Acyclic Automata with easy-to-find short regular expressions

José João Morais Nelma Moreira Rogério Reis

Technical Report Series: DCC-2005-03

Departamento de Ciência de Computadores – Faculdade de Ciências

&

Laboratório de Inteligência Artificial e Ciência de Computadores

Universidade do Porto

Rua do Campo Alegre, 823 4150 Porto, Portugal

Tel: +351+2+6078830 – Fax: +351+2+6003654

`http://www.ncc.up.pt/fcup/DCC/Pubs/treports.html`

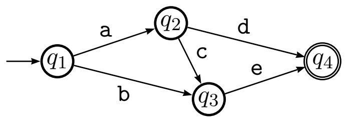
Abstract

Computing short regular expressions equivalent to a given finite automaton is a hard task. In this work we present a class of acyclic automata for which it is easy to find a regular expression that has linear size. We call those automata UDR. A UDR automaton is characterized by properties of its underlying digraph. We present a characterisation theorem and an efficient algorithm to determine if an acyclic automaton is UDR. This algorithm can be adapted to compute a short regular expression equivalent to a given UDR automaton.

Keywords: Formal languages, finite automata, regular expressions, graph theory, minimisation

1 Introduction

Computing a regular expression from a given finite automaton can be achieved by well-known algorithms based on Kleene's theorem [Kle56], establishing the equivalence between languages accepted by finite automata and languages represented by regular expressions. However the resulting regular expression depends on the order of the automaton's states considered in the conversion. In particular, this is the case if the algorithm is based on the *state elimination technique*. Consider, for example, the following automaton:



If we remove the state q_2 and then the state q_3, we obtain the regular expression $\mathtt{ad} + (\mathtt{ac} + \mathtt{b})\mathtt{e}$. But if we remove first q_3 and then q_2, we obtain the regular expression $\mathtt{be} + \mathtt{a}(\mathtt{ce} + \mathtt{d})$. In the first case, the symbol a occurs two times, and in the second, the symbol e occurs two times. The first case corresponds to the application of the distributivity rule on the right, and the second to the application of that rule on the left. Although in this example the resulting regular expressions have the same number of symbols, in general that will not be the case. If our goal is to obtain an equivalent regular expression with short size from a given automaton, the order in which we consider the automaton states is of great importance. Moreover, given an automaton with n states and k alphabetic symbols the upper bound for the size of the equivalent regular expression is $O(nk4^n)$ (and no general smaller lower bounds are known), and the problem of obtaining a minimal regular expression equivalent to a given automaton is PSPACE-complete [JR93].

In this work we present a characterisation of acyclic automata for which it is easy to find an order of state removal such that the resulting regular expressions have size linear in the number of the automata transitions.

The paper is organised as follows. In the next section, we review some basic notions and introduce notation used in this paper. In Section 3 we define a class of acyclic automata that we call UDR. Section 4 gives a characterisation theorem of UDR automata in terms of properties of their underlying digraphs. In Section 5 we show that it is possible to compute a linear size regular expression from a UDR automaton. An efficient algorithm for determining if an automaton is UDR is presented in Section 6. Relationships with other work and with specific classes of finite automata are discussed in Section 7. In the last section, we provide some final remarks.

2 Preliminaries

We recall the basics of digraphs, finite automata and regular expressions that can be found in standard books [HMU00, Har69, BJG01]. A digraph $D = (V, E)$ consists of a finite set V of vertices and a set E of ordered pairs of vertices, called *arcs*. If (u, v) in E, u is *adjacent to* v and v is *adjacent from* u. For each vertex v, the *indegree* of v is the number n_i of vertices adjacent to it and the *outdegree* of v is the number n_o of vertices adjacent from it, and we write $v(n_i; n_o)$. An arc (u, v) can be denoted by uv. A *path* between v_0 and v_n is a sequence $v_0v_1, v_1v_2, \ldots, v_{n-1}v_n$ of arcs, and is denoted by $v_0 - v_n$, or $v_0 - v_k - v_n$, for $1 \leq k < n$. A path is *simple* if all the vertices in it are distinct. The length of a path is number of arcs in the path. A path is a *cycle* if $v_0 = v_n$ and $n \geq 1$. A digraph that has no cycles is called *acyclic*. For an acyclic digraph $D = (V, E)$, there is a *topological* ordering o of its vertices, *i.e.*, such that if $(u, v) \in E$ then $o(u) < o(v)$.

An alphabet Σ is a nonempty set of symbols. A string over an alphabet Σ is a finite sequence of symbols of Σ. The empty string is denoted by ϵ. The set $\Sigma^\star$ is the set of all strings over Σ. A language L is subset of $\Sigma^\star$. If L_1 and L_2 are two languages, then $L_1 \cdot L_2 = \{xy \mid x \in L_1 \text{ and } y \in L_2\}$. The operator $\cdot$ is often omitted. A regular expression (r.e.) α over Σ represents a (regular) language $L(\alpha) \subseteq \Sigma^\star$ and is inductively defined by: $\emptyset$ is a r.e and $L(\emptyset) = \emptyset$; ϵ is a r.e and $L(\epsilon) = \{\epsilon\}$; $a \in \Sigma$ is a r.e and $L(a) = \{a\}$; if α_1 and α_2 are r.e., $\alpha_1 + \alpha_2$, $\alpha_1\alpha_2$ and $\alpha_1^\star$ are r.e., respectively with $L((\alpha_1 + \alpha_2)) = L(\alpha_1) \cup L(\alpha_2)$, $L((\alpha_1\alpha_2)) = (L(\alpha_1)L(\alpha_2))$ and $L(\alpha_1{}^\star) = L(\alpha_1)^\star$. Let R_Σ be the set of regular expressions over Σ. Two regular expressions α_1 and α_2 are equivalent if $L(\alpha_1) = L(\alpha_2)$, and we write $\alpha_1 \equiv \alpha_2$. The algebraic structure $(R_\Sigma, +, \cdot, \emptyset, \epsilon)$, forms a idempotent semi-ring. i.e., for all $\alpha, \beta, \gamma \in R_\Sigma$ we have:

$$\begin{array}{ll}
\alpha + (\beta + \gamma) \equiv (\alpha + \beta) + \gamma & \alpha + \beta \equiv \beta + \gamma \\
\alpha + \emptyset \equiv \alpha & \alpha + \alpha = \alpha \\
\alpha(\beta\gamma) \equiv (\alpha\beta)\gamma & \alpha\epsilon \equiv \epsilon\alpha \equiv \alpha \\
\alpha(\beta + \gamma) \equiv \alpha\beta + \alpha\gamma & (\alpha + \beta)\gamma \equiv \alpha\gamma + \beta\gamma \\
\alpha\emptyset \equiv \emptyset\alpha \equiv \emptyset &
\end{array}$$

In this work, we will take the *size* of a regular expression α to be the number of symbols from Σ contained in α, and we denote it by $|\alpha|$.

A *nondeterministic finite automaton* (NFA) A is a quintuple $(Q, \Sigma, \delta, q_0, F)$ where Q is finite set of states, Σ is the alphabet, $\delta \subseteq Q \times \Sigma \cup \{\epsilon\} \times Q$ the transition relation, q_0 the initial state and $F \subseteq Q$ the set of final states. A NFA is *deterministic* (DFA) if for each pair $(q, a) \in Q \times \Sigma$ there exists at most one q' such that $(q, a, q') \in \delta$. For $q \in Q$ and $a \in \Sigma$, we denote by $\delta(q, a) = \{p : (q, q, p) \in \delta\}$, and we can extend this notation to $x \in \Sigma^\star$ and $T \subseteq Q$, by $\delta(q, ax) = \delta(\delta(q, a), x)$. The *language* accepted by A is $L(A) = \{x \in \Sigma^\star \mid \delta(q_0, x) \cap F \neq \emptyset\}$. Two NFA are *equivalent* if they accept the same language. The *size* of a NFA is the number of its transitions.

The *underlying digraph* of a NFA $A = (Q, \Sigma, \delta, q_0, F)$ is the digraph $D = (Q, E)$ such that $E = \{(q, q') \mid q, q' \in Q \text{ and } \exists a \in \Sigma \cup \{\epsilon\} \text{ such that } (q, a, q') \in \delta\}$. Note that even there can be more than one symbol of Σ between two states q and q', only one arc exists in the underlying graph. We call *initial vertex* the vertex that corresponds to the initial state, *final vertices* the ones that correspond to final states and *intermediate vertices*, all the others. An automaton is *useful* if in its underlying digraph, every vertex is in a path from the initial vertex to a final vertex. An automaton is *acyclic* if its underlying digraph is acyclic. We will use the above terminology both for digraphs and for automata.

An *extended finite automaton* (EFA) A is a quintuple $(Q, \Sigma, \delta, q_0, F)$, where Q, Σ, q_0 and F are as before and $\delta : Q \times Q \to R_\Sigma$. We assume that $\delta(q, q') = \emptyset$, if the transition from q to q' is not defined. A string $x \in \Sigma$ is said to be accepted by A if $x = x_1 \cdots x_n$, for $x_1, \ldots, x_n \in \Sigma^\star$ and there is a state sequence $q_0, q_1, \ldots, q_n$ with $q_n \in F$, such that $x_1 \in L(\delta(q_0, q_1))$,..., $x_n \in L(\delta(q_{n-1}, qn))$. The language accepted by A is the set of all strings accepted by A. The *underlying digraph* of a EFA is a digraph $D = (Q, E)$ such that $(q, q') \in E$ if and only if $\delta(q, q') \neq \emptyset$. Any NFA can be easily transformed into an equivalent EFA, with the same underlying digraph: for each pair of states (q, q') one needs to construct a regular expression $a_1 + \cdots + a_n$ such that $(q, a_i, q') \in \delta$, $a_i \in \Sigma \cup \{\epsilon\}$, $1 \leq i \leq n$.

Finally, we recall the conversion of an EFA A into a regular expression α, using the *state elimination algorithm* (SEA). In each step, a non-initial and non-final state of the EFA is deleted and the transitions are changed in such way that the new EFA is equivalent to the older one. Formally, let $A = (Q, \Sigma, \delta, q_0, F)$ be a EFA. Then

1. (a) If $q_0 \in F$ or exists $q \in Q$ such that $\delta(q, q_0) \neq \emptyset$, then add a new state i to Q, define $\delta(i, q_0) = \epsilon$ and i is the new initial state.

   (b) If $|F| > 1$, then add a new state f and transition $\delta(q, f) = \epsilon$, for all $q \in F$. The set of final states becomes $\{f\}$.

   Without lost of generality, let $A' = (Q', \Sigma, \delta', i, \{f\})$ denote the new EFA. We denote by $\alpha_{qq'}$ the regular expression $\delta(q, q')$.

2. If $Q' = \{i, f\}$, then the resulting regular expression is $\alpha_{if}\alpha_{ff}^\star$, and the algorithm terminates. Otherwise continue to 3.

3. **Choose** $q \in Q' \setminus \{i, f\}$. Eliminate q from A', considering $Q' - \{q\}$ the new set of states, and for each $q_1, q_2 \in Q' - \{q\}$,

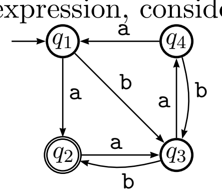
$$\delta'(q_1, q_2) = \alpha_{q_1 q_2} + \alpha_{q_1 q}\alpha_{qq}^\star\alpha_{qq_2},$$

   Continue to 2.

Let us observe that, in each step, if we have $q(k; l)$, the contribution of q for the size of the final regular expression can be measured by

$$(k-1)\sum_{i=1}^{k} |\alpha_{q_i q}| + (l-1)\sum_{j=1}^{l} |\alpha_{qq_j}| + (kl-1)|\alpha_{qq}|. \quad (1)$$

This contribution is 0 if $q(1; 1)$. To illustrate, this dependence from the order of state elimination in the resulting regular expression, consider the following automaton:



If the order of state removal is q_2, q_1, q_4 and q_3 we obtain the regular expression $a + (b + aa)(ba + a(b + a(b + aa))^*(b + aaa)$ with 16 alphabetic symbols. If order is q_3, q_4, q_2 and q_1, the resulting expression is $ba(ba)^*a + (a + bb + ba(ba)^*bb)(ab + aa(ba)^*bb)^*aa(ba)^*a)^*(a + bb + ba(ba)^*bb)(ab + aa(ba)^*bb)^*$ with 44 alphabetic symbols. In each step, we could try to simplify the regular expression obtained, but our goal is try to discover a state order that leads to shorter regular expressions. One of the advantage of this approach is to avoid the generation of bigger intermediate regular expressions.

3 UDR Automata

In this section, we will consider only useful acyclic automata with one final state. The underlying digraphs of these automata are called in the literature acyclic st-digraphs [BJG01]. In an acyclic st-digraph there exists only a vertex of indegree 0 (denoted by s), only a vertex of outdegree 0 (denoted by t), and each vertice occurs in some path from s to t. In an acyclic automaton, the vertices s and t correspond to the initial and final state, respectively.

We are going to characterise a class of automata, using the notion of digraph homeomorphism. Two digraphs are homeomorphic if both can be obtained from the same digraph by a sequence of subdivisions of arcs [Har69, BJG01].

Let consider the digraph[1]

$$\mathsf{R}^{\rightarrow} = (\{q_1, q_2, q_3, q_4\}, \{(q_1, q_2), (q_1, q_3), (q_2, q_3), (q_2, q_4), (q_3, q_4)\}),$$

represented in Figure 1.

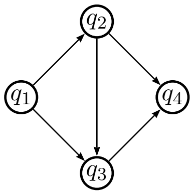


Figure 1: Digraph $\mathsf{R}^{\rightarrow}$

Definition 1. *A useful acyclic NFA with one final state is* UDR *if its underlying digraph does not contain a subgraph homeomorphic to* $\mathsf{R}^{\rightarrow}$. *We say that the underlying digraph is a* UDR digraph.

The digraph in Figure 2, is not UDR, because its underlying digraph contains a subgraph homeomorphic to $\mathsf{R}^{\rightarrow}$, namely, the one obtained by excluding the vertex q_3 and the arcs (q_4, q_6), and (q_5, q_{10}) (with dashed lines, in the figure). On the other hand, the digraph in Figure 3 is UDR. The automaton presented in the introduction is obviously not UDR.

The name UDR is an acronym of *Unique for the Distributivity Rule.* If an automaton is not UDR, there are at least two states (with outdegree or indegree greater than 1) such that the order chosen to eliminate them leads to two different regular expressions, one that results from the application of a distributivity rule to the other. In general, one of the choices will lead to a shorter expression, but is not easy to determine which. In the next section, we show that this is not the case if the automaton is UDR. In a UDR automaton, in each step we can choose to eliminate a state $q(1; 1)$.

Proposition 1. *Any acyclic NFA is equivalent to a UDR automaton.*

[1] $\mathsf{R}^{\rightarrow}$ is a directed version of Frank Harary's omnipresent *random graph* ☺.

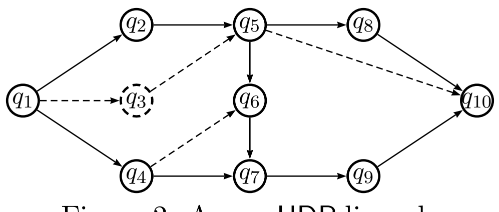


Figure 2: A non-UDRdigraph

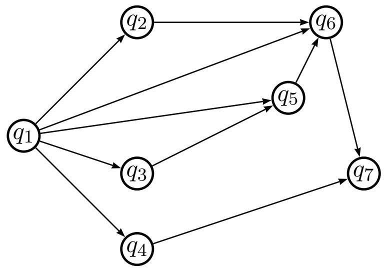


Figure 3: A UDRdigraph

Proof. Every acyclic NFA $A = (Q, \Sigma, q_0, F)$ is equivalent to a regular expression in *disjunctive normal form, i.e.*, $\alpha_1 + \cdots + \alpha_n$ where α_l is ϵ or a concatenation of symbols $a_{lj} \in \Sigma$, $\alpha_l = a_{l1} \cdots a_{lk_l}$, $1 \leq l \leq n$. From this r.e. we can construct a NFA $A' = (\{i\} \cup \{f\} \cup \cup_{1 \leq l \leq n} Q_l, \Sigma, i, \{f\})$ where $Q_l = \{q_{l1}, \ldots, q_{lk_l}\}$, $1 \leq l \leq n$, and such that $\delta'(i, a_{l1}) = \{q_{l1}\}$, $\delta'(q_{lj}, a_{lj+1}) = \{q_{lj+1}\}$, for $2 \leq j \leq k_{l-1}$, and $\delta'(q_{lk_l}, \epsilon) = \{f\}$. The automaton A' is trivially UDR. □

An interesting open problem is how to obtain from any acyclic NFA a UDR automaton with a minimal number of transitions (size).

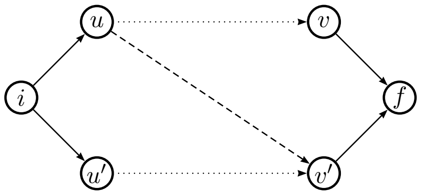
4 The Characterisation Theorem

Without loss of generality, we can just consider UDR digraphs (instead of automata), $D = (Q, E, i, f)$, where i denotes the initial vertex and f denotes the final vertex. It is obvious that,

Lemma 1. *A st-digraph subgraph of a UDR digraph is a UDR digraph.*

Lemma 2. *Let $D = (Q, E, i, f)$ be a UDR digraph. Let $i(0;k)$, $f(0;m)$, and $k, m > 1$. Suppose that u and u' are two distinct vertices adjacent from i, and v and v' are two distinct vertices adjacent to f. If there are two disjoint paths $iu - vf$ and $iu' - v'f$ then there cannot exist a path $iu - v'f$ nor a path $iu' - vf$.*

Proof. Suppose that exists a path $iu - v'f$, partially in dashed line in the picture below:



It is obvious that there will be a subgraph of D homeomorphic to $\mathsf{R}^{\rightarrow}$. This subgraph will contain the vertices $\{i, u, q', f\}$ (corresponding to the vertices of $\mathsf{R}^{\rightarrow}$), where q' is the first vertex common to the path $iu - v'f$ and to the path $iu' - v'f$ (at least v' or u'). The proof is analogous if there exists a path $iu' - vf$. □

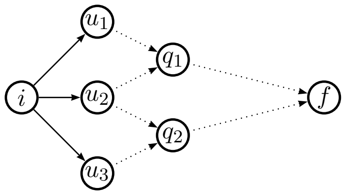
If $i(0;k)$ and $k > 1$, let U be the set of the vertices adjacent from i. For $u \in U$, let

$$X_u = \{u\} \cup \{q \mid q \in Q \setminus \{f\} \text{ and there exists a path } u - q\}.$$

Consider the binary relation $\downarrow$ in $U \times U$, such that $u \downarrow u'$ if and only if $X_u \cup X_{u'} \neq \emptyset$.

Lemma 3. *If D is a* UDR *digraph, the relation $\downarrow$ is an equivalence relation on U.*

Proof. The reflexivity and the symmetry are trivial, we only need to prove the transitivity. Let $u_1, u_2, u_3 \in U$ and $u_1 \downarrow u_2$ and $u_2 \downarrow u_3$. Then there exist $q_1 \in X_{u_1} \cup X_{u_2}$ and $q_2 \in X_{u_2} \cup X_{u_3}$. If $q_1 = q_2$ then $u_1 \downarrow u_3$. Suppose that $q_1 \neq q_2$ and that there is no path $q_1 - q_2$ nor $q_2 - q_1$. Then, we will have the following diagram:



The vertices $\{i, u_2, q_1, f\}$ define a subgraph homeomorphic to $\mathsf{R}^{\rightarrow}$, which contradicts the fact that D is UDR. □

Lemma 4. *Let $D = (Q, E, i, f)$ be a* UDR *digraph. Let $i(0; k)$, $f(0; m)$, and $k, m > 1$. Let U and X_u, for $u \in U$ be as above. If $[u]$ is an equivalence class of the relation $\downarrow$ with more than one element, then there exists $q \in Q \setminus \{f\}$ such that:*

1. *$q \in \cap_{u' \in [u]} X_{u'}$*
2. *For all $q' \in \cup_{u' \in [u]} X_{u'}$, every path $i - q' - f$ contains q.*

Proof. The existence of q, satisfying 1 is a directed consequence of the argument given in proof of the transitivity of $\downarrow$. If condition 2 does not hold, then the digraph D will have a subgraph homeomorphic to $\mathsf{R}^{\rightarrow}$. □

Now we can characterise the essential propriety of a UDR digraph.

Theorem 1. *Let $D = (Q, E, i, f)$ be a* UDR *digraph and $|Q| > 2$. Then D has at least a vertex q such that $q(1; 1)$.*

Proof. The proof is by induction on the number of vertices of the digraph, $|Q| = n$. If $n = 3$, it is trivially true. The UDR digraphs with 4 vertices are enumerated in Figure 4, and it is easy to see that all of them have one vertex q such that $q(1; 1)$. Assume that the theorem

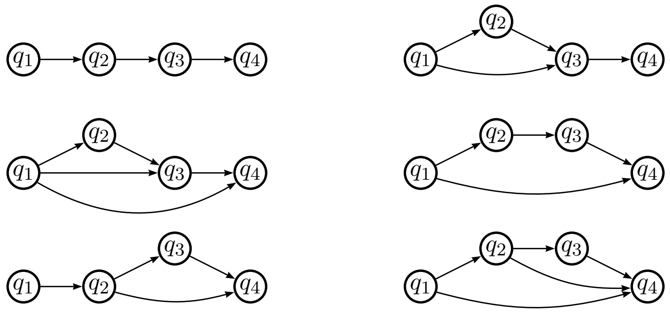


Figure 4: UDR digraphs with 4 vertices

holds for UDR digraphs with less than $n > 4$ vertices. We want to show that the same is true for a $D = (Q, E, i, F)$ with $|Q| = n$. If $i(0; 1)$, let $u \in Q \setminus \{f\}$ be the vertex adjacent from i.

Then the digraph $D' = (Q \setminus \{i\}, E \setminus \{(i,u)\}, u, f)$ is a UDR digraph with $n-1$ vertices, and by induction hypothesis it has a vertex q such that $q(1;1)$. An analogous argument can be given if $f(1;0)$. Let us suppose that $i(0;k_i)$ and $f(k_f;0)$, with $k_i, k_f > 1$. As we are seeking for an intermediate vertex, we can ignore the arc $(i,f) \in E$, if it exists. Let U, X_u and $\downarrow$ be as defined above. For every $u \in U$, we have one of the following cases:

1. The class $[u]$ has an unique element. Let $D' = (X_u \cup \{f\}, E', u, f)$, where E' has all the arcs of D with vertices in $X_u \cup \{f\}$. Then by Lemma 1, D' is a UDR digraph and has less than n vertices. If $|X_u \cup \{f\}| = 2$, then $u(1;1)$ in D. Otherwise, by induction hypothesis, D' has a vertex q' such that $q'(1;1)$, and, in D we have also $q'(1;1)$ (by Lemma 2).

2. The class $[u]$ has more than one element. Consider q as defined in Lemma 4. Let $D' = (\cup_{u' \in [u]} X_{u'}, E', i, q)$, where E' has all the arcs of D with vertices in $\cup_{u' \in [u]} X_{u'}$. Then by Lemma 4, D' is a UDR digraph and has less than n vertices. By induction hypothesis, it has a vertex q' such that $q'(1;1)$, and, in D we have also $q'(1;1)$.

□

5 Computing Regular Expressions from UDR Automata

Theorem 2. *Let $A = (Q, \Sigma, i, \delta, f)$ be a useful acyclic NFA with an unique final state. We can obtain a regular expression equivalent to A using the state elimination algorithm (SEA) in such way that in each step we remove a state q with $q(1;1)$ if and only if A is UDR.*

Proof. Suppose that A is UDR. If $Q = \{i, f\}$ then there is nothing to be proven. Otherwise, by Theorem 1, we can choose a state q to eliminate such that $q(1;1)$. The resulting EFA A' is UDR, since with that elimination step no automaton state increases its indegree nor its outdegree (at most the adjacent state to q, decreases its outdegree by one, and the adjacent state from q decreases its indegree by one).

If we apply the SEA to a useful acyclic non-UDREFA A , then the underlying digraph of A has a subgraph homeomorphic to $\mathsf{R}^{\rightarrow}$. Let q_1, q_2, q_3 and q_4 be the corresponding vertices. All of them have either indegree or outdegree greater than 1, and all those degrees cannot decrease to 1 unless one of the states is eliminated. □

Corollary 1. *Given a UDR automaton A, it is possible to construct an equivalent regular expression with size linear in the size of A.*

Proof. In the application of the SEA, in each step if q is the state to remove (and $q(1;1)$), then $\exists! q_1 \exists! q_2 : \delta'(q_1, q_2) = \alpha_{q_1 q_2} + \alpha_{q_1 q} \alpha_{q q_2}$ and all the other transitions are not changed. The size of the regular expression obtained is the number of transitions of A with alphabetic symbols (counting its multiplicities). □

6 An algorithm to decide if a digraph is UDR

Fortune and al. [FHW80] have shown that the problem of determining if an acyclic digraph $D = (V, E)$ has a subgraph homeomorphic to a fixed digraph $P = (V', E')$ has a polynomial time algorithm $O(n^{k+s})$, where $n = |V|$, $m = |E|$, $k = |E'|$ and $s = |V'|$. However, to determine if a digraph is UDR, a specialised algorithm can be designed.

Let us suppose that we have already determined that $D = (V, E, i, f)$ is an acyclic *st*-digraph, with a topological ordering o (this can be achieved in $O(n + m)$). For $v \in V$, let `AdjT(v)` be a list of vertices adjacent to v and let `AdjF(v)` be a list of vertices adjacent from `v`.

In Figure 6 we present the algorithm in pseudo-code. The vertices of the digraph are going to be traversed in topological order. Each arc (`u`,`v`)$\in E$ is annotated with a list of relevant vertices with outdegree greater than 1, that precedes `v` (in a path from `i`). Those labels are denoted by `label(u,v)` and are implemented with references to other labels. The empty list is denoted by `nil` and `l.v` represents the concatenation of `v` with the list `l`. We use $\leftarrow$ as the standard assignment operator. The semantics of an expression e_1`=` e_2 is defined by induction on the type of e_2:

$$\begin{array}{lcll} [\![e_1 = val]\!] & \equiv & e_1 \leftarrow val & \text{if } e_2 \text{ is a value } val, \\ [\![e_1 = ref(e_3)]\!] & \equiv & [\![e_1 = e_3]\!] & \text{if } e_2 \text{ is a reference } ref(e_3). \end{array}$$

We use other standard list operations as `first` (first element of the list),`last` (last element of the list) and `butlast` (the list without the last element).

The algorithm proceeds as follows. While the visited vertices have indegree less than 2, the relevant predecessors are collected (lines 19–26). If a vertice `v1` has indegree greater than 1, then either we can "resolve" all the precedent bifurcations or the digraph must be non UDR (lines 4–18). A junction is "resolved" if there are two vertices, `v` and `vi` adjacent to `v1` with labels that have equal values but which are not references to the same object. In this case, those labels can be unified and if `vp` is the last element of those labels, `outd(vp)` is decreased by 1. When that value is 1, the vertice `vp` is no longer relevant an can be deleted from the labels (lines 7–15).

Theorem 3. *The algorithm* `udrp` *is correct and has time complexity* $O(n^2 \log n)$.

Proof. (Sketch) If the digraph D is not UDR, it has a subgraph homeomorphic to $\mathsf{R}^{\rightarrow}$. Let q_1, q_2, q_3 and q_4 be the corresponding vertices. The vertice q_3 has indegree greater than 1. When q_3 is visited, there are $v, v' \in$ `AdjF(`q_3`)` such that $q_1 \in$ `label(`v`,`q_3`)` and $q_1, q_2 \in$ `label(`v'`,`q_3`)`. Those labels can never refer to the same object, thus the algorithm must return 0.

Suppose that the algorithm returns 0. Then there exists $v \in \mathtt{V}$ such that $v(\mathtt{nf}; k)$ with $\mathtt{nf} > 1$ and there exist $v', v'' \in$ `AdjF(`v`)` such that `label(`v'`,`v`)` and `label(`v''`,`v`)` are diferent. Let $q_1 \in$ (`label(`v'`,`v`)`$\backslash$`label(`v''`,`v`)`). The outdegree of q_1 is greater than 1 (otherwise it was not relevant) and $q_1 \neq i$. Then there must exist two disjoint paths $i - q_1 - v_1$ and $i - v_1$. There exists a path $q_1 - f$ that does not go through v_1 and a path $v_1 - f$ that does not go through q_1. Thus D has a subgraph homeomorphic to $\mathsf{R}^{\rightarrow}$, defined by i, q_1, v_1, q', where q' is a common ancestor of q_1 and v_1 (at least f). Thus the digraph D is not UDR.

Now, we analyse the time complexity. The total cost of executing the lines 23–26 is $O(m)$. For each $v \in V$, the lines 4–18 can be, in the worst case, executed $O(n \log n)$ (as the annotations can be sorted only once). The time complexity of `udrp` is $O(n^2 \log n)$. □

Proposition 2. *If A is UDR, the algorithm* `udrp` *can be used to compute an equivalent regular expression with size linear in the size of A.*

Proof. (Sketch) Let $A = (Q, \Sigma, \delta, i, f)$ be an EFA. In the algorithm `udrp` we can extend the algorithm annotations of the arcs to contain the automaton's transition labels. For $(q, q') \in Q$, let `label(`q`,`q'`)` be the list of vertices as before and let `regexp(`q`,`q'`)` be the

```
udrp {
  for v1 in V ordered topologically do
     nf ← |AdjF(v1)|
     while nf > 1 do
        max ← max{|label(v,v1)|: v ∈ AdjF(v1)}
        lmax ← { v ∈ AdjF(v1): |label(v,v1)| == max}
        vi ← first(lmax)
           if v in lmax\{vi} and
           (ref(label(v,v1)) <> ref(label(vi,v1)) )
              and (label(v,v1) = = label(vi,v1)) then
                  vp ← last(label(vi,v1))
                  outd(vp) ← outd(vp) - 1
                  label(v,v1) = ref(label(vi,v1))
                  if outd(vp) = = 1 then
                     label(v,v1) = butlast(label(v,v1))
                  nf ← nf - 1
                  continue
            else return 0 % is not UDR
      if v1 = = i then
         lp = nil
      else
         lp = ref(label(first(AdjF(v1)),v1))
      for v2 in AdjT(v1) do
         if outd(v1) < > 1 then
            label(v1,v2) ← lp.v1
         else label(v1,v2) ← lp
  return 1 % is UDR
}
```

Figure 5: Determining if a digraph is UDR

associated regular expression. Whenever a state $q_1(k;1)$ is visited and its incident arcs have been resolved (lines 19–26) let `rp` be the regular expression correspondent to the label `lp`. Then, in line 26, we add the instruction

$$\texttt{regexp(v1,v2)} \leftarrow \texttt{rp}\cdot\texttt{regexp(v1,v2)}.$$

Whenever a junction is resolved (in line 13) we add an instruction

```
regexp(v,v1) = regexp(v,v1) + regexp(vi,v1).
```

And in line 15, we add another concatenation:

$$\texttt{regexp(vi,v1) = rp }\cdot\texttt{regexp(vi,v1)},$$

where `rp` is the regular expression correspondent to the vertice `vp` (there must be only one). In the end we obtain the same regular expression of Corollary 1. □

7 Relationship with other works

There is no much papers in the literature on the characterization of the conversion from NFAs to regular expressions, as was pointed by Ellul and al. [ESwW02]. Giammarresi and al. [GPWZ04] characterised the automata generated by the Thompson method [Tho68] for converting regular expressions to automata. They called the underlying digraphs, Thompson digraphs. By induction on the structure of those digraphs we can prove that

Corollary 2. *Every acyclic Thompson digraph is* UDR*.*

In the same way Caron and Ziadi [CZ00] characterised the automata generated by the Glushkov method for transforming regular expressions into finite automata [Yu97]. As these automata may have more than one final state, we cannot directly compare the acyclic Glushkov digraphs and the UDR digraphs. Introducing ϵ-transitions it is possible to obtain an equivalent UDR automata equivalent to an acyclic Glushkov automaton.

8 Conclusions

In this work we have characterised a small class of acyclic automata for which it is easy to find short equivalent regular expressions. Determining the exact number of UDR digraphs (and automata) is one of our next goals. Would be interesting to investigate an algorithm to transform any acyclic automaton into a UDRequivalent with minimal size, although in general this problem is NP-complete (it is equivalent to the minimisation of regular expressions without the $\star$ operator). Finally, we hope to extend the UDR notion to acyclic automata with several final states and some restricted classes of cyclic automata.

References

[BJG01] J. Bang-Jensen and G. Gutin. *Digraphs: Theory, Algorithms, and Applications.* Monographs in Mathematics. Springer-Verlag, 2001.

[CZ00] Pascal Caron and Djelloul Ziadi. Characterization of Glushkov automata. *Theor. Comput. Sci.*, 233(1-2):75–90, 2000.

[ESwW02] Keith Ellul, Jeffrey Shallit, and Ming wei Wang. Regular expressions: New results and open problems. Talk at the DCFS 2002 conference (Descriptional Complexity of Formal Systems), London, Ontario, 2002.

[FHW80] Steven Fortune, John Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10:111–121, 1980.

[GPWZ04] Dora Giammarresi, Jean-Luc Ponty, Derick Wood, and Djelloul Ziadi. A characterization of Thompson digraphs. *Discrete Applied Mathematics*, 134(1-3):317–337, 2004.

[Har69] Frank Harary. *Graph Theory*. Addison Wesley, 6 edition, 1969.

[HMU00] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison Wesley, 2000.

[JR93] Tao Jiang and B. Ravikumar. Minimal NFA problems are hard. *SIAM Journal of Computation*, pages 1117–1141, 1993.

[Kle56] S. C. Kleene. Representation of events in nerve nets and finite automata. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 3–41. Princeton University Press, 1956.

[Tho68] K. Thompson. Regular expression search algorithm. *Communications of the ACM*, 11(6):410–422, 1968.

[Yu97] Sheng Yu. Regular languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 1. Springer-Verlag, 1997.