Applying Abstract Algebraic Logic to Classical Automata Theory: an exercise

Luís Descalço[1], Alexandre Madeira[2,1,3] and Manuel A. Martins[1]

[1] Department of Mathematics, University of Aveiro
[2] CCTC, University of Minho
[3] Critical Software S.A

Abstract. In [4], Blok and Pigozzi have shown that a deterministic finite automaton can be naturally viewed as a logical matrix. Following this idea, we use a generalisation of the matrix concept to deal with other kind of automata in the same algebraic perspective. We survey some classical concepts of automata theory using tools from algebraic logic. The novelty of this approach is the understanding of the classical automata theory within the standard abstract algebraic logic theory.

1 Introduction

There are several works in the literature about automata theory (AT) supported by results and arguments of different areas, such as, topology, algebra, quantum mechanics, etc. Following the recent works applying some tools of abstract algebraic logic (AAL) to computer sciences, namely in the formal specification and verification theory of software systems (cf. [10–12]), we survey some classical concepts of AT within an AAL perspective. AAL is an area of algebraic logic that focuses on the study of the relationship between logical equivalence and logical truth. More precisely, AAL is centered on the process of associating a class of algebras to a logical system (see [6]). A logical system, a *deductive system* as it has been called in the AAL field, is a pair formed by a signature Σ and a substitution-invariant consequence relation on the set of terms over Σ. Using deductive systems (more precisely, k-deductive systems) we can deal with sentential logics, first-order logic, equational logic and the logic of partially ordered algebras, as parts of a single unified theory (see [4]). The main paradigm in AAL is the representation of the classical propositional calculus in the equational theory of Boolean algebras by means of the so called *Lindenbaum-Tarski process*. In its traditional form, the Lindenbaum-Tarski process relies on the fact that the classical propositional calculus has a biconditional "$\leftrightarrow$" that defines logical equivalence. The set of all formulas is partitioned into logical equivalence classes and then the familiar algebraic process of forming the quotient algebra, called the *Lindenbaum-Tarski algebra*, is applied.

Blok and Pigozzi had suggested in [4] that a deterministic finite automaton can be seen as a matrix, i.e., as an algebra with a subset of designated elements. In this paper we extend this perspective to other kind of automata, namely to *Moore machines*, *Mealy machines*, *finite nondeterministic automata* and to a class of *pushdown automata*. In order to achieve this, we generalise the notion of matrix to the k-dimensional and multi-sorted case, called here a *k-dimensional state machine*. An interesting aspect of these formalisations is the fact that the concepts of equivalence between states of an automaton and minimised automata may be defined uniformly for all these automata. For example, the traditional equivalence of states in all of these automata coincides with the

Leibniz congruence defined over its corresponding k-state machines. We also sketch how *N-deterministic matrix theory* can be used to deal with finite nondeterministic automata. The theory of N-deterministic matrices has been intensively developed by A. Avron et al. (eg. [2]).

Our approach provides a different perspective from the classical semigroup-theoretic approach which looks at finite monoids (or semigroups) obtained as the quotient of the free monoid A^* by a congruence of finite index, where each element represents a congruence class of finite words and monoid multiplication corresponds to concatenation of class representatives ([9]). The logic flavor of our approach carries out also a distinct perspective from the classical relationship between generalized finite automata and Σ-algebras based on the notion of tree automata as a generalization of finite automata ([14] and [15]).

1.1 Preliminaries

In this Section, we recall some notions from *universal sorted algebra* and *abstract algebraic logic*. As a survey on these concepts we suggest [6, 16].

Universal (sorted) algebra preliminaries Let S be a non empty set whose elements are called *sorts*. An *S-sorted set* is an S-indexed family of sets $A = (A_s)_{s \in S}$. A is *nonempty* if $A_s \neq \emptyset$ for each $s \in S$. We say that an S-sorted set A is *locally finite (locally countable infinite)* if, for any $s \in S$, A_s is finite (countable infinite).

A *binary S-relation* R in an S-sorted set A consists of an S-sorted set of relations $R_s \subseteq A_s \times A_s$. R is an equivalence relation if each R_s is an equivalence relation on A. Given an element $a \in A_s$ and an equivalence relation R, we define the *equivalence class of a modulo R* as the set $a/R_s = \{b \in A_s | aR_sb\}$. The *quotient A by* an equivalence relation R is the S-sorted set $A/R = (A_s/R_s)_{s\in S}$ such that $(A/R)_s = \{a/R_s | a \in A_s\}$. A *signature Σ* is a pair $(S, \mathcal{F})$, where S is a set (of sort names) and $\mathcal{F}$ is a $(S^* \times S)$-sorted set (of operation names), where S^* is the set of all finite sequences of S elements. As usual we write $f : s_1, \cdots, s_n \to s \in \Sigma$ to mean that $s_1, \cdots, s_n, s \in S$ and $f \in \mathcal{F}_{s_1 \ldots s_n, s}$. We say that a signature $\Sigma = (S, \mathcal{F})$ is a *sentential signature* when S is a singleton. A *Σ-algebra* consists of an S-sorted set $A = (A_s)_{s\in S}$, with A nonempty, together with a function $f^A : A_{s_1} \times \cdots \times A_{s_n} \to A_s$ for each $f : s_1, \cdots, s_n \to s \in \Sigma$. The class of all Σ-algebras will be denoted by $Alg(\Sigma)$.

Given a signature $\Sigma = (S, \mathcal{F})$ and a Σ-algebra A, a *Σ-congruence on A* is an S-family of symmetric, transitive and reflexive non empty binary relations on $A \approx = (\approx_s)_{s \in S}$ such that, for any operation symbol $f : s_1, \cdots, s_n \to s \in \Sigma$ and for all $a_i, b_i \in A_{s_i}$, $1 \leq i \leq n$ if $a_1 \approx_{s_1} b_1, \ldots, a_n \approx_{s_n} b_n$, then $f^A(a_1, \ldots, a_n) \approx_s f^A(b_1, \ldots, b_n)$. The *quotient of A by* $\approx$ defined as usual and is denoted by $A/\approx$.

A *homomorphism* between two Σ-algebras A and B is a mapping $h : A \to B$, between the correspondent carrier sets, satisfying for each $f : s_1, \ldots, s_n \to s \in \Sigma$, $h(f^A_s(a_1, \ldots, a_n)) = f^B(h_{s_1}(a_1), \ldots, h_{s_n}(a_n))$ for all $a_i \in A_{s_i}, 1 \leq i \leq n$. An injective homomorphism f is called a *monomorphism.* If f is surjective, it is called an *epimorphism.* We say that h is an *isomorphism* if it is both injective and surjective.

An S-sorted set of *variables* is an S-sorted set X of pairwise disjoint sets. The elements in X_s are called *s-variables.* To say that a variable x is of sort s we write $x\!:\!s$. A set of variables for a signature $\Sigma = (S, \mathcal{F})$ is an S-sorted set of variables. Throughout the paper we distinguish, as usual, the variables from the symbols in $\mathcal{F}$. The set of *Σ-terms in the variables X* is defined as usual. We write $t(x_1, \ldots, x_n)$ to mean that the variables which occur in t are among $x_1, \dots, x_n$. In the case where $X = \emptyset$ ($X_s = \emptyset$ for all $s \in S$), we write T_Σ for $\mathrm{T}_\Sigma(\emptyset)$. The terms in T_Σ, i.e., without variables, are called *ground terms*. It is well know that if $\mathrm{T}_\Sigma(X)$ is nonempty for each sort $s \in S$, then we can define, in a standard way, operations over it to obtain a Σ-algebra which we call the *term algebra*; *algebra of ground terms* when $X = \emptyset$.

An *assignment* $h : X \to A$ is an S-family of mappings $(h_s : X_s \to A_s)_{s \in S}$. It is well known that, when $\mathrm{T}_\Sigma(X)$ is an algebra, any assignment uniquely extends to a Σ-homomorphism $\bar{h} : \mathrm{T}_\Sigma(X) \to A$; in the sequel we will write simply h instead of $\bar{h}$. Given a term $t(x_1, \dots, x_n)$ and a assignment $h : X \to A$ such that $h(x_i) = a_i$, $1 \le i \le n$, we denote $h(t)$ by $t^A(a_1, \dots, a_n)$ (in the sequel, the superscript A may be omitted). We say that an algebra A is *reachable* if for any $a \in A$ there is a ground term t such that $t^A = a$. It is not difficult to see that A is reachable iff it has no proper subalgebras. The algebra of ground terms T_Σ, when it exists, is initial in the category of Σ-algebras.

AAL preliminaries. In order to define formulas in a given signature, for each set of sorts S, we fix a locally countable infinite S-sorted set of variables X. Given a signature Σ with set of sorts S, a *Σ-formula* (a *formula* for short) is just a Σ-term in the set of variables X. The set of all formulas is denoted by Fm_Σ.

In this section we deal with sentential signatures. Let us fix a sentential signature Σ.

Definition 1. *A* matrix *over a signature Σ (a Σ-*matrix *for short) is a pair $\mathcal{A} = \langle A, F \rangle$, where A is a Σ-algebra and $F \subseteq A$. A is called the* underlying algebra *of $\mathcal{A}$ and F is called the* filter *of $\mathcal{A}$.*

Given an Σ-matrix $\mathcal{A} = \langle A, F \rangle$ and set of formulas $\Gamma \cup \{\varphi\} \subseteq \mathrm{Fm}_\Sigma$, we say that φ is a (semantical) consequence of Γ in $\mathcal{A}$, in symbols $\Gamma \models_\mathcal{A} \varphi$ if for any assignment $h : X \to A$, $h(\Gamma) \subseteq F$ implies $h(\varphi) \in F$. When $\Gamma = \emptyset$ we say that φ is *valid* in $\mathcal{A}$. The set of all valid formulas in $\mathcal{A}$ is denoted by $\mathrm{Th}(\mathcal{A})$. The subset of $\mathrm{Th}(\mathcal{A})$ of all ground terms will play an important role in the next sections and it will be denoted by $L(\mathcal{A})$.

Definition 2 (Leibniz Congruence). *Let $\mathcal{A} = \langle A, F \rangle$ be a Σ-matrix. The* Leibniz congruence *of $\mathcal{A}$, in symbols $\Omega_\mathcal{A}(F)$, is the relation defined as follows:*

$$\Omega_\mathcal{A}(F) = \{(a,b) \in A \times A \,|\, \forall \varphi(z, x_0, \dots, x_{k-1}) \in \mathrm{Fm}_\Sigma, \forall c_0, \dots, c_{k-1} \in A,$$
$$\varphi^A(a, c_0, \cdots, c_{k-1}) \in F \Leftrightarrow \varphi^A(b, c_0, \cdots, c_{k-1}) \in F\}.$$

The name Leibniz has its origin on the second order criterion of Leibniz which says that "two objects in the universe of discourse are equal if they share all the properties that can be expressed in the language of discourse". It is well known that $\Omega_\mathcal{A}(F)$ is the largest congruence on A compatible with F, i.e., the largest congruence Θ in A such that for any $a, b \in A$, if $a \Theta b$ then, $a \in F$ iff $b \in F$ (cf. [4])). We define the *reduction* of a matrix $\mathcal{A} = \langle A, F \rangle$ as the matrix $\mathcal{A}^* = \langle A/\Omega_\mathcal{A}(F), F/\Omega_\mathcal{A}(F) \rangle$, where $A/\Omega_\mathcal{A}(F)$ is the quotient algebra and $F/\Omega_\mathcal{A}(F) = \{a/\Omega_\mathcal{A}(F) : a \in F\}$.

1.2 DFA as matrices

In this section we introduce the central motivation for the present work: reasoning about deterministic finite automata as matrices. As far as we know, the first work with this approach is due to Blok and Pigozzi ([4]). Let us recall the notion of deterministic finite automaton. Although, almost all of our results still hold for infinite alphabets and infinite sets of states, in order to simplify arguments, we consider automata with a finite set of states and finite input alphabet.

Definition 3. *A* deterministic finite automaton (DFA *by short*) *is a* 5*-tuple* $\mathcal{H} = (\mathrm{Q}, \mathrm{In}, \delta, q_0, \mathrm{Q}_f)$*, where* Q *is a finite set of* states*;* In *is a finite set called* (input) alphabet*;* $\delta : \mathrm{Q} \times \mathrm{In} \to \mathrm{Q}$ *is a (total) mapping called* transition function*;* $q_0 \in \mathrm{Q}$ *called* start state *and* $Q_f \subseteq \mathrm{Q}$ *is the* set of final states*.*

As usual we extend δ to $\hat{\delta} : \mathrm{Q} \times \mathrm{In}^* \to \mathrm{Q}$ by $\hat{\delta}(q, \epsilon) = q$ for any $q \in \mathrm{Q}$; $\hat{\delta}(q, aw) = \hat{\delta}(\delta(q, a), w)$ for any $q \in \mathrm{Q}$, $a \in \mathrm{In}$ and $w \in \mathrm{In}^*$.

Let $\mathcal{F}_{\mathrm{DFA}} = \{\delta_i | i \in In\} \cup \{s_0\}$, where δ_i, $i \in \mathrm{In}$, are unary operation symbols and s_0 is a constant. Given a DFA automaton $\mathcal{H} = (\mathrm{Q}, \mathrm{In}, \delta, q_0, \mathrm{Q}_f)$, we define the $\mathcal{F}_{\mathrm{DFA}}$-matrix $\mathcal{A}_{\mathcal{H}} = \langle A, F\rangle$ over the signature $\mathcal{F}_{\mathrm{DFA}}$ as follows: the underlying algebra is $A = (\mathrm{Q}, \langle f^A\rangle_{f \in \mathcal{F}_{\mathrm{DFA}}})$ where $s_0^A = q_0$ and for each $i \in \mathrm{In}, q \in \mathrm{Q}$, $\delta_i^A(q) = \delta(i, q)$ and $F = \mathrm{Q}_f$. The matrix $\mathcal{A}_{\mathcal{H}}$ will be called the *associated matrix of* $\mathcal{H}$. On the other hand, given a finite $\mathcal{F}_{\mathrm{DFA}}$-matrix $\mathcal{A}$ there is a DFA automaton $\mathcal{H}$ such that $\mathcal{A} = \mathcal{A}_{\mathcal{H}}$.

Observe that we can "translate", in a unique way, each word $a_0 \cdots a_{n-1} \in \mathrm{In}^*$ into a term $\delta_{a_n-1}(\cdots(\delta_{a_0}(x))) \in T_{\mathcal{F}_{\mathrm{DFA}}}(\{x\})$ and vice versa. This correspondence is done by the bijection $Trans : \mathrm{In}^* \to T_{\mathcal{F}_{\mathrm{DFA}}}(\{x\})$ recursively defined by

- $Trans(\epsilon) = x$;
- $Trans(wa) = \delta_a(Trans(w))$ for any $a \in \mathrm{In}$, $w \in \mathrm{In}^*$.

In the sequel, we denote $Trans(w)$ by $t^w(x)$. A word (sequence of alphabet symbols) w is recognised by an DFA if the execution of the transition function over w, started at the initial state, halts in a final state. The *language accepted* by the DFA $\mathcal{H}$, is denoted by $\mathcal{L}(\mathcal{H})$ and consists of the set of its accepted words. Formally, $\mathcal{L}(\mathcal{H}) = \{w \in \mathrm{In}^* | \hat{\delta}(q_0, w) \in \mathrm{Q}_f\}$.

A set $L \subseteq \mathrm{In}^*$ is a *regular language* if there is a DFA $\mathcal{H}$ such that $\mathcal{L}(\mathcal{H}) = L$. Next theorem characterises the language accepted by a DFA in terms of the associated matrix. The proof follows by induction.

Theorem 1. *Let* $\mathcal{H}$ *be a* DFA*. Then for all* $w \in \mathrm{In}^*$*,* $w \in \mathcal{L}(\mathcal{H}) \Leftrightarrow t^w(s_0) \in L(\mathcal{A}_{\mathcal{H}})$*.*

Let us recall now the traditional *equivalence relation between states of a* DFA: two states $q, q' \in Q$ are said to be *equivalent*, in symbols $q \sim q'$, if for any $w \in \mathrm{In}^*$, $\hat{\delta}(q, w) \in \mathrm{Q}_f$ if and only if $\hat{\delta}(q', w) \in \mathrm{Q}_f$ (cf. [8]). In the following theorem we state the relation between this relation and the Leibniz congruence over the associated matrix.

Theorem 2 ([4]). *Let* $\mathcal{H}$ *be a* DFA *and* q, q' *states of* $\mathcal{H}$*. Then* $q \sim q'$ *iff* $q \equiv q'(\Omega_{\mathcal{A}_{\mathcal{H}}}(F))$*.*

For each reachable $\mathcal{F}_{\mathrm{DFA}}$-matrix $\mathcal{A} = \langle A, F\rangle$, there is a unique surjective homomorphism $h : \mathrm{T}_\Sigma \to A$. It is not difficult to see that $L(\mathcal{A}) = h^{-1}(F)$. These facts, lead to the so-called Myhill-Nerode theorem (cf. [4] for the details).

Theorem 3 ([4]). *Let* $L \subseteq \mathrm{In}^*$*. Then,* L *is a regular language iff* $\langle T_{\mathcal{F}_{\mathrm{DFA}_{\mathrm{In}}}}, L\rangle^*$ *is finite.*

2 *k*-dimensional algebraic machines

As we have already claimed, the principal aim of this work is to extend the perspective explained in the above section to other kind of automata. However, the matrix structure has some limitations to deal, for example, with automata with output. To achieve this, we generalise, in this section, the concept of matrix to the multi-sorted and k-dimensional case. We start by introducing some notions from universal sorted algebra. Given a signature Σ and a non zero natural k, we define a *k-formula* as a sequence of k formulas of same sort $\bar{\varphi}:s = \langle \varphi_0:s, \dots, \varphi_{k-1}:s\rangle$ and we denote the S-sorted set of k-formulas of Σ by $\mathrm{Fm}^k_\Sigma(X)$. We write $\bar{\varphi}(x_0{:}s_0, \dots, x_{n-1}{:}s_{n-1})$ to mean that the variables which occur in $\varphi_0, \dots, \varphi_{k-1}$ are among $x_0{:}s_0, \dots, x_{n-1}{:}s_{n-1}$. A *$k$-value* (of sort s) is a sequence $\bar{a}{:}s = \langle a_0{:}s, \dots, a_{k-1}{:}s\rangle$ with $a_0, \dots, a_{n-1} \in A_s$. We also extend functions $h : \mathrm{Fm}_\Sigma \to A$ to k-formulas componentwise.

Definition 4. *A* k-dimensional algebraic machine *(k-machine for short) over a (multi-sorted) signature Σ is a pair $\mathcal{A} = \langle A, F\rangle$, where A is a Σ-algebra and $F \subseteq A^k$.*

Let $\mathcal{A} = \langle A, F\rangle$ be a k-machine and $\Gamma, \{\bar{\varphi}\} \subseteq \mathrm{Fm}^k_\Sigma(X)$. The k-formula $\bar{\varphi}$ is a *(semantical) consequence* of the set of k-formulas Γ in $\mathcal{A}$, in symbols $\Gamma \models_{\mathcal{A}} \bar{\varphi}$, if for any assignment $h : X \to A$, $h(\Gamma) \subseteq F$ implies $h(\bar{\varphi}) \in F$.

The models of *classical propositional calculus* over a propositional language are examples of 1-machines, considering as underlying algebra of the machine a Boolean algebra over this signature and, as filter, the top of this algebra. The models of the (free) equational logic over a multi-sorted signature are examples of 2-machines, considering as underlying algebra of the machine an algebra over this signature and as filter (a set of 2-values) the identity of this algebra. Note that, in this case, this relation coincides with the standard satisfaction relation.

Now, we present a version of the Leibniz congruence to the k-dimensional case:

Definition 5 (Leibniz Congruence). *Let $\mathcal{A} = \langle A, F\rangle$ be a k-machine over a signature Σ. The* Leibniz congruence *of F over A, in symbols $\Omega_{\mathcal{A}}(F)$, is the binary relation defined as follows: for any $a, b \in A_s$, $a \equiv b(\Omega_{\mathcal{A}}(F)_s)$ iff for every k-formula $\bar{\varphi}(z{:}s, x_0{:}s_0, \dots, x_{n-1}{:}s_{n-1})$ and for all $c_0 \in A_{s_0}, \dots, c_{n-1} \in A_{s_{n-1}}$,*

$$\bar{\varphi}^A(a, c_0, \dots, c_{n-1}) \in F \quad \text{iff} \quad \bar{\varphi}^A(b, c_0, \dots, c_{n-1}) \in F.$$

It is straightforward to show that $\Omega_{\mathcal{A}}(F)$ is indeed a congruence on A. Moreover, it is the largest congruence on A compatible with F in the sense that, for any $a, b \in A$, $a \equiv b(\Omega_{\mathcal{A}}(F))$ implies that, $a \in F \Leftrightarrow b \in F$. The detailed study of the properties of the Leibniz congruence in the multisorted setting can be found in [10]. A k-machine $\mathcal{A} = \langle A, F\rangle$ is said to be *reduced* when $\Omega_{\mathcal{A}}(F) = id_A$. We define the *reduction* of $\mathcal{A}$ as the k-machine $\mathcal{A}^* = \langle A/\Omega_{\mathcal{A}}(F), F/\Omega_{\mathcal{A}}(F)\rangle$. It can be proved that the reduction of a matrix is always reduced (cf. [10]).

Let $\Sigma = (S, \mathcal{F})$ be a signature, $V \subseteq S$ and A a Σ-algebra. We define for each $a, b \in A_s$ the V-behavioural equivalence, in symbols $\equiv^A_V$, in the following way: $a \equiv^A_V b$ if for any $v \in V$, for every formula $\varphi(z:s, u_0:s_0, \dots, u_{n-1}:s_{n-1})$ of sort v, and for all $c_0 \in A_{s_0}, \dots, c_{n-1} \in A_{s_{n-1}}$, $\varphi^A(a, c_0, \dots, c_{n-1}) = \varphi^A(b, c_0, \dots, c_{n-1})$. It is not difficult to see that $\equiv^A_V$ is a congruence on A.

If we consider V as the set of *observable sorts* of the signature, this relation is the well known *observational equivalence* in the algebraic specifications of abstract data types setting. The next result is the theoretical support of proof methods to check behavioural equivalence (cf. [7]).

Theorem 4. *Let $\Sigma = (S, \mathcal{F})$ be a signature, $V \subseteq S$ and A a Σ-algebra. Then $\equiv^A_V$ is the largest congruence on A that coincides with the identity on V.*

Proof. Let θ be a congruence on A which coincides with the equality in all the sorts in $V \subseteq S$. Since θ is a congruence, we have that for every $a, b \in A_s$ and for every formula $\varphi(z:s, u_0:s_0, \dots, u_{n-1}:s_{n-1}) : s'$ with $s' \in V$, $a\,\theta_s\,b$ implies $\varphi(a:s, u_0:s_0, \dots, u_{n-1}:s_{n-1})\,\theta_{s'}\,\varphi(b{:}s, u_0{:}s_0, \dots, u_{n-1}{:}s_{n-1})$. Moreover, we have by hypothesis that, if $s' \in V$, we have $\varphi(a{:}s, u_0{:}s_0, \dots, u_{n-1}{:}s_{n-1}) = \varphi(b{:}s, u_0{:}s_0, \dots, u_{n-1}{:}s_{n-1})$. Therefore, $\theta \subseteq \equiv^A_V$.

The following lemma expresses that $\equiv_V^A \subseteq \Omega_{\mathcal{A}}(F)$.

Lemma 1. *Let $\Sigma = (S, \mathcal{F})$ be a signature, $V \subseteq S$ and $\mathcal{A} = \langle A, F \rangle$ a 2-machine over Σ with $F_V = id_V$ and $F_s = \emptyset$ otherwise. Then, $\equiv_V^A \subseteq \Omega_{\mathcal{A}}(F)$.*

Proof. Suppose that $\varphi^A(a, \bar{c}) = \varphi^A(b, \bar{c})$ for every formula $\varphi(z : s, \bar{u})$ and every $\bar{c} \in A_{\bar{Q}}$. Let $\langle \psi_0(z : s, \bar{u}\, \bar{Q}), \psi_1(z : s, \bar{u}\, \bar{Q}) \rangle$ be a 2-formula V, and $\bar{c} \in A_{\bar{Q}}$. Suppose that $\langle \psi_0^A(a, \bar{c}), \psi_1^A(a, \bar{c}) \rangle \in F$, i.e., $\psi_0^A(a, \bar{c}) = \psi_1^A(a, \bar{c})$. Then,

$$\psi_0^A(b, \bar{c}) = \psi_0^A(a, \bar{c}) = \psi_1^A(a, \bar{c}) = \psi_1^A(b, \bar{c}).$$

Thus for every pair of formulas and every sequence of parameters $\bar{c}$, $\psi_0^A(a, \bar{c}) = \psi_1^A(a, \bar{c})$ iff $\psi_0^A(b, \bar{c}) = \psi_1^A(b, \bar{c})$. Therefore, $\langle a, b \rangle \in \Omega_{\mathcal{A}}(F)$.

Moreover, since under the above conditions $\Omega_{\mathcal{A}}(F)$ coincides with the identity on V, by Theorem 4, $\equiv_V^A = \Omega_{\mathcal{A}}(F)$. This phenomena illustrates the strong relationship between the *Leibniz congruence* and the *observational equivalence* in the sense of [7, 3].

Theorem 5. *Let $\mathcal{A} = \langle A, F \rangle$ be a k-machine. Then, $\models_{\mathcal{A}} \; = \; \models_{\mathcal{A}^*}$.*

Proof. Suppose $\Gamma \models_{\mathcal{A}} \bar{\varphi}$. Let $h : \mathrm{Fm}_\Sigma^k(X) \to \mathcal{A}^*$ be a homomorphism such that $h(\Gamma) \subseteq F/\Omega_{\mathcal{A}}(F)$. Let $g : \mathrm{Fm}_\Sigma^k(X) \to \mathcal{A}$ such that, for any $\bar{\varphi} \in \mathrm{Fm}_\Sigma^k(X)$, $g(\bar{\varphi})/\Omega_{\mathcal{A}}(F) = h(\bar{\varphi})$. Since $\Omega_{\mathcal{A}}(F)$ is compatible with F we have that $g(\bar{\varphi}) \in F$ and, hence, that $h(\bar{\varphi}) \in F/\Omega_{\mathcal{A}}(F)$. Therefore $\Gamma \models_{\mathcal{A}^*} \bar{\varphi}$.

Suppose now $\Gamma \models_{\mathcal{A}^*} \bar{\varphi}$. Let $h : \mathrm{Fm}_\Sigma^k(X) \to \mathcal{A}$ be a homomorphism such that $h(\Gamma) \subseteq F$. Consider a $g : \mathrm{Fm}_\Sigma^k(X) \to \mathcal{A}$ defined by $g(\bar{\varphi}) = h(\bar{\varphi})/\Omega_{\mathcal{A}}(F)$. We have that $g(\Gamma) \subseteq F/\Omega_{\mathcal{A}}(F)$ and, since $\Gamma \models_{\mathcal{A}^*} \bar{\varphi}$, that $g(\bar{\varphi}) \in F$. Therefore $\Gamma \models_{\mathcal{A}} \bar{\varphi}$.

3 Automata as k-machines

We stated in Section 1.2 that a DFA can be seen as a matrix and, therefore, as an 1-machine. In fact, we may also see a DFA as a 1-machine over a 2-sorted signature by considering as universes the input alphabet and the set of states, and, as transition function, the transition function of the underlying automaton together with the set of final states (cf. [5]). In fact, using this approach we deal with all DFA as a class of algebras over the same signature, whereas in our formulation we consider a signature for each possible input alphabet. In the previous section we saw that the traditional equivalence relation between states of a DFA $\mathcal{H}$ is the Leibniz congruence in the machine $\mathcal{A}$. An DFA is *minimal* if for any $q, q' \in Q$, if $q \sim q'$ then $q = q'$; i.e., if its associated 1-machine is reduced. Moreover, it is well known that any DFA has a minimal equivalent automaton, in the sense that, for each $\mathcal{H}$ there is a minimal DFA $\mathcal{H}'$ such that $\mathcal{L}(\mathcal{H}) = \mathcal{L}(\mathcal{H}')$. Now, since $\models_{\mathcal{A}} \; = \; \models_{\mathcal{A}^*}$ (cf. Theorem 5) we have $L(\mathcal{A}_{\mathcal{H}}) = L(\mathcal{A}_{\mathcal{H}}^*)$, and we can build an equivalent minimal automaton from any other, by reducing its associate matrix. It is sometimes important to consider exclusively the reachable part of a DFA. Given it associated matrix $\mathcal{A} = \langle A, F \rangle$, we only have to consider the *reachability machine* defined as $\mathcal{A}_{Reach} = \langle A_{Reach}, F_{Reach} \rangle$ with $A_{Reach} \leq A$ the subalgebra of A generated by the constant s_0 and $F_{Reach} = F \cap A_{Reach}$. Following this perspective, we can find in the literature, some natural results from the automata theory. For example, the *global behavioural equivalence*, in sense of [16], can be seen as a generalisation of the equivalence relation between automata (cf. [13]). In the following section we formalise several kind of automata as k-machines. All of these considerations can be straightforward applied in all of these formalisations.

3.1 Automata with output

There are in the literature two kind of automata with output: the *Moore* and the *Mealy machines.* A *Moore machine* is a 6-tuple $\mathcal{H}_{Moore} = (\mathrm{Q}, \mathrm{In}, \mathrm{Out}, \delta, \lambda, q_0)$, where $\mathrm{Q}, \mathrm{In}, \delta$ and q_0 are defined as in the DFA case (see Definition 3), Out is a finite set called *output alphabet* and $\lambda : \mathrm{Q} \to \mathrm{Out}$ is a function called *output function.* We define $\hat{\lambda} : \mathrm{Q} \times In^* \to \mathrm{Out}^*$ by:

$-\hat{\lambda}(q, \epsilon) = \lambda(q)$, for any $q \in \mathrm{Q}$,

$-\hat{\lambda}(q, aw) = \lambda(q)\hat{\lambda}(\delta(q, a), w)$, for any $w \in \mathrm{In}^*, a \in \mathrm{In}, q \in \mathrm{Q}$.

Similarly as in the DFA case, we consider a Moore machine $\mathcal{H}_{Moore} = (\mathrm{Q}, \mathrm{In}, \mathrm{Out}, \delta, \lambda, q_0)$ as a 2-machine $\mathcal{A}_{\mathcal{H}_{Moore}} = \langle A, F \rangle$ over a signature with set of sorts $S_{Moore} = \{\mathrm{Q}, \mathrm{Out}\}$ and

- $\mathcal{F}_{Moore} = \{\delta_i | i \in \mathrm{In}\} \cup \{\lambda\} \cup \{s_0\}$ with δ_i and λ unary operations symbols and s_0 a constant;
- A is a 2-sorted algebra $A = (\mathrm{Q}, \mathrm{Out}; \langle f^A \rangle_{f \in \mathcal{F}_{Moore}})$;
- $\delta_i^A(q) = \delta(q, i)$, $\lambda^A(q) = \lambda(q)$ and $s_0^A = q_0$;
- $F_{\mathrm{Out}} = id_{A_{\mathrm{Out}}}$ and $F_{\mathrm{Q}} = \emptyset$.

Observe that we can not define $\mathcal{A}_{\mathcal{H}_{Moore}}$ as an ordinary matrix, since we need the multi-sorting to deal with the set of states and the output alphabet. We also need to define the filters as a set of pairs. Consider now the following technical results:

Lemma 2. *Let* $\mathcal{H} = (\mathrm{Q}, \mathrm{In}, \mathrm{Out}, \delta, \lambda, q_0)$ *be a Moore Machine* $q, q' \in \mathrm{Q}$ *and* $w = a_0 \cdots a_{k-1} \in In^*$. *Then,* $\hat{\lambda}(q, w) = \hat{\lambda}(q', w)$ *iff for any* $i < k$, $\lambda(\hat{\delta}(q, a_0 \cdots a_i)) = \lambda(\hat{\delta}(q', a_0 \cdots a_i))$.

Lemma 3. *Let* $\mathcal{H} = (\mathrm{Q}, \mathrm{In}, \mathrm{Out}, \delta, \lambda, q_0)$ *be a Moore Machine* $q, q' \in \mathrm{Q}$ *and* $w \in In^*$. *The following conditions are equivalent:*

(i) For any $w \in In^*$, $\hat{\lambda}(q, w) = \hat{\lambda}(q', w)$;

(ii) For any $w \in In^*$, $\lambda(\hat{\delta}(q, w)) = \lambda(\hat{\delta}(q', w))$.

Proof. Suppose that (i) holds. Case $w = \epsilon$, we have $\hat{\lambda}(q, \epsilon) = \hat{\lambda}(q', \epsilon)$ iff $\lambda(q) = \lambda(q')$ iff $\lambda(\hat{\delta}(q, \epsilon)) = \lambda(\hat{\delta}(q', \epsilon))$. Case $w = a_0 \cdots a_{k-1}$, we have by Lemma 2 that for any $i < k$, $\lambda(\hat{\delta}(q, a_0 \cdots a_i)) = \lambda(\hat{\delta}(q', a_0 \cdots a_i))$ and, in particular, for $i = k - 1$, we have that for any $w \in In^*$, $\lambda(\hat{\delta}(q, a_0 \cdots a_{k-1})) = \lambda(\hat{\delta}(q', a_0 \cdots a_{k-1}))$.

Suppose now that (ii) holds. Case $w = \epsilon$, we have that $\lambda(\hat{\delta}(q, \epsilon)) = \lambda(\hat{\delta}(q', \epsilon))$ iff $\lambda(q) = \lambda(q')$ iff $\hat{\lambda}(q, \epsilon) = \hat{\lambda}(q', \epsilon)$. Case $w = a_0 \cdots a_{k-1}$, $k \geq 1$, we have by hypothesis that for any $i < k$, $\lambda(\hat{\delta}(q, a_0 \cdots a_i)) = \lambda(\hat{\delta}(q', a_0 \cdots a_i))$. Therefore, by Lemma 2, $\hat{\lambda}(q, w) = \hat{\lambda}(q', w)$.

Recalling the traditional *equivalence between states of a Moore machine*, we have that two states q, q' are equivalent, in symbols $q \sim_{\mathrm{Out}} q'$, if the execution of the output function over any word, started in them, returns the same result (cf. [8]).

Theorem 6. $q \sim q'$ *iff* $q \equiv q'(\Omega_{\mathcal{A}}(F))$.

Since $\sim$ is the largest congruence on A that coincides with the identity in the sort *Out* (cf. Theorem 4) we can conductively check the equivalence between states of an automaton. For example, to prove that $q \sim q'$ we just need to define a congruence $R \subseteq A \times A$ with $R^A_{\mathrm{Out}} = id_{A_{\mathrm{Out}}}$ such that $(q, q') \in R$.

Whereas in the Moore machines, the output function tags the states of the automata, in the Mealy machines, this function tags the transitions. However, there is a procedure by which, from a Moore machine it is possible to construct an "equivalent" Mealy machine, in the sense that, for any word w, $\hat{\lambda}_{Moore}(q0_{Moore}, w) = \lambda_{Moore}(q0_{Moore})\hat{\lambda}_{Mealy}(q0_{Mealy}, w)$ (cf. [8]). In this way, we can deal with a Mealy machine as a 2-machine, building the associated machine of its equivalent Moore machine. Hence, given a *Mealy machine* $\mathcal{H}_{Mealy} = (\mathrm{Q}, \mathrm{In}, \mathrm{Out}, \delta, \lambda, q_0)$ we can build an associated 2-machine $\mathcal{A}_{\mathcal{H}_{Mealy}} = \langle A, F\rangle$ considering as underlying algebra the 2-sorted algebra $A = (\mathrm{Q} \times (\mathrm{Out} \cup \{\star\}), \mathrm{Out} \cup \{\star\}; \langle f^A\rangle_{f\in\mathcal{F}_{Moore}})$ with $\delta_i^A((q, o)) = (\delta(q, i), \lambda(q, i))$, $\lambda^A((q, o)) = o$ and $s_0^A = (q_0, \star)$ and considering as designated filter the set $F \;\; = id_{A_{\mathrm{Out}\cup\{\star\}}}$. Note that we have to consider the special alphabet symbol $\star$ in order to define the constant s_0. However, in the behaviour of the machine, this element just (and allways) appears as the first element of the output string.

3.2 Nondeterministic finite automata

A *nondeterministic finite automaton* (NFA) is a 5-tuple $\mathcal{H} = (\mathrm{Q}, \mathrm{In}, \delta, q_0, Q_f)$ defined as in Definition 3, except for the transition function δ. Here, the function $\delta : \mathrm{Q} \times \mathrm{In} \to \mathcal{P}(\mathrm{Q})$ takes a state and an input symbol and returns a set of possibles next states, whereas in the deterministic case, it returns exactly one next state. Hence, we extend the transition function to $\hat{\delta} : \mathrm{Q} \times \mathrm{In}^* \to \mathcal{P}(\mathrm{Q})$ as usual:

- $\hat{\delta}(q, \epsilon) = \{q\}$, for any $q \in \mathrm{Q}$;
- $\hat{\delta}(q, aw) = \bigcup_{q'\in\delta(q,a)} \hat{\delta}(q', w)$, for any $q \in \mathrm{Q}$, $w \in \mathrm{In}^*$, $a \in \mathrm{In}$.

A word w is accepted by a NFA if the execution of its transition function over w, starting in the initial state q_0, halts at least in one final state, i.e., if $\hat{\delta}(q_0, w) \cap \mathrm{Q}_f \neq \emptyset$ (cf. [8]). One way to deal with these automata in our approach is by using the traditional procedure to construct an equivalent DFA from a NFA. Hence, we can associate for any NFA $\mathcal{H}_{NFA} = (\mathrm{Q}, \mathrm{In}, \delta, q_0, \mathrm{Q}_f)$ a 1-machine $\mathcal{A}_{\mathcal{H}_{NFA}} = \langle A, F\rangle$ over the signature $\mathcal{F}_{\mathrm{DFA}}$, taking as underlying algebra the algebra $A = (\mathcal{P}(\mathrm{Q}); s_0, \langle f^A\rangle_{f\in\mathcal{F}})$ with $\mathcal{P}(\mathrm{Q}) = \{U | U \subseteq \mathrm{Q}\}$, for each $i \in \mathrm{In}$, $\delta_i^A(X) = \bigcup_{x\in X} \delta(x, i)$ and $s_0^A = \{q_0\}$ and considering the set $F = \{M \in \mathcal{P}(\mathrm{Q}) : M \cap \mathrm{Q}_f \neq \emptyset\}$ as the filter.

Theorem 7. $L(\mathcal{H}_{NFA}) = L(\mathcal{A}_{\mathcal{H}_{NFA}})$.

Another way to deal with NFA is using the non-deterministic matrices (or N-matrices for short) (cf. [2]):

Definition 6 (N-matrix). *A* non-deterministic matrix *over a signature Σ is a pair $\mathcal{A} = \langle (A, \langle f^A\rangle_{f\in\Sigma}), F\rangle$, where A is a set, $F \subseteq A$ and for any n-ary operation symbol $f \in \Sigma$, $f^A : A \times \cdots \times A \to \mathcal{P}(A) \setminus \{\emptyset\}$.*

A *valuation* is a function $v : \mathrm{Fm}_\Sigma \to A$ such that, for any n-ary $f \in \Sigma$ and for any $\varphi_1, \ldots, \varphi_n \in \mathrm{Fm}_\Sigma$, $v(f(\varphi_1, \ldots, \varphi_n)) \in f^A(v(\varphi_1), \ldots, v(\varphi_n))$. A valuation v is a *model* of φ in $\mathcal{A} = \langle A, F\rangle$ if $v(\varphi) \in F$. Hence, using this formalism, we can associate for any NFA $\mathcal{H}_{NFA} = (\mathrm{Q}, \mathrm{In}, \delta, q_0, \mathrm{Q}_f)$ the N-matrix $\mathcal{A}_{\mathcal{H}_{NFA}} = \langle A, F\rangle$ where $A = (\mathrm{Q}; s_0, \langle\delta_i\rangle_{i\in\mathrm{In}})$, for each $i \in \mathrm{In}$, the multi-functions satisfy $\delta_i^A(q) = \delta(q, i)$ and $s_0^A = q_0$, considering as filter the set $F = \mathrm{Q}_f$.

Note that in this case, we can not characterize accepted words on a NFA as theorems in its underlying machine: actually $\models_{\mathcal{A}_{\mathcal{H}_{\mathcal{NFA}}}} t^w(s_0)$ means that the execution halts in a final state in all of its possible paths when we just need that it halts in a final state in one of these paths, ie. that $t^w(s_0)$ has a model in $\mathcal{A}_{\mathcal{H}_{NDFA}}$. This is not surprising, since the gap between N-matrices and matrices is much larger than the gap between

DFA and NFA. For example, it proofs that for any finite N-matrix $\mathcal{A}_{\mathcal{N}\mathcal{D}}$ we can define an equivalent matrix $\mathcal{A}_{\mathcal{D}}$ (in sense that $\models_{\mathcal{A}_{\mathcal{N}\mathcal{D}}} = \models_{\mathcal{A}_{\mathcal{D}}}$). However, $\mathcal{A}_{\mathcal{D}}$ it is not necessarily finite (cf. [2]). Hence, we define the language accepted by the N-matrix $L(\mathcal{A}_{\mathcal{H}_{NDFA}})$ as the set $L(\mathcal{A}_{\mathcal{H}_{NDFA}}) = \{w \in In^* | t^w(s_0) \text{ has a model in } \mathcal{A}_{\mathcal{H}_{NDFA}}\}$. Consequently, we have that:

Theorem 8. $L(\mathcal{H}_{NDFA}) = L(\mathcal{A}_{\mathcal{H}_{NDFA}})$.

4 Conclusions and further developments

We have introduced an algebraic logic theory to treat input/output machines. The generalized algebraic logic approach to characterise known concepts of automata theory provides an elegant and unified theory. This is just preliminary work, in the sense that it lacks a deep body of results. Although the presented adaption to the work presented in [4] is straightforward, our approach is promising. Our aim with this paper is just to set the ground for developing such results, namely we discuss the way towards the development of a consistent algebraic logic theory to shed a new light on automata theory.

There are two major directions in our future work. First, we would like to co-relate results of these two fields. For example, the Leibniz congruence is a central concept of AAL that has been intensively investigated and characterised in several works (eg. [10]). Hence, one interesting issue is to look at this results from the perspective of equivalence between states of automata. On other hand, there are some efficient algorithms to minimise and proof the equivalence between automata. Hence, it can be interesting to study these methods in the context of the reduction of k-machines. Another interesting topic is the research on the applicability of *power matrix* properties to the algebraic logic context. Consider, for example, a matrix $\mathcal{A} = \langle A, F \rangle$ where A is a monadic algebra and F a singleton set $\{\top\}$. It is not difficult to see that we can reduce the verification of a theorem with one variable $\models_{\mathcal{A}} t(x)$ to the verification of a ground theorem $\models_{\mathcal{P}(\mathcal{A})} t(s_0)$ with $s_0^{\mathcal{P}(\mathcal{A})} = \{\mathrm{Q}\}$ where Q is the universe of the underlying algebra of $\mathcal{A}$ and $\mathcal{P}(\mathcal{A})$ is the power matrix of $\mathcal{A}$ eventually extended with the constant s_0. This idea is inspired by a method to check a reset word in a synchronising automata [1]. A synchronising automaton consists of a DFA $\mathcal{H}$ with a reset word, i.e., with a word w_{reset} such that, for any state $q \in \mathrm{Q}_{\mathcal{H}}$, $\widehat{\delta}(q, w_{reset}) = q_{reset}$, for some particular state $q_{reset} \in \mathrm{Q}_{\mathcal{H}}$, called *reset state* (cf. [1]). Actually there are efficient methods to search reset words and check the synchronability of automata (eg. [17]). It may be also interesting to study this methods in the view of theorem proving in matrix semantics.

The second direction is the study of other kind of automata in this algebraic perspective. Concerning Pushdown automata we have already got some improvements, namely we can deal, in a AAL perspective, with a class of *pushdown automata*: the *strongly deterministic pushdown automata* (SDPA). A *Pushdown automaton* (PA) is a 7-tuple $\mathcal{H}_{PA} = (\mathrm{Q}, \mathrm{In}, \Gamma, \delta, q_0, Z_0, \mathrm{Q}_f)$, where Q, In and q_0 are defined as in the DFA case (see Definition 3), Γ is a finite set called *stack alphabet*, Z_0 is a constant of Γ called by *start stack symbol* and δ is a mapping for $\mathrm{Q} \times (\Sigma \cup \{\epsilon\}) \times \Gamma^*$ to finite subsets of $\mathrm{Q} \times \Gamma^*$. A PA is a SDPA if for every $q \in \mathrm{Q}$, $a \in \Sigma \cup \{\epsilon\}$ and $b \in \Gamma^*$, $\delta(q, a, b)$ contains exactly one element. A PA recognise two languages: the *language recognised by final state* and the *language recognised by empty stack* (cf. [8]). Hence, given a SDPA $\mathcal{H}_{SDPA} = (\mathrm{Q}, \mathrm{In}, \Gamma, \delta, q_0, Z_0, F)$ we define the 1-machines $\mathcal{A}^{fs}_{\mathcal{H}_{SDPA}} = \langle A, F_{fs} \rangle$ and $\mathcal{A}^{es}_{\mathcal{H}_{SDPA}} = \langle A, F_{es} \rangle$ over the signature $\mathcal{F}_{DPA}$ where $A = (\mathrm{Q} \times \Gamma^*; \langle f^A \rangle_{f \in \mathcal{F}_{DPA}})$ with $\delta_i^A((q, p)) = \delta(q, i, p)$, $s_0^A = (q_0, z_0)$, $F_{fs} = \{(q, p) | q \in \mathrm{Q}_f, p \in \Gamma^*\}$ and $F_{es} = \{(q, \epsilon) | q \in \mathrm{Q}\}$. Now, we define the language recognised by final state as the language recognised by $\mathcal{A}^{fs}_{\mathcal{H}_{SDPA}}$ and the language recognised by the empty stack as the language recognised by $\mathcal{A}^{es}_{\mathcal{H}_{SDPA}}$. The work presented here is not enough to study the entire class since, in opposition to the finite automata case, the class of languages recognised by deterministic pushdown automata is a proper subclass of the languages recognised by the nondeterministic pushdown automata.

We hope to report on these and related questions in forthcoming papers.

Acknowledgment: Research supported by the Portuguese *Fundação para a Ciência e a Tecnologia (FCT)* through *Centro de Investigação e Desenvolvimento em Matemática e Aplicações* of University of Aveiro, by the project MONDRIAN (under the contract **PTDC/EIA-CCO/108302/2008**) and by the grant **SFRH/BDE/33650/2009** supported by *FCT* and by the company *Critical Software S.A., Portugal*.

References

1. Ananichev, D. and Volkov, M. V.: Synchronizing generalized monotonic automata, Theor. Comput. Sci. vol. 330,1 pag.3–13 (2005)
2. Avron, A. and Lev, I.: Non-deterministic Multiple-valued Structures. J. Log. Comput. vol. 15, 3 , pp 241–261 (2005)
3. Bidoit, M. and Hennicker, R.: Behavioural theories and the proof of behavioural properties. TCSci., vol 165(1): pp 3-55, (1996)
4. Blok, W.J. and Pigozzi, D.:Algebraic semantics for universal Horn logic without equality. Universal algebra and quasigroup theory, Lect. Conf., Jadwisin/Pol. 1989, Res. Expo. Math. 19, 1-56. (1992)
5. Denecke, K. and Wismath, S.: Universal Algebra and Applications in Theoretical Computer Science, CRC/C&H (2002)
6. Font, J.M. and Jansana, R. and Pigozzi, D.: A Survey of Abstract Algebraic Logic, Studia Logica (2003)
7. Goguen, J. and Malcolm, G.: A hidden agenda. Theo. Comput. Sci. vol. 245, 1, pp 55-101(2001)
8. Hopcroft, J. and Motwani, R. and Ullman, J.: Introduction to Automata Theory, Languages, and Computation (2nd Edition), Addison Wesley (2000)
9. Howie, John M.: Automata and languages, Oxford Science Publications. Oxford: Oxford University Press (1991)
10. Martins, M. A., Pigozzi, D.:Behavioural reasoning for conditional equations. Math. Struct. in Comp. Science., Cambridge University Press, Vol. 17, 1075–1113 (2007)
11. Martins, M., Madeira and A. Barbosa, L.: Refinement via interpretation, Seventh IEEE International Conference on Software Engineering and Formal Methods, IEEE Computer Society, pag. 250–259 (2009)
12. Martins, M., Madeira, A. and Barbosa, L.:Refinement by Interpretation, in a General Setting, ENTCS, vol 259, pag. 105–121 (2009)
13. Martins, M. A.: On the Behavioral Equivalence Between k-data Structures, Comput. J., Oxford University Press, vol. 51, 2, pag. 181–191 (2008)
14. Mezei, J., Wright, J.B., Algebraic automata and context-free sets., Inf. Control, vol. 11, pag. 3-29, (1967)
15. Niehren, J., Podelski, A., Feature Automata and Recognizable Sets of Feature Trees, In TAPSOFT'93, pag. 356–375. Springer-Verlag LNCS 668, (1993)
16. Sannella, D. and Tarlecki, A.: Foundations of Algebraic Specifications and Formal Program Development, Cambridge University Press (To appear)
17. Trahtman, A. N.: An Efficient Algorithm Finds Noticeable Trends and Examples Concerning the Cerny Conjecture, MFCS, pag. 789–800 (2006)