'_1} + \ldots + 2b_{I_n,I'_n})} \ (\mathcal{B}_{eq}/\mathbb{ACU}) \tag{8.12}$$

Since each $b_{I_i,I'_i} \in S$, by (*) $I_i(t'_{I_i,I'_i})$ and $I'_i(t'_{I_i,I'_i})$ are derivable in $\mathcal{A}/\mathbb{ACU}$. Recall that $P_i(t'_i)$ and $Q_i(t''_i)$ are derivable in $\mathcal{A}/\mathbb{ACU}$. Also $\mathcal{B}_{eq} = \mathcal{A}_{eq}$. So from (8.12) and Lemma 21 $P(t_1^\dagger + \ldots + t_m^\dagger + t^\dagger_{I_1,I'_1} + t^\ddagger_{I_1,I'_1} + \ldots + t^\dagger_{I_n,I'_n} + t^\ddagger_{I_n,I'_n})$ is derivable in $\mathcal{A}/\mathbb{ACU}$, 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 $t_1^\dagger + \ldots + t_m^\dagger + t^\dagger_{I_1,I'_1} + t^\ddagger_{I_1,I'_1} + \ldots + t^\dagger_{I_n,I'_n} + t^\ddagger_{I_n,I'_n})$. Then $t =_{\mathbb{ACUX}} t'$ and $P(t')$ is derivable in $\mathcal{A}/\mathbb{ACU}$. Similarly we can find $t''$ such that $t =_{\mathbb{ACUX}} t''$ and $Q(t'')$ is derivable in $\mathcal{A}/\mathbb{ACU}$. $\square$

**Theorem 21** *One-way $\mathbb{ACUX}$ automata are effectively closed under intersection.*

**Proof:** Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two one-way automata whose intersection we want to compute. We assume without loss of generality that they are built from disjoint sets of states $\mathbb{P}_1$ and $\mathbb{P}_2$ respectively, and that their final states are $P$ and $Q$ respectively. Define automaton $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$ on set of states $\mathbb{P} = \mathbb{P}_1 \cup \mathbb{P}_2$. We compute automaton $\mathcal{A}_{inter}$ corresponding to the automaton $\mathcal{A}$, as described in the above procedure. $\mathcal{A}_{inter}$ is not exactly a one-way automaton because of the presence of the extended epsilon clauses. However from Lemma 20, we have a one-way automaton $\mathcal{A}'_{inter}$ such that $\mathcal{L}_q(\mathcal{A}'_{inter}/\mathbb{ACU}) = \mathcal{L}_q(\mathcal{A}_{inter}/\mathbb{ACU})$ for each state $q$ of $\mathcal{A}_{inter}$.

1. First we show that $\mathcal{L}_P(\mathcal{A}_1/\mathbb{ACUX}) \cap \mathcal{L}_Q(\mathcal{A}_2/\mathbb{ACUX}) \subseteq \mathcal{L}_{(P,Q)}(\mathcal{A}'_{inter}/\mathbb{ACUX})$. Let $s \in \mathcal{L}_P(\mathcal{A}_1/\mathbb{ACUX}) \cap \mathcal{L}_Q(\mathcal{A}_2/\mathbb{ACUX})$. From Lemma 19 we have terms $t' =_{\mathbb{ACUX}} t'' =_{\mathbb{ACUX}} s$ such that $t' \in \mathcal{L}_P(\mathcal{A}_1/\mathbb{ACU}) = \mathcal{L}_P(\mathcal{A}/\mathbb{ACU})$ and $t'' \in \mathcal{L}_Q(\mathcal{A}_2/\mathbb{ACU}) = \mathcal{L}_Q(\mathcal{A}/\mathbb{ACU})$. From Lemma 31 there is a $t =_{\mathbb{ACUX}} t'$ such that $t \in \mathcal{L}_{(P,Q)}(\mathcal{A}_{inter}/\mathbb{ACU}) = \mathcal{L}_{(P,Q)}(\mathcal{A}'_{inter}/\mathbb{ACU}) \subseteq \mathcal{L}_{(P,Q)}(\mathcal{A}'_{inter}/\mathbb{ACUX})$. Since $s =_{\mathbb{ACUX}} t$ we have $s \in \mathcal{L}_{(P,Q)}(\mathcal{A}'_{inter}/\mathbb{ACUX})$.

2. Next we show that $\mathcal{L}_{(P,Q)}(\mathcal{A}'_{inter}/\mathbb{ACUX}) \subseteq \mathcal{L}_P(\mathcal{A}_1/\mathbb{ACUX}) \cap \mathcal{L}_Q(\mathcal{A}_2/\mathbb{ACUX})$. Let $s \in \mathcal{L}_{(P,Q)}(\mathcal{A}'_{inter}/\mathbb{ACUX})$. Since $\mathcal{A}'$ is a one-way automaton, by Lemma 19 there is a $t =_{\mathbb{ACUX}} s$ such that $t \in \mathcal{L}_{(P,Q)}(\mathcal{A}'_{inter}/\mathbb{ACU}) = \mathcal{L}_{(P,Q)}(\mathcal{A}_{inter}/\mathbb{ACUX})$. By Lemma 32 we have $t' =_{\mathbb{ACUX}} t'' =_{\mathbb{ACUX}} t$ such that $t' \in \mathcal{L}_P(\mathcal{A}/\mathbb{ACU}) = \mathcal{L}_P(\mathcal{A}_1/\mathbb{ACU}) \subseteq \mathcal{L}_P(\mathcal{A}_1/\mathbb{ACUX})$ and $t'' \in \mathcal{L}_Q(\mathcal{A}/\mathbb{ACU}) = \mathcal{L}_Q(\mathcal{A}_2/\mathbb{ACU}) \subseteq \mathcal{L}_Q(\mathcal{A}_2/\mathbb{ACUX})$. Since $t =_{\mathbb{ACUX}} t' =_{\mathbb{ACUX}} t''$ hence $t \in \mathcal{L}_P(\mathcal{A}_1/\mathbb{ACUX})$ and $t \in \mathcal{L}_Q(\mathcal{A}_2/\mathbb{ACUX})$. Hence $t \in \mathcal{L}_P(\mathcal{A}_1/\mathbb{ACUX}) \cap \mathcal{L}_Q(\mathcal{A}_2/\mathbb{ACUX})$.

Hence by naming $(P, Q)$ as the final state of $\mathcal{A}'_{inter}$ we have $\mathcal{L}(\mathcal{A}'/\mathbb{ACUX}) = \mathcal{L}(\mathcal{A}_1/\mathbb{ACUX}) \cap \mathcal{L}(\mathcal{A}_2/\mathbb{ACUX})$. □

### 8.2.1 One-Way $\mathbb{ACUX}_n$ Automata

The result on closure under intersection of one-way $\mathbb{ACUX}$ is easily generalized to the case of $\mathbb{ACUX}_n$ automata. Since the proofs in this case are very similar to the $\mathbb{ACUX}$ case, we merely detail the construction which ideas of proofs. Fix some $n \geq 2$, so that we are interested in the theory $\mathbb{ACUX}_n$. (The $\mathbb{ACUX}$ case corresponds to the case $n = 2$.) Whereas in the $\mathbb{ACUX}$ case, we needed to compute intersections of pairs of states, in this case, we will need to compute intersections of $n$-tuples of states.

Consider a one-way $\mathbb{ACUX}_n$ automaton $\mathcal{A}$ with predicates from some finite set $\mathbb{P}$. We introduce new predicate symbols $(P_1, \ldots, P_n)$ and $(\widehat{P_1, \ldots, P_n})$ for each $P_1, \ldots, P_n \in \mathbb{P}$, and sets of constants $S_1 = \{a_{P_1,\ldots,P_n} \mid P_1, \ldots, P_n \in \mathbb{P}\}$ and $S_2 = \{b_{P_1,\ldots,P_n} \mid P_1, \ldots, P_n \in \mathbb{P}\}$. The order of $P_1, \ldots, P_n$ in all these new predicates as well as constants is ignored. As in the $\mathbb{ACUX}$ case, instead of intersecting two distinct automata, we compute an automaton $\mathcal{A}_{inter}$ in which state $(P_1, \ldots, P_n)$ represents intersection of states $P_1, \ldots, P_n$ for all $P_1, \ldots, P_n$. $(\widehat{P_1, \ldots, P_n})$ accepts the functional terms among the terms accepted at $(P_1, \ldots, P_n)$.

Define automaton $\mathcal{B} = \mathcal{A}_{eq} \cup \{P_1(a_{P_1,\ldots,P_n}), P(b_{P_1,\ldots,P_n}) \mid P_1, \ldots, P_n \in \mathbb{P}\}$. From Theorem 16 $\mathcal{L}_P(\mathcal{B}/\mathbb{ACU})$ 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 $t \in \mathcal{L}_P(\mathcal{B}/\mathbb{ACU})$ such that each constant in $S$ occurs in $t$ a $kn$ number of times for some $k \geq 1$, 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 $S_2$. $\mathcal{L}'_{P,S}$ is again a semilinear set. Given $P_1, \ldots, P_n \in \mathbb{P}$ and $T_1, \ldots, T_n \subseteq S_2$, clearly $\mathcal{L}'_{P_1,T_1} \cap \ldots \cap \mathcal{L}'_{P_n,T_n}$ is a semilinear set. By Theorem 16, we can construct a constant-only automaton $\mathcal{A}_{P_1,\ldots,P_n,T_1,\ldots,T_n}$ with final state $F_{P_1,\ldots,P_n,T_1,\ldots,T_n}$ such that $\mathcal{L}(\mathcal{A}_{P_1,\ldots,P_n,T_1,\ldots,T_n}/\mathbb{ACU}) = \mathcal{L}'_{P_1,T_1} \cap \ldots \cap \mathcal{L}'_{P_n,T_n}$. We assume that automata $\mathcal{A}_{P_1,\ldots,P_n,T_1,\ldots,T_n}$'s are built from mutually disjoint sets of (fresh) states.

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

– for each $P_1, \ldots, P_n \in \mathbb{P}$ and each $T_1, \ldots, T_n \subseteq S_2$, the extended epsilon clause $(P_1, \ldots, P_n)(x) \Leftarrow F_{P_1,\ldots,P_n,T_1,\ldots,T_n}(x) \wedge \bigwedge_{b_{R_1,\ldots,R_n} \in T_1 \cup \ldots \cup T_n} (R_1, \ldots, R_n)(x_{R_1,\ldots,R_n})$.
– clauses of $\mathcal{A}_{P_1,\ldots,P_n,T_1,\ldots,T_n eq}$ for each $P_1, \ldots, P_n \in \mathbb{P}$ and $T_1, \ldots, T_n \subseteq S_2$.
– for each base clause $R(a_{R_1,\ldots,R_n})$ in some $\mathcal{A}_{P_1,\ldots,P_n,T_1,\ldots,T_n}$, the clause $R(x) \Leftarrow (\widehat{R_1, \ldots, R_n})(x)$.
– for each $n$-tuple of clauses

$$P^1(f(x_1, \ldots, x_m)) \Leftarrow P^1_1(x_1) \wedge \ldots \wedge P^1_m(x_m)$$
$$\ldots$$
$$P^n(f(x_1, \ldots, x_n)) \Leftarrow P^n_1(x_1) \wedge \ldots \wedge P^n_m(x_m)$$

in $\mathcal{A}_{free}$, the clause

$$(\widehat{P^1, \ldots, P^n})(f(x_1, \ldots, x_m)) \Leftarrow (P^1_1, \ldots, P^n_1)(x_1) \wedge \ldots \wedge (P^1_m, \ldots, P^n_m)(x_m)$$

Although our main purpose is to compute intersections of pairs of states, in order to take into account for the cancellations using the $\mathbb{X}_n$ equation, it is necessary to compute intersections of $n$-tuples of states. As in the $\mathbb{ACUX}$ case we prove the following two results :

**Lemma 33** *If $P_1(t_1), \ldots, P_n(t_n)$ are derivable in $\mathcal{A}/\mathbb{ACU}$ and $t_1 =_{\mathbb{ACUX}_n} \ldots =_{\mathbb{ACUX}_n} t_n$, then for some $t =_{\mathbb{ACUX}_n} t_1$, $(P_1, \ldots, P_n)(t)$ is derivable in $\mathcal{A}_{inter}/\mathbb{ACU}$.*

**Lemma 34** *For $P_1, \ldots, P_n \in \mathbb{P}$, if $(P_1, \ldots, P_n)(t)$ is derivable in $\mathcal{A}_{inter}/\mathbb{ACU}$ then for some $t_1 =_{\mathbb{ACUX}_n} \ldots =_{\mathbb{ACUX}_n} t_n =_{\mathbb{ACUX}_n} t$, $P_1(t_1), \ldots, P_n(t_n)$ are derivable in $\mathcal{A}/\mathbb{ACU}$.*

We don't need special states of the form $(P, Q)$ to represent intersection of pairs of states, since the states of the form $(P, Q, \ldots, Q)$ serve this purpose. We conclude that

**Theorem 22** *One-way $\mathbb{ACUX}_n$ automata are effectively closed under intersection.*

## 8.3 One-Way $\mathbb{ACUD}$ Automata

We show in this section that one-way $\mathbb{ACUD}$ automata are effectively closed under intersection. Fix a signature $\Sigma$ containing at least the symbols $+$ , $-$ and $0$. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be one-way equational tree automata on signature $\Sigma$. We will define an automaton $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}/\mathbb{ACUD}) = \mathcal{L}(\mathcal{A}_1/\mathbb{ACUD}) \cap \mathcal{L}(\mathcal{A}_2/\mathbb{ACUD})$. Let the set of states in $\mathcal{A}_1$ and $\mathcal{A}_2$ be $\mathbb{P}$ and $\mathbb{Q}$ respectively.

We introduce sets $S = \{a_{P,Q} \mid (P, Q) \in \mathbb{P} \times \mathbb{Q}\}$ and $\overline{S} = \{\overline{a_{P,Q}} \mid (P, Q) \in \mathbb{P} \times \mathbb{Q}\}$ of new constants. We don't distinguish between $a_{P,Q}$ and $a_{Q,P}$. Define automaton $\mathcal{B}_1 = \mathcal{A}_{1eq} \cup \{P(a_{P,Q}) \mid (P, Q) \in \mathbb{P} \times \mathbb{Q}\}$. Define automaton $\mathcal{B}_2 = \mathcal{A}_{2eq} \cup \{Q(a_{P,Q}) \mid (P, Q) \in \mathbb{P} \times \mathbb{Q}\}$. From Theorem 18, for $(P, Q) \in \mathbb{P} \times \mathbb{Q}$, $\mathcal{L}_P(\mathcal{B}_1/\mathbb{ACUD}), \mathcal{L}_Q(\mathcal{B}_2/\mathbb{ACUD})$ are semilinear sets on constants from $S \cup \overline{S}$. Hence $\mathcal{L}_P(\mathcal{B}_1/\mathbb{ACUD}) \cap \mathcal{L}_Q(\mathcal{B}_2/\mathbb{ACUD})$ is a semilinear set. Hence we can define constant-only $\mathbb{ACUD}$-automaton $\mathcal{A}_{P,Q}$ (on constants from $S$) with a final state $F_{P,Q}$ such that $\mathcal{L}_{F_{P,Q}}(\mathcal{A}_{P,Q}/\mathbb{ACUD}) = \mathcal{L}_P(\mathcal{B}_1/\mathbb{ACUD}) \cap \mathcal{L}_Q(\mathcal{B}_2/\mathbb{ACUD})$. We assume that the automata $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_{P,Q}$'s are all built from mutually disjoint sets of states.

We introduce new predicate symbols $(P, Q)$ and $\widehat{(P, Q)}$ for $(P, Q) \in \mathbb{P} \times \mathbb{Q}$. The required automaton $\mathcal{A}$ contains

1. clause $(P, Q)(x) \Leftarrow F_{P,Q}(x)$ for $(P, Q) \in \mathbb{P} \times \mathbb{Q}$.
2. clauses from $\mathcal{A}_{P,Q_{eq}}$ for $(P, Q) \in \mathbb{P} \times \mathbb{Q}$.
3. clause $R(x) \Leftarrow (P', Q')(x)$ for each base clause $R(a_{P',Q'})$ in $\mathcal{A}_{P,Q}$ for $(P, Q) \in \mathbb{P} \times \mathbb{Q}$.
4. clause

$$\widehat{(P, Q)}(f(x_1, \ldots, x_n)) \Leftarrow (P_1, Q_1)(x_1) \wedge \ldots \wedge (P_n, Q_n)(x_n)$$

for clauses

$$P(f(x_1, \ldots, x_n)) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n) \in \mathcal{A}_{1free}$$
$$Q(f(x_1, \ldots, x_n)) \Leftarrow Q_1(x_1) \wedge \ldots \wedge Q_n(x_n) \in \mathcal{A}_{2free}$$

Clearly $\mathcal{A}$ is a one-way $\mathbb{ACUD}$-automaton. That it represents the intersection of $\mathcal{A}_1$ and $\mathcal{A}_2$ is stated by Lemmas 35 and 36 respectively.

**Lemma 35** *If $P(t)$ and $Q(t)$ are derivable in $\mathcal{A}_1/\mathbb{ACUD}$ and $\mathcal{A}_2/\mathbb{ACUD}$ respectively, then $(P,Q)(t)$ is derivable in $\mathcal{A}/\mathbb{ACUD}$.*

**Proof:** We do induction of the sum of the sizes of the derivations $\pi_1$ and $\pi_2$ of $P(t)$ and $Q(t)$ respectively. We have functional terms $s_1,\ldots,s_m,t_1,\ldots,t_n$ for some $m,n \geq 0$ such that $t =_{\mathbb{ACUD}} s_1+\ldots+s_m-t_1-\ldots-t_n$. $\pi_1$ must have functional support of the form $P_1(s_1),\ldots,P_m(s_m),P'_1(t_1),\ldots,P'_n(t_n)$ and $\pi_2$ must have a functional support of the form $Q_1(s_1),\ldots,Q_m(s_m),Q'_1(t_1),\ldots,Q'_n(t_n)$ and we have

$$\pi_1 = \cfrac{\begin{matrix}\vdots & & \vdots & \vdots & & \vdots\\ P_1(s_1) & \ldots & P_n(s_m) & P'_1(t_1) & \ldots & P'_n(t_n)\end{matrix}}{P(s_1+\ldots+s_m-t_1-\ldots-t_n)}\;(\mathcal{A}_{1eq}/\mathbb{ACUD}) \tag{8.13}$$

$$\pi_2 = \cfrac{\begin{matrix}\vdots & & \vdots & \vdots & & \vdots\\ Q_1(s_1) & \ldots & Q_n(s_m) & Q'_1(t_1) & \ldots & Q'_n(t_n)\end{matrix}}{Q(s_1+\ldots+s_m-t_1-\ldots-t_n)}\;(\mathcal{A}_{2eq}/\mathbb{ACUD}) \tag{8.14}$$

By definition of $\mathcal{B}_1$, the atoms $P_1(a_{P_1,Q_1}),\ldots,P_n(a_{P_m,Q_m}),P'_1(a_{P'_1,Q'_1}),\ldots,P'_n(a_{P'_n,Q'_n})$ are derivable in $\mathcal{B}_1/\mathbb{ACUD}$. Since $\mathcal{A}_{1eq} \subseteq \mathcal{B}_1$, hence from (8.13) and using Lemma 23, $P(a_{P_1,Q_1}+\ldots+a_{P_m,Q_m}-a_{P'_1,Q'_1}-\ldots-a_{P'_n,Q'_n})$ is derivable in $\mathcal{B}_1/\mathbb{ACUD}$. Similarly $Q(a_{P_1,Q_1}+\ldots+a_{P_m,Q_m}-a_{P'_1,Q'_1}-\ldots-a_{P'_n,Q'_n})$ is derivable in $\mathcal{B}_2/\mathbb{ACUD}$ respectively. Hence $F_{P,Q}(a_{P_1,Q_1}+\ldots+a_{P_m,Q_m}-a_{P'_1,Q'_1}-\ldots-a_{P'_n,Q'_n})$ is derivable in $\mathcal{A}_{P,Q}/\mathbb{ACUD}$. This derivation $\pi_3$ must have a functional support of the form $R_1(a_{P_1,Q_1}),\ldots,R_m(a_{P_m,Q_m}),R'_1(a_{P'_1,Q'_1},\ldots,R'_n(a_{P'_n,Q'_n})$ and we must have

$$\pi_3 = \cfrac{\overline{R_1(a_{P_1,Q_1})}\;\ldots\;\overline{R_n(a_{P_m,Q_m})}\;\;\overline{R'_1(a_{P'_1,Q'_1})}\;\ldots\;\overline{R'_n(a_{P'_n,Q'_n})}}{F_{P,Q}(a_{P_1,Q_1}+\ldots+a_{P_m,Q_m}-a_{P'_1,Q'_1}-\ldots-a_{P'_n,Q'_n})}\;(\mathcal{A}_{P,Q_{eq}}/\mathbb{ACU}) \tag{8.15}$$

For each $1 \leq i \leq m$, since $s_i$ is functional we must have $s_i = f_i(s_i^1,\ldots,s_i^{k_i})$ for some free $f_i$ of arity $k_i$. As $P_i(s_i)$ is in the functional support of $\pi_1$, we must have a clause $P_i(f_i(x_1,\ldots,x_{k_i}) \Leftarrow P_i^1(x_1)\wedge\ldots\wedge P_i^{k_i}(x_{k_i})$ in $\mathcal{A}_1$ such that for $1 \leq j \leq k_i$, $P_i^j(s_i^j)$ is derivable in $\mathcal{A}_1/\mathbb{ACUD}$ with derivation strictly smaller than $\pi_1$. Similarly we have a clause $Q_i(f_i(x_1,\ldots,x_{k_i}) \Leftarrow Q_i^1(x_1)\wedge\ldots\wedge Q_i^{k_i}(x_{k_i})$ in $\mathcal{A}_2$ such that for $1 \leq j \leq k_i$, $Q_i^j(s_i^j)$ is derivable in $\mathcal{A}_2/\mathbb{ACUD}$ with derivations strictly smaller than $\pi_2$. By induction hypothesis we have derivation of $(P_i^j,Q_i^j)(s_i^j)$ in $\mathcal{A}/\mathbb{ACUD}$. Also the clause $(\widehat{P_i,Q_i})(f_i(x_1,\ldots,x_{k_i})) \Leftarrow (P_i^1,Q_i^1)(x_1)\wedge\ldots\wedge(P^{k_i},Q^{k_i})(x_{k_i}) \in \mathcal{A}$. Hence the atom $(\widehat{P_i,Q_i})(s_i)$ is derivable in $\mathcal{A}/\mathbb{ACUD}$. Since the clause $R_i(a_{P_i,Q_i})$ is in $\mathcal{A}_{P,Q}$, hence the clause $R_i(x) \Leftarrow (P_i,Q_i)(x)$ is in $\mathcal{A}$. Hence $R_i(s_i)$ is derivable in $\mathcal{A}/\mathbb{ACUD}$ for $1 \leq i \leq m$. Similarly $R'_i(t_i)$ is derivable in $\mathcal{A}/\mathbb{ACUD}$ for $1 \leq i \leq n$. Also $\mathcal{A}_{P,Q_{eq}} \subseteq \mathcal{A}$. Hence using (8.15) and Lemma 23, we get a derivation of $F_{P,Q}(s_1+\ldots+s_m-t_1-\ldots-t_n)$ in $\mathcal{A}/\mathbb{ACUD}$. Finally we use the clause $(P,Q)(x) \Leftarrow F_{P,Q}(x)$ to get a derivation of $(P,Q)(t)$ in $\mathcal{A}/\mathbb{ACUD}$. $\square$

**Lemma 36** *For $(P,Q) \in \mathbb{P}\times\mathbb{Q}$, if $(P,Q)(t)$ is derivable in $\mathcal{A}/\mathbb{ACUD}$ then $P(t)$ and $Q(t)$ are derivable in $\mathcal{A}_1/\mathbb{ACUD}$ and $\mathcal{A}_2/\mathbb{ACUD}$ respectively.*

**Proof:** We do induction on the size of the derivation $\pi$ of $(P,Q)(t)$. Since $(P,Q)(x) \Leftarrow F_{P,Q}(x)$ is the only clause which has the predicate $(P,Q)$ in the head, $\pi$ uses it as the last clause, and $F_{P,Q}(t)$ is derivable in $\mathcal{A}/\mathbb{ACU}$. Again by examining the clauses in $\mathcal{A}$, we conclude that the derivation $\pi_1$ of $F_{P,Q}(t)$ must have functional support of the form $\widehat{(P_1,Q_1)}(s_1),\ldots,\widehat{(P_n,Q_n)}(s_m),\widehat{(P'_1,Q'_1)}(t_1),\ldots,$ $\widehat{(P'_n,Q'_n)}(t_n)$ such that $t =_{\mathbb{ACUD}} s_1+\ldots+s_n-t_1-\ldots-t_n$ and $\pi_1$ is of the form

$$\dfrac{\dfrac{\vdots}{\dfrac{\widehat{(P_1,Q_1)}(s_1)}{R_1(s_1)}}(C_1)\quad\ldots\quad\dfrac{\vdots}{\dfrac{\widehat{(P_m,Q_m)}(s_m)}{R_m(s_m)}}(C_m)\quad\dfrac{\vdots}{\dfrac{\widehat{(P'_1,Q'_1)}(t_1)}{R'_1(t_1)}}(C'_1)\quad\ldots\quad\dfrac{\vdots}{\dfrac{\widehat{(P'_n,Q'_n)}(t_n)}{R'_n(t_n)}}(C'_n)}{F_{P,Q}(s_1+\ldots+s_m-t_1-\ldots-t_n)}(\mathcal{A}_{P,Q_{eq}}/\mathbb{ACUD}) \tag{8.16}$$

where $C_i = R_i(x) \Leftarrow \widehat{(P_i,Q_i)}(x))$ for $1 \le i \le m$ and $C'_i = R'_i(x) \Leftarrow \widehat{(P'_i,Q'_i)}(x))$ for $1 \le i \le n$.

Hence the clause $R_i(a_{P_i,Q_i}) \in \mathcal{A}_{P,Q}$ for $1 \le i \le m$. Hence $R_i(a_{P_i,Q_i})$ is derivable in $\mathcal{A}_{P,Q}/\mathbb{ACUD}$ for $1 \le i \le m$. Similarly $R'_i(a_{P'_i,Q'_i})$ is derivable in $\mathcal{A}_{P,Q}/\mathbb{ACUD}$ for $1 \le i \le n$. Also $\mathcal{A}_{P,Q_{eq}} \subseteq \mathcal{A}$. Hence using (8.16) and Lemma 23, $F_{P,Q}(a_{P_1,Q_1}+\ldots+a_{P_n,Q_n}-a_{P'_1,Q'_1}-\ldots-a_{P'_n,Q'_n})$ is derivable in $\mathcal{A}_{P,Q}/\mathbb{ACUD}$.

Now for $1 \le i \le m$, since $s_i$ is functional, $s_i = f_i(s_i^1,\ldots,t_i^{k_i})$ for some free $f_i$ of arity $k_i$ and some terms $s_i^1,\ldots,s_i^{k_i}$. As $\widehat{P_i,Q_i}(s_i)$ is in the functional support of $\pi_1$, there is a clause $\widehat{P_i,Q_i}(f_i(x_1,\ldots,x_{k_i})) \Leftarrow (P_i^1,Q_i^1)(x_1) \wedge \ldots \wedge (P_i^{k_i},Q_i^{k_i})(x_{k_i})$ in $\mathcal{A}$ corresponding to clauses $P_i(f_i(x_1,\ldots,x_{k_i})) \Leftarrow P_i^1(x_1)\wedge\ldots\wedge P_i^{k_i}(x_{k_i})$ and $Q_i(f_i(x_1,\ldots,x_{k_i})) \Leftarrow Q_i^1(x_1)\wedge\ldots\wedge Q_i^{k_i}(x_{k_i})$ in $\mathcal{A}_{1free}$ and $\mathcal{A}_{2free}$ respectively, and for $1 \le j \le k_i$, $(P_i^j,Q_i^j)(s_i^j)$ is derivable in $\mathcal{A}/\mathbb{ACUD}$. By induction hypothesis we get derivations of $P_i^j(s_i^j)$ and $Q_i^j(s_i^j)$ in $\mathcal{A}_1/\mathbb{ACUD}$ and $\mathcal{A}_2/\mathbb{ACUD}$ respectively. Hence using the above clauses from $\mathcal{A}_{1free}$ and $\mathcal{A}_{2free}$ respectively, $P_i(s_i)$ and $Q_i(s_i)$ are derivable in $\mathcal{A}_1/\mathbb{ACUD}$ and $\mathcal{A}_2/\mathbb{ACUD}$ respectively for $1 \le i \le m$. Similarly $P'_i(t_i)$ and $Q'_i(t_i)$ are derivable in $\mathcal{A}_1/\mathbb{ACUD}$ and $\mathcal{A}_2/\mathbb{ACUD}$ respectively for $1 \le i \le n$.

Since $F_{P,Q}(a_{P_1,Q_1}+\ldots+a_{P_m,Q_m}-a_{P'_1,Q'_1}-\ldots-a_{P'_n,Q'_n})$ is derivable in $\mathcal{A}_{P,Q}/\mathbb{ACUD}$ so $P(a_{P_1,Q_1}+\ldots+a_{P_m,Q_m}-a_{P'_1,Q'_1}-\ldots-a_{P'_n,Q'_n})$ is derivable in $\mathcal{B}_1/\mathbb{ACUD}$. This derivation (call it $\pi_2$) has a functional support $P_1(a_{P_1,Q_1}),\ldots,P_m(a_{P_m,Q_m}),P'_1(a_{P'_1,Q'_1}),\ldots,P'_n(a_{P'_n,Q'_n})$ and we have

$$\pi_2 = \dfrac{\overline{P_1(a_{P_1,Q_1})}\quad\ldots\quad\overline{P_m(a_{P_m,Q_m})}\quad\overline{P'_1(a_{P'_1,Q'_1})}\quad\ldots\quad\overline{P'_n(a_{P'_n,Q'_n})}}{P(a_{P_1,Q_1}+\ldots+a_{P_m,Q_m}-a_{P'_1,Q'_1}-\ldots-a_{P'_n,Q'_n})}(\mathcal{B}_{1eq}/\mathbb{ACUD}) \tag{8.17}$$

We know that $P_1(s_1),\ldots,P_n(s_m),P'_1(t_1),\ldots,P'_n(t_n)$ are derivable in $\mathcal{A}_1/\mathbb{ACUD}$. Also by definition $\mathcal{B}_{1eq} = \mathcal{A}_{1eq}$. Hence from (8.17) and Lemma 23 $P(s_1+\ldots+s_m-t_1-\ldots-t_n)$ is derivable in $\mathcal{A}_1/\mathbb{ACUD}$. Similarly $Q(s_1+\ldots+s_m-t_1-\ldots-t_n)$ is derivable in $\mathcal{A}_2/\mathbb{ACU}$. $\square$

Let $P$ and $Q$ be the final states of $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively. We name $(P,Q)$ as the final state of $\mathcal{A}$. From Lemmas 35 and 36, $\mathcal{L}_{(P,Q)}(\mathcal{A}/\mathbb{ACUD}) = \mathcal{L}_P(\mathcal{A}_1/\mathbb{ACUD}) \cap \mathcal{L}(\mathcal{A}_2/\mathbb{ACUD})$ implying that $\mathcal{L}(\mathcal{A}/\mathbb{ACUD}) = \mathcal{L}(\mathcal{A}_1/\mathbb{ACUD}) \cap \mathcal{L}(\mathcal{A}_2/\mathbb{ACUD})$. This allows us to conclude that

**Theorem 23** *One-way $\mathbb{ACUD}$-tree automata are effectively closed under intersection.*

## 8.4 Cancellative '$-$' Symbol : Abelian Groups Automata

We show in this section that the Abelian groups automata are effectively closed under intersection. Fix a signature $\Sigma$ containing at least the symbols $+$, $-$ and $0$.

Let $\mathcal{A}$ be a one-way $\mathbb{ACUM}$ automaton with predicates in $\mathbb{P}$. As in $\mathbb{ACUX}$ case we use new predicate symbols $(P, Q)$ and $\widehat{(P, Q)}$ for each $P, Q \in \mathbb{P}$. We will construct automaton $\mathcal{A}_{inter}$ in which state $(P, Q)$ accepts intersection of the languages accepted at $P$ 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 $\overline{S_2} = \{\overline{b_{P,Q}} \mid P, Q \in \mathbb{P}\}$.

Define automaton $\mathcal{B} = \mathcal{A}_{eq} \cup \{P(a_{P,Q}), P(b_{P,Q}) \mid P, Q \in \mathbb{P}\}$. From Theorem 18, for every $P \in \mathbb{P}$, $\mathcal{L}_P(\mathcal{B}/\mathbb{ACUD})$ 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 $t \in \mathcal{L}_P(\mathcal{B}/\mathbb{ACUD})$ 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 $t$.

Then $\mathcal{L}_{P,S}$ is a semilinear set on constants from $S_1 \cup \overline{S_1} \cup S_2 \cup \overline{S_2}$. 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 on constants from $S_1 \cup \overline{S_1}$. 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 on constants from $S_1 \cup \overline{S_1}$. Using the second part of Theorem 18, we construct a constant-only $\mathbb{ACUD}$ automaton $\mathcal{A}_{P,Q,S,T}$ on signature $S_1 \cup \{+, -, 0\}$, with a final state $F_{P,Q,S,T}$ such that $\mathcal{L}(\mathcal{A}_{P,Q,S,T}/\mathbb{ACUD}) = \mathcal{L}'_{P,S} \cap \mathcal{L}'_{Q,T}$. We assume that automata $\mathcal{A}_{P,Q,S,T}$'s are built from mutually disjoint sets of (fresh) states.

The required automaton $\mathcal{A}_{inter}$ 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 of $\mathcal{A}_{P,Q,S,T_{eq}}$ for each $P, Q \in \mathbb{P}$ and each $S, T \subseteq S_2$.
- for each base clause $R(a_{R',R''})$ in some $\mathcal{A}_{P,Q,S,T}$, the clause $R(x) \Leftarrow \widehat{(R', R'')}(x)$.
- for each pair of clauses

$$P(f(x_1, \ldots, x_n)) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n)$$
$$Q(f(x_1, \ldots, x_n)) \Leftarrow Q_1(x_1) \wedge \ldots \wedge Q_n(x_n)$$

in $\mathcal{A}_{free}$ the clause

$$\widehat{(P, Q)}(f(x_1, \ldots, x_n)) \Leftarrow (P_1, Q_1)(x_1) \wedge \ldots \wedge (P_n, Q_n)(x_n)$$

**Lemma 37** *If $P(t')$ and $Q(t'')$ are derivable in $\mathcal{A}/\mathbb{ACUD}$ and $t' =_{\mathbb{ACUM}} t''$, then for some $t =_{\mathbb{ACUM}} t'$, $(P, Q)(t)$ is derivable in $\mathcal{A}_{inter}/\mathbb{ACUD}$.*

**Proof:** We do induction on the sum of the sizes of the derivations $\pi_1$ and $\pi_2$ of $P(t')$ and $Q(t'')$ respectively. Since $t' =_{\mathbb{ACUM}} t''$ we must have

$$t' =_{\mathbb{ACU}} s'_1 + \ldots + s'_m - t'_1 - \ldots - t'_n + (u'_1 - u''_1) + \ldots + (u'_p - u''_p) \tag{8.18}$$
$$t'' =_{\mathbb{ACU}} s''_1 + \ldots + s''_m - t''_1 - \ldots - t''_n + (v'_1 - v''_1) + \ldots + (v'_q - v''_q) \tag{8.19}$$

such that $m, n, p, q \geq 0$, $s'_i, s''_i$ are functional and $s'_i =_{\mathbb{ACUM}} s''_i$ for $i \leq i \leq m$, $t'_i, t''_i$ are functional and $t'_i =_{\mathbb{ACUM}} t''_i$ for $i \leq i \leq n$, $u'_i, u''_i$ are functional and $u'_i =_{\mathbb{ACUM}} u''_i$ for $1 \leq i \leq p$, and $v_i', v_i''$ are functional and $v_i' =_{\mathbb{ACUM}} v_i''$ for $1 \leq i \leq q$. $\pi_1$ and $\pi_2$ have functional supports of the form $P_1(s_1'), \ldots, P_m(s_m'), P_1'(t_1'), \ldots, P_n'(t_n'), I_1(u_1'), I_1'(u_1''), \ldots, I_p(u_p'), I_p'(u_p'')$ and $Q_1(s_1''), \ldots, Q_m(s_m''), Q_1'(t_1''), \ldots, Q_n'(t_n''), J_1(v_1'), J_1'(v_1''), \ldots, J_q(v_q'), J_q'(v_q'')$ respectively such that $\pi_1$ and $\pi_2$ are of the form :

$$\frac{\begin{matrix}\vdots & & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots\\ P_1(s_1') & \ldots & P_m(s_m') & P_1'(t_1') & \ldots & P_n'(t_n') & I_1(u_1') & I_1'(u_1'') & \ldots & I_p(u_p') & I_p'(u_p'')\end{matrix}}{P(s_1'+\ldots+s_m'-t_1'-\ldots-t_n'+(u_1'-u_1'')+\ldots+(u_p'-u_p''))}\,(\mathcal{A}_{eq}/\mathbb{ACUD}) \tag{8.20}$$

$$\frac{\begin{matrix}\vdots & & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots\\ Q_1(s_1'') & \ldots & Q_m(s_m'') & Q_1'(t_1'') & \ldots & Q_m'(t_n'') & J_1(v_1') & J_1'(v_1'') & \ldots & J_q(v_q') & J_q'(v_q'')\end{matrix}}{P(s_1''+\ldots+s_m''-t_1''-\ldots-t_n''+(v_1'-v_1'')+\ldots+(v_q'-v_q''))}\,(\mathcal{A}_{eq}/\mathbb{ACUD}) \tag{8.21}$$

By definition of $\mathcal{B}$, the atoms $P_1(a_{P_1,Q_1}), \ldots, P_m(a_{P_m,Q_m}), P_1'(a_{P_1',Q_1'}), \ldots, P_n'(a_{P_n',Q_n'}), I_1(b_{I_1,I_1'}), I_1'(b_{I_1,I_1'}), \ldots, I_p(b_{I_p,I_p'}), I_p'(b_{I_p,I_p'})$ are derivable in $\mathcal{B}/\mathbb{ACUD}$. Also $\mathcal{A}_{eq} \subseteq \mathcal{B}$. Hence from (8.20) and Lemma 23, $P(a_{P_1,Q_1}+\ldots+a_{P_m,Q_m}-a_{P_1',Q_1'}-\ldots-a_{P_n',Q_n'}+(b_{I_1,I_1'}-b_{I_1,I_1'})+\ldots+(b_{I_p,I_p'}-b_{I_p,I_p'}))$ is derivable in $\mathcal{B}/\mathbb{ACUD}$. Similarly $Q(a_{P_1,Q_1}+\ldots+a_{P_m,Q_m}-a_{P_1',Q_1'}-\ldots-a_{P_n',Q_n'}+(b_{J_1,J_1'}-b_{J_1,J_1'})+\ldots+(b_{J_q,J_q'}-b_{J_q,J_q'}))$ is derivable in $\mathcal{B}/\mathbb{ACUD}$. Let $S=\{b_{I_1,I_1'},\ldots,b_{I_p,I_p'}\}$ and $T=\{b_{J_1,J_1'},\ldots,b_{J_q,J_q'}\}$. We have $a_{P_1,Q_1}+\ldots+a_{P_m,Q_m}-a_{P_1',Q_1'}-\ldots-a_{P_n',Q_n'}+(b_{I_1,I_1'}-b_{I_1,I_1'})+\ldots+(b_{I_p,I_p'}-b_{I_p,I_p'}) \in \mathcal{L}_{P,S}$ and $a_{P_1,Q_1}+\ldots+a_{P_m,Q_m}-a_{P_1',Q_1'}-\ldots-a_{P_n',Q_n'} \in \mathcal{L}'_{P,S}$. Similarly $a_{P_1,Q_1}+\ldots+a_{P_m,Q_m}-a_{P_1',Q_1'}-\ldots-a_{P_n',Q_n'} \in \mathcal{L}'_{Q,T}$. Hence $F_{P,Q,S,T}(a_{P_1,Q_1}+\ldots+a_{P_m,Q_m}-a_{P_1',Q_1'}-\ldots-a_{P_n',Q_n'})$ is derivable in $\mathcal{A}_{P,Q,S,T}/\mathbb{ACUD}$. This derivation $\pi_3$ must have functional support of the form $R_1(a_{P_1,Q_1}),\ldots,R_m(a_{P_m,Q_m}),R_1'(a_{P_1',Q_1'}),\ldots,R_n'(a_{P_n',Q_n'})$ and we must have

$$\pi_3 = \frac{\overline{R_1(a_{P_1,Q_1})} \;\ldots\; \overline{R_m(a_{P_m,Q_m})} \quad \overline{R_1'(a_{P_1',Q_1'})} \;\ldots\; \overline{R_n'(a_{P_n',Q_n'})}}{F_{P,Q,S,T}(a_{P_1,Q_1}+\ldots+a_{P_m,Q_m}-a_{P_1',Q_1'}-\ldots-a_{P_n',Q_n'})}\,(\mathcal{A}_{P,Q_{eq}}/\mathbb{ACUD}) \tag{8.22}$$

Since $P_i(s_i')$ and $Q_i(s_i'')$ are in the functional supports of $\pi_1$ and $\pi_2$ respectively, we have $s_i' = f_i(s_i'^1,\ldots,s_i'^{k_i})$ and $s_i'' = f_i(s_i''^1,\ldots,s_i''^{k_i})$ for some free $f_i$ of arity $k_i$ such that $s_i'^j =_{\mathbb{ACUM}} s_i''^j$, the derivation of $P_i(s_i')$ uses $P_i(f_i(x_1,\ldots,x_{k_i})) \Leftarrow P_i^1(x_1)\wedge\ldots\wedge P_i^{k_i}(x_{k_i})$ as the last clause, the derivation of $Q_i(s_i'')$ uses $Q_i(f_i(x_1,\ldots,x_{k_i})) \Leftarrow Q_i^1(x_1)\wedge\ldots\wedge Q_i^{k_i}(x_{k_i})$ as the last clause, and for $1 \leq j \leq k_i$, the atoms $P_i^j(s_i'^j)$ and $Q_i^j(s_i''^j)$ are derivable in $\mathcal{A}/\mathbb{ACUD}$. By induction hypothesis, we have $s_i^j =_{\mathbb{ACUM}} s_i'^j$ so that $(P_j^i, Q_j^i)(s_i^j)$ is derivable in $\mathcal{A}/\mathbb{ACUD}$. Let $s_i = f_i(s_i^1,\ldots,s_i^{k_i})$. $\widehat{(P_i,Q_i)}(s_i)$ is derivable in $\mathcal{A}/\mathbb{ACUD}$ using the clause $\widehat{(P_i,Q_i)}(f_i(x_1,\ldots,x_{x_i})) \Leftarrow (P_i^1,Q_i^1)(x_i^1)\wedge\ldots\wedge(P_i^{k_i},Q_i^{k_i})(x_i^{k_i})$. Since the clause $R_i(a_{P_i,Q_i}) \in \mathcal{A}P,Q$ hence the clause $R_i(x) \Leftarrow \widehat{(P_i,Q_i)}(x) \in \mathcal{A}$. Hence for $1 \leq i \leq m$, $R_i(s_i)$ is derivable in $\mathcal{A}_{inter}/\mathbb{ACUD}$ and $s_i =_{\mathbb{ACUM}} s_i'$. Similarly for $1 \leq i \leq n$ we have term $t_i$ such that $R_i'(t_i')$ is derivable in $\mathcal{A}_{inter}/\mathbb{ACUD}$ and $t_i =_{\mathbb{ACUM}} t_i'$. Also $\mathcal{A}_{P,Q_{eq}} \subseteq \mathcal{A}_{inter}$. Hence from (8.22) and Lemma 23, $F_{P,Q,S,T}(s_1+\ldots+s_m-t_1-\ldots-t_n)$ is derivable in $\mathcal{A}_{inter}/\mathbb{ACUD}$. Let the required $t$ be $s_1+\ldots+s_m-t_1-\ldots-t_n$. Also by induction hypothesis we must have $u_j =_{\mathbb{ACUM}} u_j'$, $v_k =_{\mathbb{ACUM}} v_k'$ such that $(I_j,I_j')(u_j)$ and $(J_k,J_k')(v_k)$ are derivable in $\mathcal{A}_{inter}/\mathbb{ACUD}$ for $1 \leq j \leq p$ and $1 \leq k \leq q$. Then we have the following derivation of $(P,Q)(t)$ in $\mathcal{A}_{inter}/\mathbb{ACU}$

$$\frac{F_{P,Q,S,T}(t) \quad (I_1, I_1')(u_1)\dots(I_p, I_n')(u_p) \quad (J_1, J_1')(v_1)\dots(J_q, J_q')(v_q)}{(P,Q)(t)} \ (C/\mathbb{ACUD})$$

where $C$ is the clause $(P,Q)(x) \Leftarrow F_{P,Q,S,T}(x) \wedge \bigwedge_{b_{R,R'} \in S \cup T}(R,R')(x_{R,R'})$. Also it is clear that $t =_{\mathbb{ACUM}} t'$. □

**Lemma 38** *For $P, Q \in \mathbb{P}$, if $(P,Q)(t)$ is derivable in $\mathcal{A}_{inter}/\mathbb{ACUD}$ then for some $t' =_{\mathbb{ACUM}} t'' =_{\mathbb{ACUM}} t$, $P(t')$ and $Q(t'')$ are derivable in $\mathcal{A}/\mathbb{ACUD}$.*

**Proof:** We do induction on the size of the derivation $\pi$ of $(P,Q)(t)$ in $\mathcal{A}_{inter}/\mathbb{ACUD}$. From examination of the clauses in $\mathcal{A}_{inter}$, $\pi$ must use a clause

$$C = (P,Q)(x) \Leftarrow F_{P,Q,S,T}(x) \wedge \bigwedge_{b_{R,R'} \in S \cup T} (R,R')(x_{R,R'})$$

as the last clause, for some $S, T \subseteq S_2$. Hence $F_{P,Q,S,T}(t)$ is derivable in $\mathcal{A}_{inter}/\mathbb{ACUD}$ with a derivation $\pi_1$ which is strictly smaller than $\pi$. Also we have terms $t_{R,R'}$ for each $b_{R,R'}$ in $S \cup T$, such that $(R,R')(t_{R,R'})$ is derivable in $\mathcal{A}_{inter}/\mathbb{ACUD}$ with a derivation which is strictly smaller than $\pi$. From induction hypothesis we get terms $t'_{R,R'} =_{\mathbb{ACUM}} t''_{R,R'} =_{\mathbb{ACUM}} t_{R,R'}$ such that :

$R(t'_{R,R'})$ and $R'(t''_{R,R'})$ are derivable in $\mathcal{A}/\mathbb{ACUD}$, for each $b_{R,R'} \in S \cup T$. (*)

Again from examination of the clauses in $\mathcal{A}_{inter}$, $\pi_1$ must have a functional support of the form $\widehat{(P_1,Q_1)}(s_1), \dots, \widehat{(P_m,Q_m)}(s_m), \widehat{(P_1',Q_1')}(t_1), \dots, \widehat{(P_n',Q_n')}(t_n)$ for some $m, n \geq 0$ such that $t =_{\mathbb{ACUD}} s_1 + \dots + s_m - t_1 - \dots - t_n$ and $\pi_1$ is of the form

$$\cfrac{\cfrac{\begin{matrix}\vdots\\ \widehat{(P_1,Q_1)}(s_1)\end{matrix}}{R_1(s_1)}\,(C_1) \quad \dots \quad \cfrac{\begin{matrix}\vdots\\ \widehat{(P_m,Q_m)}(s_m)\end{matrix}}{R_m(s_m)}\,(C_m) \quad \cfrac{\begin{matrix}\vdots\\ \widehat{(P_1',Q_1')}(t_1)\end{matrix}}{R_1'(t_1)}\,(C_1') \quad \dots \quad \cfrac{\begin{matrix}\vdots\\ \widehat{(P_n',Q_n')}(t_n)\end{matrix}}{R_n'(t_n)}\,(C_n')}{F_{P,Q,S,T}(s_1 + \dots + s_m - t_1 - \dots - t_n)} \ (\mathcal{A}_{P,Q,S,T_{eq}}/\mathbb{ACU}) \tag{8.23}$$

where $C_i = R_i(x) \Leftarrow \widehat{(P_i,Q_i)}(x)$ for $1 \leq i \leq m$ and $C_i' = R_i'(x) \Leftarrow \widehat{(P_i',Q_i')}(x)$ for $1 \leq i \leq n$.

Hence the clause $R_i(a_{P_i,Q_i}) \in \mathcal{A}_{P,Q,S,T}$ for $1 \leq i \leq m$. Hence $R_i(a_{P_i,Q_i})$ is derivable in $\mathcal{A}_{P,Q,S,T}/\mathbb{ACUD}$ for $1 \leq i \leq m$. Similarly $R_i'(a_{P_i',Q_i'})$ is derivable in $\mathcal{A}_{P,Q}/\mathbb{ACUD}$ for $1 \leq i \leq n$. Also $\mathcal{A}_{P,Q,S,T_{eq}} \subseteq \mathcal{A}_{inter}$. Hence using (8.23) and Lemma 23 $F_{P,Q,S,T}(a_{P_1,Q_1} + \dots + a_{P_m,Q_m} - a_{P_1',Q_1'} - \dots - a_{P_n',Q_n'})$ is derivable in $\mathcal{A}_{P,Q,S,T}/\mathbb{ACUD}$. So $a_{P_1,Q_1} + \dots + a_{P_m,Q_m} - a_{P_1',Q_1'} - \dots - a_{P_n',Q_n'} \in \mathcal{L}'_{P,S}$. Therefore we must have $b_{I_1,I_1'}, \dots, b_{I_p,I_p'} \in S (p \geq 0)$ such that $a_{P_1,Q_1} + \dots + a_{P_m,Q_m} - a_{P_1',Q_1'} - \dots - a_{P_n',Q_n'} + (b_{I_1,I_1'} - b'_{I_1,I_1'}) + \dots + (b_{I_n,I_n'} - b_{I_n,I_n'}) \in \mathcal{L}_P(\mathcal{B}/\mathbb{ACUD})$.

For $1 \leq i \leq m$, since $s_i$ is functional, hence there is a free $f_i$ of arity $k_i$, and terms $s_i^1, \dots, s_i^{k_i}$ such that $s_i = f_i(s_i^1, \dots, s_i^{k_i})$. Since $\widehat{(P_i,Q_i)}(s_i)$ is in the functional support of $\pi_1$, it uses as last clause a clause of the form $\widehat{(P_i,Q_i)}(f_i(x_1, \dots, x_{k_i})) \Leftarrow (P_i^1, Q_i^1)(x_1) \wedge \dots \wedge (P_i^{k_i}, Q_i^{k_i})(x_{k_i})$ corresponding to clauses $P_i(f_i(x_1, \dots, x_{k_i})) \Leftarrow P_i^1(x_1) \wedge \dots \wedge P_i^{k_i}(x_{k_i})$ and $Q_i(f_i(x_1, \dots, x_{k_i})) \Leftarrow Q_i^1(x_1) \wedge \dots \wedge Q_i^{k_i}(x_{k_i})$ of $\mathcal{A}_{free}$ so that $(P_i^j, Q_i^j)(s_i^j)$ is derivable in $\mathcal{A}_{inter}/\mathbb{ACUD}$ using a derivation strictly smaller than $\pi_1$. By induction hypothesis, we have $s_i'^j =_{\mathbb{ACUM}} s_i''^j =_{\mathbb{ACUM}} s_i^j$ such that $P_i^j(s_i'^j)$ and $Q_i^j(s_i''^j)$ are derivable in $\mathcal{A}/\mathbb{ACUD}$. Let $s_i' = f_i(s_i'^1, \dots, s_i'^{k_i})$ and $s_i'' = f_i(s_i''^1, \dots, s_i''^{k_i})$. For $1 \leq i \leq m$

we have $s'_i =_{\mathbb{ACUD}} s_i$ and $P_i(s'_i)$ is derivable in $\mathcal{A}/\mathbb{ACUD}$ using the clause $P_i(f_i(x_1, \ldots, x_{k_i})) \Leftarrow P_i^1(x_1) \wedge \ldots \wedge P_i^{k_i}(x_{k_i})$. Similarly for $1 \leq i \leq m$ we have $s''_i =_{\mathbb{ACUD}} s_i$ and $Q_i(t''_i)$ is derivable in $\mathcal{A}/\mathbb{ACUD}$. Similarly for $1 \leq i \leq n$ we have terms $t'_i =_{\mathbb{ACUD}} t''_i =_{\mathbb{ACUD}} t_i$ such that $P'_i(t'_i)$ and $Q'_i(t''_i)$ are derivable in $\mathcal{A}/\mathbb{ACUD}$.

Now the derivation (call it $\pi_2$) of $P(a_{P_1,Q_1} + \ldots + a_{P_m,Q_m} - a_{P'_1,Q'_1} - \ldots - a_{P'_n,Q'_n} + (b_{I_1,I'_1} - b'_{I_1,I'_1}) + \ldots + (b_{I_n,I'_n} - b_{I_n,I'_n}))$ in $\mathcal{B}/\mathbb{ACUD}$ must have a functional support of the form $P_1^\dagger(a_{P_1,Q_1})$, $\ldots, P_m^\dagger(a_{P_m,Q_m}), P_1'^\dagger(a_{P'_1,Q'_1}), \ldots, P_n'^\dagger(a_{P'_n,Q'_n}), I_1^\dagger(b_{I_1,I'_1}), I_1^\ddagger(b_{I_1,I'_1}), \ldots, I_p^\dagger(b_{I_p,I'_p}), I_p^\ddagger(b_{I_p,I'_p})$ where for $1 \leq i \leq m$ $P_i^\dagger \in \{P_i, Q_i\}$, and for $1 \leq i \leq n$ $P_i'^\ddagger \in \{P'_i, Q'_i\}$, and for $1 \leq i \leq p$ $I_i^\dagger, I_i^\ddagger \in \{I_i, I'_i\}$. We have

$$\pi_2 = \frac{(\ \overline{P_1^\dagger(a_{P_1,Q_1})}\ )_{1\leq i\leq m} \quad (\ \overline{P_1'^\dagger(a_{P'_1,Q'_1})}\ )_{1\leq i\leq n} \quad (\ \overline{I_1^\dagger(b_{I_i,I'_i})} \quad \overline{I_1^\ddagger(b_{I_i,I'_i})}\ )_{1\leq i\leq p}}{P(\sum\limits_{1\leq i\leq m} a_{P_i,Q_i} - \sum\limits_{1\leq i\leq n} a_{P'_i,Q'_i} + \sum\limits_{1\leq i\leq p} (b_{I_i,I'_i} - b'_{I_i,I'_i}))} \ (\mathcal{B}_{eq}/\mathbb{ACUD}) \tag{8.24}$$

For $1 \leq i \leq p$ since $b_{I_i,I'_i} \in S$, by (*) $I_i(t'_{I_i,I'_i})$ and $I'_i(t'_{I_i,I'_i})$ are derivable in $\mathcal{A}/\mathbb{ACUD}$ and $t'_{I_i,I'_i} =_{\mathbb{ACUM}} t''_{I_i,I'_i}$. Recall that for $1 \leq i \leq m$, $P_i(s'_i)$ and $Q_i(s''_i)$ are derivable in $\mathcal{A}/\mathbb{ACUD}$ and for $1 \leq i \leq n$, $P'_i(t'_i)$ and $Q'_i(t''_i)$ are derivable in $\mathcal{A}/\mathbb{ACUD}$. Also $\mathcal{B}_{eq} = \mathcal{A}_{eq}$. So from (8.24) and Lemma 21 $P(s_1^\dagger + \ldots + s_m^\dagger - t_1^\dagger - \ldots - t_m^\dagger + (t^\dagger_{I_1,I'_1} - t^\ddagger_{I_1,I'_1}) + \ldots + (t^\dagger_{I_n,I'_n} - t^\ddagger_{I_n,I'_n}))$ is derivable in $\mathcal{A}/\mathbb{ACUD}$, where for $1 \leq i \leq m$ we have $s_i^\dagger \in \{s'_i, s''_i\}$, for $1 \leq i \leq n$ we have $t_i^\dagger \in \{t'_i, t''_i\}$ and for $1 \leq i \leq p$ we have $t^\dagger_{I_i,I'_i}, t^\ddagger_{I_i,I'_i} \in \{t'_{I_i,I'_i}, t''_{I_i,I'_i}\}$. Let the required $t'$ be $s_1^\dagger + \ldots + s_m^\dagger - t_1^\dagger - \ldots - t_m^\dagger + (t^\dagger_{I_1,I'_1} - t^\ddagger_{I_1,I'_1}) + \ldots + (t^\dagger_{I_n,I'_n} - t^\ddagger_{I_n,I'_n})$. We have $t =_{\mathbb{ACUM}} t'$ and $P(t')$ is derivable in $\mathcal{A}/\mathbb{ACUD}$. Similarly we can find $t''$ such that $t =_{\mathbb{ACUM}} t''$ and $Q(t'')$ is derivable in $\mathcal{A}/\mathbb{ACUD}$. □

**Theorem 24** *One-way $\mathbb{ACUM}$ automata are effectively closed under intersection.*

**Proof:** Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two one-way automata whose intersection we want to compute. We assume without loss of generality that they are built from disjoint sets of states $\mathbb{P}_1$ and $\mathbb{P}_2$ respectively, and that their final states are $P$ and $Q$ respectively. Define automaton $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$ on set of states $\mathbb{P} = \mathbb{P}_1 \cup \mathbb{P}_2$. We compute automaton $\mathcal{A}_{inter}$ corresponding to the automaton $\mathcal{A}$, as described in the above procedure. $\mathcal{A}_{inter}$ is not exactly a one-way automaton because of the presence of the extended epsilon clauses. However from Lemma 20, we have a one-way automaton $\mathcal{A}'_{inter}$ such that $\mathcal{L}_q(\mathcal{A}'_{inter}/\mathbb{ACUD}) = \mathcal{L}_q(\mathcal{A}_{inter}/\mathbb{ACUD})$ for each state $q$ of $\mathcal{A}_{inter}$.

1. First we show that $\mathcal{L}_P(\mathcal{A}_1/\mathbb{ACUM}) \cap \mathcal{L}_Q(\mathcal{A}_2/\mathbb{ACUM}) \subseteq \mathcal{L}_{(P,Q)}(\mathcal{A}'_{inter}/\mathbb{ACUM})$. Let $s \in \mathcal{L}_P(\mathcal{A}_1/\mathbb{ACUM}) \cap \mathcal{L}_Q(\mathcal{A}_2/\mathbb{ACUM})$. From Lemma 19 we have terms $t' =_{\mathbb{ACUM}} t'' =_{\mathbb{ACUM}} s$ such that $t' \in \mathcal{L}_P(\mathcal{A}_1/\mathbb{ACUD}) = \mathcal{L}_P(\mathcal{A}/\mathbb{ACUD})$ and $t'' \in \mathcal{L}_Q(\mathcal{A}_2/\mathbb{ACUD}) = \mathcal{L}_Q(\mathcal{A}/\mathbb{ACUD})$. From Lemma 37 there is a $t =_{\mathbb{ACUM}} t'$ such that $t \in \mathcal{L}_{(P,Q)}(\mathcal{A}_{inter}/\mathbb{ACUD}) = \mathcal{L}_{(P,Q)}(\mathcal{A}'_{inter}/\mathbb{ACUD}) \subseteq \mathcal{L}_{(P,Q)}(\mathcal{A}'_{inter}/\mathbb{ACUM})$. Since $s =_{\mathbb{ACUM}} t$ we have $s \in \mathcal{L}_{(P,Q)}(\mathcal{A}'_{inter}/\mathbb{ACUM})$.
2. Next we show that $\mathcal{L}_{(P,Q)}(\mathcal{A}'_{inter}/\mathbb{ACUM}) \subseteq \mathcal{L}_P(\mathcal{A}_1/\mathbb{ACUM}) \cap \mathcal{L}_Q(\mathcal{A}_2/\mathbb{ACUM})$. Let $s \in \mathcal{L}_{(P,Q)}(\mathcal{A}'_{inter}/\mathbb{ACUM})$. Since $\mathcal{A}'$ is a one-way automaton, by Lemma 19 there is a $t =_{\mathbb{ACUM}} s$ such that $t \in \mathcal{L}_{(P,Q)}(\mathcal{A}'_{inter}/\mathbb{ACUD}) = \mathcal{L}_{(P,Q)}(\mathcal{A}_{inter}/\mathbb{ACUM})$. By Lemma 38 we have $t' =_{\mathbb{ACUM}} t'' =_{\mathbb{ACUM}} t$ such that $t' \in \mathcal{L}_P(\mathcal{A}/\mathbb{ACUD}) = \mathcal{L}_P(\mathcal{A}_1/\mathbb{ACUD}) \subseteq \mathcal{L}_P(\mathcal{A}_1/\mathbb{ACUM})$ and $t'' \in \mathcal{L}_Q(\mathcal{A}/\mathbb{ACUD}) = \mathcal{L}_Q(\mathcal{A}_2/\mathbb{ACUD}) \subseteq \mathcal{L}_Q(\mathcal{A}_2/\mathbb{ACUM})$. Since $t =_{\mathbb{ACUM}} t' =_{\mathbb{ACUM}} t''$ hence $t \in \mathcal{L}_P(\mathcal{A}_1/\mathbb{ACUM})$ and $t \in \mathcal{L}_Q(\mathcal{A}_2/\mathbb{ACUM})$. Hence $t \in \mathcal{L}_P(\mathcal{A}_1/\mathbb{ACUM}) \cap \mathcal{L}_Q(\mathcal{A}_2/\mathbb{ACUM})$.

Hence by naming $(P, Q)$ as the final state of $\mathcal{A}'_{inter}$ we have $\mathcal{L}(\mathcal{A}'/\mathbb{ACUM}) = \mathcal{L}(\mathcal{A}_1/\mathbb{ACUM}) \cap \mathcal{L}(\mathcal{A}_2/\mathbb{ACUM})$. $\square$

## 8.5 Idempotence Axiom : $\mathbb{ACUI}$ Automata

In this section we show that one-way $\mathbb{ACUI}$ automata are closed under intersection. In the $\mathbb{ACUI}$ case also we use techniques similar to the previous cases.

Let $\mathcal{A}$ be a one-way $\mathbb{ACUI}$ 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 $\emptyset \neq S \subseteq \mathbb{P}$.

We introduce a set of new constants $a_S$ for each $\emptyset \neq S \subseteq \mathbb{P}$. Let $\mathcal{B} = \mathcal{A}_{eq} \cup \{P(a_S) \mid P \in S\}$. From Theorem 16 $\mathcal{L}_P(\mathcal{B}/\mathbb{ACU})$ is a semilinear set for every $P$. Define $\mathcal{L}_P = \{n_1 a_{S_1} + \ldots + n_k a_{S_k} \mid$ some $m_1 a_{S_1} + \ldots + m_k a_{S_k} \in \mathcal{L}_P(\mathcal{B}/\mathbb{ACU}), 1 \leq n_i \leq m_i\}$. This 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 Theorem 16, we can construct a constant-only $\mathbb{ACU}$ automaton $\mathcal{A}_S$ with final state $F_S$ such that $\mathcal{L}(\mathcal{A}_S/\mathbb{ACU}) = \mathcal{L}_S$. We assume that automata $\mathcal{A}_S$'s are built from mutually disjoint sets of (fresh) states.

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

– the clause $S(x) \Leftarrow F_S(x)$ for each $\emptyset \neq S \subseteq \mathbb{P}$.

– clauses of $\mathcal{A}_{S eq}$ for each $\emptyset \neq S \subseteq \mathbb{P}$.

– the clause $R(x) \Leftarrow \widehat{S}(x)$ for each clause $R(a_S)$ in some $\mathcal{A}_S$.

– the clause

$$\widehat{S}(f(x_1, \ldots, x_n)) \Leftarrow S_1(x_1), \ldots, S_n(x_n)$$

for clauses

$$P^i(f(x_1, \ldots, x_n)) \Leftarrow P^i_1(x_1), \ldots, P^i_n(x_n)$$

in $\mathcal{A}_{free}$ for $1 \leq i \leq k, k \geq 1$, where $S = \{P^1, \ldots, P^k\}$ and $S_j = \{P^1_j, \ldots, P^k_j\}$.

The idea is that if say three atoms $P(t'), Q(t''), R(t''')$ are derivable in $\mathcal{A}$ modulo $\mathbb{ACU}$ and $t' =_{\mathbb{ACUI}} t'' =_{\mathbb{ACUI}} t'''$ then their functional support must be of the form $P^1_1(t'^1_1), ..., P^{p_1}_1(t'^{p_1}_1), ..., P^1_n(t'^1_n), ..., P^{p_n}_n(t'^{p_n}_n)$ and $Q^1_1(t''^1_1), ..., Q^{q_1}_1(t''^{q_1}_1), ..., Q^1_n(t''^1_n), ..., Q^{q_n}_n(t''^{q_n}_n)$ and $R^1_1(t'''^1_1), ..., R^{r_1}_1(t'''^{r_1}_1), ..., R^1_n(t'''^1_n), ..., R^{r_n}_n(t'''^{r_n}_n)$, such that $p_i, q_i, r_i \geq 1$ and $t'^1_i =_{\mathbb{ACUI}} ... =_{\mathbb{ACUI}} t'^{p_i}_i =_{\mathbb{ACUI}} t''^1_i =_{\mathbb{ACUI}} ... =_{\mathbb{ACUI}} t''^{q_i}_i =_{\mathbb{ACUI}} t'''^1_i =_{\mathbb{ACUI}} ... =_{\mathbb{ACUI}} t'''^{r_i}_i$ for $1 \leq i \leq n$, so we can contract the terms to get $t'^1_1 + ... + t'^1_n, t''^1_1 + ... + t''^1_n$ and $t'''^1_1 + ... + t'''^1_n$. Then in $\mathcal{A}_{eq}$ we can derive $P(p_1 a_{S_1} + ... + p_n a_{S_n})$, $Q(q_1 a_{S_1} + ... + q_n a_{S_n})$ and $R(r_1 a_{S_1} + ... + r_n a_{S_n})$ where $S_j = \{P^1_j, ..., P^{p_j}_j, Q^1_j, ..., Q^{q_j}_j, R^1_j, ..., R^{r_j}_j\}$. Then $a_{S_1} + ... + a_{S_n}$ is in $\mathcal{L}_P \cap \mathcal{L}_Q \cap \mathcal{L}_R$. In this way the $a_S$'s are used to account for the contractions using equation $I$.

**Lemma 39** *If $P^1(t^1), \ldots, P^N(t^N)$ are derivable in $\mathcal{A}/\mathbb{ACU}$ for some $N \geq 1$, and $t^1 =_{\mathbb{ACUI}} \ldots =_{\mathbb{ACUI}} t^N$ then for some $t =_{\mathbb{ACUI}} t^1$, $\{P^1, ..., P^N\}(t)$ is derivable in $\mathcal{A}_{inter}/\mathbb{ACU}$.*

**Proof:** We do induction on the maximum of the sizes of the derivations of $P^1(t^1), \ldots, P^N(t^N)$ in $\mathcal{A}/\mathbb{ACU}$. To avoid heavy notation we consider the case $N = 2$ : the arguments easily generalize for arbitrary $N$. So we assume that $P(t')$ and $Q(t'')$ are derivable in $\mathcal{A}/\mathbb{ACU}$ and $t' =_{\mathbb{ACUI}} t''$. We need to show that $\{P, Q\}(t)$ is derivable in $\mathcal{A}_{inter}/\mathbb{ACU}$ for some $t =_{\mathbb{ACUI}} t'$. Since $t' =_{\mathbb{ACUI}} t''$ we have

$$t' =_{\mathbb{ACU}} t'^1_1 + \ldots + t'^{p_1}_1 + \ldots + t'^1_n + \ldots + t'^{p_n}_n$$

$$t'' =_{\mathbb{ACU}} t''^1_1 + \ldots + t''^{q_1}_1 + \ldots + t''^1_n + \ldots + t''^{q_n}_n$$

where $n \geq 0$, for $1 \leq i \leq n$ we have $p_i, q_i \geq 1$ and for $1 \leq j \leq p_i, 1 \leq j' \leq q_i$ $t'^j_i$ and $t''^j_i$ are functional and $t'^j_i =_{\mathbb{ACUI}} t''^{j'}_i$. (In particular, $t'^j_i =_{\mathbb{ACUI}} t'^{j'}_i$ for $1 \leq j \leq p_i, 1 \leq j' \leq p_i$ and $t''^j_i =_{\mathbb{ACUI}} t''^{j'}_i$ for $1 \leq j \leq q_i, 1 \leq j' \leq q_i$.)

The derivation $\pi_1$ of $P(t')$ has a functional support of the form $P^1_1(t'^1_1), \ldots, P^{p_1}_1(t'^{p_1}_1), \ldots, P^1_n(t'^1_n), \ldots, P^{p_n}_n(t'^{p_n}_n)$ and we have

$$\pi_1 = \cfrac{\begin{matrix} \vdots & & \vdots & & \vdots & & \vdots \\ P^1_1(t'^1_1) & \ldots & P^{p_1}_1(t'^{p_1}_1) & \ldots & P^1_n(t'^1_n) & \ldots & P^{p_n}_n(t'^{p_n}_n) \end{matrix}}{P(t'^1_1 + \ldots + t'^{p_1}_1 + \ldots + t'^1_n + \ldots + t'^{p_n}_n)} \; (\mathcal{A}_{eq}/\mathbb{ACU}) \tag{8.25}$$

The derivation $\pi_2$ of $Q(t'')$ has a functional support of the form $Q^1_1(t''^1_1), \ldots, Q^{q_1}_1(t''^{q_1}_1), \ldots, Q^1_n(t''^1_n), \ldots, Q^{q_n}_n(t''^{q_n}_n)$ and we have

$$\pi_2 = \cfrac{\begin{matrix} \vdots & & \vdots & & \vdots & & \vdots \\ Q^1_1(t''^1_1) & \ldots & Q^{q_1}_1(t''^{q_1}_1) & \ldots & Q^1_n(t''^1_n) & \ldots & Q^{q_n}_n(t''^{q_n}_n) \end{matrix}}{Q(t''^1_1 + \ldots + t''^{q_1}_1 + \ldots + t''^1_n + \ldots + t''^{q_n}_n)} \; (\mathcal{A}_{eq}/\mathbb{ACU}) \tag{8.26}$$

For $1 \leq i \leq n$ define $S_i = \{P^1_i, \ldots, P^{p_i}_i, Q^1_i, \ldots, Q^{q_i}_i\}$. For $1 \leq i \leq n, 1 \leq j \leq p_i$, $P^j_i(a_{S_i})$ is derivable in $\mathcal{A}/\mathbb{ACU}$. From (8.25) and Lemma 21 $P(p_1 a_{S_1} + \ldots + p_n a_{S_n})$ is derivable in $\mathcal{B}$. Similarly $Q(q_1 a_{S_1} + \ldots + q_n a_{S_n})$ is derivable in $\mathcal{B}$. Hence $a_{S_1} + \ldots + a_{S_n}$ belongs to $\mathcal{L}_P \cap \mathcal{L}_Q$. Let $S = \{P, Q\}$. $F_S(a_{S_1} + \ldots + a_{S_n})$ is derivable in $\mathcal{A}_S$.

For $1 \leq i \leq n, 1 \leq j \leq p_n$ since $t'^j_i$ is functional and $t'^j_i =_{\mathbb{ACUI}} t'^k_i$ for $1 \leq k \leq p_n$ hence we have some free $f_i$ of arity $k_i$ and terms $t'^j_{i,1}, \ldots, t'^j_{i,k_i}$ such that $t'^j_i = f_i(t'^j_{i,1}, \ldots, t'^j_{i,k_i})$. For $1 \leq i \leq n, 1 \leq j \leq q_n$ since $t''^j_i =_{\mathbb{ACUI}} t'^j_1$ we have terms $t''^j_{i,1}, \ldots, t''^j_{i,k_i}$ such that $t'^j_{i,k} =_{\mathbb{ACUI}} t''^j_{i,k_i}$ for $1 \leq k \leq k_i$ and $t''^j_i = f_i(t''^j_{i,1}, \ldots, t''^j_{i,k_i})$. For $1 \leq i \leq n, 1 \leq j \leq p_n$, since $P^j_i(t'^j_i)$ is in the functional support of $\pi_1$ we have a clause $P^j_i(f_i(x_1, \ldots, x_{k_i})) \Leftarrow P^j_{i,1}(x_1) \wedge \ldots \wedge P^j_{i,k_i}(x_{k_i}) \in \mathcal{A}_{free}$ such that for $1 \leq k \leq k_i$ the atom $P^j_{i,k}(t'^j_{i,k})$ is derivable in $\mathcal{A}/\mathbb{ACU}$ using a derivation strictly smaller than that of $\pi_1$. Similarly for $1 \leq i \leq n, 1 \leq j \leq q_n$, we have a clause $Q^j_i(f_i(x_1, \ldots, x_{k_i})) \Leftarrow Q^j_{i,1}(x_1) \wedge \ldots \wedge Q^j_{i,k_i}(x_{k_i})$ such that for $1 \leq k \leq k_i$ the atom $Q^j_{i,k}(t'^j_{i,k})$ is derivable in $\mathcal{A}/\mathbb{ACU}$ using a derivation strictly smaller than that of $\pi_2$. By induction hypothesis for $1 \leq i \leq n, 1 \leq k \leq k_i$, there is some $t_{i,k} =_{\mathbb{ACUI}} t'^1_{i,k}$ such that $S_{i,k}(t_{i,k})$ is derivable in $\mathcal{A}_{inter}/\mathbb{ACU}$, where $S_{i,k} = \{P^1_{i,k}, \ldots, P^{p_i}_{i,k}, Q^1_{1,k}, \ldots, Q^{q_i}_{i,k}\}$. For $1 \leq i \leq n$, $\mathcal{A}_{inter}$ has the clause $\widehat{S}_i(f_i(x_1, \ldots, x_{k_i})) \Leftarrow S_{i,1}(x_1) \wedge \ldots \wedge S_{i,k_i}(x_{k_i})$. Let $t_i = f_i(t_{i,1}, \ldots, t_{i,k_i})$. Then $\widehat{S}_i(t_i)$ is derivable in $\mathcal{A}_{inter}/\mathbb{ACU}$.

The derivation (call it $\pi_3$) of $F_S(a_{S_1} + \ldots + a_{S_n})$ in $\mathcal{A}_S/\mathbb{ACU}$ has a functional support of the form $R_1(a_{S_1}), \ldots, R_n(a_{S_n})$ and we have

$$\pi_3 = \cfrac{\overline{R_1(a_{S_1})} \quad \ldots \quad \overline{R_n(a_{S_n})}}{F_S(a_{S_1} + \ldots + a_{S_n})} \; (\mathcal{A}_{S\,eq}/\mathbb{ACU}) \tag{8.27}$$

For $1 \leq i \leq n$ since the clause $R_i(a_{S_i}) \in \mathcal{A}_S$ hence the clause $R_i(x) \Leftarrow \widehat{S}_i(x) \in \mathcal{A}_{inter}$. Since $\widehat{S}_i(t_i)$ is derivable in $\mathcal{A}_{inter}/\mathbb{ACU}$ hence $R_i(t_i)$ is derivable in $\mathcal{A}_{inter}/\mathbb{ACU}$. Since $\mathcal{A}_{S\,eq} \subseteq \mathcal{A}_{eq}$, from (8.27) and Lemma 21 $F_S(t_1 + \ldots + t_n)$ is derivable in $\mathcal{A}_{inter}/\mathbb{ACU}$. Hence using the clause $S(x) \Leftarrow$

$F_S(x)$ we get a derivation of $S(t_1 + \ldots + t_n)$ in $\mathcal{A}_{inter}/\mathbb{ACU}$. Let the required $t = t_1 + \ldots + t_n$. Then it is easy to see that $t =_{\mathbb{ACUI}} t'$. □

**Lemma 40** *If $\{P^1, \ldots, P^N\}(t)$ is derivable in $\mathcal{A}_{inter}/\mathbb{ACU}$ for some $N \geq 1$ then for some $t' =_{\mathbb{ACUI}} t$, $P^1(t')$ is derivable in $\mathcal{A}/\mathbb{ACU}$.*

**Proof:** We do induction on the size of the derivation of $\{P^1, \ldots, P^N\}(t)$. As in the proof of Lemma 39 we only consider the case $N = 2$, and the arguments easily generalize for arbitrary $N$. So let $S = \{P, Q\}$ and $S(t)$ be derivable in $\mathcal{A}_{inter}/\mathbb{ACU}$ where $S = \{P, Q\}$. Since $S(x) \Leftarrow F_S(x)$ is the only clause with the predicate $S$ in the head, hence the derivation $\pi$ of $S(t)$ must use the clause $S(x) \Leftarrow F_S(x)$ as the last clause. Also $F_S(t)$ is derivable in $\mathcal{A}_{inter}/\mathbb{ACU}$ using a derivation $\pi_1$ which is strictly smaller than $\pi$. Again from examination of the clauses in $\mathcal{A}_{inter}$, $\pi_1$ has a functional support of the form $\widehat{S_1}(t_1), \ldots, \widehat{S_n}(t_n)$ and we have

$$\pi_1 = \cfrac{\cfrac{\cfrac{\vdots}{\widehat{S_1}(t_1)}}{R_1(t_1)}\,(R_1(x) \Leftarrow \widehat{S_1}(x)) \quad \ldots \quad \cfrac{\cfrac{\vdots}{\widehat{S_n}(t_n)}}{R_n(t_n)}\,(R_n(x) \Leftarrow \widehat{S_n}(x))}{P(t_1 + \ldots + t_n)}\,(\mathcal{A}_{S\,eq}/\mathbb{ACU}) \tag{8.28}$$

Hence for $1 \leq i \leq n$ the clause $R_i(a_{S_i}) \in \mathcal{A}_S$. Hence $R_i(a_{S_i})$ is derivable in $\mathcal{A}_S/\mathbb{ACU}$ for $1 \leq i \leq n$. Using (8.28) and Lemma 21, $F_S(a_{S_1} + \ldots + a_{S_n})$ is derivable in $\mathcal{A}_S/\mathbb{ACU}$. So $a_{S_1} + \ldots + a_{S_n} \in \mathcal{L}_P$. Then there must be some $p_1, \ldots, p_n \geq 1$ such that $p_1 a_{S_1} + \ldots + p_n a_{S_n} \in \mathcal{L}_P(\mathcal{B}_{eq}/\mathbb{ACU})$.

For $1 \leq i \leq n$, since $t_i$ is functional we have some free $f_i$ of arity $k_i$ and terms $t_{i,1}, \ldots, t_{i,k_i}$ such that $t_i = f(t_{i,1}, \ldots, t_{i,k_i})$. Since $\widehat{S}_i(t_i)$ is in the functional support of $\pi_1$ hence there is a clause $\widehat{S}_i(f_i(x_1, \ldots, x_{k_i})) \Leftarrow S_{i,1}(x_1) \wedge \ldots \wedge S_{i,k_i}(x_{k_i})$ such that for $1 \leq k \leq k_i$ the atom $S_{i,k}(t_{i,k})$ is derivable in $\mathcal{A}_{inter}/\mathbb{ACU}$ using a derivation strictly smaller than $\pi$. By induction hypothesis, for each $P' \in S_{i,k}$ there is some $u =_{\mathbb{ACUI}} t_{i,k}$ such that $P'(u)$ is derivable in $\mathcal{A}/\mathbb{ACU}$. Also by construction of $\mathcal{A}_{inter}$ for each $P' \in S_i$ there must be some clause $P'(f_i(x_1, ..., x_{k_i})) \Leftarrow P'_1(x_1) \wedge ... \wedge P'_{k_i}(x_{k_i})$ in $\mathcal{A}$ such that $P'_k \in S_{i,k}$. Hence

for each $P' \in S_i$ there is some $u =_{\mathbb{ACUI}} t_i$ such that $P'(u)$ is derivable in $\mathcal{A}/\mathbb{ACU}$ (*)

Now the derivation (call it $\pi_2$) of $P(p_1 a_{S_1} + \ldots + p_n a_{S_n})$ in $\mathcal{B}_{eq}/\mathbb{ACU}$ has a functional support of the form $P_1^1(a_{S_1}), \ldots, P_1^{p_1}(a_{S_1}), \ldots, P_n^1(a_{S_n}), \ldots, P_n^{p_n}(a_{S_n})$ such that for $1 \leq i \leq n, 1 \leq j \leq p_i$, $P_i^j \in S_i$, and we have

$$\pi_2 = \cfrac{\cfrac{\vdots}{P_1^1(a_{S_1})} \quad \ldots \quad \cfrac{\vdots}{P_1^{p_1}(a_{S_1})} \quad \ldots \quad \cfrac{\vdots}{P_n^1(a_{S_n})} \quad \ldots \quad \cfrac{\vdots}{P_n^{p_n}(a_{S_n})}}{P(p_1 a_{S_1} + \ldots + p_n a_{S_n})}\,(\mathcal{B}_{eq}) \tag{8.29}$$

From (*) for $1 \leq i \leq n, 1 \leq j \leq p_i$ we have term $t_i^j =_{\mathbb{ACUI}} t_i$ such that $P_i^j(t_i^j)$ is derivable in $\mathcal{A}/\mathbb{ACU}$. Let the require $t' = t_1^1 + \ldots + t_1^{p_1} + \ldots + t_n^1 + \ldots + t_n^{p_n}$. We know that $\mathcal{A}_{eq} = \mathcal{B}_{eq}$. Hence from (*) and Lemma 21 $P(t')$ is derivable in $\mathcal{A}/\mathbb{ACU}$. Also it is clear that $t' =_{\mathbb{ACUI}} t$. □

**Theorem 25** *One-way $\mathbb{ACUI}$ automata are effectively closed under intersection.*

**Proof:** Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two one-way automata whose intersection we want to compute. We assume without loss of generality that they are built from disjoint sets of states $\mathbb{P}_1$ and $\mathbb{P}_2$ respectively, and that their final states are $P$ and $Q$ respectively. Define automaton $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$ on set of states

$\mathbb{P} = \mathbb{P}_1 \cup \mathbb{P}_2$. We compute automaton $\mathcal{A}_{inter}$ corresponding to the automaton $\mathcal{A}$, as described in the above procedure. $\mathcal{A}_{inter}$ is a one-way automaton.

1. First we show that $\mathcal{L}_P(\mathcal{A}_1/\mathbb{ACUI}) \cap \mathcal{L}_Q(\mathcal{A}_2/\mathbb{ACUI}) \subseteq \mathcal{L}_{\{P,Q\}}(\mathcal{A}_{inter}/\mathbb{ACUI})$. Let $s \in \mathcal{L}_P(\mathcal{A}_1/\mathbb{ACUI}) \cap \mathcal{L}_Q(\mathcal{A}_2/\mathbb{ACUI})$. From Lemma 19 we have terms $t' =_{\mathbb{ACUI}} t'' =_{\mathbb{ACUI}} s$ such that $t' \in \mathcal{L}_P(\mathcal{A}_1/\mathbb{ACU}) = \mathcal{L}_P(\mathcal{A}/\mathbb{ACU})$ and $t'' \in \mathcal{L}_Q(\mathcal{A}_2/\mathbb{ACU}) = \mathcal{L}_Q(\mathcal{A}/\mathbb{ACU})$. From Lemma 39 there is a $t =_{\mathbb{ACUI}} t'$ such that $t \in \mathcal{L}_{\{P,Q\}}(\mathcal{A}_{inter}/\mathbb{ACU}) \subseteq \mathcal{L}_{\{P,Q\}}(\mathcal{A}_{inter}/\mathbb{ACUI})$. Since $s =_{\mathbb{ACUI}} t$ we have $s \in \mathcal{L}_{\{P,Q\}}(\mathcal{A}_{inter}/\mathbb{ACUI})$.
2. Next we show that $\mathcal{L}_{\{P,Q\}}(\mathcal{A}_{inter}/\mathbb{ACUI}) \subseteq \mathcal{L}_P(\mathcal{A}_1/\mathbb{ACUI}) \cap \mathcal{L}_Q(\mathcal{A}_2/\mathbb{ACUI})$. Let $s \in \mathcal{L}_{\{P,Q\}}(\mathcal{A}_{inter}/\mathbb{ACUI})$. By Lemma 19 there is a $t =_{\mathbb{ACUI}} s$ such that $t \in \mathcal{L}_{\{P,Q\}}(\mathcal{A}_{inter}/\mathbb{ACU})$. By Lemma 40 we have $t' =_{\mathbb{ACUI}} t'' =_{\mathbb{ACUI}} t$ such that $t' \in \mathcal{L}_P(\mathcal{A}/\mathbb{ACU}) = \mathcal{L}_P(\mathcal{A}_1/\mathbb{ACU}) \subseteq \mathcal{L}_P(\mathcal{A}_1/\mathbb{ACUI})$ and $t'' \in \mathcal{L}_Q(\mathcal{A}/\mathbb{ACU}) = \mathcal{L}_Q(\mathcal{A}_2/\mathbb{ACU}) \subseteq \mathcal{L}_Q(\mathcal{A}_2/\mathbb{ACUI})$. Since $t =_{\mathbb{ACUI}} t' =_{\mathbb{ACUI}} t''$ hence $t \in \mathcal{L}_P(\mathcal{A}_1/\mathbb{ACUI})$ and $t \in \mathcal{L}_Q(\mathcal{A}_2/\mathbb{ACUI})$. Hence $t \in \mathcal{L}_P(\mathcal{A}_1/\mathbb{ACUI}) \cap \mathcal{L}_Q(\mathcal{A}_2/\mathbb{ACUI})$.

Hence by naming $\{P, Q\}$ as the final state of $\mathcal{A}_{inter}$ we have $\mathcal{L}(\mathcal{A}_{inter}/\mathbb{ACUI}) = \mathcal{L}(\mathcal{A}_1/\mathbb{ACUI}) \cap \mathcal{L}(\mathcal{A}_2/\mathbb{ACUI})$. □

## 8.6 Conclusion

We have shown that modulo all the $\mathbb{AC}$ theories we deal with ($\mathbb{AC}$, $\mathbb{ACU}$, $\mathbb{ACUX}$, $\mathbb{ACUX}_n$, $\mathbb{ACUM}$, $\mathbb{ACUI}$), the one-way tree automata are closed under intersection. This result however does not generalize to arbitrary equational theories since we seen in Chapter 4 that one-way $\mathbb{A}$ tree automata are not closed under intersection. We have already shown in Chapter 7 that emptiness of one-way equational tree automata is also decidable. As a result intersection-emptiness of these one-way equational tree automata is decidable, which is the important question from the point of view of verification of cryptographic protocols. We emphasize that these results also imply decidability of membership because checking whether $t \in \mathcal{L}(\mathcal{A}/\mathbb{E})$ is equivalent to checking whether $\mathcal{L}(\{t\}) \cap \mathcal{L}(\mathcal{A}/\mathbb{E})$. The latter problem reduces to the problem of intersection-emptiness because $\mathcal{L}(\{t\})$ is accepted by a one-way $\mathbb{E}$ tree automaton by Lemma 19.

## Chapitre 9

# Complementation des automates équationnels unidirectionnels (Complementation of One-Way Equational Tree Automata)

Dans ce chapitre nous étudions la clôture pas complémentaire des automates d'arbres équationnels unidirectionnels. Nous avons vu que les automates d'arbres unidirectionnels modulo toutes les théories associatives commutatives que nous considérons sont clos par intersection. Mais la situation est très différente dans le cas du complémentaire. Nous verrons alors que les automates unidirectionnels modulo les théories $\mathbb{AC}$, $\mathbb{ACU}$ et $\mathbb{ACUD}$ sont clos par complémentaire, ceux modulo les théories $\mathbb{ACUX}$, $\mathbb{ACUX}_n$, $\mathbb{ACUM}$ et $\mathbb{ACUI}$ ne le sont pas. En ce sens ces automates d'arbres équationnels se comportent différemment des automates d'arbres non équationnels, qui sont clos par complémentaire. De même un comportement intéressant visible dans ces résultats est la coïncidence entre la clôture par complémentaire des automates équationnels unidirectionnels et la linéarité de la théorie équationnelle considérée. Ceci peut nous faire conjecturer que les automates unidirectionnels modulo $\mathbb{E}$ sont clos par complémentation ssi $\mathbb{E}$ est linéaire. Mais ceci est faux, parce que la théorie $\mathbb{A}$ est linéaire, et nous avons vu au chapitre 4 que les automates unidirectionnels modulo $\mathbb{A}$ ne sont clos ni par intersection ni par complémentaire. Notons que parmi les propriétés de clôture, la clôture par complémentaire est souvent la plus difficile à établir. Nous avons donc réussi à répondre à cette question pour les automates unidirectionnels modulo toutes les théories associatives commutatives que nous considérons.

In this chapter we study the closure under complementation of the one-way equational tree automata. We have seen that the one-way tree automata modulo all the associative commutative theories that we are considering are closed under intersection. However the situation is very different in the case of complementation. We will see that while the one-way tree automata modulo the theories $\mathbb{AC}$, $\mathbb{ACU}$ and $\mathbb{ACUD}$ are closed under complementation, those modulo the theories $\mathbb{ACUX}$, $\mathbb{ACUX}_n$, $\mathbb{ACUM}$, $\mathbb{ACUI}$ are not. In this respect these equational tree automata behave differently from non-equational tree automata which are closed under complementation. Also an interesting pattern visible in these results is the coincidence between the closure under complementation of the one-way equational tree automata and the linearity of the equational theory considered. This might lead us to conjecture that one-way $\mathbb{E}$ tree automata are closed under complementation iff $\mathbb{E}$ is linear. This is however false, since the theory $\mathbb{A}$ is linear however we have seen in Chapter 4 that that one-way $\mathbb{A}$ tree automata are not closed under intersection, nor under complementation. Note that among the closure properties, closure under complementation are often the most difficult to be established. In this respect, we have succeeded in answering this question for the one-way automata modulo all the $\mathbb{AC}$-like theories we are dealing with.

## 9.1 Complementation of $\mathbb{ACU}$ Automata

We show in this section that one-way $\mathbb{ACU}$ automata are closed under complementation.

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

Define automaton $\mathcal{B} = \mathcal{A}_{eq} \cup \{P(a_S) \mid P \in S\}$. From Theorem 16, $\mathcal{L}_P(\mathcal{B}/\mathbb{ACU})$ is a semilinear set for every $P \in \mathbb{P}$. Given $S \subseteq \mathbb{P}$, define $\mathcal{L}^S = \bigcap_{P \in S} \mathcal{L}_P(\mathcal{B}/\mathbb{ACU}) \setminus \bigcup_{P \in \mathbb{P} \setminus S} \mathcal{L}_P(\mathcal{B}/\mathbb{ACU})$. This is a semilinear set. By Theorem 16, we can construct a constant-only automaton $\mathcal{A}_S$ with final state $F_S$ such that $\mathcal{L}(\mathcal{A}_S/\mathbb{ACU}) = \mathcal{L}^S$. We assume that automata $\mathcal{A}_S$'s are built from mutually disjoint sets of fresh states.

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

1. for each $S \subseteq \mathbb{P}$ the clause $\breve{S}(x) \Leftarrow F_S(x)$.
2. clauses of $\mathcal{A}_{S_{eq}}$ for $S \subseteq \mathbb{P}$.
3. for each clause $R(a_S)$ in some $\mathcal{A}_S$, the clause $R(x) \Leftarrow \widehat{S}(x)$.
4. for each free function symbol $f$ of arity $n$, and each $S_1, \ldots, S_n \subseteq \mathbb{P}$, the clause

$$\widehat{S}(f(x_1, \ldots, x_n)) \Leftarrow \breve{S}_1(x_1) \wedge \ldots \wedge \breve{S}_n(x_n)$$

where

$$S = \{P \mid \exists P_1 \in S_1 \cdot \exists \ldots \exists P_n \in S_n \cdot P(f(x_1, \ldots, x_n)) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n) \in \mathcal{A}_{free}\}$$

This construction plays the role of the usual determinization procedure in standard tree automata. Note that it is however more complex because of the $\mathbb{ACU}$ symbol. The idea in defining $\mathcal{A}_{eq}$ and $\mathcal{A}_S$'s is to compute all possible derivations using the clauses of the equational part. The constant $a_S$ acts as 'place marker' for the terms accepted at $\widehat{S}$ for $S \subseteq \mathbb{P}$. This is made precise in the following lemma :

**Lemma 41** *For any $S \subseteq \mathbb{P}$ and any term $t$, $\breve{S}(t)$ is derivable in $\mathcal{A}^\dagger/\mathbb{ACU}$ iff $S = \{P \in \mathbb{P} \mid P(t)$ is derivable in $\mathcal{A}/\mathbb{ACU}\}$.*

**Proof:** We do induction on the size of $t$. Let $t =_{\mathbb{ACU}} t_1 + \ldots + t_n (n \geq 0)$ such that for $1 \leq i \leq n$, $t_i = f_i(t_i^1, \ldots, t_i^{k_i})$ and $f_i$ is free. Let $S = \{P \in \mathbb{P} \mid P(t)$ is derivable in $\mathcal{A}/\mathbb{ACU}\}$. First we show that *(a)* $\breve{S}(t)$ is derivable in $\mathcal{A}^\dagger/\mathbb{ACU}$; then we show that *(b)* if $\breve{S}'(t)$ is derivable in $\mathcal{A}^\dagger/\mathbb{ACU}$ for some $S' \subseteq \mathbb{P}$ then $S' = S$. This will complete our proof.

**Proof of (a) :** For $1 \leq i \leq n, 1 \leq j \leq k_i$ let $S_i^j = \{Q \mid Q(t_i^j)$ is derivable in $\mathcal{A}/\mathbb{ACU}\}$. By induction hypothesis $\breve{S}_i^j(t_i^j)$ is derivable in $\mathcal{A}^\dagger/\mathbb{ACU}$. For $1 \leq i \leq n$ let $S_i = \{Q \mid \exists Q^1 \in S_i^1 \cdot \exists \ldots \exists Q^{k_i} \in S_i^{k_i} \cdot Q(f_i(x_1, \ldots, x_{k_i})) \Leftarrow Q^1(x_i) \wedge \ldots \wedge Q^{k_i}(x_{k_i}) \in \mathcal{A}_{free}\}$. The clause $\widehat{S}_i(f_i(x_1, \ldots, x_{k_i})) \Leftarrow \breve{S}_i^1(x_1) \wedge \ldots \wedge \breve{S}_i^{k_i}(x_{k_i}) \in \mathcal{A}^\dagger$. So the atom $\widehat{S}_i(t_i)$ is derivable in $\mathcal{A}^\dagger/\mathbb{ACU}$ for $1 \leq i \leq n$.

For each $P \in S$, the derivation (call it $\pi_P$) of $P(t)$ has a functional support of the form $Q_1^P(t_1)$, $\ldots, Q_n^P(t_n)$, and we have

$$\pi_P = \cfrac{\begin{matrix} \vdots & & \vdots \\ Q_1^P(t_1) & \ldots & Q_n^P(t_n) \end{matrix}}{P(t_1 + \ldots + t_n)} \; (\mathcal{A}_{eq}/\mathbb{ACU}) \tag{9.1}$$

For $P \in S, 1 \leq i \leq n$, since $Q_i^P(t_i)$ is in the functional support of $\pi_P$, it has a derivation which uses the clause $Q_i^P(f_i(x_1, \ldots, x_{k_i})) \Leftarrow Q_i^{P,1}(x_1) \wedge \ldots \wedge Q_i^{P,k_i}(x_{k_i})$ as the last clause, and for $1 \leq j \leq k_i$, the atom $Q_i^{P,j}(t_i^j)$ is derivable in $\mathcal{A}/\mathbb{ACU}$. For $P \in S, 1 \leq i \leq n, 1 \leq j \leq k_i$, by definition of $S_i^j$, $Q_i^{P,j} \in S_i^j$. For $P \in S, 1 \leq i \leq n$, by definition of $S_i$, $Q_i^P \in S_i$. Hence the atom $Q_i^P(a_{S_i})$ is derivable in $\mathcal{B}/\mathbb{ACU}$. Also by definition $\mathcal{B}_{eq} = \mathcal{A}_{eq}$. From (9.1) and Lemma 21,

$$P(a_{S_1} + \ldots + a_{S_n}) \text{ is derivable in } \mathcal{B}/\mathbb{ACU} \text{ for each } P \in S \tag{*}$$

We also claim that

$$\text{if } P \in \mathbb{P} \setminus S \text{ then } a_{S_1} + \ldots + a_{S_n} \notin \mathcal{L}_P(\mathcal{B}/\mathbb{ACU}) \tag{**}$$

This is so because if $P \in \mathbb{P} \setminus S$ and $a_{S_1} + \ldots + a_{S_n} \in \mathcal{L}_P(\mathcal{B}/\mathbb{ACU})$, then the corresponding derivation (call it $\pi'$) of $P(a_{S_1} + \ldots + a_{S_n})$ would have a functional support of the form $Q_1(a_{S_1}), \ldots, Q_n(a_{S_n})$ where $Q_i \in S_i$ for $1 \leq i \leq n$. We would have

$$\pi' = \cfrac{\overline{Q_1(a_{S_1})} \quad \ldots \quad \overline{Q_n(a_{S_n})}}{P(a_{S_1} + \ldots + a_{S_n})} \; (\mathcal{B}/\mathbb{ACU}) \tag{9.2}$$

For $1 \leq i \leq n$, since $Q_i \in S_i$ by construction of $S_i$, $Q_i(t_i)$ would be derivable in $\mathcal{A}/\mathbb{ACU}$. Also by definition $\mathcal{B}_{eq} = \mathcal{A}_{eq}$. Hence by (9.2) and Lemma 21, $P(t_1 + \ldots + t_n)$ would be derivable in $\mathcal{A}/\mathbb{ACU}$. Hence we would have a derivation of $P(t)$ in $\mathcal{A}/\mathbb{ACU}$ which would contradict the definition of $S$. Hence claim (**) holds.

Combining *(*)* and *(**)*, $a_{S_1} + \ldots + a_{S_n} \in \mathcal{L}^S$. The derivation (call it $\pi$) of $F_S(a_{S_1} + \ldots + a_{S_n})$ in $\mathcal{A}_S/\mathbb{ACU}$ has a functional support of the form $R_1(a_{S_1}), \ldots, R_n(a_{S_n})$, and we have

$$\pi = \cfrac{\overline{R_1(a_{S_1})} \quad \ldots \quad \overline{R_n(a_{S_n})}}{F_S(a_{S_1} + \ldots + a_{S_n})} \; (\mathcal{A}_{Seq}/\mathbb{ACU}) \tag{9.3}$$

For $1 \leq i \leq n$, since the clause $R_i(a_{S_i}) \in \mathcal{A}_S$, hence the clause $R_i(x) \Leftarrow \widehat{S}_i(x) \in \mathcal{A}^\dagger$. Also we know that $\widehat{S}_i(t_i)$ is derivable in $\mathcal{A}^\dagger/\mathbb{ACU}$. Hence $R_i(t_i)$ is derivable in $\mathcal{A}^\dagger/\mathbb{ACU}$ for $1 \leq i \leq n$. By construction $\mathcal{A}_{Seq} \subseteq {\mathcal{A}^\dagger}_{eq}$. By (9.3) and Lemma 21, $F_S(t_1 + \ldots + t_n)$ is derivable in $\mathcal{A}^\dagger/\mathbb{ACU}$. Finally we use the clause $\breve{S}(x) \Leftarrow F_S(x)$ to get a derivation of $\breve{S}(t_1 + \ldots + t_n)$, i.e. of $\breve{S}(t)$ in $\mathcal{A}^\dagger/\mathbb{ACU}$. This proves *(a)*.

**Proof of (b) :** Now suppose $S' \subseteq \mathbb{P}$ such that $\breve{S}'(t)$ is derivable in $\mathcal{A}^\dagger/\mathbb{ACU}$. We need to show that $S' = S$. $\breve{S}'(x) \Leftarrow F_{S'}(x)$ is the only clause which has the predicate $\breve{S}'$ in the head. Hence the derivation $\pi_1$ of $\breve{S}'(t)$ uses the clause $\breve{S}'(x) \Leftarrow F_{S'}(x)$ as the last clause, and the atom $F_{S'}(t)$ is derivable in $\mathcal{A}^\dagger/\mathbb{ACU}$ using a derivation $\pi_2$ which is strictly smaller than $\pi_1$. Again from examination of the clauses in $\mathcal{A}^\dagger$, $\pi_2$ has a functional support of the form $\widehat{S'_1}(t_1), \ldots, \widehat{S'_n}(t_n)$ and we have

$$\pi_2 = \cfrac{\cfrac{\begin{matrix}\vdots\\ \widehat{S'_1}(t_1)\end{matrix}}{R'_1(t_1)}\,(R'_1(x) \Leftarrow \widehat{S'_1}(x)) \quad \cfrac{\begin{matrix}\vdots\\ \widehat{S'_n}(t_n)\end{matrix}}{R'_n(t_n)}\,(R'_n(x) \Leftarrow \widehat{S'_n}(x))}{F_{S'}(t_1 + \ldots + t_n)}\,(\mathcal{A}_{S'eq}/\mathbb{ACU}) \tag{9.4}$$

Hence for $1 \le i \le n$, the clause $R'_i(a_{S'_i}) \in \mathcal{A}_{S'}$. Hence the atom $R'_i(a_{S'_i})$ is derivable in $\mathcal{A}_{S'}/\mathbb{ACU}$. By (9.4) and Lemma 21, $F_{S'}(a_{S'_1} + \ldots + a_{S'_n})$ is derivable in $\mathcal{A}_{S'}/\mathbb{ACU}$. For $1 \le i \le n$ the derivation of $\widehat{S'_i}(t_i)$ in $\mathcal{A}^\dagger/\mathbb{ACU}$ uses a clause $\widehat{S'_i}(f_i(x_1, \ldots, x_{k_i})) \Leftarrow \breve{S}_i'^1(x_1) \wedge \ldots \wedge \breve{S}_i'^{k_i}(x_{k_i}) \in \mathcal{A}^\dagger$ as the last clause, and for $1 \le j \le k_i$, the atom $\breve{S}_i'^j(t_i^j)$ is derivable in $\mathcal{A}^\dagger/\mathbb{ACU}$. By induction hypothesis $S_i'^j = S_i^j$. By definition of $\mathcal{A}^\dagger$, $S'_i = S_i$. Hence $a_{S_1} + \ldots + a_{S_n} \in \mathcal{L}_{S'}$. We have already seen that $a_{S_1} + \ldots + a_{S_n} \in \mathcal{L}^S$. Since the $\mathcal{L}^S$'s are mutually disjoint by construction, so $S' = S$. This proves *(b)*. □

As a consequence we obtain :

**Theorem 26** *One-way $\mathbb{ACU}$ automata are effectively closed under complementation.*

**Proof:** Let $\mathcal{A}$ be a one-way $\mathbb{ACU}$ automaton with predicates from $\mathbb{P}$ and with final state $F$. Define automaton $\mathcal{A}^\dagger$ as above. Then pick a fresh predicate $F^\dagger$ and add to $\mathcal{A}^\dagger$ the clauses $F^\dagger(x) \Leftarrow \breve{S}(x)$ for every $S \subseteq \mathbb{P}$ such that $F \notin S$. Call this new automaton $\mathcal{A}^\ddagger$. Then by Lemma 41,

$$\begin{array}{lll} t \in \mathcal{L}_{F^\dagger}(\mathcal{A}^\ddagger/\mathbb{ACU}) & \text{iff for some } S \subseteq \mathbb{P},\ F \notin S \text{ and } t \in \mathcal{L}_{\breve{S}}(\mathcal{A}^\dagger/\mathbb{ACU}) & \\ & \text{iff } F \notin \{P \in \mathbb{P} \mid t \in \mathcal{L}_P(\mathcal{A}/\mathbb{ACU})\} & \text{(by Lemma 41)} \\ & \text{iff } t \notin \mathcal{L}_F(\mathcal{A}/\mathbb{ACU}) & \end{array}$$

We let $F^\dagger$ be the final state of $\mathcal{A}^\ddagger$. Then $\mathcal{A}^\ddagger/\mathbb{ACU}$ accepts the complement of the language accepted by $\mathcal{A}/\mathbb{ACU}$. □

The idea of using constants as abstractions for the functional terms accepted at certain states, and the correspondence between constant-only $\mathbb{ACU}$ automata and Presburger definability, have also been used in Chapter 8 to show closure under intersection of one-way equational tree automata automata for a certain number of theories. This may give the impression that the arguments for closure under complementation are similar to the ones for intersection. While there are indeed some common ideas, it is actually surprising that while similar ideas work for showing closure under intersection of the one-way automata modulo all theories under consideration, they don't work for showing closure under complementation of the one-way automata modulo the theories $\mathbb{ACUX}$, $\mathbb{ACUX}_n$, $\mathbb{ACUM}$ and $\mathbb{ACUI}$. We show later in this chapter that the latter are not closed under complementation. Unlike in the case of intersection, our results show a strong correlation between closure under complementation of one-way equational tree automata and the linearity of the equational theory involved, at least as far as the theories dealt with in this thesis are concerned.

### 9.1.1 Complementation of $\mathbb{AC}$ Automata

As in the case of intersection, we state the same result for $\mathbb{AC}$ case without repeating the proof :

**Theorem 27** *One-way $\mathbb{AC}$ automata are effectively closed under complementation.*

## 9.2 Counter-Example for $\mathbb{ACUX}$ and $\mathbb{ACUX}_n$ Automata

Contrary to the $\mathbb{ACU}$ case, one-way $\mathbb{ACUX}$ automata are not closed under complementation, as we show in this section. The result generalizes to the $\mathbb{ACUX}_n$ case for $n \geq 2$, i.e. we can show that one-way $\mathbb{ACUX}_n$ automata are not closed under complementation for any $n \geq 2$. (We get the $\mathbb{ACUX}$ case by making $n = 2$.) To show this fix some $n \geq 2$. Let our signature be $\Sigma = \{+, 0, a, f\}$ where $a$ is a constant and $f$ is a free unary symbol. Define languages $\mathcal{L}_1 = \mathbb{ACUX}_n(\{f^{k_1}(a) + \ldots + f^{k_n}(a) \mid k_1, \ldots, k_n \geq 0\})$ and $\mathcal{L}_2 = \mathbb{ACUX}_n(\{0\})$.

Then $\mathcal{L}_1 \setminus \mathcal{L}_2 = \mathbb{ACUX}_n(\{f^{k_1}(a) + \ldots + f^{k_n}(a) \mid k_1, \ldots, k_n \geq 0$ and for some $1 \leq i, j \leq n, i \neq j$ and $k_i \neq k_j\}$. $\mathcal{L}_1$ and $\mathcal{L}_2$ are clearly accepted by one-way $\mathbb{ACUX}_n$ automata. However $\mathcal{L}_1 \setminus \mathcal{L}_2$ is not :

**Lemma 42** *The language $\mathcal{L} = \mathbb{ACUX}_n(\{f^{k_1}(a) + \ldots + f^{k_n}(a) \mid k_1, \ldots, k_n \geq 0$ and for some $1 \leq i, j \leq n, i \neq j$ and $k_i \neq k_j\})$ is not accepted by any one-way $\mathbb{ACUX}_n$ automaton.*

**Proof:** Assume on the contrary that there is a one-way automaton $\mathcal{A}$ with final state $P$ such that $\mathcal{L}(\mathcal{A}/\mathbb{ACUX}_n) = \mathcal{L}$. Let $\mathbb{P}$ be the set of predicates in $\mathcal{A}$. For each $p \geq 0$ define $S_p = \{Q \in \mathbb{P} \mid Q(f^p(a))$ is derivable in $\mathcal{A}/\mathbb{ACUX}_n\}$. Since $\mathbb{P}$ is finite, it has finitely many subsets. Hence there are $p, q \in \mathbb{N}$ such that $p \neq q$ and $S_p = S_q$. Since $p \neq q$, the term $f^p(a) + \underbrace{f^q(a) + \ldots + f^q(a)}_{n-1 \text{ times}} \in \mathcal{L}$. From Lemma 19, we have some $t =_{\mathbb{ACUX}_n} f^p(a) + \underbrace{f^q(a) + \ldots + f^q(a)}_{n-1 \text{ times}}$ such that $P(t)$ is derivable in $\mathcal{A}/\mathbb{ACU}$. We have terms $t_i$ for $1 \leq i \leq n$ and terms $u_i^j$ for $1 \leq i \leq k, 1 \leq j \leq n$ for some $k \geq 0$ such that

1. $t_1$ is functional and $t_1 =_{\mathbb{ACUX}_n} f^p(a)$
2. for $2 \leq i \leq n$, $t_i$ is functional and $t_i =_{\mathbb{ACUX}_n} f^q(a)$
3. for $1 \leq i \leq k, 1 \leq j \leq n$, $u_i^j$ is functional and for $1 \leq j' \leq n$, $u_i^j =_{\mathbb{ACUX}_n} u_i^{j'}$
4. $t =_{\mathbb{ACU}} t_1 + t_2 + \ldots + t_n + u_1^1 + \ldots + u_1^n + \ldots + u_k^1 + \ldots + u_k^n$

The derivation $\pi$ of $P(t)$ has a functional support of the form $I_1(t_1), I_2(t_2), \ldots, I_n(t_n), I_1^1(u_1^1), \ldots, I_1^n(u_1^n), \ldots, I_k^1(u_k^1), \ldots, I_k^n(u_k^n)$, and we have

$$\pi = \dfrac{\begin{array}{ccccccccccc} \vdots & \vdots & & \vdots & \vdots & & \vdots & & \vdots & & \vdots \\ I_1(t_1) & I_2(t_2) & \ldots & I_n(t_n) & I_1^1(u_1^1) & \ldots & I_1^n(u_1^n) & \ldots & I_k^1(u_k^1) & \ldots & I_k^n(u_k^n) \end{array}}{P(t)} \; (\mathcal{A}_{eq}/\mathbb{ACU}) \tag{9.5}$$

We have $I_1 \in S_p$ and $I_2, \ldots, I_n \in S_q$. However $S_p = S_q$, hence $I_2, \ldots, I_n \in S_p$. Thus $I_1(f^p(a))$, $I_2(f^p(a))$, ..., $I_n(f^p(a))$ are derivable in $\mathcal{A}/\mathbb{ACUX}_n$. From (9.5) and Lemma 21, $P(\underbrace{f^p(a) + \ldots + f^p(a)}_{n \text{ times}}$

$+u_1^1+\ldots+u_1^n+\ldots+u_k^1+\ldots+u_k^n$) is derivable in $\mathcal{A}/\mathbb{ACUX}_n$. Hence $P(0)$ is derivable in $\mathcal{A}/\mathbb{ACUX}_n$ leading to contradiction. □

Since one-way $\mathbb{ACUX}_n$ automata are closed under intersection, and since $\mathcal{L}_1 \setminus \mathcal{L}_2 = \mathcal{L}_1 \cap \complement(\mathcal{L}_2)$ where $\complement(\mathcal{L}')$ denotes the complement of language $\mathcal{L}'$, we conclude :

**Theorem 28** *One-way $\mathbb{ACUX}_n$ automata are not closed under complementation for any $n \geq 2$. In particular one-way $\mathbb{ACUX}$ automata are not closed under complementation.*

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

## 9.3 Complementation of $\mathbb{ACUD}$ Automata

In the $\mathbb{ACUD}$ case the situation is similar to the $\mathbb{ACU}$ case : the automata are closed under complementation.

Fix a signature $\Sigma$ containing at least the symbols $+, -$ and $0$. Let $\mathcal{A}$ be a one-way $\mathbb{ACUD}$ automaton on signature $\Sigma$ and with predicates from $\mathbb{P}$. We introduce fresh predicate symbols $\breve{S}$ and $\widehat{S}$ for each $S \subseteq \mathbb{P}$, and sets $A = \{a_S \mid S \subseteq \mathbb{P}\}$ and $\overline{A} = \{\overline{a_S} \mid S \subseteq \mathbb{P}\}$ of fresh constants. Define automaton $\mathcal{B} = \mathcal{A}_{eq} \cup \{P(a_S) \mid P \in S\}$. $\mathcal{B}$ is an automaton on the signature $A \cup \{+, -, 0\}$. From Theorem 18, for each $P$, $\mathcal{L}_P(\mathcal{B}/\mathbb{ACUD})$ is a semilinear set on constants from $A \cup \overline{A}$. Given $S \subseteq \mathbb{P}$, define $\mathcal{L}^S = \bigcap_{P \in S} \mathcal{L}_P(\mathcal{B}/\mathbb{ACUD}) \setminus \bigcup_{P \in \mathbb{P} \setminus S} \mathcal{L}_P(\mathcal{B}/\mathbb{ACUD})$. This is a semilinear set on constants from $A \cup \overline{A}$. Using Theorem 18 we compute a constant-only $\mathbb{ACUD}$ automaton $\mathcal{A}_S$ using constants from $A$ and with final state $F_S$ such that $\mathcal{L}(\mathcal{A}_S/\mathbb{ACUD}) = \mathcal{L}^S$. We assume that the automata $\mathcal{A}_S$'s are all built from mutually disjoint sets of fresh predicates.

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

1. for each $S \subseteq \mathbb{P}$ the clause $\breve{S}(x) \Leftarrow F_S(x)$.
2. clauses of $\mathcal{A}_{S\,eq}$ for $S \subseteq \mathbb{P}$.
3. for each clause $R(a_S)$ in some $\mathcal{A}_S$, the clause $R(x) \Leftarrow \widehat{S}(x)$.
4. for each free functional symbol $f$ of arity $n$, and each $S_1, \ldots, S_n \subseteq \mathbb{P}$, the clause

$$\widehat{S}(f(x_1, \ldots, x_n)) \Leftarrow \breve{S}_1(x_1) \wedge \ldots \wedge \breve{S}_n(x_n)$$

where

$$S = \{P \mid \exists P_1 \in S_1 \cdot \exists \ldots \exists P_n \in S_n \cdot P(f(x_1, \ldots, x_n)) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n) \in \mathcal{A}_{free}\}$$

**Lemma 43** *For any $S \subseteq \mathbb{P}$ and any term $t$, $\breve{S}(t)$ is derivable in $\mathcal{A}^\dagger/\mathbb{ACUD}$ iff $S = \{P \in \mathbb{P} \mid P(t)$ is derivable in $\mathcal{A}/\mathbb{ACUD}\}$.*

**Proof:** We do induction on the size of $t$. Let $t =_{\mathbb{ACUD}} s_1 + \ldots + s_m - t_1 - \ldots - t_n (m, n \geq 0)$ such that for $1 \leq i \leq m$ we have $s_i = f_i(s_i^1, \ldots, s_i^{k_i})$ for some free $f_i$, and for $1 \leq i \leq n$ we have $t_i = g_i(t_i^1, \ldots, t_i^{l_i})$ for some free $g_i$. Let $S = \{P \in \mathbb{P} \mid P(t)$ is derivable in $\mathcal{A}/\mathbb{ACUD}\}$. First we show that *(a)* $\breve{S}(t)$ is derivable in $\mathcal{A}^\dagger/\mathbb{ACUD}$ ; then we show that *(b)* if $\breve{S}'(t)$ is derivable in $\mathcal{A}^\dagger/\mathbb{ACUD}$ for some $S' \subseteq \mathbb{P}$ then $S' = S$. This will complete our proof.

**Proof of (a) :** For $1 \leq i \leq m, 1 \leq j \leq k_i$ let $S_i^j = \{Q \mid Q(s_i^j) \text{ is derivable in } \mathcal{A}/\mathbb{ACUD}\}$. By induction hypothesis $\breve{S}_i^j(s_i^j)$ is derivable in $\mathcal{A}^\dagger/\mathbb{ACUD}$. For $1 \leq i \leq n, 1 \leq j \leq l_i$ let $T_i^j = \{Q \mid Q(t_i^j) \text{ is derivable in } \mathcal{A}/\mathbb{ACUD}\}$. By induction hypothesis $\breve{T}_i^j(t_i^j)$ is derivable in $\mathcal{A}^\dagger/\mathbb{ACUD}$. For $1 \leq i \leq m$ let $S_i = \{Q \mid \exists Q^1 \in S_i^1 \cdot \exists \ldots \exists Q^{k_i} \in S_i^{k_i} \cdot Q(f_i(x_1, \ldots, x_{k_i})) \Leftarrow Q^1(x_i) \wedge \ldots \wedge Q^{k_i}(x_{k_i}) \in \mathcal{A}_{free}\}$. The clause $\widehat{S}_i(f_i(x_1, \ldots, x_{k_i})) \Leftarrow S_i^1(x_1) \wedge \ldots \wedge S_i^{k_i}(x_{k_i}) \in \mathcal{A}^\dagger$. So the atom $\widehat{S}_i(s_i)$ is derivable in $\mathcal{A}^\dagger/\mathbb{ACUD}$ for $1 \leq i \leq m$. Similarly for $1 \leq i \leq n$ let $T_i = \{Q \mid \exists Q^1 \in T_i^1 \cdot \exists \ldots \exists Q^{l_i} \in T_i^{l_i} \cdot Q(g_i(x_1, \ldots, x_{l_i})) \Leftarrow Q^1(x_i) \wedge \ldots \wedge Q^{l_i}(x_{l_i}) \in \mathcal{A}_{free}\}$. The clause $\widehat{T}_i(g_i(x_1, \ldots, x_{l_i})) \Leftarrow T_i^1(x_1) \wedge \ldots \wedge T_i^{l_i}(x_{l_i}) \in \mathcal{A}^\dagger$. So the atom $\widehat{T}_i(t_i)$ is derivable in $\mathcal{A}^\dagger/\mathbb{ACUD}$ for $1 \leq i \leq n$.

For each $P \in S$, the derivation (call it $\pi_P$) of $P(t)$ has a functional support of the form $Q_1^P(s_1)$, $\ldots, Q_m^P(s_m), Q_1'^P(t_1), \ldots, Q_n'^P(t_n)$, and we have

$$\pi_P = \cfrac{\begin{matrix}\vdots & & \vdots & & \vdots & & \vdots \\ Q_1^P(s_1) & \ldots & Q_m^P(s_m) & \ldots & Q_1'^P(t_1) & \ldots & Q_n'^P(t_n)\end{matrix}}{P(s_1 + \ldots + s_m - t_1 - \ldots - t_n)}\ (\mathcal{A}_{eq}/\mathbb{ACUD}) \tag{9.6}$$

For $P \in S, 1 \leq i \leq m$, since $Q_i^P(s_i)$ is in the functional support of $\pi_P$, it has a derivation which uses the clause $Q_i^P(f_i(x_1, \ldots, x_{k_i})) \Leftarrow Q_i^{P,1}(x_1) \wedge \ldots \wedge Q_i^{P,k_i}(x_{k_i})$ as the last clause, and for $1 \leq j \leq k_i$ the atom $Q_i^{P,j}(s_i^j)$ is derivable in $\mathcal{A}/\mathbb{ACUD}$. For $P \in S, 1 \leq i \leq m, 1 \leq j \leq k_i$, by definition of $S_i^j$, $Q_i^{P,j} \in S_i^j$. For $P \in S, 1 \leq i \leq m$, by definition of $S_i$, $Q_i^P \in S_i$. Hence the atom $Q_i^P(a_{S_i})$ is derivable in $\mathcal{B}/\mathbb{ACUD}$ for $P \in S, 1 \leq i \leq m$. Similarly the atom $Q_i'^P(a_{T_i})$ is derivable in $\mathcal{B}/\mathbb{ACUD}$ for $P \in S, 1 \leq i \leq n$. Also by definition $\mathcal{B}_{eq} = \mathcal{A}_{eq}$. From (9.6) and Lemma 23,

$P(a_{S_1} + \ldots + a_{S_m} - a_{T_1} - \ldots - a_{T_n})$ is derivable in $\mathcal{B}/\mathbb{ACUD}$ for each $P \in S$ (*)

We also claim that

if $P \in \mathbb{P} \setminus S$ then $a_{S_1} + \ldots + a_{S_m} - a_{T_1} - \ldots - a_{T_n} \notin \mathcal{L}_P(\mathcal{B}/\mathbb{ACUD})$ (**)

This is so because if $P \in \mathbb{P} \setminus S$ and $a_{S_1} + \ldots + a_{S_m} - a_{T_1} - \ldots - a_{T_n} \in \mathcal{L}_P(\mathcal{B}/\mathbb{ACUD})$, then the corresponding derivation (call it $\pi'$) of $P(a_{S_1} + \ldots + a_{S_m} - a_{T_1} - \ldots - a_{T_n})$ would have a functional support of the form $Q_1(a_{S_1}), \ldots, Q_m(a_{S_m}), Q_1'(a_{T_1}), \ldots, Q_n'(a_{T_n})$ where $Q_i \in S_i$ for $1 \leq i \leq m$ and $Q_i' \in T_i$ for $1 \leq i \leq n$. We would have

$$\pi' = \cfrac{\overline{Q_1(a_{S_1})} \quad \ldots \quad \overline{Q_m(a_{S_m})} \quad \ldots \quad \overline{Q_1'(a_{T_1})} \quad \ldots \quad \overline{Q_n'(a_{T_n})}}{P(a_{S_1} + \ldots + a_{S_n} - a_{T_1} - \ldots - a_{T_n})}\ (\mathcal{B}/\mathbb{ACUD}) \tag{9.7}$$

For $1 \leq i \leq m$, since $Q_i \in S_i$, by construction of $S_i$, $Q_i(s_i)$ would be derivable in $\mathcal{A}/\mathbb{ACUD}$. Similarly for $1 \leq i \leq n$, $R_i(t_i)$ would be derivable in $\mathcal{A}/\mathbb{ACUD}$. Hence by (9.7 and Lemma 23, $P(s_1 + \ldots + s_m - t_1 - \ldots - t_n)$ would be derivable in $\mathcal{A}/\mathbb{ACUD}$. Hence we would have a derivation of $P(t)$ in $\mathcal{A}/\mathbb{ACUD}$ which would contradicts the definition of $S$. Hence claim (**) holds.

Combining (*) and (**), $a_{S_1} + \ldots + a_{S_m} - a_{T_1} - \ldots - a_{T_n} \in \mathcal{L}^S$. The derivation (call it $\pi$) of $F_S(a_{S_1} + \ldots + a_{S_m} - a_{T_1} - \ldots - a_{T_n})$ in $\mathcal{A}/\mathbb{ACUD}$ has a functional support of the form $R_1(a_{S_1})$, $\ldots, R_m(a_{S_m}), R_1'(a_{T_1}), \ldots, R_n'(a_{T_n})$, and we have

$$\pi_1 = \cfrac{\overline{R_1(a_{S_1})} \quad \ldots \quad \overline{R_m(a_{S_m})} \quad \ldots \quad \overline{R_1'(a_{T_1})} \quad \ldots \quad \overline{R_n'(a_{T_n})}}{F_S(a_{S_1} + \ldots + a_{S_m} - a_{T_1} - \ldots - a_{T_n})}\ (\mathcal{A}_{S eq}/\mathbb{ACU}) \tag{9.8}$$

For $1 \leq i \leq m$ since the clause $R_i(a_{S_i}) \in \mathcal{A}_S$, hence the clause $R_i(x) \Leftarrow \widehat{S}_i(x) \in \mathcal{A}^\dagger$. Also we know that $\widehat{S}_i(s_i)$ is derivable in $\mathcal{A}^\dagger/\mathbb{ACUD}$. Hence $R_i(s_i)$ is derivable in $\mathcal{A}^\dagger/\mathbb{ACUD}$ for $1 \leq i \leq m$.

Similarly $R'_i(t_i)$ is derivable in $\mathcal{A}^\dagger/\mathbb{ACUD}$ for $1 \leq i \leq n$. By construction $\mathcal{A}_{Seq} \subseteq \mathcal{A}^\dagger{}_{eq}$ By (9.8) and Lemma 23, $F_S(s_1 + \ldots + s_m - t_1 - \ldots - t_n)$ is derivable in $\mathcal{A}^\dagger/\mathbb{ACUD}$. Finally we use the clause $\breve{S}(x) \Leftarrow F_S(x)$ to get a derivation of $\breve{S}(s_1 + \ldots + s_m - t_1 - \ldots - t_n)$, i.e. of $\breve{S}(t)$ in $\mathcal{A}^\dagger/\mathbb{ACUD}$. This proves *(a)*.

**Proof of (b) :** Now suppose $S' \subseteq \mathbb{P}$ such that $\breve{S}'(t)$ is derivable in $\mathcal{A}^\dagger/\mathbb{ACUD}$. We need to show that $S' = S$. $\breve{S}'(x) \Leftarrow F_{S'}(x)$ is the only clause which has the predicate $\breve{S}'$ in the head. Hence the derivation $\pi_1$ of $\breve{S}'(t)$ uses the clause $\breve{S}'(x) \Leftarrow F_{S'}(x)$ as the last clause, and the atom $F_{S'}(t)$ is derivable in $\mathcal{A}^\dagger/\mathbb{ACUD}$ using a derivation $\pi_2$ which is strictly smaller than $\pi_1$. Again from examination of the clauses in $\mathcal{A}^\dagger$, $\pi_2$ has a functional support of the form $\widehat{S'_1}(s_1), \ldots, \widehat{S'_m}(s_m), \widehat{T'_1}(t_1), \ldots, \widehat{T'_n}(t_n)$ and we have

$$\pi_2 = \cfrac{\cfrac{\begin{matrix}\vdots\\ \widehat{S'_1}(s_1)\end{matrix}}{R''_1(s_1)}\,(C_1) \quad \ldots \quad \cfrac{\begin{matrix}\vdots\\ \widehat{S'_m}(s_m)\end{matrix}}{R''_m(s_m)}\,(C_m) \quad \ldots \quad \cfrac{\begin{matrix}\vdots\\ \widehat{T'_1}(t_1)\end{matrix}}{R'''_1(t_1)}\,(C'_1) \quad \ldots \quad \cfrac{\begin{matrix}\vdots\\ \widehat{T'_n}(t_n)\end{matrix}}{R'''_n(t_n)}\,(C'_n)}{F_{S'}(s_1 + \ldots + s_m - t_1 - \ldots - t_n)}\,(\mathcal{A}_{S'eq}/\mathbb{ACUD}) \tag{9.9}$$

where $C_i = R''_i(x) \Leftarrow S'_i(x)$ for $1 \leq i \leq m$ and $C'_i = R'''_i(x) \Leftarrow T'_i(x)$ for $1 \leq i \leq n$.

Hence for $1 \leq i \leq m$, the clause $R''_i(a_{S'_i}) \in \mathcal{A}_{S'_i}$. Hence the atom $R''_i(a_{S'_i})$ is derivable in $\mathcal{A}_{S'}/\mathbb{ACUD}$ for $1 \leq i \leq m$. Similarly the atom $R'''_i(a_{T'_i})$ is derivable in $\mathcal{A}_{S'}/\mathbb{ACUD}$ for $1 \leq i \leq n$. By (9.9) and Lemma 23, $F_{S'}(a_{S'_1} + \ldots + a_{S'_m} - a_{T'_1} - \ldots - a_{T'_n})$ is derivable in $\mathcal{A}_{S'}/\mathbb{ACUD}$. For $1 \leq i \leq m$ the derivation of $\widehat{S'_i}(s_i)$ in $\mathcal{A}^\dagger/\mathbb{ACUD}$ uses a clause $\widehat{S'_i}(f_i(x_1, \ldots, x_{k_i})) \Leftarrow S'^1_i(x_1) \wedge \ldots \wedge S'^{k_i}_i(x_{k_i}) \in \mathcal{A}^\dagger$ as the last clause, and for $1 \leq j \leq k_i$, the atom $S'^j_i(s^j_i)$ is derivable in $\mathcal{A}^\dagger/\mathbb{ACUD}$. By induction hypothesis $S'^j_i = S^j_i$. By definition of $\mathcal{A}^\dagger$, $S'_i = S_i$ for $1 \leq i \leq m$. Similarly for $T'_i = T_i$ for $1 \leq i \leq n$. Hence $a_{S_1} + \ldots + a_{S_m} - a_{T_1} - \ldots - a_{T_n} \in \mathcal{L}_{S'}$. We have already seen that $a_{S_1} + \ldots + a_{S_m} - a_{T_1} - \ldots - a_{T_n} \in \mathcal{L}^S$. Since the $\mathcal{L}^S$'s are mutually disjoint by construction, so $S' = S$. □

Then as in the $\mathbb{ACU}$ case we conclude that

**Theorem 29** *One-way $\mathbb{ACUD}$ automata are effectively closed under complementation.*

## 9.4 Counter-Example for Abelian Groups Automata

The Abelian groups ($\mathbb{ACUM}$) case is similar to the $\mathbb{ACUX}$ case : the automata are not closed under complementation. Let our signature be $\Sigma = \{+, -, 0, a, f\}$ where $a$ is a constant and $f$ is a free unary symbol. Define languages $\mathcal{L}_1 = \mathbb{ACUM}(\{f^n(a) - f^m(a) \mid n, m \geq 0\})$, $\mathcal{L}_2 = \mathbb{ACUM}(\{0\})$. Then $\mathcal{L}_1 \setminus \mathcal{L}_2 = \mathbb{ACUM}(\{f^n(a) - f^m(a) \mid n, m \geq 0 \text{ and } n \neq m\})$. $\mathcal{L}_1$ and $\mathcal{L}_2$ are clearly accepted by one-way $\mathbb{ACUM}$ automata. However $\mathcal{L}_1 \setminus \mathcal{L}_2$ is not :

**Lemma 44** *The language $\mathcal{L} = \mathbb{ACUM}(\{f^n(a) - f^m(a) \mid n, m \geq 0$ and $n \neq m\})$ is not accepted by any one-way $\mathbb{ACUM}$ automaton.*

**Proof:** Assume on the contrary that there is a one-way automaton $\mathcal{A}$ with final state $P$ such that $\mathcal{L}(\mathcal{A}/\mathbb{ACUM}) = \mathcal{L}$. Let $\mathbb{P}$ be the set of predicates in $\mathcal{A}$. For each $n \geq 0$ define $S_n = \{Q \in \mathbb{P} \mid Q(f^n(a))$ is derivable in $\mathcal{A}/\mathbb{ACUM}\}$. Since $\mathbb{P}$ is finite it has finitely many subsets. Hence there are $n, m \in \mathbb{N}$ such that $n \neq m$ and $S_n = S_m$. Since $n \neq m$ hence $f^n(a) - f^m(a) \in \mathcal{L}$. From Lemma 19 we have some $t =_{\mathbb{ACUM}} f^n(a) - f^m(a)$ such that $P(t)$ is derivable in $\mathcal{A}/\mathbb{ACUD}$. We have terms $t_1, t_2$ and terms $u_i, v_i$ for $1 \leq i \leq k$ for some $k \geq 0$ such that

1. $t_1$ is functional and $t_1 =_{\mathbb{ACUM}} f^n(a)$
2. $t_2$ is functional and $t_2 =_{\mathbb{ACUM}} f^m(a)$
3. for $1 \leq i \leq k$, $u_i$ and $v_i$ are functional and $u_i =_{\mathbb{ACUM}} v_i$
4. $t =_{\mathbb{ACUD}} t_1 - t_2 + u_1 - v_1 + \ldots + u_k - v_k$

The derivation $\pi$ of $P(t)$ has a functional support of the form $I(t_1), J(t_2), I_1(u_1), J_1(v_1), \ldots, I_k(u_k), J_k(v_k)$, and we have

$$\pi = \frac{\begin{array}{cccccc} \vdots & \vdots & \vdots & \vdots & & \vdots \quad \vdots \\ I(t_1) & J(t_2) & I_1(u_1) & J_1(v_1) & \ldots & I_k(u_k) \quad J_k(v_k) \end{array}}{P(t)} \, (\mathcal{A}_{eq}/\mathbb{ACUD}) \tag{9.10}$$

We have $I \in S_n$ and $J \in S_m$. However $S_n = S_m$, hence $J \in S_n$. Hence $I(f^n(a))$ and $J(f^n(a))$ are derivable in $\mathcal{A}/\mathbb{ACUM}$. From (9.10) and Lemma 23, $P(f^n(a) - f^n(a) + u_1 - v_1 + \ldots + u_k - v_k)$ is derivable in $\mathcal{A}/\mathbb{ACUM}$. Hence $P(0)$ is derivable in $\mathcal{A}/\mathbb{ACUM}$ leading to contradiction. □

Since one-way $\mathbb{ACUM}$ automata are closed under intersection, we have as before :

**Theorem 30** *One-way $\mathbb{ACUM}$ automata are not closed under complementation.*

## 9.5 Counter-Example for $\mathbb{ACUI}$ Automata

We show in this section that one-way $\mathbb{ACUI}$ automata are not closed under complementation either. Let the signature be $\{+, 0, a, f\}$ where $a$ is a constant and $f$ is free unary symbol. Define languages $\mathcal{L}_1 = \mathbb{ACUI}(\{f^n(a) + f^m(a) \mid n, m \geq 0\})$ and $\mathcal{L}_2 = \mathbb{ACUI}(\{f^n(a) \mid n \geq 0\})$. Then the language $\mathcal{L}_1 \setminus \mathcal{L}_2 = \mathbb{ACUI}(\{f^n(a) + f^m(a) \mid n, m \geq 0 \text{ and } n \neq m\})$ is not accepted by any one-way $\mathbb{ACUI}$ automaton :

**Lemma 45** *The language $\mathcal{L} = \mathbb{ACUI}(\{f^n(a) + f^m(a) \mid n, m \geq 0$ and $n \neq m\})$ is not accepted by any one-way $\mathbb{ACUI}$ automaton.*

**Proof:** Assume on the contrary that there is a one-way automaton $\mathcal{A}$ with final state $P$ such that $\mathcal{L}(\mathcal{A}/\mathbb{ACUI}) = \mathcal{L}$. Let $\mathbb{P}$ be the set of predicates in $\mathcal{A}$. For each $n \geq 0$ define $S_n = \{Q \in \mathbb{P} \mid Q(f^n(a))$ is derivable in $\mathcal{A}/\mathbb{ACUI}\}$. Since $\mathbb{P}$ is finite it has finitely many subsets. Hence there are $n, m \in \mathbb{N}$ such that $n \neq m$ and $S_n = S_m$. Since $n \neq m$ hence $f^n(a) + f^m(a) \in \mathcal{L}$. From Lemma 19, we have some $t =_{\mathbb{ACUI}} f^n(a) + f^m(a)$ such that $P(t)$ is derivable in $\mathcal{A}/\mathbb{ACU}$. We have terms $s_1, \ldots, s_p, t_1, \ldots, t_q$ for some $p, q \geq 1$ such that

1. for $1 \leq i \leq p$, $s_i$ is functional and $s_i =_{\mathbb{ACUI}} f^n(a)$
2. for $1 \leq i \leq q$, $t_i$ is functional and $t_i =_{\mathbb{ACUI}} f^m(a)$
3. $t =_{\mathbb{ACUI}} s_1 + \ldots + s_p + t_1 + \ldots + t_q$

The derivation $\pi$ of $P(t)$ has a functional support of the form $I_1(s_1), \ldots, I_p(s_p), J_1(t_1), \ldots, J_q(t_q)$ and we have

$$\pi = \cfrac{\begin{array}{cccccc} \vdots & & \vdots & \vdots & & \vdots \\ I_1(s_1) & \ldots & I_p(s_p) & J_1(t_1) & \ldots & J_q(t_q) \end{array}}{P(t)} \; (\mathcal{A}_{eq}/\mathbb{ACU}) \tag{9.11}$$

We have $I_1, \ldots, I_p \in S_n$ and $J_1, \ldots, J_q \in S_m$. However $S_n = S_m$, hence $J_1, \ldots, J_q \in S_n$. Hence $I_1(f^n(a)), \ldots, I_p(f^n(a)), J_1(f^n(a)), \ldots, J_q(f^n(a))$ is derivable in $\mathcal{A}/\mathbb{ACUI}$. From (9.11) and Lemma 23, $P(\underbrace{f^n(a) + \ldots + f^n(a)}_{p+q \text{ times}})$ in $\mathcal{A}/\mathbb{ACUI}$. Hence $P(f^n(a))$ is derivable in $\mathcal{A}/\mathbb{ACUI}$ leading to contradiction. □

Since one-way $\mathbb{ACUI}$ automata are closed under intersection we have :

**Theorem 31** *One-way $\mathbb{ACUI}$ automata are not closed under complementation.*

## 9.6 Conclusion

We have studied the question whether one-way equational tree automata are closed under complementation, for the $\mathbb{AC}$-like theories. The answer to this question is strikingly different than the answer to complementation. We showed that for the theories $\mathbb{AC}$, $\mathbb{ACU}$ and $\mathbb{ACUD}$, which are linear, the one-way tree automata are closed under complementation. For the remaining theories $\mathbb{ACUX}$, $\mathbb{ACUX}_n$, $\mathbb{ACUM}$ and $\mathbb{ACUM}$, which are non-linear, the one-way tree automata are not closed under complementation. Hence we have fully answered the question of closure under complementation for one-way equational tree automata. The above coincidence between linearity of equational theories and closure under complementation of one-way equational tree automata does not generalize to all theories. For example, one-way $\mathbb{A}$ tree automata are not closed under complementation.

# Quatrième partie

# Automates d'arbres équationnels bidirectionnels : réduction aux automates unidirectionnels
# (Two-Way Equational Tree Automata : Reduction to One-Way)

# Chapitre 10

# Automates d'arbres équationnels bidirectionnels (Two-Way Equational Tree Automata)

Ayant traité les propriétés de clôture et de décidabilité des automates équationnels unidirectionnels, nous nous intéressons maintenant aux automates d'arbres équationnels bidirectionnels. Nous montrons dans ce chapitre que les automates d'arbres équationnels bidirectionnels (pour toutes les théories que nous considérons sauf $\mathbb{ACUI}$) peuvent être effectivement réduits aux automates unidirectionnels. Ainsi, ils ont la même expressivité que les automates unidirectionnels. Ils ont aussi les mêmes propriétés de décidabilité et de clôture que les automates unidirectionnels. En particulier le vide d'intersection de ces automates équationnels bidirectionnels est décidable, ce dont nous avons besoin dans le contexte de la vérification de protocoles cryptographiques, par exemple, dans la modélisation du protocole de Diffie-Hellman en groupe du chapitre 5. Le cas d'automates bidirectionnels modulo $\mathbb{ACUI}$ est aujourd'hui encore ouvert.

Having dealt with the closure and decidability properties of one-way equational tree automata, we now turn our attention to two-way equational tree automata. We show in this chapter that two-way equational tree automata (for all our theories under consideration except $\mathbb{ACUI}$) can be effectively reduced to one-way automata. Thus they have the same expressiveness as the one-way automata. They also have the same closure and decidability properties as the one-way automata. In particular intersection emptiness of these two-way equational tree automata is decidable, which is what is needed in the context of verification of cryptographic protocols, e.g. in the modeling of group Diffie-Hellman protocol in Chapter 5. The two-way $\mathbb{ACUI}$ case is currently open.

## 10.1 Two-Way $\mathbb{ACU}$ Automata

We show in this section that two-way $\mathbb{ACU}$ automata can be effectively reduced to one-way $\mathbb{ACU}$ automata. We fix a signature $\Sigma$ containing at least the symbols $+$ and $0$.

We describe a saturation procedure which adds new epsilon clauses to two-way automata, until the free push clauses become redundant. Consider a two-way automaton $\mathcal{A}$ with predicates from $\mathbb{P}$. We introduce a new set $S = \{a_P \mid P \in \mathbb{P}\}$ of constants. We define automaton $\mathcal{B} = \mathcal{A}_{eq} \cup \{P(a_P) \mid P \in \mathbb{P}\}$. For $P \in \mathbb{P}$, $\mathcal{L}_P(\mathcal{B}/\mathbb{ACU})$ is a semilinear set. In particular membership of an element in $\mathcal{L}_P(\mathcal{B}/\mathbb{ACU})$ is decidable.

If there is a free push clause

$$R(x_i) \Leftarrow P(f(x_1, \ldots, x_n)) \wedge R_1^1(x_{i_1}) \wedge \ldots \wedge R_1^{n_1}(x_{i_1}) \wedge \ldots \wedge R_k^1(x_{i_k}) \wedge \ldots \wedge R_k^{n_k}(x_{i_k}) \in \mathcal{A}$$

and a free pop clause

$$Q(f(x_1, \ldots, x_n)) \Leftarrow Q_1(x_1) \wedge \ldots \wedge Q_n(x_n) \in \mathcal{A}$$

such that

1. $a_Q \in \mathcal{L}_P(\mathcal{A}/\mathbb{ACU})$
2. $\forall j \in \{1, \ldots, k\} \cdot \exists t \cdot$ each of the predicates $Q_{i_j}, R_j^1, \ldots, R_j^{n_j}$ accepts $t$ in $\mathcal{A}_{one-way}/\mathbb{ACU}$
3. $\forall j \in \{1, \ldots, n\} \setminus \{i, i_1, \ldots, i_k\} \cdot \exists t \cdot Q_j$ accepts $t$ in $\mathcal{A}_{one-way}/\mathbb{ACU}$

then we write $\mathcal{A} \rhd \mathcal{A} \cup \{R(x_i) \Leftarrow Q_i(x_i)\}$, which we take to constitute the base step of our saturation procedure. The idea is that if a free push clause is ever used then the corresponding symbol $f$ must have been introduced by some free pop clauses. However in between the applications of the free pop clause and the free pop clauses, there might be some applications of the clauses of $\mathcal{A}_{eq}$. In the above construction, the automata $\mathcal{B}$ represents the possible derivations using clauses of $\mathcal{A}_{eq}$. The constants $a_P$ are used as abstractions for functional terms accepted at $P$.

The base step of our saturation procedure is harmless :

**Lemma 46** *Let $\mathcal{A} \rhd \mathcal{A} \cup \{R(x_i) \Leftarrow Q_i(x_i)\}$ as above. Then any atom derivable in $\mathcal{A} \cup \{R(x_i) \Leftarrow Q_i(x_i)\}/\mathbb{ACU}$ is also derivable in $\mathcal{A}/\mathbb{ACU}$.*

**Proof:** It is sufficient to show that

for any $t_i$, if $Q_i(t_i)$ is derivable in $\mathcal{A}/\mathbb{ACU}$, then $R(t_i)$ is derivable in $\mathcal{A}/\mathbb{ACU}$ (*).

This is because given an arbitrary derivation $\pi$ in which the clause $R(x_i) \Leftarrow Q_i(x_i)$ is used $N \geq 1$ times, we can pick a subderivation $\pi_{sub}$ of $\pi$ which uses the clause $R(x_i) \Leftarrow Q_i(x_i)$ at the root but nowhere else. Using (*) we have a subderivation $\pi'_{sub}$ which uses only the clauses of $\mathcal{A}$ and which has the same conclusion as $\pi_{sub}$. We replace the subderivation $\pi_{sub}$ in $\pi$ by the subderivation $\pi'_{sub}$, to get a subderivation $\pi'$ which has the same conclusion as $\pi$ and which uses the clause $R(x_i) \Leftarrow Q_i(x_i)$ $N-1$ times. By iterating this process $N$ times we get a derivation which has the same conclusion as $\pi$ and which does not use the clause $R(x_i) \Leftarrow Q_i(x_i)$.

Now we proceed to prove (*). As in the definition above, let $t_{i_j}$ be some term accepted in $\mathcal{A}/\mathbb{ACU}$ at the predicates $Q_{i_j}, R_j^1, \ldots, R_j^{n_j}$ for $j \in \{1, \ldots, k\}$. Also let $t_j$ be some term accepted in $\mathcal{A}/\mathbb{ACU}$ at $Q_j$ for $j \in \{1, \ldots, n\} \setminus \{i, i_1, \ldots, i_k\}$. Then $Q(f(t_1, \ldots, t_n))$ is derivable in $\mathcal{A}/\mathbb{ACU}$ using the free pop clause. As $a_Q \in \mathcal{L}_P(\mathcal{A}/\mathbb{ACU})$, the derivation (call it $\pi_1$) of $P(a_Q)$ in $\mathcal{B}/\mathbb{ACU}$ has the functional support $Q(a_Q)$ and we have

$$\pi_1 = \cfrac{\overline{Q(a_Q)}}{\overline{\overline{P(a_Q)}}} \; (\mathcal{B}_{eq}/\mathbb{ACU}) \tag{10.1}$$

Also by definition $\mathcal{B}_{eq} = \mathcal{A}_{eq}$. Hence from the fact that $Q(f(t_1, \ldots, t_n))$ is derivable in $\mathcal{A}/\mathbb{ACU}$, and using (10.1) and from Lemma 21, we get a derivation of $P(f(t_1, \ldots, t_n))$ in $\mathcal{A}/\mathbb{ACU}$. Using the free push clause we get a derivation of $R(t_i)$ in $\mathcal{A}/\mathbb{ACU}$. This proves (*). $\square$

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

Given a two-way $\mathbb{ACU}$ automaton $\mathcal{A}$ our saturation procedure consists of (don't care non deterministically) generating a sequence $\mathcal{A}_0(= \mathcal{A}) \rhd \mathcal{A}_1 \rhd \mathcal{A}_2 \ldots$ of automata, until no new clauses can be added. This procedure always terminates because there are only a finite number of epsilon clauses possible. (If the number of predicates in $\mathbb{P}$ is $n$ then the number of epsilon clauses possible is $n^2$.) Let the final (saturated) automaton be $\mathcal{C}$. Then it is clear that $\mathcal{A}$ and $\mathcal{C}$ have the same set of derivable atoms. Then we remove clauses (3.13) from $\mathcal{C}$ to get a one-way automaton $\mathcal{C}_{one-way}$. This last step is also harmless :

**Lemma 47** *If any atom is derivable in $\mathcal{C}/\mathbb{ACU}$, then it is derivable in $\mathcal{C}_{one-way}/\mathbb{ACU}$.*

**Proof:** It is sufficient to show the following property :

(*) if $\pi$ is a derivation in $\mathcal{C}/\mathbb{ACU}$ and $\pi$ uses a push clause at the root but nowhere else, then there is a derivation $\pi'$ in $\mathcal{C}_{one-way}/\mathbb{ACU}$ which has the same conclusion as $\pi$.

This is because if there is a derivation $\pi_1$ in $\mathcal{C}/\mathbb{ACU}$ which uses free push clauses $N$ times, then we can choose a minimal sub-derivation $\pi_{sub}$ of $\pi_1$ which uses a free push clause at the root but nowhere else. Using (*) there is a derivation $\pi'_{sub}$ in $\mathcal{C}_{one-way}/\mathbb{ACU}$ which has the same conclusion as $\pi_{sub}$. Then we replace the subderivation $\pi_{sub}$ in $\pi_1$ by the derivation $\pi'_{sub}$ to get a derivation $\pi'_1$ which uses free push clauses $N-1$ times and which has the same conclusion as $\pi_1$. By iterating this procedure $N$ times we get a derivation in $\mathcal{C}_{one-way}$ which has the same conclusion as $\pi_1$.

Now we proceed to prove (*). So we assume a derivation $\pi$ in $\mathcal{C}/\mathbb{ACU}$ of $R(t_i)$ which uses the free push clause $R(x_i) \Leftarrow P(f(x_1, \ldots, x_n)) \wedge R_1^1(x_{i_1}) \wedge \ldots \wedge R_1^{n_1}(x_{i_1}) \wedge \ldots \wedge R_k^1(x_{i_k}) \wedge \ldots \wedge R_k^{n_k}(x_{i_k})$ as the last clause, and does not use a free push clause anywhere else. Hence the atoms $P(f(t_1, \ldots, t_n)), R_1^1(t_{i_1}), \ldots, R_1^{n_1}(t_{i_1}), \ldots, R_k^1(t_{i_k}), \ldots, R_k^{n_k}(t_{i_k})$ are derivable in $\mathcal{C}_{one-way}/\mathbb{ACU}$. The derivation (call it $\pi_2$) of $P(f(t_1, \ldots, t_n))$ has the functional support $Q(f(t_1, \ldots, t_n))$ for some $Q$, and we have

$$\vdots$$
$$\pi_2 = \cfrac{Q(f(t_1, \ldots .t_n))}{P(f(t_1, \ldots, t_n))} \; (\mathcal{C}_{eq}/\mathbb{ACU}) \tag{10.2}$$

Define automaton $\mathcal{D} = \mathcal{C}_{eq} \cup \{R(a_R) \mid R \in \mathbb{P}\}$. The atom $Q(a_Q)$ is derivable in $\mathcal{D}/\mathbb{ACU}$. . Since $\mathcal{C}_{eq} = \mathcal{D}_{eq}$, hence from (10.2) and Lemma 21, $P(a_Q)$ is derivable in $\mathcal{D}/\mathbb{ACU}$. Now the derivation of $Q(f(t_1, \ldots, t_n))$ in $\mathcal{C}_{one-way}$ uses some clause $Q(f(x_1, \ldots, x_n)) \Leftarrow Q_1(x_1) \wedge \ldots \wedge Q_n(x_n)$ as the last clause, and the atoms $Q_1(t_1), \ldots, Q_n(t_n)$ are derivable in $\mathcal{C}_{one-way}/\mathbb{ACU}$. In the base step of the saturation procedure described above, by replacing the automata $\mathcal{A}$ and $\mathcal{B}$ by the automata $\mathcal{C}$ and $\mathcal{D}$ we have that $\mathcal{C} \rhd \mathcal{C} \cup \{R(x_i) \Leftarrow Q_i(x_i)\}$. But $\mathcal{C}$ is already saturated, hence $R(x_i) \Leftarrow Q_i(x_i) \in \mathcal{C}$. Also $Q_i(t_i)$ is derivable in $\mathcal{C}_{one-way}/\mathbb{ACU}$. Hence $R(t_i)$ is derivable in $\mathcal{C}_{one-way}/\mathbb{ACU}$. □

The converse is trivially true. Hence $\mathcal{C}$ has the same set of derivable atoms as $\mathcal{C}_{one-way}$. Also we have already seen that $\mathcal{C}$ has the same set of derivable atoms as $\mathcal{A}$. Hence by letting the final state of the automaton $\mathcal{C}_{one-way}$ to be the same as the final state of the automaton $\mathcal{A}$, we have

**Theorem 32** *A two-way $\mathbb{ACU}$ automaton can be effectively converted to a one-way $\mathbb{ACU}$ automaton accepting the same language.*

As we have shown that one-way $\mathbb{ACU}$ automata are effectively closed under intersection and complementation, we have :

**Corollary 3** *Two-Way $\mathbb{ACU}$ automata are effectively closed under intersection and complementation.*

### 10.1.1 Two-Way $\mathbb{AC}$ Automata

As in the case of intersection and complementation of one-way $\mathbb{ACU}$ tree automata, observe that the presence of the unit symbol 0 is not at all crucial for the above proof. The same proof with minor modifications works for the $\mathbb{AC}$ case. We merely state the result without repeating the proof :

**Theorem 33** *A two-way $\mathbb{ACU}$ automaton can be effectively converted to a one-way $\mathbb{ACU}$ automaton accepting the same language. In particular they are effectively closed under intersection and complementation.*

## 10.2 Two-Way $\mathbb{ACUX}$ and $\mathbb{ACUX}_n$ Automata

We show in this section that two-way $\mathbb{ACUX}$ (resp. $\mathbb{ACUX}_n$) automata can be effectively reduced to one-way $\mathbb{ACUX}$ (resp. $\mathbb{ACUX}_n$) automata.

Consider a two-way $\mathbb{ACUX}$ automaton $\mathcal{A}$ with predicates from $\mathbb{P}$. As in the $\mathbb{ACU}$ case, to convert it to a one-way $\mathbb{ACUX}$ 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 $\mathcal{A}_{eq}$ might have been used in between to add new terms, which eventually get canceled using $\mathbb{C}$ 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 $S_1 = \{a_P \mid P \in \mathbb{P}\}$ 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 $\mathcal{B} = \mathcal{A}_{eq} \cup \{P(a_P) \mid P \in \mathbb{P}\} \cup \{P(b_{P,Q}) \mid P, Q \in \mathbb{P}\}$.

Then $\mathcal{L}_P(\mathcal{B}/\mathbb{ACU})$ is a semilinear set for every $P$. For $P, Q \in \mathbb{P}$ and $S \subseteq S_2$, define $\mathcal{L}_{P,Q,S,\mathcal{A}}$ to be the set of $t \in \mathcal{L}_P(\mathcal{B}/\mathbb{ACU})$ such that

1. $a_Q$ occurs in $t$ exactly once
2. $\forall Q' \neq Q \cdot a_{Q'}$ does not occur in $t$
3. each constant in $S$ occurs in $t$ a positive and even number of times
4. no constant from $S_2 \setminus S$ occurs in $t$

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

If $\mathcal{A}$ has a free push clause

$$R(x_i) \Leftarrow P(f(x_1, \ldots, x_n)) \wedge R_1^1(x_{i_1}) \wedge \ldots \wedge R_1^{n_1}(x_{i_1}) \wedge \ldots \wedge R_k^1(x_{i_k}) \wedge \ldots \wedge R_k^{n_k}(x_{i_k})$$

, a free pop clause

$$Q(f(x_1, \ldots, x_n)) \Leftarrow Q_1(x_1) \wedge \ldots \wedge Q_n(x_n)$$

and there is a set $S \subseteq S_2$ such that

1. $\mathcal{L}_{P,Q,S,\mathcal{A}} \neq \emptyset$
2. $\forall b_{Q',Q''} \in S \cdot \exists t \cdot$ both $Q'$ and $Q''$ accept $t$ in $\mathcal{A}_{one-way}/\mathbb{ACUX}$
3. $\forall j \in \{1, \ldots, k\} \cdot \exists t \cdot$ each of the predicates $Q_{i_j}, R_j^1, \ldots, R_j^{n_j}$ accepts $t$ in $\mathcal{A}_{one-way}/\mathbb{ACUX}$
4. $\forall j \in \{1, \ldots, n\} \setminus \{i, i_1, \ldots, i_k\} \cdot \exists t \cdot Q_j$ accepts $t$ in $\mathcal{A}_{one-way}/\mathbb{ACUX}$

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

**Lemma 48** *Let $\mathcal{A} \rhd \mathcal{A} \cup \{R(x_i) \Leftarrow Q_i(x_i)\}$ as above. Then any atom derivable in $\mathcal{A} \cup \{R(x_i) \Leftarrow Q_i(x_i)\}/\mathbb{ACUX}$ is also derivable in $\mathcal{A}/\mathbb{ACUX}$.*

**Proof:** As in the $\mathbb{ACU}$ case, it is sufficient to show that for any $t_i$, if $Q_i(t_i)$ is derivable in $\mathcal{A}/\mathbb{ACUX}$, then $R(t_i)$ is derivable in $\mathcal{A}/\mathbb{ACUX}$. As in the definition above, let $t_{i_j}$ be the term accepted at the predicates $Q_{i_j}, R_j^1, \ldots, R_j^{n_j}$ in $\mathcal{A}/\mathbb{ACUX}$ for $j \in \{1, \ldots, k\}$. Also let $t_j$ be the term recognized at $Q_j$ in $\mathcal{A}/\mathbb{ACUX}$ for $j \in \{1, \ldots, n\} \setminus \{i, i_1, \ldots, i_k\}$. $Q(f(t_1, \ldots, t_n))$ is derivable in $\mathcal{A}/\mathbb{ACUX}$ using the pop clause. As $\mathcal{L}_{P,Q,S,\mathcal{A}} \neq \emptyset$, there is an atom of the form $P(a_Q + 2b_{Q'_1,Q''_1} + \ldots + 2b_{Q'_p,Q''_p})$ derivable in $\mathcal{B}/\mathbb{ACU}$, with $p \geq 0$ and $b_{Q'_i,Q''_i} \in S$ for $1 \leq i \leq p$. Let $u_i$ be the term accepted at $Q'_i$ and $Q''_i$ in $\mathcal{A}_{one-way}/\mathbb{ACUX}$. The derivation (call it $\pi$) of $P(a_Q + 2b_{Q'_1,Q''_1} + \ldots + 2b_{Q'_p,Q''_p})$ has a functional support of the form $Q(a_Q), Q_1^\dagger(b_{Q'_1,Q''_1}), Q_1^\ddagger(b_{Q'_1,Q''_1}), \ldots, Q_p^\dagger(b_{Q'_p,Q''_p}), Q_p^\ddagger(b_{Q'_p,Q''_p})$ where $Q_i^\dagger, Q_i^\ddagger \in \{Q'_i, Q''_i\}$. We have

$$\pi = \frac{\overline{Q(a_Q)} \quad \overline{Q_1^\dagger(b_{Q'_1,Q''_1})} \quad \overline{Q_1^\ddagger(b_{Q'_1,Q''_1})} \quad \ldots \quad \overline{Q_p^\dagger(b_{Q'_p,Q''_p})} \quad \overline{Q_p^\ddagger(b_{Q'_p,Q''_p})}}{P(a_Q + 2b_{Q'_1,Q''_1} + \ldots + 2b_{Q'_p,Q''_p})} \, (\mathcal{B}/\mathbb{ACU}) \tag{10.3}$$

Also by definition $\mathcal{B}_{eq} = \mathcal{A}_{eq}$. Hence using (10.3) and Lemma 21, and the fact that the atoms $Q(f(t_1, \ldots, t_n)), Q'_1(u_1), Q''_i(u_1), \ldots, Q'_p(u_p), Q''_p(u_p)$ are derivable in $\mathcal{A}/\mathbb{ACUX}$, we get a derivation of $P(f(t_1, \ldots, t_n) + 2u_1 + \ldots + 2u_p)$ in $\mathcal{A}/\mathbb{ACUX}$. Hence $P(f(t_1, \ldots, t_n))$ is derivable in $\mathcal{A}/\mathbb{ACUX}$. Finally we use the free push clause to get a derivation of $R(t_i)$ in $\mathcal{A}/\mathbb{ACUX}$. □

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

Given a two-way $\mathbb{ACUX}$ automaton $\mathcal{A}$ our saturation procedure consists of (don't care non-deterministically) generating a sequence $\mathcal{A}_0(= \mathcal{A}) \rhd \mathcal{A}_1 \rhd \mathcal{A}_2$... 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 $\mathcal{C}$. Then we remove the free push clauses from $\mathcal{C}$ to get a one-way automaton $\mathcal{C}_{one-way}$. This step is also harmless :

**Lemma 49** *If any atom is derivable in $\mathcal{C}/\mathbb{ACUX}$, then it is derivable in $\mathcal{C}_{one-way}/\mathbb{ACUX}$.*

**Proof:** As in the $\mathbb{ACU}$ case, it is sufficient to show that for every derivation in $\mathcal{C}/\mathbb{ACUX}$, which uses a free push clause at the root and nowhere else, there is a derivation in $\mathcal{C}_{one-way}/\mathbb{ACUX}$ with the same conclusion. So we assume a derivation $\pi$ in $\mathcal{C}/\mathbb{ACUX}$ of atom $R(t_i)$ which uses a push clause $R(x_i) \Leftarrow P(f(x_1, \ldots, x_n)) \wedge R_1^1(x_{i_1}) \wedge \ldots \wedge R_1^{n_1}(x_{i_1}) \wedge \ldots \wedge R_k^1(x_{i_k}) \wedge \ldots \wedge R_k^{n_k}(x_{i_k})$ as the last clause, and does not use a free push clause anywhere else. Hence the atoms $P(f(t_1, \ldots, t_n))$, $R_1^1(t_{i_1}), \ldots, R_1^{n_1}(t_{i_1}), \ldots, R_k^1(t_{i_k}), \ldots, R_k^{n_k}(t_{i_k})$ are derivable in $\mathcal{C}_{one-way}/\mathbb{ACUX}$. By Lemma 19, there is some $t =_{\mathbb{ACUX}} f(t_1, \ldots, t_n)$ such that $P(t)$ is derivable in $\mathcal{C}_{one-way}/\mathbb{ACU}$. We have $t =_{\mathbb{ACU}} f(t'_1, \ldots, t'_n) + u_1 + v_1 + \ldots + u_p + v_p$ for some $p \geq 0$ such that for $1 \leq i \leq n$, $t_i =_{\mathbb{ACUX}} t'_i$ and for $1 \leq i \leq p$, $u_i$ and $v_i$ are functional and $u_i =_{\mathbb{ACUX}} v_i$. The derivation (call it $\pi$) of $P(t)$ has a functional support of the form $Q(f(t'_1, \ldots, t'_n)), I_1(u_1), J_1(v_1), \ldots, I_p(u_p), J_p(v_p)$, and we have

$$\pi = \dfrac{\begin{matrix} \vdots & \vdots & \vdots & & \vdots & \vdots \\ Q(f(t'_1, \ldots, t'_n)) & I_1(u_1) & J_1(v_1) & \ldots & I_p(u_p) & J_p(v_p) \end{matrix}}{P(t)} \; (\mathcal{C}_{eq}/\mathbb{ACU}) \tag{10.4}$$

Define automaton $\mathcal{D} = \mathcal{C} \cup \{P(b_{P,Q}) \mid P, Q \in \mathbb{P}\}$. The atom $Q(a_Q)$ as well as the atoms $I_i(b_{I_i,J_i})$ and $J_i(b_{I_i,J_i})$ for $1 \leq i \leq p$ are derivable in $\mathcal{D}/\mathbb{ACU}$. Also $\mathcal{D}_{eq} = \mathcal{C}_{eq}$. Hence from (10.4) and Lemma 21, $P(a_Q + 2b_{I_1,J_1} + \ldots + 2b_{I_p,J_p})$ is derivable in $\mathcal{D}/\mathbb{ACU}$. Let $S = \{(I_1, J_1), \ldots, (I_p, J_p)\}$. Then clearly $\mathcal{L}_{P,Q,S,\mathcal{C}} \neq \emptyset$. The derivation of $Q(f(t'_1, \ldots, t'_n))$ uses some clause $Q(f(x_1, \ldots, x_n)) \Leftarrow Q_1(x_1) \wedge \ldots \wedge Q_n(x_n)$ as the last clause, and the atoms $Q_1(t'_1), \ldots, Q_n(t'_n)$ are derivable in $\mathcal{C}_{one-way}/\mathbb{ACU}$. For $1 \leq i \leq n$, as $t'_i =_{\mathbb{ACUX}} t_i$, hence $Q_i(t_i)$ is derivable in $\mathcal{C}_{one-way}/\mathbb{ACUX}$. By replacing the automata $\mathcal{A}$ and $\mathcal{B}$ in the base step of the saturation procedure above by the automata $\mathcal{C}$ and $\mathcal{D}$, we have that $\mathcal{C} \rhd \mathcal{C} \cup \{R(x_i) \Leftarrow Q_i(x_i)\}$. But $\mathcal{C}$ is already saturated, hence $R(x_i) \Leftarrow Q_i(x_i) \in \mathcal{C}$. Also $Q_i(t_i)$ is derivable in $\mathcal{C}_{one-way}/\mathbb{ACUX}$. Hence $R(t_i)$ is derivable in $\mathcal{C}_{one-way}/\mathbb{ACUX}$. $\square$

The converse is trivially true. Then as in the $\mathbb{ACU}$ case, we have

**Theorem 34** *A two-way $\mathbb{ACUX}$ automaton can be effectively converted to a one-way $\mathbb{ACUX}$ automaton accepting the same language.*

### 10.2.1 Generalization to the $\mathbb{ACUX}_n$ Case

As for the results on the closure under intersection of one-way automata, the above results on two-way $\mathbb{ACUX}$ automata generalize easily to two-way $\mathbb{ACUX}_n$ automata. Again the only difference from the $\mathbb{ACUX}$ case is that in derivations, $n$-tuples of equal terms cancel together instead of pairs of equal terms. We give the saturation procedure without repeating the proofs.

Consider a two-way $\mathbb{ACUX}_n$ automaton $\mathcal{A}$ with predicates from $\mathbb{P}$. In the construction 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 $S_1 = \{a_P \mid P \in \mathbb{P}\}$ and $S_2 = \{b_{P_1,\ldots,P_n} \mid P_1,\ldots,P_n \in \mathbb{P}\}$. The order of the predicates $P_1,\ldots,P_n$ in $b_{P_1,\ldots,P_n}$ is ignored. We define $\mathcal{B} = \mathcal{A}_{eq} \cup \{P(a_P) \mid P \in \mathbb{P}\} \cup \{P_1(b_{P_1,\ldots,P_n}) \mid P_1,\ldots,P_n \in \mathbb{P}\}$.

Then $\mathcal{L}_P(\mathcal{B}/\mathbb{ACU})$ is a semilinear set for every $P$. For $P, Q \in \mathbb{P}$ and $S \subseteq S_2$, define $\mathcal{L}_{P,Q,S,\mathcal{A}}$ to be the set of $t \in \mathcal{L}_P(\mathcal{B}/\mathbb{ACU})$ such that

1. $a_Q$ occurs in $t$ exactly once
2. $\forall Q' \neq Q \cdot a_{Q'}$ does not occur in $t$
3. each constant in $S$ occurs in $t$ a $kn$ number of times for some $k \geq 1$
4. no constant from $S_2 \setminus S$ occurs in $t$

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

If $\mathcal{A}$ has a free push clause

$$R(x_i) \Leftarrow P(f(x_1,\ldots,x_n)) \wedge R_1^1(x_{i_1}) \wedge \ldots \wedge R_1^{n_1}(x_{i_1}) \wedge \ldots \wedge R_k^1(x_{i_k}) \wedge \ldots \wedge R_k^{n_k}(x_{i_k})$$

a free pop clause

$$Q(f(x_1,\ldots,x_n)) \Leftarrow Q_1(x_1) \wedge \ldots \wedge Q_n(x_n)$$

and there is a set $S \subseteq S_2$ such that

1. $\mathcal{L}_{P,Q,S,\mathcal{A}} \neq \emptyset$
2. $\forall b_{Q'_1,\ldots,Q'_n} \in S \cdot \exists t \cdot$ each of the predicates $Q'_1,\ldots,Q'_n$ accept $t$ in $\mathcal{A}_{one-way}/\mathbb{ACUX}_n$
3. $\forall j \in \{1,\ldots,k\} \cdot \exists t \cdot$ each of the predicates $Q_{i_j}, R_j^1,\ldots,R_j^{n_j}$ accepts $t$ in $\mathcal{A}_{one-way}/\mathbb{ACUX}_n$
4. $\forall j \in \{1,\ldots,n\} \setminus \{i, i_1,\ldots,i_k\} \cdot \exists t \cdot Q_j$ accepts $t$ in $\mathcal{A}_{one-way}/\mathbb{ACUX}_n$

then we will write $\mathcal{A} \rhd \mathcal{A} \cup \{R(x_i) \Leftarrow Q_i(x_i)\}$, which we take to constitute one step of our saturation procedure. This can be effectively decided because of the fact that one-way $\mathbb{ACUX}_n$-automata are closed under intersection and hence their intersection emptiness is decidable. The rest works as in the $\mathbb{ACUX}$ case. We merely state the final result :

**Theorem 35** *A Two-way $\mathbb{ACUX}_n$ automaton can be effectively converted to one-way $\mathbb{ACUX}_n$ automaton accepting the same language.*

## 10.3 Two-Way $\mathbb{ACUD}$ Automata

In this section we show that two-way $\mathbb{ACUD}$ automata can be effectively reduced to one-way $\mathbb{ACUD}$ automata. We fix a signature $\Sigma$ containing at least the symbols $+, -$ and $0$. Consider a two-way $\mathbb{ACUD}$ automaton $\mathcal{A}$ with predicates from $\mathbb{P}$. We describe the base step of our saturation procedure that adds an epsilon clause. The new sets of constants used are $S = \{a_P \mid P \in \mathbb{P}\}$ and $\overline{S} = \{\overline{a_P} \mid P \in \mathbb{P}\}$. We define the automaton $\mathcal{B} = \mathcal{A}_{eq} \cup \{P(a_P) \mid P \in \mathbb{P}\}$. From Theorem 18, for every $P \in \mathbb{P}$, $\mathcal{L}_P(\mathcal{B}/\mathbb{ACUD})$ is a semilinear set on the constants from $S \cup \overline{S}$. In particular we can effectively check whether $a_Q \in \mathcal{L}_P(\mathcal{B}/\mathbb{ACUD})$ for $P, Q \in \mathbb{P}$.

If $\mathcal{A}$ contains a free push clause

$$R(x_i) \Leftarrow P(f(x_1,\ldots,x_n)) \wedge R_1^1(x_{i_1}) \wedge \ldots \wedge R_1^{n_1}(x_{i_1}) \wedge \ldots \wedge R_k^1(x_{i_k}) \wedge \ldots \wedge R_k^{n_k}(x_{i_k})$$

a free pop clause

$$Q(f(x_1, \ldots, x_n)) \Leftarrow Q_1(x_1), \ldots, Q_n(x_n)$$

and a set $S \subseteq S_2$ such that

1. $a_Q \in \mathcal{L}_P(\mathcal{B}/\mathbb{ACUD})$
2. $\forall j \in \{1, \ldots, k\} \cdot \exists t \cdot$ each of the predicates $Q_{i_j}, R_j^1, \ldots, R_j^{n_j}$ accepts $t$ in $\mathcal{A}_{one-way}/\mathbb{ACUD}$
3. $\forall j \in \{1, \ldots, n\} \setminus \{i, i_1, ..., i_k\} \cdot \exists t \cdot Q_j$ accepts $t$ in $\mathcal{A}_{one-way}/\mathbb{ACUD}$

then we write $\mathcal{A} \rhd \mathcal{A} \cup \{R(x_i) \Leftarrow Q_i(x_i)\}$, which we take to constitute the base step of our saturation procedure.

The base step of our saturation procedure is harmless :

**Lemma 50** *Let $\mathcal{A} \rhd \mathcal{A} \cup \{R(x_i) \Leftarrow Q_i(x_i)\}$ as above. Then any atom derivable in $\mathcal{A} \cup \{R(x_i) \Leftarrow Q_i(x_i)\}/\mathbb{ACUD}$ is also derivable in $\mathcal{A}/\mathbb{ACUD}$.*

**Proof:** As for the previous theories it is sufficient to show that for any $t_i$, if $Q_i(t_i)$ is derivable in $\mathcal{A}/\mathbb{ACUD}$, then $R(t_i)$ is derivable in $\mathcal{A}/\mathbb{ACUD}$. As in the definition above, let $t_{i_j}$ be some term accepted in $\mathcal{A}/\mathbb{ACUD}$ at the predicates $Q_{i_j}, R_j^1, \ldots, R_j^{n_j}$ for $j \in \{1, \ldots, k\}$. Also let $t_j$ be some term accepted in $\mathcal{A}/\mathbb{ACUD}$ at $Q_j$ for $j \in \{1, \ldots, n\} \setminus \{i, i_1, \ldots, i_k\}$. $Q(f(t_1, \ldots, t_n))$ is derivable in $\mathcal{A}/\mathbb{ACUD}$ using the free pop clause. As $a_Q \in \mathcal{L}_P(\mathcal{A}/\mathbb{ACUD})$, the derivation (call it $\pi_1$) of $P(a_Q)$ in $\mathcal{B}/\mathbb{ACUD}$ has the functional support $Q(a_Q)$ and we have

$$\pi_1 = \frac{\overline{Q(a_Q)}}{P(a_Q)}\ (\mathcal{B}_{eq}/\mathbb{ACUD}) \tag{10.5}$$

Also by definition $\mathcal{B}_{eq} = \mathcal{A}_{eq}$. Hence from the fact that $Q(f(t_1, \ldots, t_n))$ is derivable in $\mathcal{A}/\mathbb{ACUD}$, and using (10.5) and from Lemma 23, we get a derivation of $P(f(t_1, \ldots, t_n))$ in $\mathcal{A}/\mathbb{ACUD}$. Using the free push clause we get a derivation of $R(t_i)$ in $\mathcal{A}/\mathbb{ACUD}$. This proves (*). □

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

Given a two-way $\mathbb{ACUD}$ automaton $\mathcal{A}$ our saturation procedure consists of (don't care non deterministically) generating a sequence $\mathcal{A}_0(= \mathcal{A}) \rhd \mathcal{A}_1 \rhd \mathcal{A}_2$... of automata, until no new clauses can be added. This procedure always terminates because there are only a finite number of epsilon clauses possible. (If the number of predicates in $\mathbb{P}$ is $n$ then the number of epsilon clauses possible is $n^2$.) Let the final (saturated) automaton be $\mathcal{C}$. Then it is clear that $\mathcal{A}$ and $\mathcal{C}$ have the same set of derivable atoms. Then we remove clauses (3.13) from $\mathcal{C}$ to get a one-way automaton $\mathcal{C}_{one-way}$. This last step is also harmless :

**Lemma 51** *If any atom is derivable in $\mathcal{C}/\mathbb{ACUD}$, then it is derivable in $\mathcal{C}_{one-way}/\mathbb{ACUD}$.*

**Proof:** As for the previous theories it is sufficient to show that for every derivation in $\mathcal{C}/\mathbb{ACUD}$ which uses a free push clause at the root but nowhere else, there is a derivation in $\mathcal{C}_{one-way}/\mathbb{ACUD}$ with the same conclusion. So we assume a derivation $\pi$ in $\mathcal{C}/\mathbb{ACUD}$ of atom $R(t_i)$ which uses the free push clause $R(x_i) \Leftarrow P(f(x_1, \ldots, x_n)) \wedge R_1^1(x_{i_1}) \wedge \ldots \wedge R_1^{n_1}(x_{i_1}) \wedge \ldots \wedge R_k^1(x_{i_k}) \wedge \ldots \wedge R_k^{n_k}(x_{i_k})$ as the last clause, and does not use a free push clause anywhere else. Hence the atoms $P(f(t_1, \ldots, t_n))$, $R_1^1(t_{i_1}), \ldots, R_1^{n_1}(t_{i_1}), \ldots, R_k^1(t_{i_k}), \ldots, R_k^{n_k}(t_{i_k})$ are derivable in $\mathcal{C}_{one-way}/\mathbb{ACUD}$. The derivation (call it $\pi_2$) of $P(f(t_1, \ldots, t_n))$ has the functional support $Q(f(t_1, \ldots, t_n))$ for some $Q$, and we have

$$\pi_2 = \cfrac{\vdots}{\cfrac{Q(f(t_1, \ldots .t_n))}{P(f(t_1, \ldots, t_n))}} \; (\mathcal{C}_{eq}/\mathbb{ACUD}) \tag{10.6}$$

Define automaton $\mathcal{D} = \mathcal{C}_{eq} \cup \{R(a_R) \mid R \in \mathbb{P}\}$. The atom $Q(a_Q)$ is derivable in $\mathcal{D}/\mathbb{ACUD}$. Since $\mathcal{C}_{eq} = \mathcal{D}_{eq}$, hence from (10.2) and Lemma 21, $P(a_Q)$ is derivable in $\mathcal{D}/\mathbb{ACUD}$. Now the derivation of $Q(f(t_1, \ldots, t_n))$ in $\mathcal{C}_{one-way}/\mathbb{ACUD}$ uses some clause $Q(f(x_1, \ldots, x_n)) \Leftarrow Q_1(x_1) \wedge \ldots \wedge Q_n(x_n)$ as the last clause, and the atoms $Q_1(t_1), \ldots, Q_n(t_n)$ are derivable in $\mathcal{C}_{one-way}/\mathbb{ACUD}$. In the base step of the saturation procedure described above, by replacing the automata $\mathcal{A}$ and $\mathcal{B}$ by the automata $\mathcal{C}$ and $\mathcal{D}$ we have that $\mathcal{C} \rhd \mathcal{C} \cup \{R(x_i) \Leftarrow Q_i(x_i)\}$. But $\mathcal{C}$ is already saturated, hence $R(x_i) \Leftarrow Q_i(x_i) \in \mathcal{C}$. Also $Q_i(t_i)$ is derivable in $\mathcal{C}_{one-way}/\mathbb{ACUD}$. Hence $R(t_i)$ is derivable in $\mathcal{C}_{one-way}/\mathbb{ACUD}$. □

**Theorem 36** *Two-way $\mathbb{ACUD}$ automata can be effectively converted to one-way $\mathbb{ACUD}$ automata.*

## 10.4 Two-Way $\mathbb{ACUM}$ Automata

In this section we show that two-way $\mathbb{ACUM}$ automata can be effectively reduced to one-way $\mathbb{ACUM}$ automata. Fix a signature $\Sigma$ containing at least the symbols $+, -$ and $0$. Consider a two-way $\mathbb{ACUM}$ automaton $\mathcal{A}$ on signature $\Sigma$ with predicates in $\mathbb{P}$. 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 $\mathcal{B} = \mathcal{A}_{eq} \cup \{P(a_P) \mid P \in \mathbb{P}\} \cup \{P(b_{P,Q}) \mid P, Q \in \mathbb{P}\}$. Then from Theorem 18, for every $P$, $\mathcal{L}_P(\mathcal{B}/\mathbb{ACUD})$ 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 $\mathcal{L}_{P,Q,S,\mathcal{A}}$ to be the set of $t \in \mathcal{L}_P(\mathcal{B}/\mathbb{ACUD})$ such that

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

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

If $\mathcal{A}$ has a free push clause

$$R(x_i) \Leftarrow P(f(x_1, \ldots, x_n)) \wedge R_1^1(x_{i_1}) \wedge \ldots \wedge R_1^{n_1}(x_{i_1}) \wedge \ldots \wedge R_k^1(x_{i_k}) \wedge \ldots \wedge R_k^{n_k}(x_{i_k})$$

a pop clause

$$Q(f(x_1, \ldots, x_n)) \Leftarrow Q_1(x_1), \ldots, Q_n(x_n)$$

and a set $S \subseteq S_2$ such that

1. $\mathcal{L}_{P,Q,S,\mathcal{A}} \neq \emptyset$
2. $\forall b_{Q',Q''} \in S \cdot \exists t \cdot$ both $Q'$ and $Q''$ accept $t$ in $\mathcal{A}_{one-way}/\mathbb{ACUM}$
3. $\forall j \in \{1, \ldots, k\} \cdot \exists t \cdot$ each of the predicates $Q_{i_j}, R_j^1, \ldots, R_j^{n_j}$ accepts $t$ in $\mathcal{A}_{one-way}/\mathbb{ACUM}$
4. $\forall j \in \{1, \ldots, n\} \setminus \{i, i_1, \ldots, i_k\} \cdot \exists t \cdot Q_j$ accepts $t$ in $\mathcal{A}_{one-way}/\mathbb{ACUM}$

then we write $\mathcal{A} \rhd \mathcal{A} \cup \{R(x_i) \Leftarrow Q_i(x_i)\}$, which is one step of our saturation procedure.

**Lemma 52** *Let $\mathcal{A} \rhd \mathcal{A} \cup \{R(x_i) \Leftarrow Q_i(x_i)\}$ as above. Then any atom derivable in $\mathcal{A} \cup \{R(x_i) \Leftarrow Q_i(x_i)\}/\mathbb{ACUM}$ is also derivable in $\mathcal{A}/\mathbb{ACUM}$.*

**Proof:** As for the previous theories, it is sufficient to show that for any $t_i$, if $Q_i(t_i)$ is derivable in $\mathcal{A}/\mathbb{ACUM}$, then $R(t_i)$ is derivable in $\mathcal{A}/\mathbb{ACUM}$. As in the definition above, let $t_{i_j}$ be the term accepted at the predicates $Q_{i_j}, R_j^1, \ldots, R_j^{n_j}$ in $\mathcal{A}/\mathbb{ACUM}$ for $j \in \{1, \ldots, k\}$. Also let $t_j$ be the term recognized at $Q_j$ in $\mathcal{A}/\mathbb{ACUM}$ for $j \in \{1, \ldots, n\} \setminus \{i, i_1, \ldots, i_k\}$. $Q(f(t_1, \ldots, t_n))$ is derivable in $\mathcal{A}/\mathbb{ACUM}$ using the pop clause. As $\mathcal{L}_{P,Q,S,\mathcal{A}} \neq \emptyset$, there is an atom of the form $P(a_Q + (b_{Q_1',Q_1''} - b_{Q_1',Q_1''}) + \ldots + (b_{Q_p',Q_p''} - b_{Q_p',Q_p''}))$ derivable in $\mathcal{B}/\mathbb{ACUD}$, with $p \geq 0$ and $b_{Q_i',Q_i''} \in S$ for $1 \leq i \leq p$. Let $u_i$ be the term accepted at $Q_i'$ and $Q_i''$ in $\mathcal{A}_{one-way}/\mathbb{ACUM}$. The derivation (call it $\pi$) of $P(a_Q + (b_{Q_1',Q_1''} - b_{Q_1',Q_1''}) + \ldots + (b_{Q_p',Q_p''} - b_{Q_p',Q_p''}))$ has a functional support of the form $Q(a_Q), Q_1^\dagger(b_{Q_1',Q_1''}), Q_1^\ddagger(b_{Q_1',Q_1''}), \ldots, Q_p^\dagger(b_{Q_p',Q_p''}), Q_p^\ddagger(b_{Q_p',Q_p''})$ where $Q_i^\dagger, Q_i^\ddagger \in \{Q_i', Q_i''\}$. We have

$$\pi = \frac{\overline{Q(a_Q)} \quad \overline{Q_1^\dagger(b_{Q_1',Q_1''})} \quad \overline{Q_1^\ddagger(b_{Q_1',Q_1''})} \quad \ldots \quad \overline{Q_p^\dagger(b_{Q_p',Q_p''})} \quad \overline{Q_p^\ddagger(b_{Q_p',Q_p''})}}{P(a_Q + (b_{Q_1',Q_1''} - b_{Q_1',Q_1''}) + \ldots + (b_{Q_p',Q_p''} - b_{Q_p',Q_p''}))} \, (\mathcal{B}/\mathbb{ACU}) \tag{10.7}$$

Also by definition $\mathcal{B}_{eq} = \mathcal{A}_{eq}$. Hence using (10.7) and Lemma 23, and the fact that the atoms $Q(f(t_1, \ldots, t_n)), Q_1'(u_1), Q_i''(u_1), \ldots, Q_p'(u_p), Q_p''(u_p)$ are derivable in $\mathcal{A}/\mathbb{ACUM}$, we get a derivation of $P(f(t_1, \ldots, t_n) + (u_1 - u_1) + \ldots + (u_p - u_p))$ in $\mathcal{A}/\mathbb{ACUM}$. Hence $P(f(t_1, \ldots, t_n))$ is derivable in $\mathcal{A}/\mathbb{ACUM}$. Finally we use the free push clause to get a derivation of $R(t_i)$ in $\mathcal{A}/\mathbb{ACUM}$.
$\square$

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

Given a two-way $\mathbb{ACUM}$ automaton $\mathcal{A}$ our saturation procedure consists of (don't care non-deterministically) generating a sequence $\mathcal{A}_0 (= \mathcal{A}) \rhd \mathcal{A}_1 \rhd \mathcal{A}_2 ...$ 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 $\mathcal{C}$. Then we remove the free push clauses from $\mathcal{C}$ to get a one-way automaton $\mathcal{C}_{one-way}$. This step is also harmless :

**Lemma 53** *If any atom is derivable in $\mathcal{C}/\mathbb{ACUM}$, then it is derivable in $\mathcal{C}_{one-way}/\mathbb{ACUM}$.*

**Proof:** As before, it is sufficient to show that for every derivation in $\mathcal{C}/\mathbb{ACUM}$, which uses a free push clause at the root and nowhere else, there is a derivation in $\mathcal{C}_{one-way}/\mathbb{ACUM}$ with the same conclusion. So we assume a derivation $\pi$ in $\mathcal{C}/\mathbb{ACUM}$ of atom $R(t_i)$ which uses a free push clause $R(x_i) \Leftarrow P(f(x_1, \ldots, x_n)) \wedge R_1^1(x_{i_1}) \wedge \ldots \wedge R_1^{n_1}(x_{i_1}) \wedge \ldots \wedge R_k^1(x_{i_k}) \wedge \ldots \wedge R_k^{n_k}(x_{i_k})$ as the last clause, and does not use a free push clause anywhere else. Hence the atoms $P(f(t_1, \ldots, t_n)), R_1^1(t_{i_1}), \ldots, R_1^{n_1}(t_{i_1}), \ldots, R_k^1(t_{i_k}), \ldots, R_k^{n_k}(t_{i_k})$ are derivable in $\mathcal{C}_{one-way}/\mathbb{ACUM}$. By Lemma 19, there is some $t =_{\mathbb{ACUM}} f(t_1, \ldots, t_n)$ such that $P(t)$ is derivable in $\mathcal{C}_{one-way}/\mathbb{ACUD}$. We have $t =_{\mathbb{ACUD}} f(t_1', \ldots, t_n') + u_1 - v_1 + \ldots + u_p - v_p$ for some $p \geq 0$ such that for $1 \leq i \leq n$, $t_i =_{\mathbb{ACUM}} t_i'$ and for $1 \leq i \leq p$, $u_i$ and $v_i$ are functional and $u_i =_{\mathbb{ACUM}} v_i$. The derivation (call it $\pi$) of $P(t)$ has a functional support of the form $Q(f(t_1', \ldots, t_n')), I_1(u_1), J_1(v_1), \ldots, I_p(u_p), J_p(v_p)$, and we have

$$\pi = \frac{\begin{array}{c}\vdots \\ Q(f(t_1', \ldots, t_n'))\end{array} \quad \begin{array}{c}\vdots \\ I_1(u_1)\end{array} \quad \begin{array}{c}\vdots \\ J_1(v_1)\end{array} \quad \ldots \quad \begin{array}{c}\vdots \\ I_p(u_p)\end{array} \quad \begin{array}{c}\vdots \\ J_p(v_p)\end{array}}{P(t)} \, (\mathcal{C}_{eq}/\mathbb{ACUD}) \tag{10.8}$$

Define automaton $\mathcal{D} = \mathcal{C} \cup \{P(b_{P,Q}) \mid P, Q \in \mathbb{P}\}$. The atom $Q(a_Q)$ as well as the atoms $I_i(b_{I_i,J_i})$ and $J_i(b_{I_i,J_i})$ for $1 \leq i \leq p$ are derivable in $\mathcal{D}/\mathbb{ACUD}$. Also $\mathcal{D}_{eq} = \mathcal{C}_{eq}$. Hence from (10.8) and Lemma 23, $P(a_Q + (b_{I_1,J_1} - b_{I_1,J_1}) + \ldots + (b_{I_p,J_p} - b_{I_p,J_p}))$ is derivable in $\mathcal{D}/\mathbb{ACUD}$. Let $S = \{(I_1, J_1), \ldots, (I_p, J_p)\}$. Then clearly $\mathcal{L}_{P,Q,S,\mathcal{C}} \neq \emptyset$. The derivation of $Q(f(t'_1, \ldots, t'_n))$ uses some clause $Q(f(x_1, \ldots, x_n)) \Leftarrow Q_1(x_1) \wedge \ldots \wedge Q_n(x_n)$ as the last clause, and the atoms $Q_1(t'_1), \ldots, Q_n(t'_n)$ are derivable in $\mathcal{C}_{one-way}/\mathbb{ACUD}$. For $1 \leq i \leq n$, as $t'_i =_{\mathbb{ACUM}} t_i$, hence $Q_i(t_i)$ is derivable in $\mathcal{C}_{one-way}/\mathbb{ACUM}$. By replacing the automata $\mathcal{A}$ and $\mathcal{B}$ in the base step of the saturation procedure above by the automata $\mathcal{C}$ and $\mathcal{D}$, we have that $\mathcal{C} \rhd \mathcal{C} \cup \{R(x_i) \Leftarrow Q_i(x_i)\}$. But $\mathcal{C}$ is already saturated, hence $R(x_i) \Leftarrow Q_i(x_i) \in \mathcal{C}$. Also $Q_i(t_i)$ is derivable in $\mathcal{C}_{one-way}/\mathbb{ACUM}$. Hence $R(t_i)$ is derivable in $\mathcal{C}_{one-way}/\mathbb{ACUM}$. $\square$

The converse is trivially true. Hence as before we have

**Theorem 37** *A two-way $\mathbb{ACUM}$ automaton can be effectively converted to a one-way $\mathbb{ACUM}$ automaton accepting the same language.*

## 10.5 Two-Way $\mathbb{ACUI}$ Automata

While the one-way $\mathbb{ACUI}$ automata are known to be closed under intersection and not close under complementation, 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 $\mathbb{ACUI}$ automata are powerful enough to express alternation : the clause $P(x) \Leftarrow P_1(x) \wedge P_2(x)$ can be translated 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 $Q_1, Q_2, Q$ and unary free symbol $f$. We have already seen that alternation produces undecidability for $\mathbb{ACU}$, $\mathbb{ACUD}$ and $\mathbb{ACUM}$ theories. This suggests that this problem is difficult and might require new techniques.

## 10.6 Conclusion

We have shown that except for the theory $\mathbb{ACUI}$, for all other $\mathbb{AC}$ theories we deal with ($\mathbb{AC}$, $\mathbb{ACU}$, $\mathbb{ACUX}$, $\mathbb{ACUX}_n$, $\mathbb{ACUM}$), the two-way automata are reducible to equivalent one-way automata. The constructions involved saturation procedures which added new epsilon clauses until the free push clauses in the automata become redundant. The $\mathbb{ACUI}$ case has been left open. In particular these automata (except in $\mathbb{ACUI}$ case) have the same decidability and closure properties as the corresponding one-way automata. In particular intersection-emptiness of these two-way automata is decidable, which is what is needed from the point of view of cryptographic protocols. For example, recall that the modeling of the group Diffie Hellman protocol in Chapter 5 is done using two-way $\mathbb{ACU}$ and $\mathbb{ACUX}$ automata. Also the security property was expressed as intersection emptiness of two-way $\mathbb{AC}$ automata. From the results in this Chapter, we know that these properties are decidable.

# Chapitre 11

# VASS étendus et automates avec clauses $+$-push (Extended VASS and Automata with $+$-Push Clauses)

Les clauses push considérées dans les automates bidirectionnels en chapitre 10 ne font apparaître que des symboles de fonction non équationnels (c'est-à-dire ne contenant pas le symbole $+$). Dans ce chapitre nous étudions les automates avec clauses push contenant $+$. Pour les traiter, nous devons d'abord définir une extension des *systèmes d'addition de vecteurs à états* (SAVE, ou VASS en anglais) [Reu89], que nous appelons *VASS étendus (EVASS)*. Nous montrons que les *arbres de Karp-Miller* définis de sorte à calculer les limites de configurations accessibles des VASS peut se généraliser aux EVASS. Grâce à des traductions des automates $\mathbb{AC}$ en EVASS, nous sommes capables de montrer que les automates que l'on obtient en ajoutant des *clauses $+$-push standard* aux automates $\mathbb{AC}$ unidirectionnels et bidirectionnels se réduisent effectivement aux automates $\mathbb{AC}$ unidirectionnels. Cependant comme la construction de Karp et Miller (même pour les VASS) n'est pas primitive récursive, ceci ne nous donne pas un algorithme tractable. À l'opposé nous montrons que dans le cas $\mathbb{ACU}$ (et non $\mathbb{AC}$), il n'est pas besoin de passer par les EVASS, et nous donnons des réductions étonnamment simples des automates obtenus par l'ajout de clauses $+$-push standard aux automates $\mathbb{ACU}$ unidirectionnels et bidirectionnels vers les automates $\mathbb{ACU}$ unidirectionnels. En ce qui concerne les *clauses $+$-push*, qui sont strictement plus expressives que les clauses $+$-push standard, et bien que ces clauses puissent être trivialement éliminées des automates $\mathbb{ACUX}$ et $\mathbb{ACUM}$, le test du vide pour les automates contenant ces clauses modulo $\mathbb{ACU}$ est au moins aussi difficile que le problème de l'accessibilité des VASS ou des réseaux de Petri, qui est un problème bien connu pour être difficile. Nous montrons que le problème du vide de l'intersection pour cette classe d'automates se réduit au problème du vide pour la même classe d'automates, ce qui donne une indication du pouvoir expressif des clauses $+$-push. Ceci suggère que les automates $\mathbb{AC}$ et $\mathbb{ACU}$ avec clauses $+$-push sont difficiles à traiter, et la question de leur décidabilité est aujourd'hui ouverte.

We saw in Chapter 6 that general two-way automata have undecidable emptiness problem for theories $\mathbb{ACU}$, $\mathbb{AC}$, $\mathbb{ACUD}$ and $\mathbb{ACUM}$. This led us to study a restricted class called two-way automata, and we showed that two-way automata can be effectively reduced to one-way automata modulo all our theories except $\mathbb{ACUI}$. The two-way automata are obtained from one-way automata by adding free push clauses of the form

$$P(x_i) \Leftarrow Q(f(x_1, ..., x_n)) \wedge P_1(x_{i_1}) \wedge ... \wedge P_k(x_{i_k}),$$
$$f \text{ being free, } 1 \leq i_1, ..., i_k \leq n, i \in \{1, ..., n\} \setminus \{i_1, ..., i_k\}$$

Note that one of the side-conditions in the above clause is that $f$ is free. In particular we have not studied till now the properties of automata which have push clauses involving the symbol $+$. While the class of automata studied in Chapter 10 are sufficient for modeling cryptographic protocols, as illustrated by the example in Chapter 5, it is natural to ask what properties the automata have in the presence of push clauses involving the $+$-symbol. We introduced two-such clauses in Chapter 3 :

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

which we called $+$-push clause and standard $+$-push clause. In this chapter we study the automata obtained by adding these kinds of clauses. To deal with certain classes of these automata, in particular automata modulo $\mathbb{AC}$ containing standard $+$-push clauses, we need to define and study an extension of the traditional notion of Vector Addition Systems with States (VASS). This extension is obtained by adding a new form of transition rule which makes the runs branching tree-like structures instead of linear ones. We show that the Karp-Miller tree construction for VASS can be extended to deal with the extended VASS, to get extended Karp-Miller trees. This allows us to compute the 'limits' of reachable configurations in an extended VASS. We then use these results on extended VASS to show decidability and closure properties of our automata.

First it is easy to observe that $+$-push clauses are more expressive than standard $+$-push clauses. A standard $+$-push clause $P(x) \Leftarrow Q(x+y)$ can we coded using the clauses $P(x) \Leftarrow Q(x+y) \wedge Q_{all}(y)$ where $Q_{all}$ is a new state, together with a set of clauses which allow $Q_{all}$ to accept all terms built from our signature.

Hence if we know how to deal with $+$-push clauses in our automata, then we know how to deal with standard $+$-push clauses. This is possible at least for two theories, namely $\mathbb{ACUX}$, the theory of xor, and $\mathbb{ACUM}$, the theory of Abelian groups. In the xor case, observe that for any terms $s, t, u$ we have $s + t =_{\mathbb{ACUX}} u \Leftrightarrow s =_{\mathbb{ACUX}} u + t$. As a result, the $+$-push clause $P(x) \Leftarrow P_1(x+y) \wedge P_2(y)$ can be replaced in an $\mathbb{ACUX}$ automaton by the $+$-pop clause $P(x+y) \Leftarrow P_1(x) \wedge P_2(y)$ without changing the language accepted modulo $\mathbb{ACUX}$. Hence

**Observation 12** *Modulo the theory $\mathbb{ACUX}$, an automaton which contains two-way (resp. one-way) $\mathbb{ACUX}$ automata clauses as well as $+$-push clauses, can be converted in linear time to an equivalent two-way (resp. one-way) $\mathbb{ACUX}$ automaton.*

In the case of Abelian groups theory, for any terms $s, t, u$ we have $s+t =_{\mathbb{ACUM}} u \Leftrightarrow s = u+(-t)$. Hence the $+$-push clause $P(x) \Leftarrow P_1(x+y) \wedge P_2(y)$ can be replaced in an $\mathbb{ACUM}$ automaton by the pair of clauses $\overline{P_2}(-x) \Leftarrow P_2(x)$ and $P(x+y) \Leftarrow P_1(x) \wedge \overline{P_2}(y)$, for some fresh predicate $\overline{P_2}$, without changing the language accepted modulo $\mathbb{ACUM}$. Hence

**Observation 13** *Modulo the theory $\mathbb{ACUM}$, an automaton which contains two-way (resp. one-way) $\mathbb{ACUM}$ automata clauses as well as $+$-push clauses, can be converted in linear time to an equivalent two-way (resp. one-way) $\mathbb{ACUM}$ automaton.*

## 11.1 $\mathbb{ACU}$ Automata with Standard $+$-Push Clauses

We showed above that $+$-push clauses (and consequently the standard $+$-push clauses) are redundant in the case of the theories $\mathbb{ACUX}$ and $\mathbb{ACUM}$, and that these clauses can be easily eliminated from our automata. We now come to the theories $\mathbb{AC}$ and $\mathbb{ACU}$ for which the situation is rather different. First in this section we deal with one of the simpler cases : the case of one-way $\mathbb{ACU}$ automata extended by adding standard $+$-push clauses. We show that adding standard $+$-push clauses to one-way $\mathbb{ACU}$ automata does not increase their expressiveness, and that they can be effectively eliminated. Let the signature be $\Sigma$ and let $+, 0 \in \Sigma$ and $- \notin \Sigma$.

First we show that it is easy to decide emptiness of a state in such an automaton. This is done by Algorithm 1 which computes the set of all reachable (or non-empty) states in an automaton $\mathcal{A}$ which contains one-way $\mathbb{ACU}$ automata clauses and standard $+$-push clauses. It works in a way similar to the algorithm for computing the set of reachable states of one-way automata. Recall that the algorithm for computing the set of reachable states in a one-way (equational) automaton works by marking all the reachable states. Initially all the states all are unmarked. If there is a pop clause or an epsilon clause $C$ such that all states in the body of $C$ are already marked, then the algorithm marks state in the head of $C$. Algorithm 1 extends this algorithm by treating standard push clauses just like pop clauses and epsilon clauses, i.e. if the state in the body of a standard $+$-push clause is already marked, then the algorithm marks the state in the head of the clause.

To prove termination of the algorithm, observe that at the end of each iteration of the **Repeat-Until** loop, if $StateAdded = True$, then it means that at least one new state was added in the set $Reachable$. Since there are only finitely many states in $\mathcal{A}$, Algorithm 1 is guaranteed to terminate.

Now we show that each state in the output of Algorithm 1 is reachable in $\mathcal{A}/\mathbb{ACU}$. Clearly if $P(0)$ is a clause then $P$ is reachable. Next we argue that if $P \in Reachable$ after any number of iterations of the **Repeat-Until** loop, then $P$ is reachable in $\mathcal{A}/\mathbb{ACU}$. Consider one iteration of this loop. We know that if a free pop clause $P(f(x_1, \ldots, x_n)) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n) \in \mathcal{A}$ and $P_1, \ldots, P_n$ are reachable in $\mathcal{A}/\mathbb{ACU}$, then $P$ is reachable in $\mathcal{A}/\mathbb{ACU}$. Hence the states added to $Reachable$ using the first **For** loop inside the **Repeat-Until** loop are actually reachable. Similar arguments hold for $+$-pop clauses and epsilon clauses. Now if a standard $+$-push $P(x) \Leftarrow Q(x + y) \in \mathcal{A}$ and $Q$ is reachable, then we have some term $t$ which is accepted at $\mathcal{A}/\mathbb{ACU}$. Then using this clause, $t$ is also accepted at $P$. Hence $P$ is also reachable. Thus the states added to $Reachable$ using the last **For** loop of the **Repeat-Until** loop are reachable. Hence we have shown that each state in the output of Algorithm 1 is reachable.

Algorithm 1 computes all the states that are reachable in $\mathcal{A}/\mathbb{ACU}$ because if there is a derivation $\pi$ in $\mathcal{A}/\mathbb{ACU}$ which leads to some term term being accepted at $P$, then the clause $C$ used by $\pi$ as the last clause has $P$ in the head, and for every $P'$ in the body of $C$, there is a derivation strictly smaller than $\pi$ which leads to some term being accepted at $P'$. Hence using an inductive argument we can show that all states reachable in $\mathcal{A}/\mathbb{ACU}$ are computed by the algorithm. We conclude that

**Lemma 54** *The emptiness of a state in $\mathcal{A}/\mathbb{ACU}$, where $\mathcal{A}$ is an automaton modulo $\mathbb{ACU}$ containing one-way $\mathbb{ACU}$ automata clauses and standard $+$-push clauses is decidable is decidable in polynomial time.*

**Algorithm 1** Reachable States of Automata Modulo $\mathbb{ACU}$ Containing One-Way $\mathbb{ACU}$ Automata Clauses and $+$-Push Clauses

```
# Input : automaton A modulo ACU, on signature Σ, and containing
#        one-way ACU automata clauses and standard +-push clauses
# Output : a set of states
#     Reachable := ∅ ;
#     For each zero clause P(0) ∈ A
#         Reachable := Reachable ∪ {P} ;
#     Repeat
#         StateAdded := False ;
#         For each free pop clause P(f(x_1, ..., x_n)) ⇐ P_1(x_1) ∧ ... ∧ P_n(x_n) ∈ A
#             if {P_1, ..., P_n} ⊆ Reachable and P ∉ Reachable
#             then
#                 Reachable := Reachable ∪ {P} ;
#                 StateAdded := True ;
#         For each +-pop clause P(x + y) ⇐ P_1(x) ∧ P_2(y) ∈ A
#             if P_1, P_2 ∈ Reachable and P ∉ Reachable
#             then
#                 Reachable := Reachable ∪ {P} ;
#                 StateAdded := True ;
#         For each epsilon clause P(x) ⇐ P_1(x) ∈ A
#             if P_1 ∈ Reachable and P ∉ Reachable
#             then
#                 Reachable := Reachable ∪ {P} ;
#                 StateAdded := True ;
#         For each standard +-push clause P(x) ⇐ Q(x + y) ∈ A
#             if Q ∈ Reachable and P ∉ Reachable
#             then
#                 Reachable := Reachable ∪ {P} ;
#                 StateAdded := True ;
#     Until (StateAdded = False) ;
#     Return Reachable ;
```

Note that the key idea in this algorithm which makes it work is that fact that if $P(x) \Leftarrow Q(x+y)$ is a clause, and $Q$ is reachable in $\mathcal{A}/\mathbb{ACU}$ then we can conclude that $P$ is reachable in $\mathcal{A}/\mathbb{ACU}$. This is because we can instantiate the variable $y$ in the above clause to the term $0$. Hence the presence of an unit for the $+$ symbol is crucial for the correctness of this algorithm. Observe that the same algorithm becomes incorrect when the theory we are working with is $\mathbb{AC}$ instead of $\mathbb{ACU}$. Hence the correctness of this algorithm is strongly dependent on the equational theory, as against the usual reachability algorithm for one-way equational tree automata which works for all equational theories. Still we prove later that emptiness is also decidable for one-way $\mathbb{AC}$ automata extended by adding standard $+$-push clauses.

Now we show that one-way $\mathbb{ACU}$ automata extended with standard $+$-push clauses can be reduced to one-way $\mathbb{ACU}$ automaton. Let $\mathcal{A}$ be a one-way $\mathbb{ACU}$ automaton, together with standard $+$-push clauses, on signature $\Sigma$. Let $\mathbb{P}$ be the set of predicates in $\mathcal{A}$. We now construct a one-way $\mathbb{ACU}$

automaton $\mathcal{B}$ (without standard $+$-push clauses) equivalent to $\mathcal{A}$. Because of Lemma 54, we can assume the no state in $P$ in empty in $\mathcal{A}/\mathbb{ACU}$, otherwise we can (effectively) remove that state (an related clauses) from the automaton. (If the final state of $\mathcal{A}$ itself is empty, then we let $\mathcal{B}$ be trivial one-way automaton which has no clauses.) To compute the required one-way automaton we need new predicate symbols $P^\dagger$ for each predicate $P \in \mathbb{P}$. For $s, t \in T(\Sigma)$ define

$$s \leq_+ t \text{ iff } s + u =_{\mathbb{ACU}} t \text{ for some } u \in T(\Sigma)$$

For $P \in \mathbb{P}$, we intend $P$ to accept the same language in both $\mathcal{A}$ and $\mathcal{B}$, whereas we intend $P^\dagger$ to accept all terms $s$ such that $s \leq_+ t$ for some term $t$ accepted at $P$. We define the automaton $\mathcal{B}$ to contain clauses corresponding to clauses of $\mathcal{A}$ as defined by the following table :

| Clause of $\mathcal{A}$ | Clauses of $\mathcal{B}$ |
|---|---|
| $P(0)$ | $P(0)$<br>$P^\dagger(0)$ |
| $P(x) \Leftarrow P_1(x)$ | $P(x) \Leftarrow P_1(x)$<br>$P^\dagger(x) \Leftarrow P_1^\dagger(x)$ |
| $P(x+y) \Leftarrow P_1(x) \wedge P_2(y)$ | $P(x+y) \Leftarrow P_1(x) \wedge P_2(y)$<br>$P^\dagger(x+y) \Leftarrow P_1^\dagger(x) \wedge P_2^\dagger(x)$ |
| $P(f(x_1, \ldots, x_n)) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n)$<br>($f$ is free) | $P(f(x_1, \ldots, x_n)) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n)$<br>$P^\dagger(f(x_1, \ldots, x_n)) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n)$<br>$P^\dagger(0)$ |
| $P(x) \Leftarrow Q(x+y)$ | $P(x) \Leftarrow Q^\dagger(x)$<br>$P^\dagger(x) \Leftarrow Q^\dagger(x)$ |

The relationship between $\mathcal{A}$ and $\mathcal{B}$ are stated by the next two results :

**Lemma 55** *If $P(t)$ is derivable in $\mathcal{A}/\mathbb{ACU}$ then $P(t)$ is derivable in $\mathcal{B}/\mathbb{ACU}$ and for all $s \leq_+ t$, $P^\dagger(s)$ is derivable in $\mathcal{B}/\mathbb{ACU}$.*

**Proof:** We do induction on the size of the derivation $\pi$ of $P(t)$ in $\mathcal{A}/\mathbb{ACU}$. We have the following cases :

(i)

$$\pi = \frac{}{P(t)}\,(P(0)/\mathbb{ACU})$$

Clearly $t =_{\mathbb{ACU}} 0$. We have the derivation

$$\frac{}{P(t)}\,(P(0)/\mathbb{ACU})$$

in $\mathcal{B}/\mathbb{ACU}$. Also if $s \leq_+ t$ then $s =_{\mathbb{ACU}} 0$. We have the derivation

$$\frac{}{P^\dagger(s)}\,(P^\dagger(0)/\mathbb{ACU})$$

in $\mathcal{B}/\mathbb{ACU}$.

(ii)

$$\pi = \frac{\begin{array}{c}\vdots\\ P_1(t_1)\end{array}}{P(t)}\,(P(x) \Leftarrow P_1(x)/\mathbb{ACU})$$

Clearly $t =_{\mathbb{ACU}} t_1$. By induction hypothesis we have a derivation $\pi_1$ of $P_1(t_1)$ in $\mathcal{B}/\mathbb{ACU}$. Hence we have the derivation

$$\cfrac{\cfrac{\pi_1}{P_1(t_1)}}{P(t)}\,(P(x) \Leftarrow P_1(x)/\mathbb{ACU})$$

in $\mathcal{B}/\mathbb{ACU}$. Also if $s \leq_+ t$ then $s \leq_+ t_1$ and by induction hypothesis we have a derivation $\pi_2$ of $P_1^\dagger(s)$ in $\mathcal{B}/\mathbb{ACU}$. Hence we have the derivation

$$\cfrac{\cfrac{\pi_2}{P_1^\dagger(s)}}{P^\dagger(s)}\,(P^\dagger(x) \Leftarrow P_1^\dagger(x)/\mathbb{ACU})$$

in $\mathcal{B}/\mathbb{ACU}$.

(iii)

$$\pi = \cfrac{\begin{matrix}\vdots & & \vdots\\ P_1(t_1) & & P_2(t_2)\end{matrix}}{P(t)}\,(P(x+y) \Leftarrow P_1(x) \wedge P_2(y)/\mathbb{ACU})$$

Clearly $t =_{\mathbb{ACU}} t_1 + t_2$. By induction hypothesis we have derivations $\pi_1$ and $\pi_2$ of $P_1(t_1)$ and $P_2(t_2)$ respectively in $\mathcal{B}/\mathbb{ACU}$. Hence we have the derivation

$$\cfrac{\cfrac{\pi_1}{P_1(t_1)} \quad \cfrac{\pi_2}{P_2(t_2)}}{P(t)}\,(P(x+y) \Leftarrow P_1(x) \wedge P_2(y)/\mathbb{ACU})$$

in $\mathcal{B}/\mathbb{ACU}$. Also if $s \leq_+ t$ then we have some terms $s_1$ and $s_2$ such that $s =_{\mathbb{ACU}} s_1 + s_2$ and $s_1 \leq_+ t_1$ and $s_2 \leq_+ t_2$. By induction hypothesis we have derivations $\pi_3$ and $\pi_4$ of $P_1^\dagger(s_1)$ and $P_2^\dagger(s_2)$ in $\mathcal{B}/\mathbb{ACU}$. Hence we have the derivation

$$\cfrac{\cfrac{\pi_3}{P_1^\dagger(s_1)} \quad \cfrac{\pi_4}{P_2^\dagger(s_2)}}{P^\dagger(s)}\,(P^\dagger(x+y) \Leftarrow P_1^\dagger(x) \wedge P_2^\dagger(y)/\mathbb{ACU})$$

in $\mathcal{B}/\mathbb{ACU}$.

(iv)

$$\pi = \cfrac{\begin{matrix}\vdots & & \vdots\\ P_1(t_1) & \ldots & P_n(t_n)\end{matrix}}{P(t)}\,(P(f(x_1,\ldots,x_n)) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n)/\mathbb{ACU})$$

where $f$ is free. Clearly $t =_{\mathbb{ACU}} f(t_1,\ldots,t_n)$. By induction hypothesis we have derivations $\pi_1,\ldots,\pi_n$ of $P_1(t_1),\ldots,P_n(t_n)$ respectively in $\mathcal{B}/\mathbb{ACU}$. Hence we have the derivation

$$\cfrac{\cfrac{\pi_1}{P_1(t_1)} \quad \ldots \quad \cfrac{\pi_n}{P_n(t_n)}}{P(t)}\,(P(f(x_1,\ldots,x_n)) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n)/\mathbb{ACU})$$

in $\mathcal{B}/\mathbb{ACU}$. Also if $s \leq_+ t$ then either $s =_{\mathbb{ACU}} 0$ or $s =_{\mathbb{ACU}} t$. We have the derivations

$$\frac{}{P^\dagger(0)}\,(P^\dagger(0)/\mathbb{ACU})$$

and

$$\frac{\dfrac{\pi_1}{P_1(t_1)} \quad \ldots \quad \dfrac{\pi_n}{P_n(t_n)}}{P^\dagger(t)}\,(P^\dagger(f(x_1,\ldots,x_n)) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n)/\mathbb{ACU})$$

in $\mathcal{B}/\mathbb{ACU}$. Hence both the cases mentioned above are taken care of.

(v)

$$\pi = \cfrac{\begin{array}{c}\vdots\\ Q(t_1)\end{array}}{P(t)}\,(P(x) \Leftarrow Q(x+y)/\mathbb{ACU})$$

This is the most interesting case. Clearly $t \leq_+ t_1$. Hence by induction hypothesis we have a derivation $\pi_1$ of $Q^\dagger(t)$ in $\mathcal{B}/\mathbb{ACU}$. Hence we have the derivation

$$\cfrac{\cfrac{\pi_1}{Q^\dagger(t)}}{P(t)}\,(P(x) \Leftarrow Q^\dagger(x)/\mathbb{ACU})$$

Also if $s \leq_+ t$ then $s \leq_+ t_1$ also. By induction hypothesis we have a derivation $\pi_2$ of $Q^\dagger(s)$ in $\mathcal{B}/\mathbb{ACU}$. Hence we have the derivation

$$\cfrac{\cfrac{\pi_2}{Q^\dagger(2)}}{P^\dagger(s)}\,(P^\dagger(x) \Leftarrow Q^\dagger(x)/\mathbb{ACU})$$

in $\mathcal{B}/\mathbb{ACU}$.

□

We now state the converse :

**Lemma 56** *For $P \in \mathbb{P}$*

*(i) If $P(t)$ is derivable in $\mathcal{B}/\mathbb{ACU}$ then $P(t)$ is derivable in $\mathcal{A}/\mathbb{ACU}$.*

*(ii) If $P^\dagger(t)$ is derivable in $\mathcal{B}/\mathbb{ACU}$ then for some $s$ we have $t \leq_+ s$ and $P(s)$ is derivable in $\mathcal{A}/\mathbb{ACU}$.*

**Proof:** We prove results (i) and (ii) simultaneously by induction on derivations $\pi$ in $\mathcal{B}/\mathbb{ACU}$. We have the following cases :

(i)

$$\pi = \frac{}{P(t)}\,(P(0)/\mathbb{ACU})$$

where $P \in \mathbb{P}$. Clearly $t =_{\mathbb{ACU}} 0$. Then we have the derivation

$$\frac{}{P(t)}\,(P(0)/\mathbb{ACU})$$

in $\mathcal{A}/\mathbb{ACU}$

(ii)

$$\pi = \frac{}{P^{\dagger}(t)}\,(P^{\dagger}(0)/\mathbb{ACU})$$

where the clause $P^{\dagger}(0)$ is in $\mathcal{B}$ corresponding to the clause $P(0)$ of $\mathcal{A}$. Clearly $t =_{\mathbb{ACU}} 0$. Then we have the derivation

$$\frac{}{P(t)}\,(P(0)/\mathbb{ACU})$$

in $\mathcal{A}/\mathbb{ACU}$.

(iii)

$$\pi = \frac{\begin{array}{c}\vdots\\ P_1(t_1)\end{array}}{P(t)}\,(P(x) \Leftarrow P_1(x)/\mathbb{ACU})$$

where $P, P_1 \in \mathbb{P}$. Clearly $t =_{\mathbb{ACU}} t_1$. By induction hypothesis we have a derivation $\pi_1$ of $P_1(t_1)$ in $\mathcal{A}/\mathbb{ACU}$. Hence we have the derivation

$$\frac{\begin{array}{c}\vdots\\ P_1(t_1)\end{array}}{P(t)}\,(P(x) \Leftarrow P_1(x)/\mathbb{ACU})$$

(iv)

$$\pi = \frac{\begin{array}{c}\vdots\\ P_1^{\dagger}(t_1)\end{array}}{P^{\dagger}(t)}\,(P^{\dagger}(x) \Leftarrow P_1^{\dagger}(x)/\mathbb{ACU})$$

where the clause $P^{\dagger}(x) \Leftarrow P_1^{\dagger}(x)$ is in $\mathcal{B}$ corresponding to the clause $P(x) \Leftarrow P_1(x)$ of $\mathcal{A}$. Clearly $t =_{\mathbb{ACU}} t_1$. By induction hypothesis we have some $s$ such that $t_1 \leq_+ s$ and we have a derivation $\pi_1$ of $P_1(s)$ in $\mathcal{A}/\mathbb{ACU}$. Then $t \leq_+ s$ and we have the derivation

$$\frac{\begin{array}{c}\vdots\\ P_1(s)\end{array}}{P(s)}\,(P(x) \Leftarrow P_1(x)/\mathbb{ACU})$$

in $\mathcal{A}/\mathbb{ACU}$.

(v)

$$\pi = \frac{\begin{array}{c}\vdots\\ P_1(t_1)\end{array} \quad \begin{array}{c}\vdots\\ P_2(t_2)\end{array}}{P(t)}\,(P(x+y) \Leftarrow P_1(x) \wedge P_2(y)/\mathbb{ACU})$$

where $P, P_1, P_2 \in \mathbb{P}$. Clearly $t =_{\mathbb{ACU}} t_1 + t_2$. By induction hypothesis we have derivations $\pi_1$ and $\pi_2$ of $P_1(t_1)$ and $P_2(t_2)$ in $\mathcal{A}/\mathbb{ACU}$. Hence we have the derivation

$$\pi = \frac{\dfrac{\pi_1}{P_1(t_1)} \quad \dfrac{\pi_2}{P_2(t_2)}}{P(t)}\,(P(x+y) \Leftarrow P_1(x) \wedge P_2(y)/\mathbb{ACU})$$

(vi)

$$\pi = \dfrac{\begin{array}{cc}\vdots & \vdots\\ P_1^\dagger(t_1) & P_2^\dagger(t_2)\end{array}}{P^\dagger(t)}\,(P^\dagger(x+y) \Leftarrow P_1^\dagger(x) \wedge P_2^\dagger(y)/\mathbb{ACU})$$

where $P, P_1, P_2 \in \mathbb{P}$. Clearly $t =_{\mathbb{ACU}} t_1 + t_2$. By induction hypothesis we have some $s_1$ and $s_2$ such that $t_1 \leq_+ s_1$, $t_2 \leq_+ s_2$ and we have derivations $\pi_1$ and $\pi_2$ of $P_1(s_1)$ and $P_2(s_2)$ respectively in $\mathcal{A}/\mathbb{ACU}$. Then clearly $t \leq_+ s_1 + s_2$ and we have the derivation

$$\pi = \dfrac{\dfrac{\pi_1}{P_1(s_1)} \quad \dfrac{\pi_2}{P_2(s_2)}}{P(s_1 + s_2)}\,(P(x+y) \Leftarrow P_1(x) \wedge P_2(y)/\mathbb{ACU})$$

in $\mathcal{A}/\mathbb{ACU}$.

(vii)

$$\pi = \dfrac{\begin{array}{ccc}\vdots & & \vdots\\ P_1(t_1) & \ldots & P_n(t_n)\end{array}}{P(t)}\,(P(f(x_1, \ldots, x_n)) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n)/\mathbb{ACU})$$

where $f$ is free and $P, P_1, \ldots, P_n \in \mathbb{P}$. Clearly $t =_{\mathbb{ACU}} f(t_1, \ldots, t_n)$. By induction hypothesis we have derivations $\pi_1, \ldots, \pi_n$ of $P_1(t_1), \ldots, P_n(t_n)$ respectively in $\mathcal{A}/\mathbb{ACU}$. Hence we have the derivation

$$\dfrac{\dfrac{\pi_1}{P_1(t_1)} \quad \ldots \quad \dfrac{\pi_n}{P_n(t_n)}}{P(t)}\,(P(f(x_1, \ldots, x_n)) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n)/\mathbb{ACU})$$

in $\mathcal{A}/\mathbb{ACU}$.

(viii)

$$\pi = \dfrac{\begin{array}{ccc}\vdots & & \vdots\\ P_1(t_1) & \ldots & P_n(t_n)\end{array}}{P^\dagger(t)}\,(P^\dagger(f(x_1, \ldots, x_n)) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n)/\mathbb{ACU})$$

where $P, P_1, \ldots, P_n \in \mathbb{P}$. By induction hypothesis we have derivations $\pi_1, \ldots, \pi_n$ of $P_1(t_1), \ldots, P_n(t_n)$ respectively in $\mathcal{A}/\mathbb{ACU}$. We know that $t \leq_+ t$ and also we have the derivation

$$\dfrac{\dfrac{\pi_1}{P_1(t_1)} \quad \ldots \quad \dfrac{\pi_n}{P_n(t_n)}}{P(t)}\,(P(f(x_1, \ldots, x_n)) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n)/\mathbb{ACU})$$

in $\mathcal{A}/\mathbb{ACU}$.

(ix)

$$\pi = \dfrac{}{P(t)}\,(P^\dagger(0)/\mathbb{ACU})$$

where the clause $P^\dagger(0)$ is in $\mathcal{B}$ corresponding to the clause $P(f(x_1, \ldots, x_n)) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n)$ in $\mathcal{A}$. Clearly $t =_{\mathbb{ACU}} 0$. Since we assumed that each state in $\mathcal{A}$ is non-empty hence we have terms $t_1, \ldots, t_n$ such that we have derivations $\pi_1, \ldots, \pi_n$ of atoms $P_1(t_1), \ldots, P_n(t_n)$ respectively. Clearly $t \leq_+ f(t_1, \ldots, t_n)$ and we also have the derivation

$$\frac{\dfrac{\pi_1}{P_1(t_1)} \quad \ldots \quad \dfrac{\pi_n}{P_n(t_n)}}{P(f(t_1, \ldots, t_n))} \, (P(f(x_1, \ldots, x_n)) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n)/\mathbb{ACU})$$

in $\mathcal{A}/\mathbb{ACU}$. Next we have the two most interesting cases (x) and (xi).

(x)

$$\pi = \frac{\begin{matrix}\vdots \\ Q^\dagger(t_1)\end{matrix}}{P(t)} \, (P(x) \Leftarrow Q^\dagger(x)/\mathbb{ACU})$$

where $P, Q \in \mathbb{P}$. Clearly $t =_{\mathbb{ACU}} t_1$. By induction hypothesis we have some $s$ such that $t_1 \leq_+ s$ and there is a derivation $\pi_1$ of $Q(s)$ in $\mathcal{A}/\mathbb{ACU}$. Since $t \leq_+ s$ we have the derivation

$$\frac{\dfrac{\pi_1}{Q(s)}}{P(t)} \, (P(x) \Leftarrow Q(x+y)/\mathbb{ACU})$$

in $\mathcal{A}/\mathbb{ACU}$.

(xi)

$$\pi = \frac{\begin{matrix}\vdots \\ Q^\dagger(t_1)\end{matrix}}{P^\dagger(t)} \, (P^\dagger(x) \Leftarrow Q^\dagger(x)/\mathbb{ACU})$$

where the clause $P^\dagger(x) \Leftarrow Q^\dagger(x)$ is in $\mathcal{B}$ corresponding to the clause $P(x) \Leftarrow Q(x+y)$ in $\mathcal{A}$. Clearly $t =_{\mathbb{ACU}} t_1$. By induction hypothesis we have some $s$ such that $t_1 \leq_+ s$ and we have a derivation $\pi_1$ of $Q(s)$ in $\mathcal{A}/\mathbb{ACU}$. We have $t \leq_+ s$. Hence we have the derivation

$$\frac{\dfrac{\pi_1}{Q(s)}}{P(t)} \, (P(x) \Leftarrow Q(x+y)/\mathbb{ACU})$$

in $\mathcal{A}/\mathbb{ACU}$ and we know that $t \leq_+ t$.

□

Now if the final state of $\mathcal{A}$ is $P$ then we name $P$ to be also the final state of $\mathcal{B}$. From Lemmas 55 and 56 we have $\mathcal{L}(\mathcal{A}/\mathbb{ACU}) = \mathcal{L}(\mathcal{B}/\mathbb{ACU})$. We conclude that

**Theorem 38** *The automata modulo $\mathbb{ACU}$ containing one-way $\mathbb{ACU}$ automata clauses and standard $+$-push clauses can be converted in polynomial time to one-way $\mathbb{ACU}$ automata accepting the same language.*

The construction described above takes linear time. This construction is based on the assumption that the empty states of the automaton have been already removed. This step can be carried out in polynomial time by Lemma 54.

While we are able to do the above construction for $\mathbb{ACU}$ theory in linear time, these ideas fail to work in the case of $\mathbb{AC}$ (even after the assumption that all the empty states of the automaton have already been removed.) It is actually an open question whether such a translation can be done for $\mathbb{AC}$ theory in linear time. We now explain why the above ideas fail in the case of $\mathbb{AC}$ theory. Define an operator $Cl$ as $Cl(S) = \{s \mid \exists t \in S \cdot s \leq_+ t\}$. In the $\mathbb{ACU}$ case, a standard $+$-push clause $P(x) \Leftarrow Q(x+y)$ intuitively tells us to accept at $P$ the terms in $Cl(S)$ where $S$ is the set of terms accepted at $Q$. In the above construction, the states $Q^\dagger$ accept the terms in $Cl(S)$ where $S$ is the set of terms accepted at $Q$. The operator $Cl$ has the nice properties that $S \subseteq Cl(S)$ and $Cl(Cl(S)) = Cl(S)$. This justifies the translation of the clause $P(x) \Leftarrow Q(x+y)$ in the automaton $\mathcal{A}$ above by the pair of clauses $P(x) \Leftarrow Q^\dagger(x)$ and $P^\dagger(x) \Leftarrow Q^\dagger(x)$.

However in the $\mathbb{AC}$ case the situation is somewhat different. Here we need to define an operator $Cl'$ as $Cl'(S) = \{s \mid \exists t \in S \cdot \exists u \cdot s + u =_{\mathbb{AC}} t\}$. Then a standard $+$-push clause $P(x) \Leftarrow Q(x+y)$ intuitively tells us to accept at $P$ the terms in $Cl'(S)$ where $S$ is the set of terms accepted at $Q$. We don't have the property $S \subseteq Cl'(S)$ since we don't have an unit symbol for the $+$ operator. Neither do we have the property $Cl'(Cl'(S)) = Cl'(S)$. As an example if all the free symbols in the signature are constants, then $Cl'(S)$ contains all those terms which are obtained by removing at least one symbol from some term in $S$. $Cl'(Cl'(S))$ contains all those terms which are obtained by removing at least two symbols from some term in $S$. Hence there is some kind of counting implicit in the $Cl'$ operator, which makes it difficult to define clauses clauses corresponding to states of the form $Q^\dagger$ such that $Q^\dagger$ accepts the terms in $Cl'(S)$ where $S$ is the set of terms accepted at $Q$.

In order to solve the problem in the $\mathbb{AC}$ case, we take a detour through an extension of *Vector Addition Systems with States (VASS)* defined and studied in the next section.

## 11.2 Extended VASS

We first try to motivate our study of VASS and its extension by giving some ideas as to how they are connected to equational tree automata. Vector Addition Systems with States (or VASS) can be thought of as automata on tuples of natural numbers. A VASS has a finite number of states each of which accepts a set of tuples. First of all we fix a number $p \in \mathbb{N}$ for the rest of this section, so that we will be interested in (extensions of) VASS which accept tuples from $\mathbb{N}^p$. Now observe that if we have an $\mathbb{AC}$ or $\mathbb{ACU}$ automaton on a signature in which the set of free symbols are exactly the constants $a_1, \ldots, a_p$, then the terms on this signature can be thought of as tuples from $\mathbb{N}^p$. Hence it is natural to think of possibilities of translation of our automata to VASS. To be more precise :

**Definition 9 (VASS)** *A* Vector Addition System with States (VASS) *consists of a finite set of states and a finite set of transitions of the form* $q \xrightarrow{\nu} q'$ *where* $q$ *and* $q'$ *are states and* $\nu \in \mathbb{Z}^p$. Configurations *of a VASS are of the form* $q(\nu)$ *where* $q$ *is a state and* $\nu \in \mathbb{N}^p$*. A VASS in addition has an initial configuration.*

Intuitively the transition $q \xrightarrow{\nu} q'$ says that if $\nu_1 \in \mathbb{N}^p$ is accepted at $q$ and $\nu_1 + \nu \in \mathbb{N}^p$ then $\nu_1 + \nu$ is accepted at $q'$. Thus moves of the VASS are defined by the binary relation $\rightarrow$ where $q(\nu_1) \rightarrow q'(\nu_2)$ iff for some transition $q \xrightarrow{\nu} q'$ we have $\nu_1 + \nu = \nu_2$. (As mentioned above, we assume $\nu_1, \nu_2 \in \mathbb{N}^p$.) Its reflexive transitive closure $\rightarrow^*$ is the reachability relation on configurations. Configurations reachable from the initial configuration are said to be the reachable configurations of the VASS.

Now if we try to translate our automata to VASS, then a simple clause $q'(x) \Leftarrow q(x+a_i)$ can easily be translated by the clause $q \xrightarrow{\nu} q'$ with $\nu = (0, \ldots, 0, -1, 0, \ldots, 0)$ where the '$-1$' is put in the $i$th component of $\nu$. The clause intuitively tells us to delete a constant $a_i$ from some term accepted at $q$ and the resulting term is accepted at $q'$. Deleting $a_i$ from a term corresponds to subtracting $1$ from the $i$th component of a tuple. Now a standard $+$-push clause $P(x) \Leftarrow Q(x+y)$ in the $\mathbb{AC}$ case simply tells us to delete arbitrary number of (but at least one) constants from some term accepted at $Q$ and the resulting term is accepted at $P$. Such a clause can easily be expressed using clause of the form $q'(x) \Leftarrow q(x+a_i)$ by having a loop. Similarly an epsilon clause $q'(x) \Leftarrow q(x)$ is translated as $q'(x) \Leftarrow q(x+a_i)$ where $\nu = (0, \ldots, 0)$. A clause of the form $q'(x+a) \Leftarrow q(x)$ which is a restricted version of the $+$-pop clause, can be translated as the transition $q \xrightarrow{\nu} q'$ with $\nu = (0, \ldots, 0, 1, 0, \ldots, 0)$ where the '1' is put in the $i$th component of $\nu$. Note that some care needs to be taken in the translation described above if we are working modulo $\mathbb{AC}$ in order to avoid deriving the '0' term, which corresponds to the tuple $(0, \ldots, 0)$.

However the main problem occurs when we try to translate $+$-pop clauses of the form $P(x+y) \Leftarrow P_1(x) \wedge P_2(y)$. Such a clause has no obvious translation to the transitions of VASS described above. If we try to equate the derivable atoms of our automata with the reachable configurations of a VASS, then observe that the derivations in our automata are branching tree like structures (precisely because of the $+$-pop clauses) whereas a VASS can be thought of as being able to move in a linear fashion from one configuration to another. It is to deal with such clauses that we define an extension of VASS in Definition 10 below. Observe also that we have changed our representation slightly so that the transitions are now written in the form of clauses, which makes them look more like our automata.

**Definition 10 (Extended VASS)** *An* extended VASS $\mathcal{V}$ *is a set of clauses of the form*

$$P(\nu), \quad \nu \in \mathbb{N}^p \tag{11.1}$$

$$P(x+\nu) \Leftarrow P_1(x), \quad \nu \in \mathbb{Z}^p \tag{11.2}$$

$$P(x+y) \Leftarrow P_1(x) \wedge P_2(y) \tag{11.3}$$

*where $P, P_1, P_2$ are predicates.*

Clauses (11.1) are called *constant clauses*. Clauses (11.2) are called *constant-addition clauses*. Clauses (11.3) are called *addition clauses*.

We clarify that for our discussion, $P(x+y) \Leftarrow P_1(x) \wedge P_2(y)$, $P(x+y) \Leftarrow P_2(x) \wedge P_1(y)$, and $P(y+x) \Leftarrow P_1(x) \wedge P_2(y)$ are three different notations for the same mathematical object.

**Definition 11 (Configuration)** Configurations *are of the form $P(\nu)$ where $\nu \in \mathbb{N}^p$.* Generalized configurations *are of the form $P(\nu)$ where $\nu \in (\mathbb{N} \cup \{\infty\})^p$.*

For $i \leq i \leq p$, $\nu(i)$ denotes the $i$th component of $\nu$. If $I = \{i_1, ..., i_n\}$ where $1 \leq i_1 < ... < i_n \leq p$ then $\nu(I) = (\nu(i_1), ..., \nu(i_n))$.

**Definition 12 (Derivation in Extended VASS)** *The set of configurations which are* derivable *in an extended VASS $\mathcal{V}$ is defined inductively as*

1. *If $P(\nu)$ is a constant clause then $P(\nu)$ is derivable in $\mathcal{V}$.*
2. *If $P_1(\nu_1)$ is derivable in $\mathcal{V}$, $P(x+\nu) \Leftarrow P_1(x)$ is a constant-addition clause, and $\nu_1 + \nu \geq 0$, then $P(\nu_1 + \nu)$ is derivable in $\mathcal{V}$.*

3. *If $P_1(\nu_1)$ and $P_2(\nu_2)$ are derivable in $\mathcal{V}$, and $P(x+y) \Leftarrow P_1(x) \wedge P_2(y)$ is an addition clause, then $P(\nu_1+\nu_2)$ is derivable.*

In the last rule, it is guaranteed that $\nu_1 \geq 0$ and $\nu_2 \geq 0$, hence $\nu_1 + \nu_2 \geq 0$. We will simply say that a configuration $P(\nu)$ is derivable, when the extended VASS involved is obvious. The constant-addition clauses of the form $P(x+\nu) \Leftarrow P_1(x)$ correspond to the transitions $P_1 \xrightarrow{\nu} P$ in VASS. While this is the only kind of transition in VASS, in EVASS, we have the addition clauses (11.2) for which there is no equivalent in VASS. To make the translation from VASS to EVASS complete, if initial configuration of the VASS is $q(\nu)$ then we add the constant clause $q(\nu)$ to the translated EVASS. In this way the set of reachable configuration of the VASS is the same as that of the EVASS we get from this translation. Hence a VASS can be thought of as an EVASS which has no addition clause and which has exactly one constant clause.

Next we are going to extend the notion of Karp and Miller trees for our extended VASS's. This poses some complications. A path in the Karp and Miller tree for a VASS corresponds to a path in the VASS. For extended VASS's, instead of paths, we have the corresponding notion of derivation, which is actually a tree like structure (because of the addition clauses) instead of a sequence. Hence the structure corresponding to Karp and Miller trees would be a tree like structure branching in upward as well as downward direction, which would be cumbersome to reason about. To simplify the presentation, we have broken the construction into two phases. First we define a notion of *covering derivation* which is a tree structure and vaguely corresponds to the finite paths of Karp and Miller trees. We do all our reasoning on these derivations, instead of reasoning on a structure containing all such derivations. Finally we just show that there are only a finite number of such derivations, by showing how to construct a (finite) forest of all such derivations. This last step corresponds to the result that the Karp and Miller trees for VASS's are finite. (We have a forest instead of a tree because we have arbitrarily many constant clauses, whereas in VASS's we consider a single starting configuration).

**Definition 13 (Covering Derivation)** *Assume fixed an extended VASS $\mathcal{V}$. A* covering derivation *$\delta$ is a finite tree, each of whose nodes is labeled with a generalized configuration and a clause, constructed using the following rules :*

1. *If $P(\nu)$ is a constant clause then the tree with a single node $N$ labeled with the generalized configuration $P(\nu)$ and with the same clause is a covering derivation.*
2. *If*
   - *there is a covering derivation $\delta_1$ whose root is labeled with generalized configuration $P_1(\nu_1)$*
   - *there is no other node in $\delta_1$ labeled with generalized configuration $P_1(\nu_1)$*
   - *there is a clause $C = P(x+\nu_2) \Leftarrow P_1(x)$*
   - $\nu_1 + \nu_2 \geq 0$

   *then the tree whose root $N$ is mapped to the clause $C$ and generalized configuration $P(\nu)$, and has as unique child $\delta_1$, is a covering derivation, where $\nu(i)$ is defined for each $1 \leq i \leq p$ as follows :*

   (a) *If there is some node $N'$ in $\delta_1$ labeled with the generalized configuration $P(\nu')$ such that $\nu' \leq \nu_1 + \nu_2$ and $\nu'(i) < \nu_1(i) + \nu_2(i)$, then $\nu(i) = \infty$.*
   (b) *Otherwise $\nu(i) = \nu_1(i) + \nu_2(i)$.*

3. *If*
   - $\delta_1$ *is a covering derivation whose root is labeled with generalized configuration* $P_1(\nu_1)$
   - *no other node in* $\delta_1$ *is labeled with the generalized configuration* $P_1(\nu_1)$
   - $\delta_2$ *is a covering derivation whose root is labeled with generalized configuration* $P_2(\nu_2)$
   - *no other node in* $\delta_2$ *is labeled with* $P_2(\nu_2)$
   - *there is a clause* $C = P(x+y) \Leftarrow P_1(x) \wedge P_2(y)$

   *then the tree whose root* $N$ *is labeled with clause* $C$ *and generalized configuration* $P(\nu)$*, and has two children* $\delta_1$ *and* $\delta_2$*, is a covering derivation where* $\nu(i)$ *is defined for each* $1 \leq i \leq p$ *as follows :*

   (a) *If there is some node* $N'$ *in* $\delta_1$ *or* $\delta_2$*, labeled with a generalized configuration* $P(\nu')$*, such that* $\nu' \leq \nu_1 + \nu_2$ *and* $\nu'(i) < \nu_1(i) + \nu_2(i)$*, then* $\nu(i) = \infty$*.*

   (b) *Otherwise* $\nu(i) = \nu_1(i) + \nu_2(i)$*.*

Intuitively, covering derivations compute 'limits' of configurations derivable in an extended VASS. This is made precise by Theorems 39 and 40 below.

**Theorem 39** *Let* $\mathcal{V}$ *be an extended VASS. If a configuration* $P(\nu)$ *is derivable, then there is a covering derivation* $\delta$ *with root labeled with some generalized configuration* $P(\nu')$ *such that for all* $1 \leq i \leq p$*, if* $\nu'(i) < \infty$ *then* $\nu'(i) = \nu(i)$*.*

**Proof:** We do induction on the size of the derivation of $P(\nu)$. We have the following cases :

(i) If $P(\nu)$ is derivable using the constant clause $P(\nu)$, then by using Rule 1 of Definition 13, we have a covering derivation $\delta$ with root labeled with configuration $P(\nu)$ which satisfies the requirements.

(ii) Suppose $P(\nu_1 + \nu_2)$ is derivable from the derivation of $P_1(\nu_1)$ using the clause $P(x + \nu_2) \Leftarrow P_1(x)$. By induction hypothesis we have a covering derivation $\delta_1$, with root labeled with some $P_1(\nu_1')$ such that if $\nu_1'(i) < \infty$ then $\nu_1'(i) = \nu_1(i)$. We pick a minimal such $\delta_1$. Consequently no other node in $\delta_1$ is labeled with $P_1(\nu_1')$ (otherwise we would have picked the covering derivation rooted at the latter node instead.) Clearly we have $\nu_1 + \nu_2 \geq 0$ and hence $\nu_1' + \nu_2 \geq 0$. By using Rule 2 of Definition 13, we get a covering derivation $\delta$ with root labeled by a generalized configuration $P(\nu')$ with the property that if $\nu'(i) < \infty$ then $\nu'(i) = \nu_1'(i) + \nu_2(i)$. But then if $\nu'(i) < \infty$ then $\nu_1'(i) < \infty$ and hence $\nu_1'(i) = \nu_1(i)$, so $\nu'(i) = \nu_1(i) + \nu_2(i)$. Hence $\delta$ is the required covering derivation.

(iii) Suppose $P(\nu_1 + \nu_2)$ is derivable from the derivations of $P_1(\nu_1)$ and $P_2(\nu_2)$ using the clause $P(x + y) \Leftarrow P_1(x) \wedge P_2(y)$. By induction hypothesis, we have covering derivations $\delta_1$ and $\delta_2$ with roots labeled with $P_1(\nu_1')$ and $P_2(\nu_2')$ respectively such that for all $i$, if $\nu_1'(i) < \infty$ then $\nu_1'(i) = \nu_1(i)$, and if $\nu_2'(i) < \infty$ then $\nu_2'(i) = \nu_2(i)$. As in the previous case we may assume that no non-root node in $\delta_1$ is labeled with the generalized configuration $P_1(\nu_1')$ and no non-root node in $\delta_2$ is labeled with the generalized configuration $P_2(\nu_2')$. By using Rule 3 of Definition 13, we get a covering derivation $\delta$ with root labeled by a generalized configuration $P(\nu')$ with the property that if $\nu'(i) < \infty$, then $\nu'(i) = \nu_1'(i) + \nu_2'(i)$. But then if $\nu'(i) < \infty$, then $\nu_1'(i), \nu_2'(i) < \infty$, and hence $\nu_1'(i) = \nu_1(i)$ and $\nu_2'(i) = \nu_2(i)$, implying that $\nu'(i) = \nu_1(i) + \nu_2(i)$. Hence $\delta$ is the required covering derivation.

□

**Definition 14 (Linear Path)** *Assume fixed an extended VASS $\mathcal{V}$. a* linear path $\pi$ *is a sequence of nodes $N_1, ..., N_n$ for some $n \geq 1$, and edges $e_i$ from $N_i$ to $N_{i+1}$ for $1 \leq i < n$, with each $N_i$ labeled by a predicate $P_i$, each $e_i$ being labeled by*

*(i) either a constant-addition clause of the form $P_{i+1}(x + \nu) \Leftarrow P_i(x)$*

*(ii) or by a pair of the form $(C, Q(\nu))$ where $C$ is an addition clause of the form $P_{i+1}(x + y) \Leftarrow P_i(x) \wedge Q(y)$, and $Q(\nu)$ is derivable in $\mathcal{V}$.*

In both cases (i) and (ii) in Definition 14 we say that the *valuation* of the edge is $v(e) = \nu$. The *valuation* of the linear path is defined as $v(\pi) = \Sigma_{1 \leq i < n} v(e_i)$. We also say that $\pi$ is a linear path *from $P_1$ to $P_n$*. We sometimes denote $\pi$ as

$$P_1 \xrightarrow{e_1} P_2 \xrightarrow{e_2} ...P_{n-1} \xrightarrow{e_{n-1}} P_n$$

If

$$\pi_1 = P_1 \xrightarrow{e_1} P_2 \xrightarrow{e_2} \dots P_{n-1} \xrightarrow{e_{n-1}} P_n$$

and

$$\pi_2 = P_n \xrightarrow{e_n} P_{n+1} \xrightarrow{e_{n+1}} \dots P_{n+m-1} \xrightarrow{e_{n+m-1}} P_{n+m}$$

then we define the *concatenation* of $\pi_1$ and $\pi_2$ as the linear path path

$$\pi_1\pi_2 = P_1 \xrightarrow{e_1} P_2 \xrightarrow{e_2} ...P_{n-1} \xrightarrow{e_{n-1}} P_n \xrightarrow{e_n} P_{n+1} \xrightarrow{e_{n+1}} ...P_{n+m-1} \xrightarrow{e_{n+m-1}} P_{n+m}$$

Note that $\pi_1\pi_2$ is defined only when the last predicate of $\pi_1$ is equal to the first predicate of $\pi_2$. In such a case, we clearly have the equality

$$v(\pi_1\pi_2) = v(\pi_1) + v(\pi_2)$$

**Definition 15 (Admissible Linear Path)** *Given a tuple $\nu \in (\mathbb{N} \cup \{\infty\})^p$, the linear path $\pi$ is said to be* admissible *for $\nu$ if for each prefix $\pi'$ of $\pi$, we have $\nu + v(\pi') \geq 0$. It is* admissible with respect to *$I \subseteq \{1, \dots, p\}$ if $(\nu + v(\pi'))(I) \geq 0$ for all such $\pi'$.*

Then it is easy to see that if $\pi$ is a linear path from $P_1$ to $P_n$ and is admissible for $\nu$ and $P_1(\nu)$ is derivable in $\mathcal{V}$ (in which case $\nu$ would have no infinite coordinate by definition,) then $P_n(\nu + v(\pi))$ is derivable. Also if $\pi_1$ is admissible for $\nu$ and $\pi_2$ is admissible for $\nu + v(\pi_1)$, then $\pi_1\pi_2$ is admissible for $\nu$.

**Example 8** *Consider an EVASS with the following set of clauses*

$$\begin{aligned} C_1 &= P_1(2,5) \\ C_2 &= P_2(3,4) \\ C_3 &= P(x+y) \Leftarrow P_1(x) \wedge P_2(y) \\ C_4 &= P_1(x + (-2,-4)) \Leftarrow P_3(x) \\ C_5 &= P_2(x + (2,-5)) \Leftarrow P_3(x) \end{aligned}$$

*Some configurations derivable in this EVASS are $P_1(2,5), P_2(3,4), P_3(5,9), P_1(2+n,5), P_3(5+n,9), P_2(3+4n,4)$ for all $n \geq 0$. Figure 11.1 shows an example of a covering derivation for this EVASS. Figure 11.2 is an example of a linear path. Note that the configurations $P_2(3,4)$ and $P_1(50,5)$ are derivable in the EVASS.*

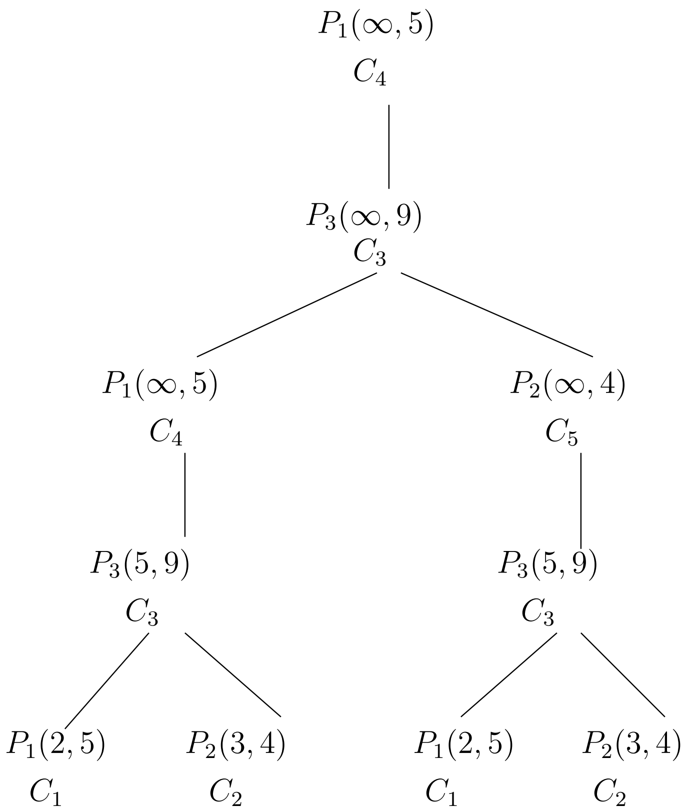


FIG. 11.1 – A covering derivation for the EVASS in Example 8

$$P_1 \xrightarrow{(C_3, P_2(3,4))} P_3 \xrightarrow{C_4} P_2 \xrightarrow{C_3, P_1(50,5))} P_3$$

FIG. 11.2 – A linear path for the EVASS in Example 8

We require the following auxiliary lemma to prove Theorem 40 which is the most crucial result of our discussion on extended VASS's.

**Lemma 57** *Let $\mathcal{V}$ be an extended VASS. Let $\delta$ be a covering-derivation such that given any generalized configuration $P(\nu)$ occurring in this derivation, we can find a $\nu'$ such that*

– *$P(\nu')$ is derivable*

– *for each $i \in \{1, ..., p\}$, if $\nu(i) < \infty$ then $\nu'(i) = \nu(i)$*

*Suppose that $N_1$, labeled with generalized configuration $P_1(\nu_1)$, is a descendent of $N_2$, labeled with generalized configuration $P_2(\nu_2)$. Let $I$ be such that for each $i \in I$, $\nu_2(i) < \infty$. Then we can find a linear path $\pi$ from $P_1$ to $P_2$ such that $\pi$ is admissible for $\nu_1$ wrt $I$ and $(\nu_1 + v(\pi))(I) = \nu_2(I)$.*

**Proof:** We induct on the distance between the nodes $N_1$ and $N_2$ in the tree $\delta$. We have the following cases :

(i) Suppose the distance is 0 ($N_1$ is same as $N_2$). Then the trivial linear path $P_1$ suffices.

(ii) Suppose $N_2$ is labeled with clause $P_2(x + \nu_4) \Leftarrow P_3(x)$ and has child $N_3$, labeled with generalized configuration $P_3(\nu_3)$, and $N_1$ is descendent of $N_3$. Clearly we must have $(\nu_3 + \nu_4)(I) = \nu_2(I)$. Also for all $i \in I$, $\nu_3(i) < \infty$. By induction hypothesis we get a linear path $\pi'$ from $P_1$ to $P_3$ which is admissible for $\nu_1$ wrt $I$, and such that $(\nu_1 + v(\pi'))(I) = \nu_3(I)$. Let the required linear path $\pi$ be the concatenation of $\pi'$ with the linear path $P_3 \xrightarrow{P_2(x+\nu_4)\Leftarrow P_3(x)} P_2$ . We have $(\nu_1+v(\pi))(I) = (\nu_1+v(\pi')+\nu_4)(I) = (\nu_3+\nu_4)(I) = \nu_2(I)$. In particular $(\nu_1+v(\pi))(I) \geq 0$ and hence $\pi$ is admissible for $\nu_1$ wrt $I$.

(iii) Suppose $N_2$ is labeled with clause $P_2(x + y) \Leftarrow P_3(x) \wedge P_4(x)$ and has children $N_3$ and $N_4$ labeled with $P_3(\nu_3)$ and $P_4(\nu_4)$ respectively, and without loss of generality $N_1$ is descendent of $N_3$. Clearly $\nu_2(I) = (\nu_3 + \nu_4)(I)$. Also for all $i \in I$, $\nu_3(i), \nu_4(i) < \infty$. By induction hypothesis we get a linear path $\pi'$ from $P_1$ to $P_3$ which is admissible for $\nu_1$ wrt $I$, and such that $(\nu_1+v(\pi'))(I) = \nu_3(I)$. Also, by assumption we have a $\nu_4'$ such that $P_4(\nu_4')$ is derivable, and for all $i$, if $\nu_4(i) < \infty$ then $\nu_4'(i) = \nu_4(i)$. In particular $\nu_4(I) = \nu_4'(I)$. Let the required linear path $\pi$ be the concatenation of $\pi'$ with the linear path $P_3 \xrightarrow{(P_2(x+y)\Leftarrow P_3(x)\wedge P_4(y), P_4(\nu_4'))} P_2$ . This is a well-defined linear path. We have $(\nu_1 + v(\pi))(I) = (\nu_1 + v(\pi') + \nu_4')(I) = (\nu_3 + \nu_4')(I) = (\nu_3 + \nu_4)(I) = \nu_2(I)$. In particular $(\nu_1 + v(\pi))(I) \geq 0$ and hence $\pi$ is admissible for $\nu_1$ wrt $I$.

□

Now we are ready to prove the required result :

**Theorem 40** *Let $\mathcal{V}$ be an extended VASS. If there is a covering-derivation $\delta$ with root labeled by the generalized configuration $P(\nu)$ then for any $K \geq 0$ there is a tuple $\nu' \in \mathbb{N}^p$ such that*

– *for every i, if $\nu(i) = \infty$ then $\nu'(i) \geq K$*

– *for every i, if $\nu(i) < \infty$ then $\nu'(i) = \nu(i)$*

– *$P(\nu')$ is derivable*

**Proof:** We do induction on the size of $\delta$. We have the following cases :

(i) If $\delta$ is just a single node then $P(\nu)$ is a constant clause, and the tuple $\nu' = \nu$ is easily seen to satisfy the requirements.

(ii) Suppose $\delta$ is constructed using the Rule 2 of Definition 13. Let

$$\begin{aligned} I &= \{i \mid \nu(i) = \infty\} \\ I_1 &= \{i \mid \nu_1(i) = \infty\} \\ J &= \{1, ..., p\} \setminus I \\ J_1 &= \{1, ..., p\} \setminus I_1 \\ I_a &= I \setminus I_1 \end{aligned}$$

For each $i \in I_a$ we define a path $\pi_i$ as follows. By definition of $I_a$ we have a node $N^i$ in $\delta_1$ labeled with a generalized configuration $P(\nu^i)$ such that

$$\nu^i \leq \nu_1 + \nu_2$$
$$\text{and } \nu^i(i) < \nu_1(i) + \nu_2(i).$$

Because of the induction hypothesis, the covering derivation rooted at $N_1$ satisfies the assumptions of Lemma 57. Hence from this lemma we get a linear path $\pi'_i$ from $P$ to $P_1$ which is admissible for $\nu^i$ wrt $J_1$ and

$$(\nu^i + v(\pi'_i))(J_1) = \nu_1(J_1).$$

Let $\pi_i$ be the linear path obtained by concatenating $\pi'_i$ with the linear path $P_1 \xrightarrow{P(x+\nu_2) \Leftarrow P_1(x)} P$ . We have

$$(\nu^i + v(\pi_i))(J_1) = (\nu^i + v(\pi'_i) + \nu_2)(J_1) = (\nu_1 + \nu_2)(J_1) \geq 0.$$

Hence $\pi_i$ is admissible for $\nu^i$ wrt $J_1$. Since $\nu^i \leq \nu_1 + \nu_2$, $\pi_i$ is admissible for $\nu_1 + \nu_2$ wrt $J_1$. If $j \in J$, then $\nu^i(j) = (\nu_1 + \nu_2)(j)$, otherwise by construction of the covering derivation, we would have the contradiction that $\nu(j) = \infty$. But since $J \subseteq J_1$, we have

$$v(\pi_i)(J) = 0.$$

Also since $\nu^i \leq \nu_1 + \nu_2$, therefore we have

$$v(\pi_i)(J_1) \geq 0.$$

Since $\nu^i(i) < (\nu_1 + \nu_2)(i)$, we have

$$v(\pi_i)(i) \geq 1.$$

Let

$$\pi = \Pi_{i \in I_a} \pi_i$$

where the product operation is concatenation of paths (the order in which the $\pi_i$'s are concatenated is not important.) Since each $\pi_i$ starts from $P$ and ends at $P$, the linear path $\pi$ is well-defined. Since $v(\pi_i)(J_1) \geq 0$ for each $i$, and each $\pi_i$ is admissible for $\nu_1 + \nu_2$ wrt $J_1$, it is easy to see that $\pi$ is admissible for $\nu_1 + \nu_2$ wrt $J_1$. Also we have

$$v(\pi)(J) = \textstyle\sum_{i \in I_a} v(\pi_i)(J) = \sum_{i \in I_a} 0 = 0.$$

Since $v(\pi_i)(i) \geq 1$ for each $i \in I_a$, we have

$$v(\pi)(I_a) \geq 1.$$

We can again see that

$$\text{the path } \pi^K \text{ is admissible for } \nu_1 + \nu_2 \text{ wrt } J_1 \qquad (\theta)$$
$$v(\pi^K)(J) = 0 \qquad (\Omega)$$
$$\text{and } v(\pi^K)(I_a) \geq (K, ..., K). \qquad (\zeta)$$

Choose a $K_1 \geq 0$ such that for each prefix $\pi'$ of $\pi^K$, we have

$$((K_1, ..., K_1) + v(\pi'))(I_1) \geq 0 \qquad (\beta)$$
$$\text{and } ((K_1, ..., K_1) + v(\pi^K))(I_1) \geq (K, ..., K) \qquad (\gamma)$$

Consider $K_2$ such that

$$((K_2, , , ., K_2) + \nu_2)(I_1) \geq (K_1, ..., K_1) \qquad (\alpha)$$

By induction hypothesis we have a $\nu_1'$ such that

$$P_1(\nu_1') \text{ is derivable} \quad \text{(a)}$$
$$\nu_1'(I_1) \geq (K_2, ..., K_2) \quad \text{(b)}$$
$$\nu_1'(J_1) = \nu_1(J_1) \quad \text{(c)}$$

Then we have that $P(\nu_1' + \nu_2)$ is derivable.

$$\begin{aligned}(\nu_1' + \nu_2)(I_1) &= \nu_1'(I_1) + \nu_2(I_1) & \\ &\geq (K_2, ..., K_2) + \nu_2(I_1) & \text{by (b)} \\ &\geq (K_1, ..., K_1). & \text{by } (\alpha)\end{aligned}$$

So from ($\beta$) $\pi^K$ is admissible for $(\nu_1' + \nu_2)$ wrt $I_1$. Since $\nu_1'(J_1) = \nu_1(J_1)$ so by ($\theta$) $\pi^K$ is admissible for $\nu_1' + \nu_2$ wrt $J_1$. Since $I_1 \cup J_1 = \{1, ..., p\}$ so $\pi^K$ is admissible for $\nu_1' + \nu_2$. This means that $P(\nu_1' + \nu_2 + v(\pi^K))$ is derivable.

$$\begin{aligned}(\nu_1' + \nu_2 + v(\pi^K))(J) &= (\nu_1' + \nu_2)(J) & \text{by } (\Omega) \\ &= (\nu_1 + \nu_2)(J) & \text{by (c) and since } J \subseteq J_1 \\ &= \nu(J). &\end{aligned}$$

$$\begin{aligned}(\nu_1' + \nu_2 + v(\pi^K))(I_a) &= (\nu_1 + \nu_2 + v(\pi^K))(I_a) & \text{by (c) and since } J \subseteq J_1 \\ &\geq v(\pi^K)(I_a) & \\ &\geq (K, ..., K). & \text{by } (\zeta)\end{aligned}$$

$$\begin{aligned}(\nu_1' + \nu_2 + v(\pi^K))(I_1) &\geq ((K_2, ..., K_2) + \nu_2 + v(\pi^K))(I_1) & \text{by (b)} \\ &\geq ((K_1, ..., K_1) + v(\pi^K))(I_1) & \text{by } (\alpha) \\ &\geq (K, ..., K). & \text{by } (\gamma)\end{aligned}$$

Hence the required tuple is $\nu' = \nu_1' + \nu_2 + v(\pi^K)$.

(iii) Suppose $\delta$ is constructed using the Rule 3 of Definition 13. Let

$$\begin{aligned}I &= \{i \mid \nu(i) = \infty\} \\ I_1 &= \{i \mid \nu_1(i) = \infty\} \\ I_2 &= \{i \mid \nu_2(i) = \infty\} \\ J &= \{1, ..., p\} \setminus I \\ J_1 &= \{1, ..., p\} \setminus I_1 \\ J_2 &= \{1, ..., p\} \setminus I_2 \\ I_a &= I \setminus (I_1 \cup I_2)\end{aligned}$$

By induction hypothesis we have $\nu_1'$ and $\nu_2'$ such that $P_1(\nu_1')$ and $P_2(\nu_2')$ are derivable, $\nu_1'(i) = \nu_1(i)$ for all $i \in J_1$ and $\nu_2'(i) = \nu_2(i)$ for all $i \in J_2$.

For each $i \in I_a$, we define a path $\pi_i$ as follows. By definition of $I_a$ there is a node $N^i$ in $\delta_1$ or $\delta_2$, labeled with $P(\nu^i)$ such that

$$\nu^i \leq \nu_1 + \nu_2$$
$$\text{and } \nu^i(i) < \nu_1(i) + \nu_2(i).$$

Let $b_i \in \{1, 2\}$ be such that $N^i$ is in $\delta_{b_i}$. Let $\overline{b_i} = \{1, 2\} \setminus \{b_i\}$. Because of the induction hypothesis, the covering derivation $\delta_{b_i}$ satisfies the assumptions of Lemma 57. Hence from Lemma 57 we get a linear path $\pi_i'$ from $P$ to $P_{b_i}$ which is admissible for $\nu^i$ wrt $J_{b_i}$ and

$$(\nu^i + v(\pi_i'))(J_{b_i}) = \nu_{b_i}(J_{b_i}).$$

Let $\pi_i$ be the linear path obtained by concatenating $\pi_i'$ with the linear path

$$P_{b_i} \xrightarrow{(P(x+y)\Leftarrow P_{b_i}(x)\wedge P_{\overline{b_i}}(y), P_{\overline{b_i}}(\nu'_{\overline{b_i}}))} P\ .$$

$\pi_i$ is a well defined linear path. We claim that

$$\pi_i \text{ is admissible for } \nu_1+\nu_2 \text{ wrt } J_1 \cap J_2. \qquad (\kappa)$$

Since $J_1 \cap J_2 \subseteq J_{b_i}$, so $\pi'_i$ is admissible for $\nu^i$ wrt $J_1 \cap J_2$ and

$$(\nu^i + v(\pi'_i))(J_1 \cap J_2) = \nu_{b_i}(J_1 \cap J_2).$$

Since $\nu'_{\overline{b_i}}(J_{\overline{b_i}}) = \nu_{\overline{b_i}}(J_{\overline{b_i}})$ and $J_1 \cap J_2 \subseteq J_{\overline{b_i}}$, hence

$$\nu'_{\overline{b_i}}(J_1 \cap J_2) = \nu_{\overline{b_i}}(J_1 \cap J_2).$$

$$\begin{aligned}(\nu^i + v(\pi_i))(J_1 \cap J_2) &= (\nu^i + v(\pi'_i) + \nu'_{\overline{b_i}})(J_1 \cap J_2)\\ &= (\nu_{b_i} + \nu'_{\overline{b_i}})(J_1 \cap J_2)\\ &= (\nu_{b_i} + \nu_{\overline{b_i}})(J_1 \cap J_2)\\ &= (\nu_1 + \nu_2)(J_1 \cap J_2)\end{aligned}$$

Hence we have

$$(\nu^i + v(\pi_i))(J_1 \cap J_2) \geq 0. \qquad (\Lambda)$$

Hence $\pi_i$ is admissible for $\nu^i$ wrt $J_1 \cap J_2$. As $\nu^i \leq \nu_1 + \nu_2$, so $\pi_i$ is admissible for $\nu_1 + \nu_2$ wrt $J_1 \cap J_2$. This proves claim $(\kappa)$.

If $j \in J$ then $\nu^i(j) = (\nu_1 + \nu_2)(j)$, otherwise by construction of the covering derivation, we would have the contradiction that $\nu(j) = \infty$. Since $J \subseteq J_1 \cap J_2$, hence by $(\Lambda)$,

$$v(\pi_i)(J) = 0.$$

Also since $\nu^i \leq \nu_1 + \nu_2$ we have that

$$v(\pi_i)(J_1 \cap J_2) \geq 0.$$

Since $\nu^i(i) < (\nu_1 + \nu_2)(i)$, we have

$$v(\pi_i)(i) \geq 1.$$

Let

$$\pi = \Pi_{i \in I_a} \pi_i$$

Since each $\pi_i$ starts at $P$ and ends at $P$, the linear path $\pi$ is well defined. Since $v(\pi_i)(J_1 \cap J_2) \geq 0$ for each $i$, and each $\pi_i$ is admissible for $\nu_1 + \nu_2$ wrt $J_1 \cap J_2$, it is easy to see that $\pi$ is admissible for $\nu_1 + \nu_2$ wrt $J_1 \cap J_2$. Also we have

$$v(\pi)(J) = \textstyle\sum_{i \in I_a} v(\pi_i)(J) = \sum_{i \in I_a} 0 = 0.$$

Since $v(\pi_i)(i) \geq 1$ for each $i \in I_a$, we have

$$v(\pi)(I_a) \geq 1.$$

We can again see that $\pi^K$ is admissible for $\nu_1 + \nu_2$ wrt $J_1 \cap J_2$,

$$v(\pi^K)(J) = 0 \qquad (\alpha)$$

$$\text{and } v(\pi^K)(I_a) \geq (K, ..., K). \qquad (\beta)$$

Choose a $K_1 \geq 0$ such that

$$\text{for each prefix } \pi' \text{ of } \pi^K,\ ((K_1, ..., K_1) + v(\pi'))(I_1 \cup I_2) \geq 0 \qquad (\lambda)$$

$$\text{and } ((K_1, ..., K_1) + v(\pi^K))(I_1 \cup I_2) \geq (K, ..., K). \qquad (\theta)$$

By induction hypothesis we have $\nu''_1, \nu''_2$ such that

$$P_1(\nu''_1),\ P_2(\nu''_2) \text{ are derivable} \qquad \text{(a)}$$

$$\nu''_1(I_1), \nu''_2(I_2) \geq (K_1, ..., K_1) \qquad \text{(b)}$$

$$\nu''_1(J_1) = \nu_1(J_1) \text{ and } \nu''_2(J_2) = \nu_2(J_2) \qquad \text{(c)}$$

Then we have that $P(\nu_1'' + \nu_2'')$ is derivable. Since $\nu_1''(I_1) \geq (K_1, ..., K_1)$ so $(\nu_1'' + \nu_2'')(I_1) \geq (K_1, ..., K_1)$. Similarly $(\nu_1'' + \nu_2'')(I_2) \geq (K_1, ..., K_1)$. So we get

$$(\nu_1'' + \nu_2'')(I_1 \cup I_2) \geq (K_1, ..., K_1). \qquad (\gamma)$$

So from $(\lambda)$, we have that $\pi^K$ is admissible for $\nu_1'' + \nu_2''$ wrt $I_1 \cup I_2$. Since $\nu_1''(J_1) = \nu_1(J_1)$ and $\nu_2''(J_2) = \nu_2(J_2)$ so $(\nu_1''+\nu_2'')(J_1 \cap J_2) = (\nu_1+\nu_2)(J_1 \cap J_2)$. Since $\pi^K$ is admissible for $\nu_1+\nu_2$ wrt $J_1 \cap J_2$, so $\pi^K$ is admissible for $\nu_1''+\nu_2''$ wrt $J_1 \cap J_2$. Since $(I_1 \cup I_2) \cup (J_1 \cap J_2) = \{1, ..., p\}$ so $\pi^K$ is admissible for $\nu_1'' + \nu_2''$. This means that $P(\nu_1'' + \nu_2'' + v(\pi^K))$ is derivable.

Now

$$\begin{aligned}
(\nu_1'' + \nu_2'' + v(\pi^K))(J) &= (\nu_1'' + \nu_2'')(J) && \text{by } \alpha \\
&= (\nu_1 + \nu_2)(J). && \text{by (c), and since } J \subseteq J_1 \cap J_2 \\
&= \nu(J)
\end{aligned}$$

$$\begin{aligned}
(\nu_1'' + \nu_2'' + v(\pi^K))(I_a) &\geq v(\pi^K)(I_a) \\
&\geq (K, ..., K) && \text{by } \beta
\end{aligned}$$

$$\begin{aligned}
(\nu_1'' + \nu_2'' + v(\pi^K))(I_1 \cup I_2) &\geq ((K_1, ..., K_1) + v(\pi^K))(I_1 \cup I_2) && \text{by } \gamma \\
&\geq (K, ..., K) && \text{by } \theta
\end{aligned}$$

Hence the required tuple is $\nu' = \nu_1'' + \nu_2'' + v(\pi^K)$.

□

Now we are left to prove that there are only finitely many covering derivations. This is Theorem 41 below.

**Remark 1** *Let $\mathcal{V}$ be an extended VASS, $\delta$ a covering derivation, and $N_1$ and $N_2$ two nodes in $\delta$ such that $N_1$ is descendent of $N_2$. Let $N_1$ and $N_2$ be labeled by generalized configurations $P_1(\nu_1)$ and $P_2(\nu_2$. Then for any $i$, if $\nu_1(i) = \infty$ then $\nu_2(i) = \infty$.*

**Lemma 58** *Let $\mathcal{V}$ be an extended VASS. Given a covering derivation $\delta$ with root node $N$, and another $N'$ in it such that $N$ and $N'$ are labeled with the generalized configurations $P(\nu)$ and $P(\nu')$ (same $P$ in both cases) such that $\nu' < \nu$. Then $\nu$ has strictly more infinite coordinates than $\nu'$.*

**Proof:** Assume the contrary. Then from Remark 1, $\nu$ and $\nu'$ have the same infinite coordinates. Clearly $\delta$ must have been constructed by using Rule 2 or Rule 3 of Definition 13. Accordingly we have the following two cases :

– $N$ has a child $N_1$ labeled with generalized configuration $P_1(\nu_1)$ and $N$ is labeled with clause $P(x + \nu_2) \Leftarrow P_1(x)$. From Remark 1 we have that $\nu_1$ has same infinite coordinates as $\nu$. Then from the construction in Rule 2 of Definition 13, we must have $\nu = \nu_1 + \nu_1$. But then we have $\nu' \leq \nu_1 + \nu_2$ and for some $i$, $\nu'(i) < \nu_1(i) + \nu_2(i)$. Then we must have $\nu(i) = \infty$, whereas we have $\nu'(i) < \infty$ because of the fact that $\nu'(i) < \nu_1(i) + \nu_2(i)$. Hence we have a contradiction.

– $N$ is labeled with clause $P(x + y) \Leftarrow P_1(x), P_2(y)$ and has children $N_1$ and $N_2$ labeled with $P_1(\nu_1)$ and $P_2(\nu_2)$. Without loss of generality we assume $N'$ is descendant of $N_1$. Then from Remark 1 we have that $\nu_1$ and $\nu$ have the same infinite coordinates. Then from the construction in Rule 3 of Definition 13, we have that $\nu = \nu_1 + \nu_2$. But then we must have $\nu' \leq \nu_1 + \nu_2$ and for some $i$, $\nu'(i) < \nu_1(i) + \nu_2(i)$. Then we must have $\nu(i) = \infty$, whereas we have $\nu'(i) < \infty$ because of the fact that $\nu'(i) < \nu_1(i) + \nu_2(i)$. Hence we have a contradiction.

□

The height $H(\delta)$ of covering derivation $\delta$ is the height of the corresponding tree.

We define a partial order $<_d$ on covering derivations as $\delta_1 <_d \delta_2$ if $H(\delta_1) <_d H(\delta_2)$. Let $\prec$ be any total extension of $<_d$. e.g. $\delta_1 \prec \delta_2$ iff $H(\delta_1) < H(\delta_2)$ or $H(\delta_1) = H(\delta_2)$ and $\delta_1 \sqsubset \delta_2$ for some arbitrary total order $\sqsubset$ on derivations of the same height. Then it is clear that for any $\delta$, there are only finite number of covering derivations $\delta'$ such that $\delta' \prec \delta$. We define $\delta_1 \preceq \delta_2$ iff $\delta_1 \prec \delta_2$ or $\delta_1 = \delta_2$.

**Theorem 41** *There are only finitely many covering derivations. In particular they can be effectively computed.*

**Proof:** We will construct a forest of all possible derivations (i.e., each node in this forest will be a derivation.) A forest is a finite set of trees, whom we shall call *components*. This forest is constructed iteratively by adding one node at a time, using the following rules :

1. Each covering derivation constructed using the Rule 1 of Definition 13 is a root node. These are the only root nodes, so that we will have only a finite number of trees.
2. Assume $\delta$ is constructed using the derivation $\delta_1$ as defined in Rule 2 of Definition 13, and $\delta_1$ has been added in the forest, and $\delta$ has not been added. Then we add $\delta$ as a child of $\delta_1$.
3. Suppose $\delta_1$ and $\delta_2$ have been added in the forest, and $\delta$ is constructed using them as defined in Rule 3 of Definition 13 and has not been added in the forest. Let $\delta_1 \preceq \delta_2$, which we may assume wlog, as $\preceq$ is total. Then we add $\delta$ to the forest as a child of $\delta_2$.

It is clear that in this way all the derivations are added in the forest eventually. We claim that this process ends (i.e. the forest is finite.) Assume the contrary. Since the number of trees is finite, one of the trees would be infinite. Call it $T$.

We see that the trees are finitely branching. For any covering derivation $\delta$, the number of children created using Rule 2 above is limited by the number of clauses, and the number of children created using Rule 3 above is limited by the number of clauses times the (finite) number of $\delta_1$'s such that $\delta_1 \leq \delta$.

Then by König's lemma, there would be an infinite path in $T$. Consider the corresponding sequence of generalized configurations labeling the roots of the derivations.

First we note that if $P_1(\nu_1)$ occurs (not necessarily immediately) before $P_2(\nu_2)$ in this sequence, then it means that there is a covering derivation with root labeled $P_2(\nu_2)$ with some other node in the derivation labeled $P_1(\nu_1)$.

Since the number of predicates is finite, we would have an infinite subsequence of the form $P(\nu'_1), P(\nu'_2), ...$ for some $P$. Then we can find a subsequence $P(\nu_1), P(\nu_2), ...$ such that $\nu_i \leq \nu_{i+1}$ for all $i$, since $\leq$ is a wqo on $(\mathbb{N} \cup \{\infty\})^p$, an easy consequence of Dickson's lemma.

We cannot have $\nu_i = \nu_{i+1}$. Otherwise it would mean that the derivation $\delta$ corresponding to $P(\nu_{i+1})$ has a non-root node labeled with $P(\nu_i)$ $(= P(\nu_{i+1}))$. But then $\delta$ cannot be used further to create new derivations, which means that it would have no children in the forest, giving a contradiction.

Hence we have $\nu_i < \nu_{i+1}$ for all $i$. Consider the derivation corresponding to the generalized configuration $P(\nu_{i+1})$. The root $N$ has a descendent $N'$ labeled with atom $P(\nu_i)$. Then by Lemma 58, $\nu_{i+1}$ has strictly more infinite coordinates than $\nu_i$. This holds for each $i$, which gives us a contradiction. □

## 11.3 Adding Standard +-Push Clauses to $\mathbb{AC}$ Automata

Having studied the EVASS, now we ready to deal with standard +-push clauses in automata modulo $\mathbb{AC}$.

### 11.3.1 Constant-Only $\mathbb{AC}$ Automata with Standard $+$-Push Clauses

First of all we deal with the simplest of these automata, namely the constant-only $\mathbb{AC}$ automata extended by adding standard $+$-push clauses. From the discussion at the beginning of Section 11.2, this translation is almost straight-forward, except for a minor problem. Since we are working modulo $\mathbb{AC}$ instead of $\mathbb{ACU}$, we are not allowed to derive the empty summation in our automata, whereas in the extended VASS's we are allowed to derive the tuple $(0, ..., 0)$. Hence the translation requires some care. We assume that the constants in our signature are $a_1, \ldots, a_p$ so that the EVASS we will construct work on tuples in $\mathbb{N}^p$.

**Lemma 59** *Let $\mathcal{A}$ be an automata modulo $\mathbb{AC}$ containing clauses of constant-only $\mathbb{AC}$ automata, as well as standard $+$-push clauses. Then we can effectively construct an extended VASS $\mathcal{V}$, such that for every state $P$ in the automaton $\mathcal{A}$ :*

*(i) If the atom $P(\sum_{i=1}^{p} n_i a_i)$ is derivable in automaton $\mathcal{A}/\mathbb{AC}$ then the configuration $P(n_1, \ldots, n_p)$ is derivable in EVASS $\mathcal{V}$.*

*(ii) If the configuration $P(n_1, \ldots, n_p)$ is derivable in EVASS $\mathcal{V}$ then $(n_1, \ldots, n_p) > 0$ and the atom $P(\sum_{i=1}^{p} n_i a_i)$ is derivable in automaton $\mathcal{A}/\mathbb{AC}$.*

**Proof:** The states of $\mathcal{V}$ are of the form $P, P_{aux}, P^1_{aux}, \ldots, P^p_{aux}$ where $P$ is a state in $\mathcal{A}$. We add clauses to $\mathcal{V}$ corresponding to clauses of $\mathcal{A}$. For $+$-pop clause $P(x + y) \Leftarrow P_1(x) \wedge P_2(y) \in \mathcal{A}$, we add the addition clause $P(x+y) \Leftarrow P_1(x) \wedge P_2(y)$ in $\mathcal{V}$. For epsilon clauses $P(x) \Leftarrow P_1(x) \in \mathcal{A}$ we add the constant-addition clause $P(x + (0, \ldots, 0)) \Leftarrow P_1(x)$ in $\mathcal{V}$. For base clauses $P(a_i) \in \mathcal{A}$ we add the constant clause $P(0, \ldots, 0, 1, 0, \ldots, 0)$ where the '1' is present in the $i$th coordinate.

Finally we come to the interesting case, namely that of a standard $+$-push clause of the form $P(x) \Leftarrow Q(x + y) \in \mathcal{A}$. For $1 \leq i \leq p$ define $\nu_i = (0, \ldots, 0, 1, 0, \ldots, 0)$ and $\nu'_i = (0, \ldots, 0, -1, 0, \ldots, 0)$ where the '1' and '$-1$' have been put in the $i$th positions in $\nu_i$ and $\nu'_i$ respectively. For $1 \leq i \leq p$ we add the clauses

$$\begin{array}{rl} Q_{aux}(x + \nu'_i) & \Leftarrow Q(x) \\ Q_{aux}(x + \nu'_i) & \Leftarrow Q_{aux}(x) \\ Q^i_{aux}(x + \nu'_i) & \Leftarrow Q_{aux}(x) \\ P(x + \nu_i) & \Leftarrow Q^i_{aux}(x) \end{array}$$

to $\mathcal{V}$. The translation of the $+$-pop clauses, epsilon clauses and base clauses are self evident. Now we explain the translation of the standard $+$-push clause $P(x) \Leftarrow Q(x + y) \in \mathcal{A}$. This clause intuitively tells us to go from state $Q$ to state $P$ by deleting as many (but at least one) constants as we like, provided at least one constant is left in the term. The clauses $Q_{aux}(x+\nu'_i) \Leftarrow Q(x)$ for $1 \leq i \leq p$ allow us to go from $Q$ to $Q_{aux}$ by deleting exactly one constant $a_i$ for some $1 \leq i \leq p$, but in the process we may get the term 0 corresponding to the tuple $(0, \ldots, 0)$. The clauses $Q_{aux}(x + \nu'_i) \Leftarrow Q_{aux}(x)$ for $1 \leq i \leq p$ allow us to delete as many constants as we like, but remaining in the state $Q_{aux}$. In this process we may also get the term 0 corresponding to the tuple $(0, \ldots, 0)$. It is clear that now we just need to accept all the non-zero tuples of $Q_{aux}$ at $P$. For $1 \leq i \leq p$ the clause $Q^i_{aux}(x+\nu'_i) \Leftarrow Q_{aux}(x)$ allows us to go from $Q_{aux}$ to $Q^i_{aux}$ by decreasing the $i$th coordinate by 1. This step is possible only if $i$th coordinate of the concerned tuple in $Q_{aux}$ is strictly positive. Now to get back the original tuple of $_{aux}$ we have the clause $P(x+\nu_i) \Leftarrow Q^i_{aux}(x)$ which allows us to go from $Q^i_{aux}$ to $Q_{aux}$ by increasing the $i$th coordinate by 1. $\square$

This gives us a procedure for eliminating standard $+$-push clauses. For that we first we make the following observation :

**Remark 2** *Given any $\nu \in (\mathbb{N} \cup \infty)^p$, the set $L_<(\nu) = \{\nu' \in \mathbb{N}^p \mid (0, \ldots, 0) < \nu' < \nu\}$ is semilinear. We will use $L'_<(\nu)$ to denote the corresponding set $\mathbb{AC}(\{\sum_{i=1}^p n_i a_i \mid (n_1, ..., n_p) \in L_<(\nu)\})$.*

**Lemma 60** *Given an automaton $\mathcal{A}$ modulo $\mathbb{AC}$ containing constant-only $\mathbb{AC}$ automata clauses and standard $+$-push clauses, we can compute a constant-only $\mathbb{AC}$ automaton $\mathcal{B}$ such that for every state $P$ in automaton $\mathcal{A}$, $\mathcal{L}_P(\mathcal{A}/\mathbb{AC}) = \mathcal{L}_P(\mathcal{B}/\mathbb{AC})$.*

**Proof:** Using Lemma 59 we construct an EVASS $\mathcal{V}$ such that each state $P$ in $\mathcal{A}$ accepts exactly the terms $\sum_{i=1}^p n_i a_i$ such that $P(n_1, \ldots, n_p)$ is derivable in $\mathcal{V}$. From Theorems 39 and 40, if $P(x) \Leftarrow Q(x + y)$ is any clause then the set of terms accepted at $P$ using this clause is $L^Q_< = \bigcup L'_<(\nu)$ where the union is taken for all $\nu$ such that there is some covering derivation for $\mathcal{V}$, with root labeled by generalized configuration $Q(\nu)$. From Theorem 41, since the number of such $\nu$'s is finite, $L^Q_<$ is semilinear, and can be accepted at a new state $Q^\dagger$ using other new states, and only clauses of constant-only $\mathbb{AC}$ automata. Then we can replace the clause $P(x) \Leftarrow Q(x + y)$ by the clause $P(x) \Leftarrow Q^\dagger(x)$. $\square$

However note that while in the $\mathbb{ACU}$ case we are able to do this in polynomial time, in the $\mathbb{AC}$ case this algorithm is non-primitive recursive because the Karp-Miller construction (even for VASS) is not primitive recursive. The presence of standard $+$-push clauses is the only case in this thesis where the algorithms for automata modulo $\mathbb{AC}$ and $\mathbb{ACU}$ have such widely different complexities. Still we don't know of any case where the decidability question has different answers for automata modulo $\mathbb{AC}$ and $\mathbb{ACU}$. We conclude that

**Theorem 42** *The automata modulo $\mathbb{AC}$ containing constant-only $\mathbb{AC}$ automata clauses and standard $+$-push clauses accept exactly those semilinear sets which don't contain the term $0$.*

### 11.3.2 Reusing $\mathbb{AC}$ Derivations with Standard $+$-Push Clauses

We now prove an equivalent of Lemmas 21 and 23 which allow us to reuse $\mathbb{ACU}$ and $\mathbb{ACUD}$ derivations. We show how to reuse $\mathbb{AC}$ derivations which also involve standard $+$-push clauses. However the statement of result has a slightly different form which is more suited for the $\mathbb{AC}$ case. Given a $\mathbb{AC}$ derivation we show that some clause is a consequence of the clauses used in the derivation. Such a clause can then be used to derive new atoms. Hence the result indirectly shows what new atoms can be derived by reusing a given derivation.

**Lemma 61** *Let $S$ be a set of epsilon clauses (3.3), $+$-pop clauses (3.6), and standard $+$-push clauses (3.15). Let $\pi$ be a derivation of an atom $P(t)$ of the form*

$$\pi = \frac{\dfrac{\pi_1}{P_1(t_1)} \quad \ldots \quad \dfrac{\pi_n}{P_n(t_n)}}{P(t)} \; (S/\mathbb{AC})$$

*where each $\pi_i$ uses a free pop clauses as the last clause (in particular $t_i$ is functional). Then for some $1 \leq i_1 < \ldots < i_k \leq n$*

*(i)* $t =_{\mathbb{AC}} t_{i_1} + \ldots + t_{i_k}$

*(ii)* $S \models_{\mathbb{AC}} P(x_{i_1} + \ldots + x_{i_k}) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n)$.

*(iii) If no standard $+$-push clause (3.15) is in $S$ then $k = n$, i.e. $t =_{\mathbb{AC}} t_1 + \ldots + t_n$.*

**Proof:** We do induction on the size of the derivation $\pi$. We have the following cases :

(i) $n = 1$ and $\pi = \pi_1$. Clearly $t = t_1$ and $P = P_1$. Trivially we have $\models_{\mathbb{AC}} P_1(x_1) \Leftarrow P_1(x_1)$.

(ii)

$$\pi = \dfrac{\dfrac{\dfrac{\pi_1}{P_1(t_1)} \;\ldots\; \dfrac{\pi_m}{P_m(t_m)}}{P'(t')}\,(S/\mathbb{AC}) \quad \dfrac{\dfrac{\pi_{m+1}}{P_{m+1}(t_{m+1})} \;\ldots\; \dfrac{\pi_n}{P_n(t_n)}}{P''(t'')}\,(S/\mathbb{AC})}{P(t)}\,(P(x+y) \Leftarrow P'(x) \wedge P''(y)/\mathbb{AC})$$

Clearly $t =_{\mathbb{AC}} t' + t''$. By induction hypothesis we have $1 \leq i_1 < \ldots < i_p \leq m$, $t' =_{\mathbb{AC}} t_{i_1} + \ldots + t_{i_p}$, $m+1 \leq i_{p+1} < \ldots < i_k \leq n$, $t'' =_{\mathbb{AC}} t_{i_{p+1}} + \ldots + t_{i_k}$,

$$S \models_{\mathbb{AC}} P'(x_{i_1} + \ldots + x_{i_p}) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_m(x_m), \quad (*)$$

$$S \models_{\mathbb{AC}} P''(x_{i_{p+1}} + \ldots + x_{i_k}) \Leftarrow P_{m+1}(x_{m+1}) \wedge \ldots \wedge P_n(x_n). \quad (**)$$

Hence we have $1 \leq i_1 < \ldots < i_k \leq n$ and $t =_{\mathbb{AC}} t' + t'' =_{\mathbb{AC}} t_{i_1} + \ldots + t_{i_k}$. Since $P(x+y) \Leftarrow P'(x), P''(y) \in S$, from (*) and (**) we have $S' \models_{\mathbb{AC}} P(x_{i_1} + \ldots + x_{i_k}) \Leftarrow P_1(x_1), \ldots, P_n(x_n)$. Also if no standard $+$-push clause (3.15) is in $S$ then $t' =_{\mathbb{AC}} t_1 + \ldots + t_m$ and $t'' =_{\mathbb{AC}} t_{m+1} + \ldots + t_n$ so that $t =_{\mathbb{AC}} t_1 + \ldots + t_n$.

(iii)

$$\pi = \dfrac{\dfrac{\dfrac{\pi_1}{P_1(t_1)} \;\ldots\; \dfrac{\pi_n}{P_n(t_n)}}{P'(t')}\,(S/\mathbb{AC})}{P(t)}\,(P(x) \Leftarrow P'(x)/\mathbb{AC})$$

Clearly $t =_{\mathbb{AC}} t'$. By induction hypothesis we have $1 \leq i_1 < \ldots < i_k \leq n$ such that $t =_{\mathbb{AC}} t_{i_1} + \ldots + t_{i_k}$, and $S \models_{\mathbb{AC}} P'(x_{i_1} + \ldots + x_{i_k}) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_k(x_k)$. Since $P(x) \Leftarrow P'(x) \in S'$ we have $S' \models_{\mathbb{AC}} P(x_{i_1} + \ldots + x_{i_k}) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_k(x_k)$. Also, if no standard $+$-push clause (3.15) is in $S$ then by induction hypothesis we have $t =_{\mathbb{AC}} t_1 + \ldots + t_n$.

(iv)

$$\pi = \dfrac{\dfrac{\dfrac{\pi_1}{P_1(t_1)} \;\ldots\; \dfrac{\pi_n}{P_n(t_n)}}{P'(t')}\,(S/\mathbb{AC})}{P(t)}\,(P(x) \Leftarrow P'(x+y)/\mathbb{AC})$$

By induction hypothesis we have $1 \leq j_1 < \ldots < j_p \leq n$ such that $t' =_{\mathbb{AC}} t_{j_1} + \ldots + t_{j_p}$, and

$$S \models_{\mathbb{AC}} P'(x_{j_1} + \ldots + x_{j_p}) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n). \quad (*)$$

From the structure of the derivation, and since each $t_i$ is functional, we have $1 \leq i_1 < \ldots < i_k \leq n$ such that $\{i_1, \ldots, i_k\} \subsetneq \{j_1, \ldots, j_p\}$ and $t =_{\mathbb{AC}} t_{i_1} + \ldots + t_{i_k}$. Also $P(x) \Leftarrow P'(x+y) \in S$, hence from (*) we have $S \models_{\mathbb{AC}} P(x_{i_1} + \ldots + x_{i_k}) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n)$.

□

Note that the corresponding Lemmas 21, 22 and 23 for reusing derivations, do not have the condition that the last clause used by each $\pi$ should be a free pop clause. This condition is needed in Lemma 61 because of the standard $+$-push clauses (see (iv) in the proof of the Lemma).

We now define equivalent notion of functional support in the presence of standard $+$-push clauses for the $\mathbb{AC}$ case :

**Definition 16 ($\mathbb{AC}$-Standard Functional Support)** *Let $\pi$ be a derivation modulo $\mathbb{AC}$ in an automaton $\mathcal{A}$ containing one-way $\mathbb{AC}$ automata clauses as well as standard $+$-push clauses. If*

$$\pi = \cfrac{\cfrac{\pi_1}{P_1(t_1)} \quad \ldots \quad \cfrac{\pi_n}{P_n(t_n)}}{P(t)} \; (\mathcal{A}_{eq}/\mathbb{AC})$$

*where each $\pi_i$ uses a free pop clause at the root. Then the (unordered) list of atoms $P_1(t_1), \ldots, P_n(t_n)$ is called the* $\mathbb{AC}$-standard functional support *of* $\pi$.

From Lemma 61 we also have some $1 \leq i_1 < \ldots < i_k \leq n$ such that

(i) $t =_{\mathbb{AC}} t_{i_1} + \ldots + t_{i_k}$

(ii) $S \models_{\mathbb{AC}} P(x_{i_1} + \ldots + x_{i_k}) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n)$ where $S$ is the set of clauses used in $\pi$.

(iii) If no standard +-push clause (3.15) is in $S$ then $k = n$, i.e. $t =_{\mathbb{AC}} t_1 + \ldots + t_n$.

As for the $\mathbb{ACU}$-functional supports and $\mathbb{ACUD}$ functional supports, we observe that $\mathbb{AC}$-standard functional support is uniquely defined (upto $\mathbb{AC}$ equivalence on terms) for derivations modulo $\mathbb{AC}$ in automata modulo $\mathbb{AC}$ which contain clauses of one-way $\mathbb{AC}$ automata as well as standard +-push clauses.

We also have the following result :

**Corollary 4** *Let $S$ be a set of +-pop clauses (3.6), epsilon clauses (3.3) and standard +-push clauses (3.15). Then given a clause $C$ of the form*

$$C = P(x_{i_1} + \ldots + x_{i_k}) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n)$$

*where $x_1, \ldots, x_n$ are distinct variables, and $1 \leq i_1 < \ldots < i_k \leq n$, it is decidable whether $S \models_{\mathbb{AC}} C$.*

**Proof:** Introduce constants $a_1, \ldots, a_n$. Define automaton $\mathcal{A} = S \cup \{P_i(a_i) \mid 1 \leq i \leq n\}$. We claim that

$$S \models_{\mathbb{AC}} C \text{ iff } P(a_{i_1} + \ldots + a_{i_k}) \text{ is derivable in } \mathcal{A}/\mathbb{AC} \qquad (*)$$

The latter is decidable because from Theorem 42 $\mathcal{A}$ accepts a semilinear set.

Now we prove claim (*). If $S \models_{\mathbb{AC}} C$ then $\mathcal{A} \models_{\mathbb{AC}} C$. Also $\mathcal{A} \models_{\mathbb{AC}} P_i(a_i)$ for $1 \leq i \leq n$. Hence $\mathcal{A} \models_{\mathbb{AC}} P(a_{i_1} + \ldots + a_{i_k})$. Conversely suppose $\mathcal{A} \models_{\mathbb{AC}} P(a_{i_1} + \ldots + a_{i_k})$. The derivation $\pi$ of $P(a_{i_1} + \ldots + a_{i_k})$ in $\mathcal{A}/\mathbb{AC}$ has a functional support of the form $\underbrace{P_{j_1}(a_{j_1}), \ldots, P_{j_1}(a_{j_1})}_{n_1 \text{ times}}, \ldots, \underbrace{P_{j_l}(a_{j_l}), \ldots, P_{j_l}(a_{j_l})}_{n_l \text{ times}}$ where $1 \leq j_1 < \ldots < j_l \leq n$ and $n_i \geq 1$ for $1 \leq i \leq l$.. We have

$$\pi = \cfrac{\overline{P_{j_1}(a_{j_1})} \quad \ldots \quad \overline{P_{j_1}(a_{j_1})} \quad \ldots \quad \overline{P_{j_l}(a_{j_l})} \quad \ldots \quad \overline{P_{j_l}(a_{j_l})}}{P(a_{i_1} + \ldots + a_{i_k})} \; (S/\mathbb{AC})$$

By Lemma 61 we have $\{i_1, \ldots, i_k\} \subseteq \{j_1, \ldots, j_N\}$ and $S \models_{\mathbb{AC}} C_1$ where

$$C_1 = P(x_{i_1} + \ldots + x_{i_k}) \Leftarrow P_{j_1}(x_{j_1}) \wedge P_{j_1}(y_1^1) \wedge \ldots \wedge P_{j_1}(y_1^{n_1-1})$$
$$\wedge \ldots \wedge P_{j_l}(x_l) \wedge P_{j_l}(y_l^1) \wedge \ldots \wedge P_{j_l}(y_l^{n_l-1})$$

The variables in $C_1$ have been suitably named according to our convenience. Let $\sigma$ be a substitution which maps each $y_i^p$ to $x_i$ for $1 \leq i \leq l, 1 \leq p \leq n_i - 1$ and is the identity function on other variables. Since $S \models_{\mathbb{AC}} C_1$ we have $S \models_{\mathbb{AC}} C_1\sigma$. We have $C_1\sigma = P(x_{i_1} + \ldots + x_{i_k}) \Leftarrow P_{j_1}(x_{j_1}) \wedge \ldots \wedge P_{j_l}(x_l)$. Also since $\{j_1, \ldots, j_N\} \subseteq \{i, \ldots, n\}$ we have $\models P_1(x_1) \wedge \ldots \wedge P_n(x_n) \Rightarrow P_{j_1}(x_{j_1}) \wedge \ldots \wedge P_{j_l}(x_l)$. Hence we get $S \models_{\mathbb{AC}} C$. This proves claim (*). $\square$

### 11.3.3 Elimination of Standard + Push Clauses

We now show that adding standard + push clauses does not increase expressiveness of one-way automata. We have already proved this result in Section 11.3.1 for the case where all free symbols are constants, i.e. we have shown that the extension of constant-only $\mathbb{AC}$ automata by adding standard +-push clauses are as expressive as constant-only $\mathbb{AC}$ automata. We now consider the general case where we have free symbols of arbitrary arity. This case has already been considered for the $\mathbb{ACU}$ case in Section 11.1 where we showed standard +-push clauses can be eliminated in polynomial time from automata modulo $\mathbb{ACU}$ containing one-way $\mathbb{ACU}$ automata clauses as well as standard +-push clauses. We now show the same result, although we don't provide a polynomial time algorithm.

We let $\Sigma$ be a signature such that $+ \in \Sigma$ and $0, - \notin \Sigma$. Let $\mathcal{A}$ be an automaton modulo $\mathbb{ACU}$ containing one-way $\mathbb{AC}$ automata clauses and standard +-push clauses. $\mathcal{A}$ is assumed to be an automaton on signature $\Sigma$ with predicates from $\mathbb{P}$. We will construct an equivalent one-way $\mathbb{AC}$ automaton $\mathcal{C}$ which also contain extended epsilon clauses. However we know that extended epsilon clauses don't increase expressiveness of one-way equational tree automata.

Introduce a set $A = \{a_P \mid P \in \mathbb{P}\}$ of new constants. For each $S \subseteq \mathbb{P}$ define automaton $\mathcal{B}_S = \mathcal{A}_{eq} \cup \{P(a_P) \mid P \in S\}$. $\mathcal{B}_S$ is a constant-only $\mathbb{AC}$ automaton with standard +-push clauses. Hence $\mathcal{L}_P(\mathcal{B}_S/\mathbb{AC})$ is a semilinear set for each $P \in \mathbb{P}$. Therefore we can construct a constant-only $\mathbb{AC}$ automaton $\mathcal{A}_{P,S}$ with final state $F_{P,S}$ such that $\mathcal{L}_{F_{P,S}}(\mathcal{A}_{P,S}/\mathbb{AC}) = \mathcal{L}_P(\mathcal{B}_S/\mathbb{AC})$. We assume that the automata $\mathcal{A}_{P,S}$'s are all based on mutually disjoint sets of fresh states.

The required one-way automaton $\mathcal{C}$ consists of

1. the extended epsilon clause $P(x) \Leftarrow F_{P,S}(x) \wedge \bigwedge_{Q \in S} Q(x_Q)$ for each $P \in \mathbb{P}, S \subseteq \mathbb{P}$.
2. clauses of $\mathcal{A}_{P,S_{eq}}$ for each $P \in \mathbb{P}, S \subseteq \mathbb{P}$.
3. clause $Q(x) \Leftarrow R(x)$ for each constant clause $Q(a_R)$ in some $\mathcal{A}_P$.
4. free pop clauses $P(f(x_1, \ldots, x_n)) \Leftarrow P_1(x_1), \ldots, P_n(x_n)$ of $\mathcal{A}$.

**Lemma 62** *If $P(t)$ is derivable in $\mathcal{A}/\mathbb{AC}$ then it is derivable in $\mathcal{C}/\mathbb{AC}$.*

**Proof:** We do induction on the size of the derivation $\pi$ of of $P(t)$ in $\mathcal{A}/\mathbb{AC}$. Let $\pi$ have $\mathbb{AC}$-standard functional support $P_1(t_1), \ldots, P_n(t_n)$. We have

$$\pi = \dfrac{\begin{matrix}\vdots & & \vdots\\ P_1(t_1) & \ldots & P_n(t_n)\end{matrix}}{P(t)}\ (\mathcal{A}_{eq}/\mathbb{AC})$$

From Lemma 61 we have $1 \leq i_1 < \ldots < i_k \leq n$ such that $t =_{\mathbb{AC}} t_{i_1} + \ldots + t_{i_k}$ and $\mathcal{A}_{eq} \models_{\mathbb{AC}} P(x_{i_1} + \ldots + x_{i_k}) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n)$. Let $S = \{P_1, \ldots, P_n\}$. Since $\mathcal{A}_{eq} = \mathcal{B}_{S_{eq}}$ hence

$$\mathcal{B}_{S_{eq}} \models_{\mathbb{AC}} P(x_{i_1} + \ldots + x_{i_k}) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n). \quad (*)$$

Also the atoms $P_1(a_{P_1}), \ldots, P_n(a_{P_n})$ are derivable in $\mathcal{B}_S/\mathbb{AC}$. Hence from (*), $P(a_{P_{i_1}} + \ldots + a_{P_{i_k}})$ is derivable in $\mathcal{B}_S/\mathbb{AC}$. Hence $F_{P,S}(a_{P_{i_1}} + \ldots + a_{P_{i_k}})$ is derivable in $\mathcal{A}_{P,S}/\mathbb{AC}$. Since $\mathcal{A}_{P,S}$ has no clause standard +-push (3.14), by Lemma 61 this derivation has a functional support of the form $R_1(a_{P_{i_1}}), \ldots, R_k(a_{P_{i_k}})$ and $\mathcal{A}_{P,S_{eq}} \models_{\mathbb{AC}} F_{P,S}(x_1 + \ldots + x_k) \Leftarrow R_1(x_1) \wedge \ldots \wedge R_k(x_k)$. Since $\mathcal{A}_{P,S_{eq}} \subseteq \mathcal{C}_{eq}$ hence

$$\mathcal{C}_{eq} \models_{\mathbb{AC}} F_{P,S}(x_1 + \ldots + x_k) \Leftarrow R_1(x_1) \wedge \ldots \wedge R_k(x_k). \quad (**)$$

Also since the clause $R_j(a_{P_{i_j}}) \in \mathcal{A}_{P,S}$ hence $R_j(x) \Leftarrow P_{i_j}(x) \in \mathcal{C}$ for $1 \leq j \leq k$.

For $1 \leq i \leq n$ since $t_i$ is functional we have some free $f_i$ of arity $k_i$ and terms $t_i^1, \ldots, t_i^{k_i}$ such that $t_i = f_i(t_i^1, \ldots, t_i^{k_i})$. Since $P_i(t_i)$ is in the functional support of the derivation of $P(t)$, hence there is some clause $P_i(f_i(x_1, \ldots, x_{k_i})) \Leftarrow P_i^1(x_1), \ldots, P_i^{k_i}(x_{k_i})$ such that for $1 \leq j \leq k_i$, the atom $P_i^j(t_i^j)$ is derivable in $\mathcal{A}/\mathbb{AC}$ using a derivation strictly smaller than $\pi$. By induction hypothesis $P_i^j(t_i^j)$ is derivable in $\mathcal{C}/\mathbb{AC}$ for $1 \leq j \leq k_i$. Hence for $1 \leq i \leq n$ $P_i(t_i)$ is derivable in $\mathcal{C}/\mathbb{AC}$ using the clause $P_i(f_i(x_1, \ldots, x_{k_i})) \Leftarrow P_i^1(x_1) \wedge \ldots \wedge P_i^{k_i}(x_{k_i})$. Hence for $1 \leq j \leq k$, $R_j(t_{i_j})$ is derivable in $\mathcal{C}/\mathbb{AC}$ using the clause $R_j(x) \Leftarrow P_{i_j}(x)$.

Hence from (**), $F_{P,S}(t_{i_1} + \ldots + t_{i_k})$ (or $F_{P,S}(t)$) is derivable in $\mathcal{C}/\mathbb{AC}$. Since $S = \{P_1, \ldots, P_n\}$ hence the extended epsilon clause $P(x) \Leftarrow F_{P,S}(x) \wedge P_1(x_1) \wedge \ldots \wedge P_n(x_n) \in \mathcal{C}$. Also the atoms $P_1(t_1), \ldots, P_n(t_n)$ are derivable in $\mathcal{C}/\mathbb{AC}$. Hence using the extended epsilon clause, $P(t)$ is derivable in $\mathcal{C}/\mathbb{AC}$. $\square$

**Lemma 63** *For $P \in \mathbb{P}$, if $P(t)$ is derivable in $\mathcal{C}/\mathbb{AC}$ then it is derivable in $\mathcal{A}/\mathbb{AC}$.*

**Proof:** We do induction on the size of the derivation $\pi$ of $P(t)$ in $\mathcal{C}/\mathbb{AC}$. From examination of the clauses in $\mathcal{C}$, we have the following two cases :

(i)

$$\pi = \cfrac{\begin{array}{ccc}\vdots & & \vdots \\ P_1(t_1) & \ldots & P_n(t_n)\end{array}}{P(t)} \; (P(f(x_1, \ldots, x_n)) \Leftarrow P_1(x_1) \wedge \ldots \wedge P_n(x_n)/\mathbb{AC})$$

where $f$ is free. Clearly $t =_{\mathbb{AC}} f(t_1, \ldots, t_n)$. By induction hypothesis, the atoms $P_1(t_1), \ldots, P_n(t_n)$ are derivable in $\mathcal{A}/\mathbb{AC}$. Hence using the same free pop clause the atom $P(t)$ is derivable in $\mathcal{A}/\mathbb{AC}$.

(ii)

$$\pi = \cfrac{\cfrac{\pi'}{F_{P,S}(t)} \quad \begin{array}{c}\vdots \\ (Q(t_Q))_{Q \in S}\end{array}}{P(t)} \; (P(x) \Leftarrow F_{P,S}(x) \wedge \textstyle\bigwedge_{Q \in S} Q(x_Q)/\mathbb{AC})$$

By induction hypothesis

$$Q(t_Q) \text{ is derivable in } \mathcal{A}/\mathbb{AC} \text{ for } Q \in S. \qquad (*)$$

Let $t =_{\mathbb{AC}} t_1 + \ldots + t_n$ where each $t_i$ is functional. Since $\mathcal{C}$ has no standard +-push clauses, by examination of the clauses in $\mathcal{C}$, $\pi'$ has functional support of the form $R_1(t_1), \ldots, R_n(t_n)$ and we have

$$\pi' = \cfrac{\cfrac{\cfrac{\pi_1}{R_1(t_1)}}{Q_1(t_1)} \, (Q_1(x) \Leftarrow R_1(x)/\mathbb{AC}) \quad \ldots \quad \cfrac{\cfrac{\pi_n}{R_n(t_n)}}{Q_n(t_n)} \, (Q_n(x) \Leftarrow R_n(x)/\mathbb{AC})}{F_{P,S}(t)} \; (\mathcal{A}_{P,S_{eq}}/\mathbb{AC})$$

where each $\pi_i$ uses a free pop clause at the root. By induction hypothesis $R_i(t_i)$ is derivable in $\mathcal{A}/\mathbb{AC}$ for $1 \leq i \leq n$. For $1 \leq i \leq n$, since the clause $Q_i(x) \Leftarrow R_i(x) \in \mathcal{C}$, hence the clause $Q_i(a_{R_i}) \in \mathcal{A}_{P,S}$. Also from Lemma 61 and the structure of $\pi'$ we have $\mathcal{A}_{P,S_{eq}} \models_{\mathbb{AC}} F_{P,S}(x_1 + \ldots + x_n) \Leftarrow Q_1(x_1) \wedge \ldots \wedge Q_n(x_n)$. Hence the atom $F_{P,S}(a_{R_1} + \ldots + a_{R_n})$ is derivable in $\mathcal{A}_{P,S}/\mathbb{AC}$. Hence $P(a_{R_1} + \ldots + a_{R_n})$ is derivable in $\mathcal{B}_S/\mathbb{AC}$. By Lemma 61, this derivation has a functional support of the form $P_1(a_{P_1}), \ldots, P_n(a_{P_n}), Q_1(a_{Q_1}), \ldots, Q_p(a_{Q_p})$

$(p \geq 0)$ and $\mathcal{B}_{Seq} \models_{\mathbb{AC}} P(x_1 + \ldots + x_n) \Leftarrow P_1(x_1), \ldots, P_n(x_n) \wedge Q_1(y_1), \ldots, Q_p(y_p)$. By definition $\mathcal{B}_{eq} = \mathcal{A}_{eq}$, hence

$$\mathcal{A}_{eq} \models_{\mathbb{AC}} P(x_1 + \ldots + x_n) \Leftarrow P_1(x_1), \ldots, P_n(x_n) \wedge Q_1(y_1), \ldots, Q_p(y_p). \quad (**)$$

For $1 \leq i \leq p$ since the clause $Q_i(a_{Q_i}) \in \mathcal{B}_S$, hence $Q_i \in S$. Hence from (*) we have a term $t_{Q_i}$ such that $Q_i(t_{Q_i})$ is derivable in $\mathcal{A}/\mathbb{AC}$. So from (**), $P(t_1 + \ldots + t_n)$ (or $P(t)$) is derivable in $\mathcal{A}/\mathbb{AC}$.

□

If $P$ is the final state of $\mathcal{A}$ then we let $P$ be the final state of $\mathcal{C}$. From Lemmas 62 and 63, $\mathcal{L}_P(\mathcal{C}/\mathbb{AC}) = \mathcal{L}_P(\mathcal{A}/\mathbb{AC})$. We conclude that

**Theorem 43** *One-way $\mathbb{AC}$ tree automata extended by adding standard $+$ push clauses can be effectively reduced to equivalent one-way $\mathbb{AC}$ tree automata.*

### 11.3.4 Elimination of Free Push Clauses

We have seen that we can add standard $+$ push clauses to one-way $\mathbb{AC}$ automata without increasing their expressiveness. Now we show that we can further add free push clauses without increasing expressiveness, by showing how to eliminate the free push clauses. In other words this also means that we can add standard $+$-push clauses to two-way $\mathbb{AC}$ tree automata without increasing their expressiveness.

To eliminate the free push clauses from our automata, we use a saturation procedure that iteratively adds new epsilon clauses so that finally the free push clauses become redundant.

We first define one-step of the saturation procedure. Let $\mathcal{A}$ be an automaton modulo $\mathbb{AC}$ containing two-way $\mathbb{AC}$ automata clauses and standard $+$-push clauses. We assume that the predicates are from the set $\mathbb{P}$. Let $\mathcal{A}_{std}$ be the part of $\mathcal{A}$ without the free push clauses. ($\mathcal{A}_{std}$ is contains one-way $\mathbb{AC}$ automata clauses and standard $+$-push clauses.) Trivially $\mathcal{A}_{eq} \subseteq \mathcal{A}_{std}$.

If

1 $\mathcal{A}$ has a free push clause $R(x_i) \Leftarrow P(f(x_1, \ldots, x_n)), P_1^1(x_{i_1}) \wedge \ldots \wedge P^{n_1}(x_{i_1}) \ldots \wedge P_k^1(x_{i_k}) \wedge \ldots \wedge P^{n_k}(x_{i_k})$

2 $\mathcal{A}$ has a free pop clause $Q(f(x_1, \ldots, x_n)) \Leftarrow Q_1(x_1), \ldots, Q_n(x_n)$

3 for some clause $C = P(x) \Leftarrow Q(x), R_1(x_1), \ldots, R_p(x_p)$ $(p \geq 0)$ we have $\mathcal{A}_{eq} \models_{\mathbb{AC}} C$

4 for each $j \in \{1, \ldots, p\}$ there is some $s_j$ such that $R_j$ accepts $s_j$ in $\mathcal{A}_{std}/\mathbb{AC}$

5 for each $j \in \{1, \ldots, k\}$ there is some $t_{i_j}$ such that each of the states $Q_{i_j}, P_j^1, \ldots, P_j^{n_j}$ accept $t_{i_j}$ in $\mathcal{A}_{std}$.

6 for each $j \in \{1, \ldots, n\} \setminus \{i, i_1, \ldots, i_k\}$ there is some $t_j$ such that $Q_j$ accepts $t_j$ in $\mathcal{A}_{std}/\mathbb{AC}$.

then we write $\mathcal{A} \rhd \mathcal{A} \cup \{R(x_i) \Leftarrow Q_i(x_i)\}$, which is one step of our saturation procedure. Some remarks are necessary. Firstly in step (3), it is sufficient to consider the (finitely many) clauses in which $R_1, \ldots, R_p$ are mutually distinct. This is because given a clause of the form $C_1 = P(x) \Leftarrow Q(x) \wedge R_1(x_1) \wedge R_1(y_1^1) \ldots \wedge R_1(y_1^{N_1}) \wedge \ldots \wedge R_p(x_p) \wedge R_p(y_p^1) \ldots \wedge R_p(y_p^{N_p})$ let $C_2 = P(x) \Leftarrow Q(x) \wedge R_1(x_1) \wedge \ldots \wedge R_p^1(x_p)$. It is easy to check that $C_1 \models C_2$ and $C_2 \models C_1$. Secondly the condition $\mathcal{A}_{eq} \models_{\mathbb{AC}} C$ of step (3) can be decided using Corollary 4. Thirdly from Theorems 20 and 43 emptiness and intersection emptiness problems are decidable for one-way $\mathbb{AC}$ tree automata extended by adding standard $+$-push clauses. Hence we can effectively check whether $\mathcal{A} \rhd \mathcal{A} \cup \{R(x_i) \Leftarrow Q_i(x_i)\}$.

This saturation step is harmless :

**Lemma 64** *If $\mathcal{A} \rhd \mathcal{A} \cup \{R(x_i) \Leftarrow Q_i(x_i)\}$ as in the definition above, then any atom derivable in $\mathcal{A} \cup \{R(x_i) \Leftarrow Q_i(x_i)\}$ is also derivable in $\mathcal{A}$.*

**Proof:** It is sufficient to show that $R(t_i)$ is derivable in $\mathcal{A}/\mathbb{AC}$ assuming $Q_i(t_i)$ is derivable in $\mathcal{A}/\mathbb{AC}$. For $j \in \{1, \ldots, n\} \setminus \{i\}$ let $t_j$ be as above. For $j \in \{1, \ldots, p\}$ let $s_j$ be as above. $Q(f(t_1, \ldots, t_n))$ is derivable in $\mathcal{A}/\mathbb{AC}$ using the free pop clause. $P(f(t_1, \ldots, t_n))$ is derivable in $\mathcal{A}/\mathbb{AC}$ using the clause $P(x) \Leftarrow Q(x), R_1(x_1) \wedge \ldots \wedge R_p(x_p)$. $R(t_i)$ is derivable using the free push clause. □

The converse is trivially true. Hence $\mathcal{A}$ and $\mathcal{A} \cup \{R(x_i) \Leftarrow Q_i(x_i)\}$ have the same set of derivable atoms modulo $\mathbb{AC}$.

Given a two-way automaton $\mathcal{A}$ our saturation procedure consists of (don't care non-deterministically) generating a sequence $\mathcal{A}_0 (= \mathcal{A}) \rhd \mathcal{A}_1 \rhd \mathcal{A}_2 ...$ until non new clauses can be added. This always terminates because there are only a finite number of epsilon clauses possible. Let the final (saturated) automaton be $\mathcal{B}$. Then we remove the free push clauses from $\mathcal{B}$ to get $\mathcal{B}_{std}$. This step is also harmless :

**Lemma 65** *If any atom is derivable in $\mathcal{B}/\mathbb{AC}$ then it is derivable in $\mathcal{B}_{std}/\mathbb{AC}$.*

**Proof:** It is sufficient to show that a derivation in $\mathcal{B}$ which uses a free push clause only at the root and nowhere else, can be converted to a derivation in $\mathcal{B}_{std}$. Suppose we have a derivation of $R(t_i)$ using the free push clause $R(x_i) \Leftarrow P(f(x_1, \ldots, x_n) \wedge P_1^1(x_{i_1}) \wedge \ldots \wedge P_1^{n_1}(x_{i_1}) \wedge \ldots \wedge P_k^1(x_{i_k}) \wedge \ldots \wedge P_k^{n_k}(x_{i_k})$ as the last clause. Hence the atoms $P(f(t_1, \ldots, t_n)), P_1(t_{i_1}), \ldots, P_k(t_{i_k})$ are derivable in $\mathcal{B}_{std}/\mathbb{AC}$. From Lemma 61 the derivation of $P(f(t_1, \ldots, t_n))$ has a functional support of the form $Q(f(t_1, \ldots, t_n)), R_1(s_1), \ldots, R_p(s_p)$ $(p \geq 0)$ such that $\mathcal{B}_{eq} \models_{\mathbb{AC}} P(x) \Leftarrow Q(x), R_1(x_1), \ldots, R_p(x_p)$. The derivation of $Q(f(t_1, \ldots, t_n))$ uses some clause $Q(f(x_1, \ldots, x_n)) \Leftarrow Q_1(x_1), \ldots, Q_p(x_p)$ as the last clause. Hence $Q_1(t_1), \ldots, Q_n(t_n)$ are derivable in $\mathcal{B}_{std}/\mathbb{AC}$. Then we have that $\mathcal{B} \rhd \mathcal{B} \cup \{R(x_i) \Leftarrow Q_i(x_i)\}$. But $\mathcal{B}$ is saturated. Hence $R(x_i) \Leftarrow Q_i(x_i) \in \mathcal{B}$. Also $Q_i(t_i)$ is derivable in $\mathcal{B}_{std}/\mathbb{AC}$. Hence $R(t_i)$ is derivable in $\mathcal{B}_{std}/\mathbb{AC}$. □

The converse is trivially true. Hence the automaton $\mathcal{A}$ is equivalent to the automaton $\mathcal{B}_{std}$. Hence free push clauses can be effectively eliminated from a two-way $\mathbb{AC}$ automaton with standard +-push clauses, to get a one-way $\mathbb{AC}$ automaton with standard +-push clauses. We have also seen that standard +-push clauses don't increase the expressiveness of one-way $\mathbb{AC}$ automata. We conclude that

**Theorem 44** *Two-way $\mathbb{AC}$ tree automata extended by adding standard +-push clauses have the same expressiveness as one-way $\mathbb{AC}$ automata.*

### 11.3.5 Two-Way $\mathbb{ACU}$ Automata with Standard +-Push Clauses

We saw above that two-way $\mathbb{AC}$ automata extended by adding standard +-push clauses are as expressive as one-way $\mathbb{AC}$ automata. To do this, we used two results : firstly we showed that standard +-push clauses can be eliminated from one-way $\mathbb{AC}$ automata extended by adding standard +-push clauses. Secondly we showed we showed how free push clauses can be eliminated from two-way $\mathbb{AC}$ automata extended by adding standard +-push clauses.

Now as far as the $\mathbb{ACU}$ case is concerned, we have already shown the first of the two results : standard +-push clauses can be eliminated from one-way $\mathbb{ACU}$ automata extended by adding standard +-push clauses. For this we were actually able to give a polynomial time algorithm without taking a detour through EVASS, unlike in the $\mathbb{AC}$ case. However we have not yet looked at the second problem, that of two-way $\mathbb{ACU}$ automata with standard +-push clauses. However we observe that the construction and proofs of Section 11.3.4 can be easily modified to work for the $\mathbb{ACU}$ case. The presence or absence of an identity element does not play a crucial role in this translation. The only new ingredients to the proof are the notion of $\mathbb{AC}$-standard functional support, and Lemma 61 which allows us to reuse parts of derivations of a one-way $\mathbb{AC}$ automata extended by adding standard $+$-push clauses. It should be clear that both of them have their counterparts in the $\mathbb{ACU}$ case, i.e. we can define a notion of $\mathbb{ACU}$-standard functional support which takes into account standard $+$-push clauses, and we can also prove a result allowing us to reuse parts of $\mathbb{ACU}$ derivations involving standard $+$-push clauses. Without repeating the obvious constructions and proofs we merely state the result :

**Theorem 45** *Two-way $\mathbb{ACU}$ tree automata extended by adding standard $+$-push clauses have the same expressiveness as one-way $\mathbb{ACU}$ automata. In particular this class of automata accepts the same class of languages as one-way $\mathbb{ACU}$ automata.*

Also observe that the elimination of free push clauses in Section 11.3.4 does not involve taking any detours through EVASS. This means that to translate an automaton modulo $\mathbb{ACU}$ containing two-way automata clauses and standard $+$-push clauses to one-way $\mathbb{ACU}$ automata does not require the detours through EVASS which were used in the $\mathbb{AC}$ case. In particular this can be done in exponential time unlike the algorithm for the $\mathbb{AC}$ case which is not primitive recursive.

## 11.4 $+$-Push Clauses in $\mathbb{AC}$ and $\mathbb{ACU}$ Cases

We have seen that standard $+$-push clauses can be added to one-way and two-way $\mathbb{AC}$ and $\mathbb{ACU}$ automata without increasing their expressiveness. We started this Chapter with a discussion on $+$-push clauses, which are more powerful than standard $+$-push clauses, and we showed that these clauses can be trivially eliminated from $\mathbb{ACUX}$ and $\mathbb{ACUM}$ automata. However we don't have a similar result in the $\mathbb{AC}$ or $\mathbb{ACU}$ case. In fact the decidability of question for automata modulo $\mathbb{AC}$ and $\mathbb{ACU}$ with $+$-push clauses is still open.

Even in the very restricted case where all free symbols are constants, we don't know whether the emptiness of automata modulo $\mathbb{AC}$ or $\mathbb{ACU}$ containing $+$-push clauses is decidable. There is no obvious translation from such automata to EVASS, unlike in the case of standard $+$-push clauses where such a translation was possible. However it is possible to translate these automata to a further extension of EVASS with the following new kind of clauses :

$$P(x - y) \Leftarrow P_1(x) \wedge P_2(y)$$

Call these new clauses *subtraction clauses*. The semantics of this clause is : "if configurations $P_1(\nu_1)$ and $P_2(\nu_2)$ are reachable and $\nu_1 - \nu_2 \geq 0$ then the configuration $P(\nu_1 - \nu_2)$ is reachable." In the presence of such clauses, the constant-addition clauses become redundant since they can be encoded using addition and subtraction clauses. Hence we are left with the following set of clauses

$$\begin{array}{l} P(\nu) \text{ where } \nu \in \mathbb{N}^p \\ P(x + y) \Leftarrow P_1(x) \wedge P_2(y) \\ P(x - y) \Leftarrow P_1(x) \wedge P_2(y) \end{array}$$

With such a set of clauses, it can be seen that these EVASS with subtraction clauses are exactly constant-only $\mathbb{ACU}$ automata extended by adding $+$-push clauses. (The translation of automata modulo $\mathbb{AC}$ requires some care to avoid deriving the zero tuples.) However we don't know whether the construction of Karp-Miller trees can be further extended to deal with subtraction clauses.

While the questions whether emptiness and intersection emptiness of constant-only $\mathbb{ACU}$ automata is decidable is open, we do know that decidability of emptiness problem implies the decidability of intersection emptiness problem for such automata. This is because given an set $\mathcal{V}$ of EVASS $\mathcal{V}$ clauses and subtraction clauses, if we want to know whether two states $P_1$ and $P_2$ accept at least one common tuple, it suffices to add the clauses :

$$\begin{array}{l} P_3(x-y) \Leftarrow P_1(x) \wedge P_2(y) \\ P_0((0,\ldots,0)) \\ P(x-y) \Leftarrow P_0(x) \wedge P_3(y) \end{array}$$

Then there is some common tuple accepted at both $P_1$ and $P_2$ iff there is some tuple accepted at $P$. This result further shows the power of subtraction clauses which allows us to reduce the intersection-emptiness problem to emptiness problem, implying that the decidability proofs, if any exist, in this case would not be simple.

We also don't know whether adding these new clauses to EVASS strictly increases the expressivess of EVASS. Similarly while EVASS are more expressive than VASS, we don't know whether they are strictly more expressive.

# Chapitre 12

# Conclusion

Nous avons étudié les automates d'arbres bidirectionnels modulo plusieurs variantes de la théorie de l'associativité et commutativité. Les automates bidirectionnels nous donnent un mécanisme approprié pour la modélisation des protocoles cryptographiques. Notre motivation en ce qui concerne ces variantes équationnelles des automates d'arbres est de pouvoir modéliser des protocoles cryptographiques qui utilisent des primitives cryptographiques non parfaites. Les théories équationnelles sont utilisées pour modéliser les propriétés algébriques de ces primitives. Ce travail est la première étude générale des automates d'arbres équationnels unidirectionnels et bidirectionnels pour les théories $\mathbb{AC}$ et ses variantes. Alors qu'on a des résultats connus dans le cas de $\mathbb{AC}$ unidirectionnel, les extensions aux autres théories associatives et commutatives, comme la théorie du ou exclusif ou la théorie des groupes abéliens, n'ont jamais été considérées. Dans le cas bidirectionnel, il ne semble pas y avoir eu de publications étudiant les propriétés de décidabilité ou de clôture en présence de théories équationnelles. Les théories équationnelles que nous traitons sont $\mathbb{AC}$ (associativité et commutativité de $+$), $\mathbb{ACU}$ ($\mathbb{AC}$ avec unité $0$ de l'opération $+$), $\mathbb{ACUX}$ (la théorie du ou exclusif), $\mathbb{ACUX}_n$ (une généralisation de la théorie du ou exclusif, avec la règle d'annullation généralisée $nx = 0$ où $n \geq 2$), $\mathbb{ACUM}$ (la théorie des groupes abéliens), $\mathbb{ACUD}$ (la théorie d'un symbole $-$ distributif), et $\mathbb{ACUI}$ ($\mathbb{ACU}$ plus l'axiome $x + x = x$, c'est-à-dire la théorie de l'idempotence).

Inclues dans cette liste sont les théories $\mathbb{AC}$, $\mathbb{ACU}$, la théorie du ou exclusif et la théorie des groupes abéliens, qui figurent souvent en vérification de protocoles cryptographiques. Pour illustrer l'utilité des automates d'arbres équationnels en modélisation de protocoles cryptographiques, nous montrons la modélisation du protocole de Diffie-Hellman en groupe par des fragments décidables de nos automates d'arbres modulo les théories $\mathbb{ACU}$ et $\mathbb{ACUM}$ (la théorie de groupes abéliens). Notons que nous étudions aussi les théories $\mathbb{AC}$ et $\mathbb{ACUX}$ (la théorie d'ou exclusif) qui figurent souvent dans les protocoles cryptographiques.

Le cœur de cette thèse porte sur l´étude des propriétés algorithmiques des automates équationnels. Nous étudions si la vacuité de langages acceptés par nos automates d'arbres équationnels est décidable, si ces langages sont clos par les opérations booléennes, et si les variantes bidirectionnelles des automates d'arbres équationnels peuvent être réduits aux automates d'arbres équationnels unidirectionnels.

Nous avons d'abord montré que l'ajout de l'alternance aux automates d'arbres équationnels unidirectionnels entraîne l'indécidabilité du vide, dans les cas des théories $\mathbb{AC}$, $\mathbb{ACU}$, $\mathbb{ACUD}$ et $\mathbb{ACUM}$. En particulier, les automates d'arbres équationnels ont des propriétés remarquablement différentes de celles des automates d'arbres non équationnels où l'alternance est essentiellement bénigne. Ceci implique aussi que les automates bidirectionnels généraux modulo ces théories ont un problème du vide indécidable.

Du côté positif, nous avons montré que le vide des automates d'arbres équationnels unidirectionnels est décidable pour une théorie arbitraire. Nous avons aussi montré que les automates "constant-only" modulo les théories $\mathbb{AC}$, $\mathbb{ACU}$ et $\mathbb{ACUD}$ acceptent les ensembles semilinéaires modulo des codages.

En ce qui concerne les propriétés de clôture des automates équationnels unidirectionnels, nous avons montré que les automates unidirectionnels modulo toutes les théories ci-dessus sont clos par union et intersection. A l'opposé, les résultats de complémentation sont remarquablement différents. Nous avons montré alors que les automates d'arbres unidirectionnels modulo les théories $\mathbb{AC}$, $\mathbb{ACU}$, $\mathbb{ACUD}$ sont clos par complémentaire, ceux modulo les théories $\mathbb{ACUX}$, $\mathbb{ACUX}_n$, $\mathbb{ACUM}$ et $\mathbb{ACUI}$ ne le sont pas. Nous avons donné des contre-exemples dans ces derniers cas. Ces résultats suggèrent l'équivalence entre la linéarité de la théorie équationnelle et la clôture par complémentation des automates d'arbres équationnels unidirectionnels. Mais nous avons vu que cette équivalence était fausse en général. La clôture par intersection et la décidabilité du vide impliquent en plus la décidabilité du vide d'intersection et de l'appartenance.

À cause des résultats d'indécidabilité mentionnés ci-dessus, nous avons dû identifier une sous-classe appropriée des automates d'arbres équationnels bidirectionnels généraux, que nous avons appelés "automates d'arbres équationnels bidirectionnels". Mais cette classe est assez générale pour la modélisation de protocoles cryptographiques, ce qui est illustré par le fait que la modélisation du protocole de Diffie-Hellman en groupe mentionné ci-dessus se fait de façon exacte dans cette classe. Nous avons montré que modulo toutes les théories mentionnées ci-dessus sauf $\mathbb{ACUI}$, ces automates bidirectionnels peuvent être effectivement réduits aux automates unidirectionnels modulo les mêmes théories. (Le cas de $\mathbb{ACUI}$ est ouvert.) Par conséquent, les résultats de décidabilité et de clôture par opératons booléennes des automates unidirectionnels se généralisent à ces automates bidirectionnels. En particulier, le problème du vide d'intersection de ces automates bidirectionnels est décidable, ce dont nous avons besoin en vérification de protocoles cryptographiques, par exemple, dans la modélisation du protocole de Diffie-Hellman en groupe.

Les clauses push considérées dans ces automates bidirectionnels ne font apparaître que des symboles de fonction non équationnels (c'est-à-dire ne contenant pas le symbole $+$). Ensuite nous avons étudié l'effet de l'ajout de clauses push contenant $+$. Pour traiter les clauses $+$-push standard, nous avons dû définir une extension des systèmes d'addition de vecteurs à états (SAVE, ou VASS en anglais) [Reu89], que nous avons appelés VASS étendus (EVASS). Nous avons montré que la construction d'arbres de Karp-Miller pour les VASS, définis de sorte à calculer les limites de configurations accessibles des VASS peut se généraliser aux EVASS. Grâce à des traductions des automates contenant des clauses $+$-push standard en EVASS, nous montrons que l'on peut éliminer des clauses $+$-push standard des automates modulo $\mathbb{AC}$. Cependant cet algorithme n'est pas primitif récursif, parce que la construction de Karp et Miller (même pour les VASS) ne l'est pas. Cependant dans le cas $\mathbb{ACU}$ (à l'opposé de $\mathbb{AC}$), nous avons, de façon peut-être surprenante, pu éviter le pasaage par les EVASS. Par conséquent nous avons un algorithme seulement exponentiel dans le cas $\mathbb{ACU}$. Pour résumer, nous avons montré que les automates modulo $\mathbb{AC}$ et $\mathbb{ACU}$ qui contiennent des clauses d'automates bidirectionnels et clauses $+$-push standard peuvent être effectivement réduits aux automates unidirectionnels modulo $\mathbb{AC}$ et $\mathbb{ACU}$ respectivement. (Dans le cas de $\mathbb{ACUX}$ et $\mathbb{ACUM}$ ces clauses peuvent être trivialement éliminés.) Nous avons montré que les clauses $+$-push, qui sont plus générales que les clauses $+$-push standard, sont encore triviales dans le cas de $\mathbb{ACUX}$ et de $\mathbb{ACUM}$, mais sont difficiles dans le cas de $\mathbb{AC}$ et de $\mathbb{ACU}$, pour lesquelles la question de la décidabilité est aujourd'hui encore ouverte.

## Perspectives

Plusieurs problèmes ont été laissés ouverts dans cette thèse. Alors que nous avons montré que l'alternance entraîne l'indécidabilité pour les théories $\mathbb{AC}$, $\mathbb{ACU}$, $\mathbb{ACUD}$ et $\mathbb{ACUM}$, la question est ouverte pour les théories $\mathbb{ACUX}$, $\mathbb{ACUX}_n$ et $\mathbb{ACUI}$. La question de savoir si les automates bidirectionnels modulo $\mathbb{ACUI}$ (qui codent aussi l'alternance) peuvent être traduits dans les automates unidirectionnels modulo $\mathbb{ACUI}$ est aussi ouverte. La question de la décidabilité en présence des clauses $+$-push dans le cas de $\mathbb{AC}$ et de $\mathbb{ACU}$ est elle encore ouverte. Nous avons montré que ces derniers automates pouvaient se traduire en une extension des EVASS contenant des "clauses soustraction", mais la question de la décidabilité de ces dernières est ouverte, et inclut celle de l'accessibilité dans les réseaux de Petri. Donc, dans cette thèse nous avons vu deux extensions successives des VASS, mais la question de savoir si l'une des deux ou les deux sont strictement plus expressives que les VASS est ouverte. De même, alors que nous avons pu traiter les clauses $+$-push standard dans le cas de $\mathbb{ACU}$ sans le passage par les EVASS, ce qui nous permet de rester dans le domaine des algorithmes exponentiels, nous ne savons pas si l'on peut faire de même pour les automates $\mathbb{AC}$.

Finalement, une étude fine de la complexité des problèmes de décision étudiés dans cette thèse reste à faire, ainsi qu'une évaluation empirique de l'efficacité des algorithmes.

We have studied two-way equational tree automata for several variants of the theory of associativity and commutativity. Two-way tree automata provide a suitable mechanism for modeling of cryptographic protocols. The motivation behind defining these equational variants of tree automata is to be able to model cryptographic protocols which use non-perfect cryptographic primitives. Equational theories are used to model algebraic properties of these primitives. We believe that this work is the first general treatment of one-way and two-way equational tree automata for $\mathbb{AC}$-like theories. While there are known results in the one-way $\mathbb{AC}$ case, the extensions to other $\mathbb{AC}$ theories like the theory of exclusive-or or that of Abelian groups have not been considered so far. In the two-way case, we are not aware of any previous work dealing with decidability or closure properties in the presence of equational theories. The equational theories we deal with are $\mathbb{AC}$ (associativity and commutativity of a $+$ symbol), $\mathbb{ACU}$ ($\mathbb{AC}$ together with a unit 0 for the $+$ operator), $\mathbb{ACUX}$ (the theory of exclusive-or), $\mathbb{ACUX}_n$ (generalization of the theory of exclusive-or by the generalized cancellation rule $nx = 0$, where $n \geq 2$), $\mathbb{ACUD}$ (the theory of a distributive $-$ symbol), $\mathbb{ACUM}$ (the theory of Abelian groups) and $\mathbb{ACUI}$ ($\mathbb{ACU}$ with idempotence axiom $x + x = x$).

Included in this list are the theories $\mathbb{AC}$, $\mathbb{ACU}$, the theory of exclusive-or and the theory Abelian groups, which occur often in verification of cryptographic protocols. To illustrate the use of equational tree automata in modeling cryptographic protocols, we show the modeling of the group Diffie-Hellman protocol into decidable fragments of our tree automata modulo theories $\mathbb{ACU}$ and $\mathbb{ACUM}$ (Abelian groups theory). Note that we also deal with the theories $\mathbb{AC}$ and $\mathbb{ACUX}$ (the theory of exclusive-or) which occur frequently in cryptographic protocols.

The core of this thesis is concerned with algorithmic properties of equational tree automata. We study whether emptiness of languages accepted by our equational tree automata is decidable, whether these languages are closed under Boolean operations, and whether the two-way variants of equational tree automata can be reduced to one-way equational tree automata.

We first showed that adding alternation to one-way equational tree automata leads to undecidability of emptiness in theories $\mathbb{AC}$, $\mathbb{ACU}$ $\mathbb{ACUD}$ and $\mathbb{ACUM}$. In particular equational tree automata have remarkably different properties than non-equational tree automata in which alternation is harmless. This also means that general two-way automata modulo these theories have undecidable emptiness problem.

On the positive side we showed that emptiness of one-way equational tree automata is decidable for an arbitrary theory. Also we showed that the constant-only automata modulo the theories $\mathbb{AC}$, $\mathbb{ACU}$ and $\mathbb{ACUD}$ accept semilinear or Presburger-definable sets, modulo some encoding.

Coming to closure properties of one-way equational tree automata, we showed that one-way automata for all the theories above are closed under union and intersection. On the other hand the results on complementation are remarkably different. We showed that while the one-way tree automata modulo the theories $\mathbb{AC}$, $\mathbb{ACU}$ and $\mathbb{ACUD}$ are closed under complementation, those modulo the theories $\mathbb{ACUX}$, $\mathbb{ACUX}_n$, $\mathbb{ACUM}$ and $\mathbb{ACUI}$ are not closed under complementation. We gave counterexamples for the latter cases. These results suggest equivalence between linearity of the equational theory and closure under complementation of equational tree automata. However this equivalence does not work for all theories. Closure under intersection and decidability of emptiness further imply decidability of intersection-emptiness as well as of membership test.

Because of the undecidability results mentioned above, we identified a suitable subclass of the general two-way equational tree automata which we called "two-way equational tree automata". This class is however general enough to for purposes of modeling cryptographic protocols, illustrated by the fact that the modeling of the group Diffie-Hellman protocol mentioned above is done using this class. We showed that modulo all theories mentioned above except $\mathbb{ACUI}$, these two-way automata can be effectively reduced to one-way automata modulo the same theory. (The $\mathbb{ACUI}$ case is open.) As a result, the results about decidability and closure under Boolean operations of one-way automata also generalize to these two-way automata. In particular intersection-emptiness problem for these two-way automata is decidable, which is what is required in verification of cryptographic protocols, e.g. in the modeling of the group Diffie-Hellman protocol mentioned above.

The push clauses considered in these two-way automata involved only free functional symbols (i.e. they didn't contain the symbol $+$). Next we looked at the effect of adding push clauses involving $+$. In order to deal with standard $+$-push clauses, we needed to define a extension of Vector Addition Systems with States (VASS) which we called Extended VASS (EVASS). We showed that the usual construction of the Karp-Miller trees for VASS which is used to compute limits of the reachable configurations, can be extended to the case of EVASS. By translating automata with standard $+$-push clauses to EVASS, we were able to show how to eliminate standard $+$-push clauses from $\mathbb{AC}$ automata. However this algorithm is not primitive recursive because the Karp-Miller tree construction (even for VASS) is not primitive recursive. However in the $\mathbb{ACU}$ case (as against $\mathbb{AC}$ case), we were surprisingly able to avoid the detours through EVASS. As a result we have an exponential time algorithm in the $\mathbb{ACU}$ case. To sum up, we showed that $\mathbb{AC}$ and $\mathbb{ACU}$ automata which contain two-way automata clauses and standard $+$-push clauses can be effectively reduced to one-way automata $\mathbb{AC}$ and $\mathbb{ACU}$ automata respectively. (In the $\mathbb{ACUX}$ and $\mathbb{ACUM}$ cases these clauses can be trivially eliminated). We showed that $+$-push clauses, which are more general than standard $+$-push clauses are again trivial for the $\mathbb{ACUX}$ and $\mathbb{ACUM}$ theory, but are difficult in the $\mathbb{AC}$ and $\mathbb{ACU}$ cases for which the decidability problem is open.

## Perspectives

Several problems have been left open in this thesis. Although we have shown that alternation leads to undecidability for theories $\mathbb{AC}$, $\mathbb{ACU}$, $\mathbb{ACUD}$ and $\mathbb{ACUM}$, the question is left open for the theories $\mathbb{ACUX}$, $\mathbb{ACUX}_n$ and $\mathbb{ACUI}$. The question whether two-way $\mathbb{ACUI}$ tree automata (which also encode alternation) can be translated to one-way $\mathbb{ACUI}$ automata is also left open. The decidability problem in the presence of $+$-push clauses in the $\mathbb{AC}$ and $\mathbb{ACU}$ case has also been left open. These automata are shown to be translatable to an extension of EVASS with so-called subtraction clauses, however the decidability question for the latter is open and includes that of Petri net reachability. Hence in this thesis we have looked at two successive extensions of VASS, however the question whether any of the two is strictly more expressive than VASS is open. Also while we were able to deal with standard $+$-push clauses in the $\mathbb{ACU}$ case without going through EVASS, thus remaining in the realm of exponential algorithms, we do not know whether the same can be done for $\mathbb{AC}$ automata.

Finally, it would be interesting to study the precise complexity of the decision problems studied in this thesis, and to evaluate the efficiency of algorithms empirically.

# Bibliographie


[AG97] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols : The spi calculus. In *4th ACM Conference on Computer and Communications Security (CCS'97)*, pages 36–47, Zurich, Switzerland, April 1997. ACM Press.

[BAN89] Michael Burrows, Martín Abadi, and Roger Needham. A logic of authentication. *Proceedings of the Royal Society*, 426(1871) :233–271, December 1989.

[Bie90] Pierre Bieber. A logic of communication in a hostile environment. In Iliano Cervesato, editor, *3rd Computer Security Foundations Workshop (CSFW'90)*, pages 14–22, Franconia, New Hampshire, USA, June 1990. IEEE Computer Society Press.

[BJ97] Adel Bouhoula and Jean-Pierre Jouannaud. Automata-driven automated induction. In Glynn Winskel, editor, *12th Annual IEEE Symposium on Logic in Computer Science (LICS'97)*, pages 14–25, Warsaw, Poland, June 1997. IEEE Computer Society Press.

[Bol96] Dominique Bolignano. An approach to the formal verification of cryptographic protocols. In *3rd ACM Conference on Computer and Communication Security (CCS'96)*, pages 106–118, New Delhi, India, March 1996. ACM Press.

[CCM01] Hubert Comon, Véronique Cortier, and John Mitchell. Tree automata with one memory, set constraints and ping-pong protocols. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *28th International Colloquium on Automata, Languages, and Programming (ICALP'2001)*, volume 2076 of *LNCS*, pages 682–693, Crete, Greece, July 2001. Springer-Verlag.

[CDG+97] Hubert Comon, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Denis Lugiez, Sophie Tison, and Marc Tommasi. Tree automata techniques and applications. `http://www.grappa.univ-lille3.fr/tata`, 1997.

[CJ97] Hubert Comon and Florent Jacquemard. Ground reducibility is EXPTIME-complete. In Glynn Winskel, editor, *12th Annual IEEE Symposium on Logic in Computer Science (LICS'97)*, pages 26–34, Warsaw, Poland, June 1997. IEEE Computer Society Press.

[CKRT03] Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP decision procedure for protocol insecurity with XOR. In Phokion G. Kolaitis, editor, *18th Annual IEEE Symposium on Logic in Computer Science (LICS'03)*, pages 261–270, Ottawa, Canada, June 2003. IEEE Computer Society Press.

[CLC03] Hubert Comon-Lundh and Véronique Cortier. New decidability results for fragments of first-order logic and application to cryptographic protocols. In Robert Nieuwenhuis, editor, *14th International Conference on Rewriting Techniques and Applications (RTA'03)*, volume 2706 of *LNCS*, pages 148–164, Valencia, Spain, June 2003. Springer-Verlag.

[CLS03] Hubert Comon-Lundh and Vitaly Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In Phokion G. Kolaitis, editor, *18th IEEE*

*Symposium on Logic in Computer Science (LICS'03)*, pages 271–280, Ottawa, Canada, June 2003. IEEE Computer Society Press.

[Col02] Thomas Colcombet. Rewriting in the partial algebra of typed terms modulo AC. In Antonin Kucera and Richard Mayr, editors, *Electronic Notes in Theoretical Computer Science*, volume 68. Elsevier Science Publishers, 2002.

[Cor03] Véronique Cortier. *Vérification Automatique des Protocoles Cryptographiques*. Phd thesis, ENS Cachan, Cachan, France, 2003.

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6) :644–654, November 1976.

[DY83] Danny Dolev and Andrew C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, IT-29(2) :198–208, March 1983.

[DZL03] Silvano Dal Zilio and Denis Lugiez. XML schema, tree logic and sheaves automata. In Robert Nieuwenhuis, editor, *14th International Conference on Rewriting Techniques and Applications (RTA'2003)*, volume 2706 of *LNCS*, pages 483–498, Valencia, Spain, June 2003. Springer-Verlag.

[EJ88] E. Allen Emerson and Charanjit S. Jutla. The complexity of tree automata and logics of programs (extended abstract). In *29th Annual Symposium on Foundations of Computer Science (FOCS'88)*, pages 328–337, White Plains, New York, October 1988. IEEE.

[FSVY91] Thom Frühwirth, Ehud Shapiro, Moshe Y. Vardi, and Eyal Yardeni. Logic programs as types for logic programs. In *6th Annual IEEE Symposium on Logic in Computer Science (LICS'91)*, Amsterdam, The Netherlands, July 1991. IEEE Computer Society Press.

[Gen98] Thomas Genet. Decidable approximations of sets of descendants and sets of normal forms. In Tobias Nipkow, editor, *9th International Conference on Rewriting Techniques and Applications (RTA'98)*, volume 1379 of *LNCS*, pages 151–165, Tsukuba, Japan, 1998. Springer Verlag.

[GK99] Thomas Genet and Francis Klay. Rewriting for cryptographic protocol verification (extended version). Technical report, CNET-France Telecom, 1999. `http://www.loria.fr/~genet/Publications/GenetKlay-RR99.ps`.

[GL00] Jean Goubault-Larrecq. A method for automatic cryptographic protocol verification. In *Fifth International Workshop on Formal Methods for Parallel Programming : Theory and Applications (FMPPTA'2000)*, volume 1800 of *LNCS*, pages 977–984, Cancun, Mexico, May 2000. Springer Verlag.

[GLM97] Jean Goubault-Larrecq and Ian Mackie. *Proof Theory and Automated Deduction*, volume 6 of *Applied Logic Series*. Kluwer, 1997.

[GLV] Jean Goubault-Larrecq and Kumar Neeraj Verma. Alternating two-way AC-tree automata. In preparation. Available as Research Report LSV-02-11 at `http ://www.lsv.ens-cachan.fr/Publis/RAPPORTS_LSV/rr-lsv-2002-11.rr.ps`.

[GS66] Seymour Ginsburg and Edwin H. Spanier. Semigroups, Presburger formulas and languages. *Pacific Journal of Mathematic*, 16(2) :285–296, 1966.

[GS97] Ferenc Gécseg and Magnus Steinby. Tree languages. In Grzegorz Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, chapter 1, pages 1–68. Springer Verlag, 1997.

[Han94] Michael Hanus. The integration of functions into logic programming : From theory to practice. *Journal of Logic Programming*, 19 & 20 :583–628, 1994.

[HU79] John E. Hopcroft and Jeffery D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

[ISD+02] Oscar H. Ibarra, Jianwen Su, Zhe Dang, Tevfik Bultan, and Richard A. Kemmerer. Counter machines and verification problems. *Theoretical Computer Science*, 289(1) :165–189, 2002.

[KFK97] Yuichi Kaji, Toru Fujiwara, and Tadao Kasami. Solving a unification problem under constrained substitutions using tree automata. *Journal of Symbolic Computation*, 23(1) :79–117, January 1997.

[KNW03] D. Kapur, P. Narendran, and L. Wang. An E-unification algorithm for analyzing protocols that use modular exponentiation. In Robert Nieuwenhuis, editor, *14th International Conference on Rewriting Techniques and Applications (RTA'03)*, volume 2706 of *LNCS*, pages 165–179, Valencia, Spain, June 2003. Springer-Verlag.

[Li88] P. Li. Pattern matching in trees. Master's Thesis CS-88-23, University of Waterloo, May 1988.

[LM94] Denis Lugiez and J. L. Moysset. Tree automata help one to solve equational formulae in AC-theories. *Journal of Symbolic Computation*, 18(4) :297–318, 1994.

[Low95] Gavin Lowe. An attack on the Needham-Schroeder public-key protocol. *Information Processing Letters*, 56(3) :131–133, 1995.

[Low96] Gavin Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In Tiziana Margaria and Bernhard Steffen, editors, *Second International Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, volume 1055 of *LNCS*, pages 147–166, Passau, Germany, March 1996. Springer Verlag.

[Lug98] Denis Lugiez. A good class of tree automata. Application to inductive theorem proving. In Kim Guldstrand Larsen, Sven Skyum, and Glynn Winskel, editors, *25th International Colloquium on Automata, Languages, and Programming (ICALP'98)*, volume 1443 of *LNCS*, pages 409–420, Aalborg, Denmark, July 1998. Springer Verlag.

[Lug03] Denis Lugiez. Counting and equality constraints for multitree automata. In Andrew D. Gordon, editor, *6th International Conference on Foundations of Software Science and Computational Structures (FoSSaCS'03)*, volume 2620 of *LNCS*, pages 328–342, Warsaw, Poland, 2003. Springer Verlag.

[MCJ97] Will Marrero, Edmund M. Clarke, and S. Jha. Model checking for security protocols. Technical Report CMU-SCS-97-139, Carnegie Mellon University, 1997.

[MD02] Jon Millen and Grit Denker. CAPSL and muCAPSL. *Journal of Telecommunications and Information Technology*, 4 :16–27, 2002.

[Mea92] Cathy A. Meadows. Applying formal methods to the analysis of a key management protocol. *Journal of Computer Security*, 1(1) :5–36, 1992.

[Mea96] Catherine A. Meadows. The NRL protocol analyzer : An overview. *Journal of Logic Programming*, 26(2) :113–131, 1996.

[Min61] Marvin 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.

[Mon99] David Monniaux. Abstracting cryptographic protocols with tree automata. In Agostino Cortesi and Gilberto Filé, editors, *6th International Static Analysis Symposium (SAS'99)*, volume 1694 of *LNCS*, pages 149–163, Venice, Italy, September 1999. Springer-Verlag.

[New42] M. H. A. Newman. On theories with a combinatorial definition of equivalence. *Annals of Mathematics*, 43(2) :223–242, 1942.

[Ohs01] Hitoshi Ohsaki. Beyond regularity : Equational tree automata for associative and commutative theories. In Laurent Fribourg, editor, *10th Annual Conference of the European Association for Computer Science Logic (CSL'01)*, volume 2142 of *LNCS*, pages 539–553, Paris, France, September 2001. Springer Verlag.

[OST03] Hitoshi Ohsaki, Hiroyuki Seki, and Toshinori Takai. Recognizing boolean closed A-tree languages with membership conditional rewriting mechanism. In Robert Nieuwenhuis, editor, *14th International Conference on Rewriting Techniques and Applications (RTA'03)*, volume 2706 of *LNCS*, pages 483–498, Valencia, Spain, June 2003. Springer-Verlag.

[OT02] Hitoshi Ohsaki and Toshinori Takai. Decidability and closure properties of equational tree languages. In Sophie Tison, editor, *13th International Conference on Rewriting Techniques and Applications (RTA'02)*, volume 2378 of *LNCS*, pages 114–128, Copenhagen, Denmark, July 2002. Springer-Verlag.

[Par66] Rohit J. Parikh. On context-free languages. *Journal of the ACM*, 13(4) :570–581, October 1966.

[Pau97] Lawrence C. Paulson. Mechanized proofs for a recursive authentication protocol. In Iliano Cervesato, editor, *10th IEEE Computer Security Foundations Workshop (CSFW'97)*, pages 84–95, Rockport, Massachusetts, USA, June 1997. IEEE Computer Society Press.

[Pau98] Larry C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6 :85–128, 1998.

[Pel97] Nicolas Peltier. Tree automata and automated model building. *Fundamenta Informaticae*, 30(1) :59–81, 1997.

[PQ01] Olivier Pereira and Jean-Jacques Quisquater. A security analysis of the cliques protocols suites. In Iliano Cervesato, editor, *14th IEEE Computer Security Foundations Workshop (CSFW'2001)*, pages 73–81, Cape Breton, Novia Scotia, Canada, June 2001. IEEE Computer Society Press.

[Reu89] Christophe Reutenauer. *Aspects Mathématiques des Réseaux de Pétri*. Masson, 1989.

[RS98] P. Ryan and S. Schneider. An attack on a recursive authentication protocol : A cautionary tale. *Information Processing Letters*, 65(1) :7–10, 1998.

[Slu85] G. Slutzki. Alternating tree automata. *Theoretical Computer science*, 41 :305–318, 1985.

[STW00] Michael Steiner, Gene Tsudik, and Michael Waidner. Key agreement in dynamic peer groups. *IEEE Transactions on Parallel and Distributed Systems*, 11(8) :769–780, 2000.

[Tho90] Wolfgang Thomas. Automata on infinite objects. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 4, pages 133–191. Elsevier Science, 1990.

[Var98] Moshe Y. Vardi. Reasoning about the past with two-way automata. In Kim Guldstrand Larsen, Sven Skyum, and Glynn Winskel, editors, *25th International Colloquium on Automata, Languages, and Programming (ICALP'98)*, volume 1443 of *LNCS*, pages 628–641, Aalborg, Denmark, July 1998. Springer Verlag.

[Ver03a] Kumar Neeraj Verma. On closure under complementation of equational tree automata for theories extending AC. In Moshe Vardi and Andrei Voronkov, editors, *10th International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR'03)*, volume 2850 of *LNAI*, pages 183–195, Almaty, Kazakhstan, September 2003. Springer Verlag.

[Ver03b] Kumar Neeraj Verma. Two-way equational tree automata for AC-like theories : Decidability and closure properties. In Robert Nieuwenhuis, editor, *14th International Conference on Rewriting Techniques and Applications (RTA'2003)*, volume 2706 of *LNCS*, pages 180–196, Valencia, Spain, June 2003. Springer Verlag.