Small NFAs from Regular Expressions: Some Experimental Results*

Hugo Gouveia**, Nelma Moreira, Rogério Reis

DCC-FC & LIACC, Universidade do Porto
R. do Campo Alegre 1021/1055, 4169-007 Porto, Portugal

Abstract. Regular expressions (REs), because of their succinctness and clear syntax, are the common choice to represent regular languages. However, efficient pattern matching or word recognition depend on the size of the equivalent nondeterministic finite automata (NFA). We present the implementation of several algorithms for constructing small ε-free NFAs from REs within the **FAdo** system, and a comparison of regular expression measures and NFA sizes based on experimental results obtained from uniform random generated REs. For this analysis, nonredundant REs and reduced REs in star normal form were considered.

1 Introduction

Regular expressions (REs), because of their succinctness and clear syntax, are the common choice to represent regular languages. Equivalent deterministic finite automata (DFA) would be the preferred choice for pattern matching or word recognition as these problems can be solved efficiently by DFAs. However, minimal DFAs can be exponentially bigger than REs. Nondeterministic finite automata (NFA) obtained from REs can have the number of states linear with respect to (w.r.t) the size of the REs. Because NFA minimization is a PSPACE-complete problem other methods must be used in order to obtain small NFAs usable for practical purposes. Conversion methods from REs to equivalent NFAs can produce NFAs without or with transitions labelled with the empty word (ε-NFA). Here we consider several constructions of small ε-free NFAs that were recently developed or improved [Mir66,Ant96,CZ02,HSW01,IY03a,COZ07], and that are related with the one of Glushkov and McNaughton-Yamada [Glu61,MY60]. The NFA size can be reduced by merging equivalent states [IY03b,ISOY05]. Another solution is to simplify the REs before the conversion [EKSW05]. Gruber and Gulan [GG09] showed that REs in reduced star normal form (snf) achieve some conversion lower bounds. Our experimental results corroborate that REs must be converted to reduced snf. In this paper we present the implementation within the **FAdo** system [FAd10] of several algorithms for constructing small ε-free

* This work was partially funded by Fundação para a Ciência e Tecnologia (FCT) and Program POSI, and by the project ASA (PTDC/MAT/65481/2006).

** Hugo Gouveia passed away in December, 2009. Through 2009 he was funded by a LIACC-FCT scholarship for young undergraduated researchers.

NFAs from REs, and a comparison of regular expression measures and NFA sizes based on experimental results obtained from uniform random generated REs. We consider nonredundant REs and REs in reduced snf in particular.

2 Regular Expressions and Finite Automata

Let Σ be an *alphabet* (set of *letters*). A *word* w over Σ is any finite sequence of letters. The *empty word* is denoted by ε. Let $\Sigma^\star$ be the set of all words over Σ. A *language* over Σ is a subset of $\Sigma^\star$. The set R of *regular expressions* (RE) over Σ is defined by:

$$\alpha := \emptyset \mid \varepsilon \mid \sigma \in \Sigma \mid (\alpha + \alpha) \mid (\alpha \cdot \alpha) \mid \alpha^\star,$$

where the operator $\cdot$ (concatenation) is often omitted. The language $\mathcal{L}(\alpha)$ associated to $\alpha \in \mathsf{R}$ is inductively defined as follows: $\mathcal{L}(\emptyset) = \emptyset$, $\mathcal{L}(\varepsilon) = \{\varepsilon\}$, $\mathcal{L}(\sigma) = \{\sigma\}$ for $\sigma \in \Sigma$, $\mathcal{L}((\alpha+\beta)) = \mathcal{L}(\alpha) \cup \mathcal{L}(\beta)$, $\mathcal{L}((\alpha\cdot\beta)) = \mathcal{L}(\alpha)\cdot\mathcal{L}(\beta)$, and $\mathcal{L}(\alpha^\star) = \mathcal{L}(\alpha)^\star$. Two regular expressions α and β are *equivalent* if $\mathcal{L}(\alpha) = \mathcal{L}(\beta)$, and we write $\alpha = \beta$. The algebraic structure $(\mathsf{R}, +, \cdot, \emptyset, \varepsilon)$ constitutes an idempotent semiring, and with the unary operator $\star$, a Kleene algebra. There are several ways to measure the size of a regular expression. The *size* (or *ordinary length*) $|\alpha|$ of $\alpha \in \mathsf{R}$ is the number of symbols in α, including parentheses (but not the operator $\cdot$); the *alphabetic size* $|\alpha|_\Sigma$ (or $\mathrm{alph}(\alpha)$) is its number of letters (multiplicities included); and the *reverse polish notation size* $\mathrm{rpn}(\alpha)$ is the number of nodes in its syntactic tree. The *alphabetic size* is considered in the literature [EKSW05] the most useful measure, and will be the one we consider here for several RE measure comparisons. Moreover all these measures are identical up a constant factor if the regular expression is reduced [EKSW05, Th. 3]. Let $\varepsilon(\alpha)$ be ε if $\varepsilon \in \mathcal{L}(\alpha)$, and $\emptyset$ otherwise. A regular expression α is *reduced* if it is normalised w.r.t the following equivalences (rules):

$$\begin{array}{lll}
\varepsilon \cdot \alpha = \alpha \cdot \varepsilon = \alpha & & \varepsilon + \alpha = \alpha + \varepsilon = \alpha, \text{ where } \varepsilon(\alpha) = \varepsilon \\
\emptyset \cdot \alpha = \alpha \cdot \emptyset = \emptyset & & \alpha^{\star\star} = \alpha^\star \\
\emptyset + \alpha = \alpha + \emptyset = \alpha & & \emptyset^\star = \varepsilon^\star = \varepsilon
\end{array}$$

A RE can be transformed into an equivalent reduced RE in linear time.

A *nondeterministic automaton* (NFA) $\mathcal{A}$ is a quintuple $(Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of states, Σ is the alphabet, $\delta \subseteq Q \times \Sigma \times Q$ the transition relation, q_0 the initial state, and $F \subseteq Q$ the set of final states. The *size* of an NFA is $|Q| + |\delta|$. For $q \in Q$ and $\sigma \in \Sigma$, we denote by $\delta(q, \sigma) = \{p \mid (q, \sigma, p) \in \delta\}$, and we can extend this notation to $w \in \Sigma^\star$, and to $R \subseteq Q$. The *language* accepted by $\mathcal{A}$ is $\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^\star \mid \delta(q_0, w) \cap F \neq \emptyset\}$. Two NFAs are *equivalent*, if they accept the same language. If two NFAs A and B are isomorphic, and we write $A \simeq B$. An NFA is *deterministic* (DFA) if for each pair $(q, \sigma) \in Q \times \Sigma$ there exists at most one q' such that $(q, \sigma, q') \in \delta$. A DFA is *minimal* if there is no equivalent DFA with fewer states. Minimal DFA are unique up to isomorphism. Given an equivalence relation E on Q, for $q \in Q$ let $[q]_E$ be the class of q w.r.t E, and for

$T \subseteq Q$ let $T/_E = \{[q]_E \mid q \in T\}$. The equivalence relation E is *right invariant* w.r.t an NFA $\mathcal{A}$ if $E \subseteq (Q \setminus F)^2 \cup F^2$ and for any $p, q \in Q$, $\sigma \in \Sigma$ if $p\ E\ q$, then $\delta(p,\sigma)/_E = \delta(q,\sigma)/_E$. The quotient automaton $\mathcal{A}/_E = (Q/_E, \Sigma, \delta_E, [q_0]_E, F/_E)$, where $\delta_E = \{([p]_E, \sigma, [q]_E) \mid (p,\sigma,q) \in \delta\}$, satisfies $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}/_E)$. Given two equivalence relations over a set Q, G and H, we say that G is *finer* than H (and H *coarser* than G) if and only if $G \subseteq H$.

3 Small NFAs from Regular Expressions

We consider three methods for constructing small NFAs $\mathcal{A}$ from a regular expression α such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\alpha)$, i.e., they are *equivalent*.

3.1 Position Automata

The position automaton construction was independently proposed by Glushkov, and McNaughton and Yamada [Glu61,MY60]. Let $\mathrm{Pos}(\alpha) = \{1, 2, \ldots, |\alpha|_\Sigma\}$ for $\alpha \in \mathsf{R}$, and let $\mathrm{Pos}_0(\alpha) = \mathrm{Pos}(\alpha) \cup \{0\}$. We consider the expression $\overline{\alpha}$ obtained by marking each letter σ with its position i in α, σ_i. The same notation is used to remove the markings, i.e., $\overline{\overline{\alpha}} = \alpha$. For $\alpha \in \mathsf{R}$ and $i \in \mathrm{Pos}(\alpha)$, let $\mathsf{first}(\alpha) = \{j \mid \exists w \in \overline{\Sigma}^\star, \sigma_j w \in \mathcal{L}(\overline{\alpha})\}$, $\mathsf{last}(\alpha) = \{j \mid \exists w \in \overline{\Sigma}^\star, w\sigma_j \in \mathcal{L}(\overline{\alpha})\}$, and $\mathsf{follow}(\alpha, i) = \{j \mid \exists u, v \in \overline{\Sigma}^\star, u\sigma_i\sigma_j v \in \mathcal{L}(\overline{\alpha})\}$. Let $\mathsf{follow}(\alpha, 0) = \mathsf{first}(\alpha)$. The *position automaton* for $\alpha \in \mathsf{R}$ is $\mathcal{A}_{\mathrm{pos}}(\alpha) = (\mathrm{Pos}_0(\alpha), \Sigma, \delta_{\mathrm{pos}}, 0, F)$, with $\delta_{\mathrm{pos}} = \{(i, \overline{\sigma_j}, j) \mid \ j \in \mathsf{follow}(\alpha, i)\}$ and $F = \mathsf{last}(\alpha) \cup \{0\}$ if $\varepsilon(\alpha) = \varepsilon$, and $F = \mathsf{last}(\alpha)$, otherwise. We note that the number of states of $\mathcal{A}_{\mathrm{pos}}(\alpha)$ is exactly $|\alpha|_\Sigma + 1$. Other interesting property is that $\mathcal{A}_{\mathrm{pos}}$ is *homogeneous*, i.e., all transitions arriving at a given state are labelled by the same letter. Brüggemann-Klein [BK93] showed that the construction of $\mathcal{A}_{\mathrm{pos}}$ can be obtained in $\mathcal{O}(n^2)$ ($n = |\alpha|$) if the regular expression α is in the so called *star normal form* (snf), i.e., if for each subexpression $\beta^\star$ of α, $\forall x \in \mathsf{last}(\beta)$, $\mathsf{follow}(\beta, x) \cap \mathsf{first}(\beta) = \emptyset$ and $\varepsilon(\beta) = \emptyset$. For every $\alpha \in \mathsf{R}$ there is an equivalent RE in star normal form $\alpha^\bullet$ that can be computed in linear time and such that $\mathcal{A}_{\mathrm{pos}}(\alpha) \simeq \mathcal{A}_{\mathrm{pos}}(\alpha^\bullet)$.

3.2 Follow Automata

Ilie and Yu [IY03a] introduced the construction of the follow automaton from a RE. Their initial algorithm begins by converting $\alpha \in \mathsf{R}$ into an equivalent ε-NFA from which the follow automaton $\mathcal{A}_f(\alpha)$ is obtained. For efficiency reasons we implemented that method in the **FAdo** library. The *follow automaton* is a quotient of the position automaton w.r.t the right-invariant equivalence given by the *follow relation* $\equiv_f \subseteq \mathrm{Pos}_0^2$ that is defined by:

$$\forall x, y \in \mathrm{Pos}_0(\alpha), x \equiv_f y \text{ if } \begin{array}{l}\text{(i) both } x, y \text{ or none belong to } \mathsf{last}(\alpha) \text{ and} \\ \text{(ii) } \mathsf{follow}(\alpha, x) = \mathsf{follow}(\alpha, y)\end{array}$$

Proposition 1 (Ilie and Yu, Thm. 23). $\mathcal{A}_f(\alpha) \simeq \mathcal{A}_{\mathrm{pos}}(\alpha)/_{\equiv_f}$.

3.3 Partial Derivative Automata

Let $S \cup \{\beta\}$ be a set of regular expressions. Then $S \odot \beta = \{\alpha\beta \,|\, \alpha \in S\}$ if $\beta \neq \emptyset$ and $S \odot \emptyset = \emptyset$. For $\alpha \in \mathsf{R}$ and $\sigma \in \Sigma$, the set $\partial_\sigma(\alpha)$ of *partial derivatives* of α w.r.t. σ is defined inductively as follows:

$$\begin{array}{ll} \partial_\sigma(\emptyset) = \partial_\sigma(\varepsilon) = \emptyset & \partial_\sigma(\alpha+\beta) = \partial_\sigma(\alpha) \cup \partial_\sigma(\beta) \\ \partial_\sigma(\sigma') = \begin{cases} \{\varepsilon\} & \text{if } \sigma' \equiv \sigma \\ \emptyset & \text{otherwise} \end{cases} & \partial_\sigma(\alpha\beta) = \begin{cases} \partial_\sigma(\alpha) \odot \beta \cup \partial_\sigma(\beta) & \text{if } \varepsilon(\alpha) = \varepsilon \\ \partial_\sigma(\alpha) \odot \beta & \text{otherwise.} \end{cases} \\ \partial_\sigma(\alpha^\star) = \partial_\sigma(\alpha) \odot \alpha^\star & \end{array}$$

This definition can be extended to sets of regular expressions, words, and languages. Given $\alpha \in \mathsf{R}$ and $\sigma \in \Sigma$, $\partial_\sigma(S) = \cup_{\alpha \in S} \partial_\sigma(\alpha)$ for $S \subseteq \mathsf{R}$, $\partial_\varepsilon(\alpha) = \{\alpha\}$, $\partial_{w\sigma}(\alpha) = \partial_\sigma(\partial_w(\alpha))$ for $w \in \Sigma^\star$, and $\partial_L(\alpha) = \cup_{w \in L} \partial_w(\alpha)$ for $L \subseteq \Sigma^\star$. The *set of partial derivatives* of α is denoted by $\mathsf{PD}(\alpha) = \{\partial_w(\alpha) \mid w \in \Sigma^\star\}$.

Given a regular expression α, the partial derivative automaton $\mathcal{A}_{\mathrm{pd}}(\alpha)$, introduced by Mirkin and Antimirov [Mir66,Ant96], is defined by

$$\mathcal{A}_{\mathrm{pd}}(\alpha) = (\mathsf{PD}(\alpha), \Sigma, \delta_{pd}, \alpha, \{q \in \mathsf{PD}(\alpha) \mid \varepsilon(q) = \varepsilon\}),$$

where $\delta_{pd}(q, \sigma) = \partial_\sigma(q)$, for all $q \in \mathsf{PD}(\alpha)$ and $\sigma \in \Sigma$.

Proposition 2 (Mirkin and Antimirov). $\mathcal{L}(\mathcal{A}_{\mathrm{pd}}(\alpha)) = \mathcal{L}(\alpha)$.

Champarnaud and Ziadi [CZ02] showed that the partial derivative automaton is also a quotient of the position automaton. Champarnaud *et al.* [COZ07] proved that for RE reduced and in star normal form the size of its partial derivative automaton $\mathcal{A}_{\mathrm{pd}}$ is always smaller than the one of its follow automaton $\mathcal{A}_f$.

3.4 Complexity

The automata here presented $\mathcal{A}_{\mathrm{pos}}$, $\mathcal{A}_f$ and $\mathcal{A}_{\mathrm{pd}}$ can in worst-case be constructed in time and space $\mathcal{O}(n^2)$, and have, in worst-case, size $\mathcal{O}(n^2)$, where n is the size of the RE. Recently, Nicaud [Nic09] showed that on the average-case the size of the $\mathcal{A}_{\mathrm{pos}}$ automata is linear. The best worst case construction of ε-free NFAs from RE is the one presented by Hromkovic *et al.* [HSW01] that can be constructed and have size $\mathcal{O}(n(\log n^2))$. However this construction is not considered here.

4 NFAs Reduction with Equivalences

It is possible to obtain in time $\mathcal{O}(n \log n)$ a (unique) minimal DFA equivalent to a given one. However NFA state minimization is PSPACE-complete and, in general, minimal NFAs are not unique. Considering the exponential succinctness of NFAs w.r.t DFAs, it is important to have methods to obtain small NFAs. Any right-invariant equivalence relation over Q w.r.t $\mathcal{A}$ can be used to diminish the size of $\mathcal{A}$ (by computing the quotient automaton). The *coarsest right-invariant equivalence* $\equiv_R$ can be computed by an algorithm similar to the one used to minimize DFAs [IY03b]. This coincides with the notion of (auto)-bisimulation,

widely applied to transition systems and which can be computed efficiently (in almost linear time) by the Paige and Tarjan algorithm [PT87]. A *left-invariant* equivalence relation on Q w.r.t $\mathcal{A}$ is any right-invariant equivalence relation on the reversed automaton of $\mathcal{A}$, $\mathcal{A}^r = (Q, \Sigma, \delta_r, F, \{q_0\})$, where $q \in \delta^r(p, \sigma)$ if $p \in \delta(q, \sigma)$ (and we allow multiple initial states). The *coarsest left-invariant equivalence* on Q w.r.t $\mathcal{A}$, $\equiv_L$, is $\equiv_R$ of $\mathcal{A}^r$.

5 FAdo Implementations

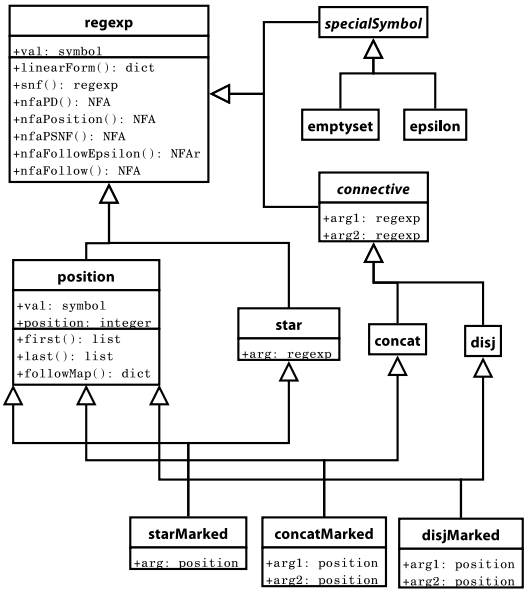

Fig. 1. FAdo classes for REs

FAdo [FAd10,MR05,AAA+09] is an ongoing project that aims to provide a set of tools for symbolic manipulation of formal languages. To allow high-level programming with complex data structures, easy prototyping of algorithms, and portability are its main features. It is mainly developed in the `Python` programming language. In **FAdo**, regular expressions and finite automata are implemented as `Python` classes.

Figure 1 presents the classes for REs and the main methods described in this paper. The `regexp` class is the base class for all REs and the class `position` is the base class for marked REs. The methods `first()`, `last()` and `followMap()` (where $\mathsf{follow}(\alpha, x) = \{\beta \mid (x, \beta) \in \texttt{followMap()}\}$) are coded for each subclass. The method `nfaPosition()` implements a construction of the $\mathcal{A}_{\text{pos}}$ automaton without reduction to snf. Brüggemann-Klein algorithm is implemented by the

`nfaPSNF()` method. The methods `nfaFollowEpsilon()` and `nfaFollow()` implement the construction of the $\mathcal{A}_f$ via an ε-NFA. The exact text of all these algorithms is too long to present here. The method `nfaPD()` computes the $\mathcal{A}_{\text{pd}}$ and uses the method `linearForm()`. This method implements the function lf() defined by Antimirov [Ant96] to compute the partial derivatives of a RE w.r.t all letters. Algorithm 1 presents the computation of the $\mathcal{A}_{\text{pd}}$.

Algorithm 1 Computation of $\mathcal{A}_{\text{pd}}$

$Q \leftarrow \{\alpha\}$
$\delta \leftarrow \emptyset$
$F \leftarrow \emptyset$
stack $\leftarrow \{\alpha\}$
while pd $\leftarrow$ POP(stack) **do**
  for (head,tail) $\in$ lf(pd) **do**
    if $\neg$ tail $\in Q$ **then**
      $Q \leftarrow Q \cup \{\text{tail}\}$
      PUSH(stack,pd)
    end if
    δ(pd,head) $\leftarrow \delta$(pd,head) $\cup \{\text{tail}\}$
  end for
  if ε(pd) **then**
    $F \leftarrow F \cup \{\text{pd}\}$
  end if
end while

Figure 2 presents the classes for finite automata. `FA` is the abstract class for finite automata. The class `NFAr` includes the inverse of the transition relation, that is not included in the `NFA` class for efficiency reasons. In the `NFA` class the method `autobisimulation()` implements a naïve version for compute $\equiv_R$, as presented in Algorithm 2. Given an equivalence relation the method `equivReduced()` builds the quotient automaton. Given an NFA **A**, **A**.`rEquiv()` corresponds to $\mathcal{A}/_{\equiv_R}$, **A**.`lEquiv()` to $\mathcal{A}/_{\equiv_L}$ and **A**.`lrEquiv()` to $(\mathcal{A}/_{\equiv_L})/_{\equiv_R}$. We refer the reader to Gouveia [Gou09] and to **FAdo** webpage [FAd10] for more implementation details.

Algorithm 2 Computation of the set R corresponding to $\equiv_R$.

$\overline{R} \leftarrow \emptyset$
for $(p, q) \in Q \times Q$ **do**
  if $p \in F \not\Leftrightarrow q \in F$ **then**
    $\overline{R} \leftarrow \overline{R} \cup \{(p, q)\}$
  end if
end for
while $\exists (x, y) \notin \overline{R}\colon \exists \sigma \in \Sigma\colon \exists z \in \delta(x, \sigma)\colon \forall w \in \delta(y, \sigma)\colon z\overline{R}w$ **do**
  $\overline{R} \leftarrow \overline{R} \cup \{(x, y), (y, x)\}$
end while
$R \leftarrow (Q \times Q) \setminus \overline{R}$
Return R

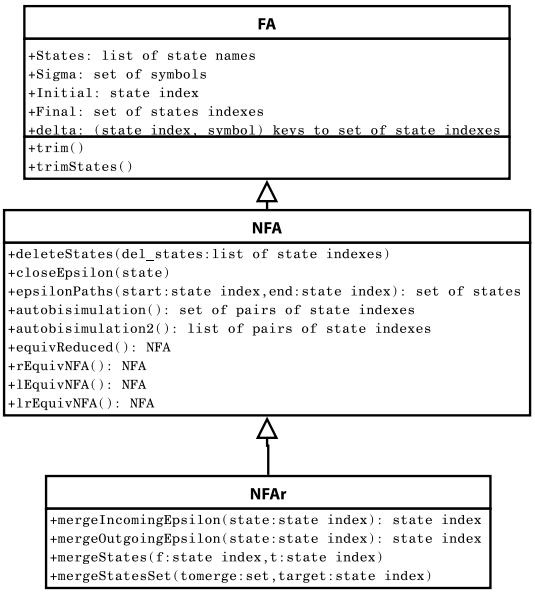

Fig. 2. FAdo classes for NFAs

6 RE Random Generator

Uniform random generators are essential to obtain reliable experimental results that can provide information about the average-case analysis of both computational and descriptional complexity. For general regular expressions, the task is somehow simplified because they can be described by small unambiguous context-free grammars from which it is possible to build uniform random generators [Mai94]. In the **FAdo** system we implemented the method described by Mairson [Mai94] for the generation of context-free languages. The method accepts as input a context-free grammar and the size of the words to be uniformly random generated.

The random samples need to be consistent and large enough to ensure statistically significant results. To have these samples readily available, the **FAdo** system includes a dataset of random RE, that can be accessed online. The current dataset was obtained using a grammar for REs given by Lee and Shallit [LS05], and that is presented in Figure 3. This grammar generates REs normalized by rules that define reduced REs, except for certain cases of the rule: $\varepsilon + \alpha$, where $\varepsilon(\alpha) = \varepsilon$. The database makes available random samples of REs with different sizes between 10 and 500 and with alphabet sizes between 2 and 50.

7 Experimental Results

In order to experiment with several properties of REs and NFAs we developed a generic program to ease to add/remove the methods to be applied and to specify

$$
\begin{aligned}
S &:= A \mid C \mid E \mid \Sigma \mid \varepsilon \mid \emptyset \\
C &:= C\ R \mid R\ R \\
R &:= (\ A\ ) \mid E \mid \Sigma \\
E &:= (\ A\ )^\star \mid (\ C\ )^\star \mid \Sigma^\star \\
A &:= \varepsilon\ +\ X \mid Y + Z \\
X &:= T \mid T\ +\ X \\
T &:= C \mid \Sigma \\
Y &:= Z \mid Y\ +\ Z \\
Z &:= C \mid E \mid \Sigma
\end{aligned}
$$

Fig. 3. Grammar for almost reduced REs. The start symbol is S.

the data, from the database, to be used. Here we are interested in the comparison of several REs descriptional measures with measures of the NFAs obtained using the methods earlier described.

For REs we considered the following properties: the alphabetic size (alph); the rpn size (rpn); test if it is in snf (snf); if not in snf, compute the snf and its measures (alph,rpn); test if it is reduced; if not reduced, reduce it and compute its measures (alph,rpn); the number of states (sc) and number of transitions (tc) of the equivalent minimal DFA.

For each NFA ($\mathcal{A}_{\text{pos}}$, $\mathcal{A}_f$, and $\mathcal{A}_{\text{pd}}$) we considered the following properties: the number of states ($|Q|$); the number of transitions ($|\delta|$); if it is deterministic (det); and if it is homogeneous (hom). All these properties were also considered for the case where the REs are in snf, and for the NFAs obtained after applying the invariant equivalences $\equiv_R$, $\equiv_L$, and their composition.

All tests were performed on samples of $10,000$ uniformly random generated REs. Each sample contains REs of size 50, 100, 200 and 300, respectively.

Table 1 shows some results concerning REs. The ratio of alphabetic size to rpn size is almost constant for all samples. Almost all REs are in snf, so we do not presented the measures after transforming into snf. This fact is relevant as the REs were generated only *almost reduced*. The column snfr contains the percentage of REs for which their snf are reduced. It is interesting to note that the average number of states of the minimal DFA (sc) is near alph (i.e., near the number of states of $\mathcal{A}_{\text{pos}}$). The standard deviation is here very high. For the sample of size 300, however, 99% of the REs have $160 \leq \text{sc} \leq 300$. More theoretical work is needed for a deeper understanding of these results.

Table 1. Statistical values for RE measures, where (avg) is the average and (std) the standard deviation.

size	alph		rpn		$\frac{\text{rpn}}{\text{alph}}$	snf	snfr	sc		tc		$\frac{\text{sc}}{\text{alph}}$	$\frac{\text{tc}}{\text{alph}}$
	avg	std	avg	std				avg	std	avg	std		
50	42	6.39	85	10.80	2.04	97%	99%	38	9.42	44	6.39	0.92	1.05
100	77	10.26	161	17.41	2.08	93%	98%	69	20.00	89	37.47	0.89	1.15
200	165	25.75	340	43.83	2.06	90%	97%	160	91.58	203	186.10	0.97	1.24
300	247	38.06	511	64.96	2.06	87%	95%	258	300.01	343	617.51	1.04	1.4

Table 2 and Table 3 show some results concerning the NFAs obtained from REs. In Table 2 the values not in percentage are average values. If $\mathcal{A}_{\rm pos}$ is deterministic then the REs is unambiguous (and strong unambiguous, if in snf) [BK93]. The results obtained suggest that perhaps 25% of the reduced REs are strong unambiguous. Note that if $\mathcal{A}_{\rm pos}$ is not deterministic, almost certainly, neither $\mathcal{A}_{\rm pd}$ nor $\mathcal{A}_f$ are. For reasonable sized REs, although $\mathcal{A}_{\rm pos}$ are homogeneous it is unlikely that either $\mathcal{A}_{\rm pd}$ or $\mathcal{A}_f$ will be so. It is not significant the difference between $|Q_f|$ and $|Q_{\rm pd}|$. On average $|\delta_{\rm pos}|$ seems linear in the size of the RE, and that fact was recently proved by Nicaud [Nic09].

Table 2. NFA measures.

size	$\mathcal{A}_{\rm pos}$				$\mathcal{A}_f$				$\mathcal{A}_{\rm pd}$			
	$\lvert Q_{\rm pos}\rvert$	$\lvert\delta_{\rm pos}\rvert$	det	hom	$\lvert Q_f\rvert$	$\lvert\delta_f\rvert$	det	hom	$\lvert Q_{\rm pd}\rvert$	$\lvert\delta_{\rm pd}\rvert$	det	hom
50	43	51	49.1%	100%	38	44	49.3%	13.7%	38	44	49.4%	13.6%
100	78	104	16.0%	100%	67	84	17.0%	1.0%	66	83	17.0%	1.0%
200	166	211	27.6%	100%	148	175	27.7%	1.5%	146	173	27.7%	1.4%
300	248	317	23.9%	100%	222	262	23.9%	0.5%	220	260	23.9%	0.5%

Table 3. Ratios of NFA measures.

size	$\frac{\lvert\delta_{\rm pos}\rvert}{\rm alph+1}$	$\frac{\lvert Q_f\rvert}{/\rm alph+1}$	$\frac{\lvert\delta_f\rvert}{\rm alph+1}$	$\frac{\lvert Q_{\rm pd}\rvert}{\rm alph+1}$	$\frac{\lvert\delta_{\rm pd}\rvert}{\rm alph+1}$	$\frac{\lvert\delta_{\rm pd}\rvert}{\lvert\delta_{\rm pos}\rvert}$	$\frac{\lvert Q_{\rm pd}\rvert}{\lvert Q_f\rvert}$	$\frac{\lvert\delta_{\rm pd}\rvert}{\lvert\delta_f\rvert}$
50	1.18	0.90	1.02	0.89	1.02	0.86	0.99	0.99
100	1.33	0.85	1.07	0.84	1.05	0.79	0.98	0.99
200	1.27	0.89	1.06	0.88	1.05	0.82	0.99	0.99
300	1.28	0.89	1.06	0.88	1.05	0.82	0.99	0.99

Reductions by $\equiv_R$ and $\equiv_L$ (or $\equiv_R \circ \equiv_L$) decrease by less than 2% the size of the considered NFAs ($\mathcal{A}_{\rm pos}$, $\mathcal{A}_f$, and $\mathcal{A}_{\rm pd}$). In particular the quotient automata of $\mathcal{A}_{\rm pos}$ are less than 1% smaller than $\mathcal{A}_{\rm pd}$. In general, we can hypothesize that reductions by the coarsest invariant equivalences are not significant when REs are reduced (and/or are in snf).

8 Conclusion

We presented a set of tools within the **FAdo** system to uniformly random generate REs, to convert REs into ε-free NFAs and to simplify both REs and NFAs. These tools can be used to obtain experimental results about the relative descriptional complexity of regular language representations on the average case. Our experimental data corroborate some previous experimental and theoretical results, and suggest some new hypotheses to be theoretically proved. We highlight the two following conjectures. Reduced REs have high probability of being in snf. And the $\mathcal{A}_{\rm pd}$ obtained from REs in reduced snf seems to almost coincide with quotient automata of $\mathcal{A}_{\rm pos}$ by $\equiv_R \circ \equiv_L$.

We would like to thank the anonymous referees for their comments that helped to improve this paper.

References

[AAA+09] A. Almeida, M. Almeida, J. Alves, N. Moreira, and R. Reis. FAdo and GUItar: tools for automata manipulation and visualization. In S. Maneth, editor, *14th CIAA'09*, volume 5642 of *LNCS*, pages 65–74. Springer, 2009.

[Ant96] V. M. Antimirov. Partial derivatives of regular expressions and finite automaton constructions. *Theoret. Comput. Sci.*, 155(2):291–319, 1996.

[BK93] A. Brüggemann-Klein. Regular expressions into finite automata. *Theoret. Comput. Sci.*, 48:197–213, 1993.

[COZ07] J.-M. Champarnaud, F. Ouardi, and D. Ziadi. Normalized expressions and finite automata. *Intern. Journ. of Alg. and Comp.*, 17(1):141–154, 2007.

[CZ02] J. M. Champarnaud and D. Ziadi. Canonical derivatives, partial derivatives and finite automaton constructions. *Theoret. Comput. Sci.*, 289:137–163, 2002.

[EKSW05] K. Ellul, B. Krawetz, J. Shallit, and M. Wang. Regular expressions: New results and open problems. *J. Aut., Lang. and Combin.*, 10(4):407–437, 2005.

[FAd10] Project FAdo. FAdo: tools for formal languages manipulation. `http://www.ncc.up.pt/FAdo`, Access date:1.1.2010.

[GG09] H. Gruber and S. Gulan. Simplifying regular expressions: A quantitative perspective. Technical report, IFIG Research Report, 2009.

[Glu61] V. M. Glushkov. The abstract theory of automata. *Russian Math. Surveys*, 16:1–53, 1961.

[Gou09] H. Gouveia. Obtenção de autómatos finitos não determinísticos pequenos. Technical report, DCC-FCUP, 2009.

[HSW01] J. Hromkovic, S. Seibert, and T. Wilke. Translating regular expressions into small epsilon-free nondeterministic finite automata. *J. Comput. System Sci.*, 62(4):565–588, 2001.

[ISOY05] L. Ilie, R. Solis-Oba, and S. Yu. Reducing the size of NFAs by using equivalences and preorders. In A. Apostolico, M. Crochemore, and K. Park, editors, *16th CPM 2005*, volume 3537 of *LNCS*, pages 310–321. Springer, 2005.

[IY03a] L. Ilie and S. Yu. Follow automata. *Inf. Comput.*, 186(1):140–162, 2003.

[IY03b] L. Ilie and S. Yu. Reducing NFAs by invariant equivalences. *Theoret. Comput. Sci.*, 306(1-3):373–390, 2003.

[LS05] J. Lee and J. Shallit. Enumerating regular expressions and their languages. In M. Domaratzki, A. Okhotin, K. Salomaa, and S. Yu, editors, *9th CIAA 2004*, volume 3314 of *LNCS*, pages 2–22. Springer, 2005.

[Mai94] H. G. Mairson. Generating words in a context-free language uniformly at random. *Information Processing Letters*, 49:95–99, 1994.

[Mir66] B. G. Mirkin. An algorithm for constructing a base in a language of regular expressions. *Engineering Cybernetics*, 5:51—57, 1966.

[MR05] N. Moreira and R. Reis. Interactive manipulation of regular objects with FAdo. In *ITiCSE 2005*, pages 335–339. ACM, 2005.

[MY60] R. McNaughton and H. Yamada. Regular expressions and state graphs for automata. *IEEE Transactions on Electronic Computers*, 9:39–47, 1960.

[Nic09] C. Nicaud. On the average size of Glushkov's automata. In A. Dediu, A.-M.i Ionescu, and C. M. Vide, editors, *3rd LATA 2009*, volume 5457 of *LNCS*, pages 626–637. Springer, 2009.

[PT87] R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM J. Comput.*, 16(6):973–989, 1987.