Random Generation of Nondeterministic Finite-State Tree Automata

Thomas Hanneforth
Universität Potsdam, Department Linguistik
Karl-Liebknecht-Straße 24–25, 14476 Potsdam, Germany
thomas.hanneforth@uni-potsdam.de

Andreas Maletti* Daniel Quernheim*
Universität Stuttgart, Institut für Maschinelle Sprachverarbeitung
Pfaffenwaldring 5b, 70569 Stuttgart, Germany
{maletti,daniel}@ims.uni-stuttgart.de

Algorithms for (nondeterministic) finite-state tree automata (FTAs) are often tested on random FTAs, in which all internal transitions are equiprobable. The run-time results obtained in this manner are usually overly optimistic as most such generated random FTAs are trivial in the sense that the number of states of an equivalent minimal deterministic FTA is extremely small. It is demonstrated that nontrivial random FTAs are obtained only for a narrow band of transition probabilities. Moreover, an analytic analysis yields a formula to approximate the transition probability that yields the most complex random FTAs, which should be used in experiments.

1 Introduction

Nondeterministic finite-state tree automata (FTAs) play a major role in several areas of natural language processing. For example, the BERKELEY parser [10] uses a (weighted) FTA as do syntax-based approaches to statistical machine translation. Toolkits [9, 7] for FTAs allow users to easily run experiments. However, algorithms like determinization typically cannot be tested on real-world examples (like the FTA of the BERKELEY parser) due to their complexity. In such cases, the inputs are often random FTAs, which are typically created by fixing densities, which are the probabilities that any given potential transition (of a certain group) is indeed a transition of the generated FTA (see [12] and [8] for the generation of random finite-state string automata).

It is known [2] that for string automata most random automata are trivial in the sense that the equivalent minimal deterministic finite-state string automaton is extremely small. Here, we observe the same effect for FTAs, which means that testing algorithms on random FTAs also has to be done carefully to avoid vastly underestimating their actual run-time. To simplify such experiments and to make them more representative we provide both empirical and analytic evidence for the nontrivial (difficult) cases. In the empirical evaluation we create many random FTAs with a given number n of (useful) states using a given transition density d. We then determinize and minimize them, and we record the number of states of the resulting canonical FTA (i.e., the equivalent minimal deterministic FTA). As expected, outside a narrow density band the randomly generated FTAs yield very small canonical FTA, which means that they are in a sense trivial.[1] It can be observed that if the density is above the upper limit of the band, then the FTAs accept almost everything, whereas FTAs with densities below the lower limit accept almost nothing. This observation also justifies calling them trivial.

*Financially supported by the German Research Foundation (DFG) grant MA / 4959 / 1-1.

[1] Trivial here does not mean that the recognized tree language is uninteresting, but rather it only relates to its complexity.

S. Maneth (Ed.): Proceedings Workshop Trends in Tree Automata and Tree Transducers.
TTATT 2013, EPTCS 134, pages 11–16, 2013

In the analytic evaluation, given a number n of states, we compute densities $d(n)$, for which we expect the most difficult FTAs. Looking at the empirical results, the formula predicts the narrow density band belonging to complex random FTAs very well. Consequently, we promote experiments with random FTAs that use exactly these predicted densities in order to avoid experiments with (only) trivial FTAs. Finally, we discuss how parameter changes affect our results. For example, adding another binary input symbol does not move the interesting narrow density band, but it generally does increase the sizes of the obtained canonical FTA. Moreover, it shows that whenever we obtain large deterministic FTA after determinization (i.e., before minimization), then the corresponding canonical FTA are also large (i.e., after minimization). This demonstrates that our implementation of determinization is rather efficient.

2 Finite tree automata

The power set of a set S is $\mathscr{P}(S) = \{S' \mid S' \subseteq S\}$. The set of nonnegative integers is denoted by $\mathbb{N}$. An alphabet is simply a finite set of symbols. A ranked alphabet (Σ, rk) consists of an alphabet Σ and a mapping $\mathrm{rk}\colon \Sigma \to \mathbb{N}$, which assigns a rank to each symbol. For every $k \in \mathbb{N}$, we let $\Sigma_k = \{\sigma \in \Sigma \mid \mathrm{rk}(\sigma) = k\}$ be the set of symbols of rank k. We also write $\sigma^{(k)}$ to indicate that the symbol σ has rank $\mathrm{rk}(\sigma) = k$. To keep the presentation simple, we typically write just Σ for the ranked alphabet (Σ, rk) and assume that the ranking 'rk' is clear from the context. Moreover, we often drop obvious universal quantifications like '$k \in \mathbb{N}$' in expressions like 'for all $k \in \mathbb{N}$ and $\sigma \in \Sigma_k$'. Our trees have node labels taken from a ranked alphabet Σ and leaves can also be labeled by elements of a finite set Q. The rank of a symbol $\sigma \in \Sigma$ determines the number of direct children of all nodes labeled σ. Given a set T, we write $\Sigma(T)$ for the set $\{\sigma(t_1, \dots, t_k) \mid \sigma \in \Sigma_k, t_1, \dots, t_k \in T\}$. The set $T_\Sigma(Q)$ of Σ-trees indexed by Q is defined as the smallest set T such that $\Sigma(T) \cup Q \subseteq T$. We write T_Σ for $T_\Sigma(\emptyset)$.

Next, we recall finite-state tree automata [4, 5] and the required standard constructions. In general, finite-state tree automata (FTAs) offer an efficient representation of the regular tree languages. We distinguish a (bottom-up) deterministic variant called deterministic FTA, which will be used in our size measurements. A *finite-state tree automaton* (FTA) is a system (Q, Σ, F, P), where (i) Q is a finite set of *states*, (ii) Σ is a ranked alphabet of *input symbols*, (iii) $F \subseteq Q$ is a set of *final* states, and (iv) $P \subseteq \Sigma(Q) \times Q$ is a finite set of *transitions*. It is *deterministic* if for every $t \in \Sigma(Q)$ there exists at most one $q \in Q$ such that $(t, q) \in P$. We often write a transition $(t, q) \in \Sigma(Q) \times Q$ as $t \to q$. The size of the FTA $M = (Q, \Sigma, F, P)$ is $|M| = |Q|$. This is arguably a crude measure for the 'size', but it will mostly be used for deterministic FTAs, where it is commonly used.

Example 1. *As illustration we consider the FTA $M_{ex} = (\{0,1,2,3\}, \Sigma, \{3\}, P)$, where $\Sigma = \{\sigma^{(2)}, \alpha^{(0)}\}$ and P contains the following transitions. Clearly, this FTA is not deterministic.*

$$\alpha \to 0 \qquad \alpha \to 2 \qquad \sigma(0,0) \to 1 \qquad \sigma(1,0) \to 1 \qquad \sigma(1,2) \to 3 \qquad \sigma(1,3) \to 3$$

In the following, let $M = (Q, \Sigma, F, P)$ be an FTA. For every $\sigma \in \Sigma_k$, let $\overline{\sigma}\colon \mathscr{P}(Q)^k \to \mathscr{P}(Q)$ be such that $\overline{\sigma}(Q_1, \dots, Q_k) = \{q \in Q \mid \forall 1 \leq i \leq k, \exists q_i \in Q_i\colon \sigma(q_1, \dots, q_k) \to q \in P\}$ for all $Q_1, \dots, Q_k \subseteq Q$. Next, we define the action of the transitions on a tree $t \in T_\Sigma(Q)$. Let $P\colon T_\Sigma(Q) \to \mathscr{P}(Q)$ be such that $P(q) = \{q\}$ for every $q \in Q$ and $P(\sigma(t_1, \dots, t_k)) = \overline{\sigma}(P(t_1), \dots, P(t_k))$ for every $\sigma \in \Sigma_k$ and $t_1, \dots, t_k \in T_\Sigma$. The FTA M accepts the tree language $L(M)$, which is given by $L(M) = \{t \in T_\Sigma \mid P(t) \cap F \neq \emptyset\}$. Two FTAs M_1 and M_2 are equivalent if $L(M_1) = L(M_2)$. For example, for the FTA of Example 1 we have $P(\alpha) = \{0,2\}$ and $P(\sigma(\alpha, \alpha)) = \{1\}$. Moreover, $\sigma(\sigma(\sigma(\alpha, \alpha), \alpha), \alpha) \in L(M_{\mathrm{ex}})$. For our experiments, we generate a random FTA, then determinize it and compute the number of states of the canonical

FTA, which the equivalent minimal deterministic FTA. For an FTA $M = (Q, \Sigma, F, P)$, we construct the deterministic FTA $\mathscr{P}(M) = (\mathscr{P}(Q), \Sigma, F', P')$ such that $F' = \{Q' \subseteq Q \mid Q' \cap F \neq \emptyset\}$ and

$$P' = \{\sigma(Q_1, \dots, Q_k) \to \overline{\sigma}(Q_1, \dots, Q_k) \mid \sigma \in \Sigma_k, Q_1, \dots, Q_k \subseteq Q\} \ .$$

Theorem 2 (see [3, Theorem 1.10])**.** *$\mathscr{P}(M)$ is a deterministic FTA that is equivalent to M.*

Example 3. *For the FTA M_{ex} of Example 1 an equivalent deterministic FTA is $\mathscr{P}(M_{ex})$, which is given by $\mathscr{P}(M_{ex}) = (\mathscr{P}(Q), \Sigma, F', P')$ with $F' = \{3, 13, 23, 123\}$ and P' contains the (non-trivial) transitions*[2]

$$\alpha \to 02 \quad \sigma(02, 02) \to 1 \quad \sigma(1, 02) \to 13 \quad \sigma(13, 02) \to 13 \quad \sigma(1, 13) \to 3 \quad \sigma(1, 3) \to 3 \ .$$

A deterministic FTA is *minimal* if there is no strictly smaller equivalent deterministic FTA. For example, the deterministic FTA in Example 3 is minimal. It is known [1] that for every deterministic FTA we can compute an equivalent minimal deterministic FTA, called the *canonical* FTA.

3 Random generation of FTAs

First, we describe how we generate random FTAs.[3] We closely follow the random generation outlined in [12], augmented by density parameters d_2 and d_0, which is similar to the setup of [8]. Both methods [12, 8] are discussed in [2], where they are applied to finite-state string automata [11]. For an event E, let $\pi(E)$ be the probability of E. To keep the presentation simple, we assume that the ranked alphabet Σ of input symbols is binary (i.e., $\Sigma = \Sigma_2 \cup \Sigma_0$).[4] Note that all regular tree languages [4, 5] can be encoded using a binary ranked alphabet. In order to randomly generate an FTA $M = (Q, \Sigma, F, P)$ with $n = |Q|$ states and binary and nullary transition densities d_2 and d_0, each transition (incl. the target state) is a random variable and each state is a random variable representing whether it is final or not. More precisely, we use the following approach:

- $Q = \{1, \dots, n\}$,
- $\pi(q \in F) = \frac{1}{2}$ for all $q \in Q$ (i.e., for each state q the probability that it is final is $\frac{1}{2}$),
- $\pi(\alpha \to q \in P) = d_0$ for all nullary $\alpha \in \Sigma_0$ and $q \in Q$, and
- $\pi(\sigma(q_1, q_2) \to q \in P) = d_2$ for all binary symbols $\sigma \in \Sigma_2$ and all states $q_1, q_2, q \in Q$.

If the such created FTA is not trim[5], then we start over and generate a new FTA. Thus, all our randomly generated FTAs indeed have n useful states.

4 Analytic analysis

In this section, we present a short analytic analysis and compute densities, for which we expect the randomly generated FTAs to be non-trivial. More precisely, we estimate for which densities the determinization (and subsequent minimization) returns the largest canonical FTAs. It is known from the

[2] We abbreviate sets like $\{0, 2\}$ to just 02.

[3] These FTAs shall serve as test inputs for algorithms that operate on FTAs such as determinization, bisimulation minimization, etc. Naturally, the size of an equivalent minimal FTA would be an obvious complexity measure for them, but it is PSPACE-complete to determine it, and the size of the canonical FTA is naturally always bigger, so trivial FTAs according to our measure are also trivial under the minimal FTA size measure.

[4] Our approach can easily be adjusted to accommodate non-binary ranked alphabets. We can imagine a model in which only one density d governs all transitions, but this model requires a slightly more difficult analytic analysis.

[5] The FTA M is trim if for every $q \in Q$ there exist $t \in T_\Sigma$ and $t' \in T_\Sigma(\{q\}) \setminus T_\Sigma$ such that (i) $q \in P(t)$ and (ii) $P(t') \cap F \neq \emptyset$.

n	d_2	d_2'	conf. interval
2	.6364	.6264	[.5769,.6804]
3	.2965	.2570	[.2091,.3159]
4	.1696	.1334	[.1024,.1737]
5	.1094	.0855	[.0642,.1138]
6	.0763	.0635	[.0475,.0848]
7	.0562	.0501	[.0380,.0662]
8	.0431	.0408	[.0317,.0526]
9	.0341	.0342	[.0272,.0430]
10	.0276	.0282	[.0231,.0343]
11	.0228	.0251	[.0208,.0303]
12	.0192	.0212	[.0182,.0248]
13	.0164	.0189	[.0162,.0219]

Table 1: Expected density d_2 (see Theorem 4) and observed density d_2' (see Section 5) for the most complex FTAs in Setting (A). We also report confidence intervals for the confidence level $p > .95$.

generation of random finite-state string automata [2] that the largest deterministic automata are obtained during determinization if each state $q \in Q$ occurs with probability $\frac{1}{2}$ in the transition target of a transition, in which the source states are selected uniformly at random. This observation was empirically confirmed multiple times by independent research groups [8, 12, 2]. In addition, all transition target states are equiprobable for input states that are drawn uniformly at random. This latter observation supports the optimality claim by arguments from information theory because if all target states of a transition are equiprobable, then the entropy of the transition is maximal. While these facts support our hypothesis (and subsequent conclusions), we also rely on an empirical evaluation in Section 5 to validate it.

The determinization (see Section 2) constructs the state set $\mathscr{P}(Q)$. Let $\mathscr{P}(M) = (\mathscr{P}(Q), \Sigma, F', P')$ be the deterministic FTA given the random FTA M with $n = |Q|$.[6] According to our intuition, the probability that a state $Q' \in \mathscr{P}(Q)$ is the transition target of a given transition $\sigma(Q_1, Q_2)$, where Q_1 and Q_2 are uniformly selected at random from $\mathscr{P}(Q)$, should be 2^{-n} [i.e., $\pi(\sigma(Q_1, Q_2) \to Q' \in P') = 2^{-n}$]. Thus, in particular, each given state $q \in Q$ is in the (real) successor state Q'' of the given transition $\sigma(Q_1, Q_2)$ with probability $\frac{1}{2}$ [i.e., $\pi(q \in Q'') = \frac{1}{2}$]. Given $\sigma \in \Sigma_2$ and $q \in Q$, let $\pi_{\sigma,q} = \pi(q \in Q'')$, where Q_1 and Q_2 are uniformly selected at random from $\mathscr{P}(Q)$ and $Q'' = \overline{\sigma}(Q_1, Q_2)$. Similarly, given $\alpha \in \Sigma_0$ and $q \in Q$, let $\pi_{\alpha,q} = \pi(q \in \overline{\alpha})$. It is easily seen that $\pi_{\sigma,q} = \pi_{\sigma',q'}$ for all $\sigma, \sigma' \in \Sigma_2$ and $q, q' \in Q$ because in our generation model all σ-transitions in M with $\sigma \in \Sigma_2$ are equiprobable.[7] Thus, we simply write π_2 instead of $\pi_{\sigma,q}$. The same property holds for nullary symbols, so we henceforth write π_0 for $\pi_{\alpha,q}$. Moreover, as in the previous section, let n be the number of states of the original random FTA, and let d_2 and d_0 be the transition densities of it. For $n = 1$ the presented intuition cannot be met.

Theorem 4. *Let $n > 1$. If $d_2 = 4(1 - \sqrt[n^2]{.5})$ and $d_0 = \frac{1}{2}$, then $\pi_2 = \pi_0 = \frac{1}{2}$.*

Proof. We start with π_0. Let $\alpha \in \Sigma_0$ and $q \in Q$. Then $\pi(q \in \overline{\alpha}) = \pi(\alpha \to q \in P) = d_0 = \frac{1}{2}$ as required. For π_2 let $Q_1, Q_2 \in \mathscr{P}(Q)$ be selected uniformly at random, $\sigma \in \Sigma_2$, and $q \in Q$. Then

$$\begin{aligned}\pi(q \in \overline{\sigma}(Q_1, Q_2)) &= 1 - \pi(q \notin \overline{\sigma}(Q_1, Q_2)) \\ &= 1 - \prod_{q_1, q_2 \in Q} \Big(1 - \pi(q_1 \in Q_1) \cdot \pi(q_2 \in Q_2) \cdot \pi(\sigma(q_1, q_2) \to q \in P)\Big) \\ &= 1 - \big(1 - \frac{d_2}{4}\big)^{n^2} = 1 - \big(1 - 1 + \sqrt[n^2]{.5}\big)^{n^2} = 1 - (\sqrt[n^2]{.5})^{n^2} = \frac{1}{2} \ . \qquad \square\end{aligned}$$

Table 1 lists some values d_2 computed according to Theorem 4 for the sizes $n \in \{2, \dots, 13\}$.

[6] Note that the transitions of this DTA are random variables distributed according to the determinization construction of Section 2 applied to M.

[7] Note that the individual σ-transitions of $\mathscr{P}(M)$ are not equiprobable. For example, the transition $\sigma(\emptyset, \emptyset) \to Q$ is impossible.

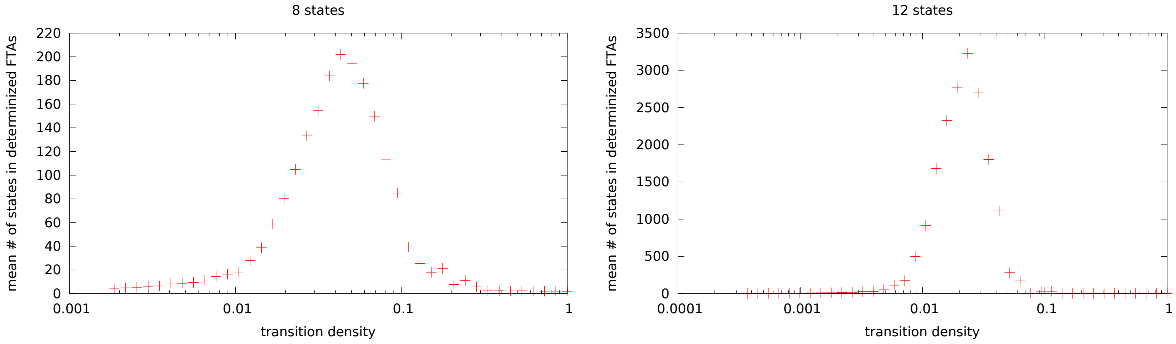

Figure 1: Graphs plotting the mean size of the determinized FTAs obtained by determinization over the density for FTAs of size 8 and 12. Minimization reduces the mean size, but does not move the peak.

5 Empirical analysis

In this section, we want to confirm that the computed densities indeed represent the most difficult instances for the random FTAs constructed in Section 3. We use two settings:

(A) $\Sigma = \{\alpha^{(0)}, \sigma^{(2)}\}$ and
(B) $\Sigma = \{\alpha^{(0)}, \sigma^{(2)}, \delta^{(2)}\}$.

For both settings (A) and (B) and varying densities $d_2 = e^{\frac{x \cdot \log D_n}{20}}$ and $d_0 = \frac{1}{2}$, where $D_n = 4(1 - \sqrt[n^2]{.5})$, for all $0 \leq x \leq 40$ and sizes $2 \leq n \leq 13$, we generated at least 40 trim FTAs. The ratio of trim FTAs for various densities and state set sizes can be found in Table 2. Generally, larger state sets and higher densities increase the chance of obtaining a trim FTA. The choice of densities we made ensures that sufficiently many data points (in equally-spaced steps on a logarithmic scale) will exist on both sides of the density that is predicted to generate the most difficult instances. We will discuss why we favored the logarithmic scale over a linear scale in the next paragraph. These FTAs were subsequently determinized, minimized, and the sizes of the canonical FTAs were recorded. These operations were performed inside our new tree automata toolkit TALIB[8].

Our experiments confirm the theoretical predictions. A peak in the mean size of the determinized FTAs can be observed where it is predicted. Exemplary graphs for setting (A) and $n \in \{8, 12\}$ are presented in Figure 1 on a logarithmic scale. Since these graphs appear to be log-normal distributions[9], we computed the mean and the variance of these log-normal distributions, interpreting the density as the random variable and the number of states as the frequency. The relevant statistics are reported in Table 1 for setting (A). All predicted densities are in the confidence interval for the confidence level $p > .95$ (that is, the predicted density is within $1.96 \cdot \sigma$ distance of the observed mean, where σ is the standard deviation).

It is worth noting that the location of the peak does not change between settings (A) and (B). This means that the size of the alphabet of binary symbols does not influence the hardness of the problem; the only difference is the size of the resulting determinized FTAs, which is generally larger in setting (B). Also, minimization does not change the location of the peak, which means that hard instances for de-

[8] Additional information about TALIB is available at `http://www.ims.uni-stuttgart.de/forschung/ressourcen/werkzeuge/talib.en.html`.

[9] A log-normal distribution is a distribution of a random variable whose logarithm is normally distributed. A log-normal distribution usually arises as the product of independent normal distributions. We leave the question of how exactly this distribution can be derived for further research.

$d_2 \backslash n$	2	4	6	7	8	9	10	11	12	13
.01				7%	11%	18%	27%	38%	50%	64%
.05			68%	82%	92%	98%	99%	100%	100%	100%
.10		54%	90%	96%	99%	99%	100%	100%	100%	100%
.25		83%	97%	98%	99%	100%	100%	100%	100%	100%
.50	47%	88%	96%	98%	100%	100%	100%	100%	100%	100%

Table 2: Ratio of trim FTAs in a set of randomly generated FTAs parameterized by density d_2 (rows) and number n of states (columns). Blank entries indicate that no such experiment was performed.

terminization are also hard instances for minimization. In addition, we also performed experiments that confirmed the similar result of [2] for finite-state string automata using the string automata toolkit FSM<2.0> of [6].

References

[1] Walter S. Brainerd (1968): *The Minimalization of Tree Automata.* *Inform. and Control* 13(5), pp. 484–491, doi:10.1016/S0019-9958(68)90917-0.

[2] Jean-Marc Champarnaud, Georges Hansel, Thomas Paranthoën & Djelloul Ziadi (2004): *Random Generation Models for NFAs.* *J. Autom. Lang. Combin.* 9(2/3), pp. 203–216.

[3] John Doner (1970): *Tree Acceptors and Some of Their Applications.* *J. Comput. System Sci.* 4(5), pp. 406–451, doi:10.1016/S0022-0000(70)80041-1.

[4] Ferenc Gécseg & Magnus Steinby (1984): *Tree Automata.* Akadémiai Kiadó, Budapest.

[5] Ferenc Gécseg & Magnus Steinby (1997): *Tree Languages.* In Grzegorz Rozenberg & Arto Salomaa, editors: *Beyond Words*, chapter 1, *Handbook of Formal Languages* 3, Springer, pp. 1–68, doi:10.1007/978-3-642-59126-6_1.

[6] Thomas Hanneforth (2010): *fsm2 — A Scripting Language Interpreter for Manipulating Weighted Finite-State Automata.* In: *Proc. FSMNLP*, LNCS 6062, Springer, pp. 13–30, doi:10.1007/978-3-642-14684-8_3.

[7] Ondrej Lengál, Jirí Simácek & Tomás Vojnar (2012): *VATA: A Library for Efficient Manipulation of Non-deterministic Tree Automata.* In: *Proc. TACAS*, *LNCS* 7214, Springer, pp. 79–94, doi:10.1007/978-3-642-28756-5_7.

[8] Ted Leslie (1995): *Efficient Approaches to Subset Construction.* Technical Report, University of Waterloo, Canada.

[9] Jonathan May & Kevin Knight (2006): *Tiburon: A Weighted Tree Automata Toolkit.* In: *Proc. CIAA*, LNCS 4094, Springer, pp. 102–113, doi:10.1007/11812128_11.

[10] Slav Petrov, Leon Barrett, Romain Thibaux & Dan Klein (2006): *Learning Accurate, Compact, and Interpretable Tree Annotation.* In: *Proc. COLING-ACL*, Association for Computational Linguistics, pp. 433–440, doi:10.3115/1220175.1220230.

[11] Sheng Yu (1997): *Regular Languages.* In Grzegorz Rozenberg & Arto Salomaa, editors: *Word, Language, Grammar*, chapter 2, *Handbook of Formal Languages* 1, Springer, pp. 41–110, doi:10.1007/978-3-642-59136-5_2.

[12] Lynette van Zijl (1997): *Generalized Nondeterminism and the Succinct Representation of Regular Languages.* Ph.D. thesis, Stellenbosch University, South Africa.