Random Generation of Nondeterministic Tree Automata

Thomas Hanneforth[1], Andreas Maletti[2,*] and Daniel Quernheim[2,*]

[1] Universität Potsdam, Department Linguistik
Karl-Liebknecht-Straße 24–25, 14476 Potsdam, Germany
`thomas.hanneforth@uni-potsdam.de`

[2] Universität Stuttgart, Institut für Maschinelle Sprachverarbeitung
Pfaffenwaldring 5b, 70569 Stuttgart, Germany
`{maletti,daniel}@ims.uni-stuttgart.de`

Abstract

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

1 Introduction

Nondeterministic finite-state tree automata (NTA) play a major role in several areas of natural language processing. For example, the grammars of the Berkeley parser [9] are trivially (weighted) NTA. Moreover, syntax-based approaches to statistical machine translation commonly use weighted NTA. Toolkits for NTA exist by now [8] and allow users to easily run experiments. However, algorithms like determinization typically cannot be tested on real-word examples (like the Berkeley parser grammars) due to their complexity. In such cases, the inputs are often random NTA, 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 NTA [see [10] and [7] 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 NTA, which means that testing algorithms on random NTA 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 analytical evidence for the nontrivial (difficult) cases.

In the empirical evaluation we create many random NTA with a given number n of (useful) states using a given transition density d. We then determinize and minimize them, and record the number of states of the resulting minimal deterministic finite-state tree automaton (DTA). As expected, outside a narrow density band the randomly generated NTA yield very small equivalent minimal DTA, which means that they are in a sense trivial.[1] It can be observed that

[*]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.

if the density is above the upper limit of the band, then the NTA accept almost everything, whereas NTA with densities below the lower limit accept almost nothing.[2]

In the analytical evaluation, given a number n of states, we compute densities $d_2(n)$ and $d_0(n)$, for which we expect the most difficult NTA. Looking at the empirical results, the formula predicts the narrow density band belonging to complex random NTA very well. Consequently, we promote experiments with random NTA that use exactly these predicted densities in order to avoid experiments with (only) trivial NTA.

Finally, we discuss how parameter changes affect our results. For example, the addition of another binary input symbol does not move the interesting narrow density band, but it generally does increase the sizes of the obtained minimal DTA. Moreover, it shows that whenever we obtain large DTA before minimization, then those DTA remain large even after minimization. This demonstrates that our implementation of determinization is rather efficient.

2 Finite tree automata

The power set of a set S is $\mathcal{P}(S) = \{S' \mid S' \subseteq S\}$. 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 $\sigma \in \Sigma_k$. Our trees have node labels taken from a ranked alphabet Σ. 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_Σ of Σ-trees is defined as the smallest set T such that $\Sigma(T) \subseteq T$.

Next, we recall finite tree automata [4, 5] and the required standard constructions. In general, finite tree automata (NTA) offer an efficient representation of the regular tree languages. We distinguish a (bottom-up) deterministic variant called deterministic tree automaton (DTA), which will be used in our size measurements. A *finite tree automaton* (NTA) 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 NTA $M = (Q, \Sigma, F, P)$ is $|M| = |Q|$. This is arguably a crude measure for the 'size', but it will mostly be used for DTA, where it is commonly used.

Example 1. *As illustration we consider the NTA* $M_{ex} = (\{0, 1, 2, 3\}, \Sigma, \{3\}, P)$*, where* $\Sigma = \{\sigma^{(2)}, \alpha^{(0)}\}$ *and* P *contains the transitions*

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

This NTA is not deterministic.

In the following, let $M = (Q, \Sigma, F, P)$ be an NTA. For every $\sigma \in \Sigma_k$, let $\overline{\sigma}\colon \mathcal{P}(Q)^k \to \mathcal{P}(Q)$ be the mapping defined for all $Q_1, \dots, Q_k \subseteq Q$ by

$$\overline{\sigma}(Q_1, \dots, Q_k) = \{q \mid \forall 1 \leq i \leq k, \exists q_i \in Q_i \colon \sigma(q_1, \dots, q_k) \to q \in P\} \ .$$

[2]This observation also justifies calling them trivial.

Next, we define the action of the transitions on a tree $t \in T_\Sigma$. Let $P \colon T_\Sigma \to \mathcal{P}(Q)$ be such that $P(\sigma(t_1, \dots, t_k)) = \overline{\sigma}(P(t_1), \dots, P(t_k))$ for every $\sigma \in \Sigma_k$ and $t_1, \dots, t_k \in T_\Sigma$. The NTA 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 NTA M_1 and M_2 are equivalent if $L(M_1) = L(M_2)$.

Example 2. *For the NTA of Example 1 we have $P(\alpha) = \{0, 2\}$ and $P(\sigma(\alpha, \alpha)) = \{1\}$. Moreover, $\sigma(\sigma(\sigma(\alpha, \alpha), \alpha), \alpha) \in L(M_{ex})$.*

For our experiments, we generate a random NTA, then determinize it and compute the number of states of an equivalent minimal dta. Let us quickly recall the relevant constructions.

Definition 3. *For an NTA $M = (Q, \Sigma, F, P)$, we construct the DTA $\mathcal{P}(M) = (\mathcal{P}(Q), \Sigma, F', P')$ such that $F' = \{Q' \subseteq Q \mid Q' \cap F \neq \emptyset\}$ and*

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

Theorem 4 (see [3, Theorem 1.10])**.** *$\mathcal{P}(M)$ is a DTA that is equivalent to M.*

Example 5. *For the NTA of Example 1 an equivalent DTA is $\mathcal{P}(M_{ex}) = (\mathcal{P}(Q), \Sigma, F', P')$,[3] where $F' = \{3, 13, 23, 123\}$ and the following (non-trivial) transitions in P':*

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

A DTA is minimal if there is no strictly smaller equivalent DTA. The DTA in Example 5 is minimal. It is known [1] that for every DTA we can compute an equivalent minimal DTA.

3 Random NTA generation

First, we describe how we generate random NTA.[4] We closely follow the random generation outlined in [10], augmented by density parameters d_2 and d_0, which is similar to the setup of [7]. Both methods [10, 7] 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., $\Sigma = \Sigma_2 \cup \Sigma_0$).[5] Note that all regular tree languages [4, 5] can be encoded using a binary ranked alphabet. In order to randomly generate an NTA $M = (Q, \Sigma, F, P)$ 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 NTA is not trim[6], then we start over and generate a new NTA. Thus, all our randomly generated NTA indeed have n useful states.

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

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

[5]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 analytical analysis.

[6]The NTA M is trim if for every $q \in Q$ there exists $t \in T_\Sigma$ such that $q \in P(t)$.

4 Analytical analysis

In this section, we present a short analytical analysis and compute densities, for which we expect the randomly generated NTA to be non-trivial. More precisely, we estimate for which densities the determinization (and subsequent minimization) returns the largest minimal DTA. It is known from the 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 [7, 10, 2]. In addition, all transition target states are equiprobable for input states that are drawn uniformly at random. This latter observation supports the maximality 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 our findings.

As demonstrated in Section 2 the determinization constructs the state set $\mathcal{P}(Q)$. Let $M' = (\mathcal{P}(Q), \Sigma, F', P')$ be the constructed DTA given the random NTA M with $n = |Q|$.[7] According to our intuition, the probability that a state $Q' \in \mathcal{P}(Q)$ 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 $\mathcal{P}(Q)$, 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 $\mathcal{P}(Q)$ 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.[8] 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 NTA, and let d_2 and d_0 be the transition densities of it.

Theorem 6. *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 $Q_1, Q_2 \in \mathcal{P}(Q)$ 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 6 for the sizes $n \in \{2, \dots, 13\}$.

5 Empirical analysis

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

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

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

n	d_2	d'_2	CI
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 6) and observed density d'_2 (see Section 5) for the most complex NTA in Setting (A). We also report confidence intervals for the confidence level $p > .95$.

	2	4	6	7	8	9	10	11	12	13
.01	n/a	n/a	n/a	6.9 %	11.1 %	17.7 %	26.8 %	38.0 %	50.0 %	64.3 %
.05	n/a	n/a	67.6 %	81.5 %	92.2 %	97.7 %	99.1 %	99.5 %	100.0 %	100.0 %
.1	n/a	54.1 %	90.1 %	96.0 %	98.6 %	98.6 %	99.7 %	99.9 %	99.8 %	100.0 %
.25	n/a	82.6 %	96.5 %	97.9 %	99.0 %	99.6 %	99.6 %	100.0 %	100.0 %	100.0 %
.5	46.9 %	88.0 %	96.3 %	97.7 %	99.5 %	99.6 %	99.8 %	99.7 %	99.9 %	100.0 %

Table 2: Ratio of trim NTA in a set of randomly generated NTA. Rows indicate various densities, columns indicate the number of states. Where no data is available, "n/a" is given.

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

For both settings (A) and (B) and varying densities $d_2 = e^{\frac{x \log d_n}{20}}$ 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 NTA. The ratio of trim NTA 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 NTA. 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 NTA were subsequently determinized, minimized, and the size of the DTA obtained by determinization as well as the size of the minimal equivalent DTA were recorded. These operations were performed inside our new tree automata toolkit TALib[9].

Our experiments confirm the theoretical predictions. A peak in the mean size of the DTA obtained by determinization 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[10], 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). We found all predicted densities to be in the confidence interval for the confidence level $p > .95$ (that is, the predicted density is within 1.96σ 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

[9]For more information about TALib, visit `http://www.ims.uni-stuttgart.de/forschung/ressourcen/werkzeuge/talib.en.html`.

[10]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.

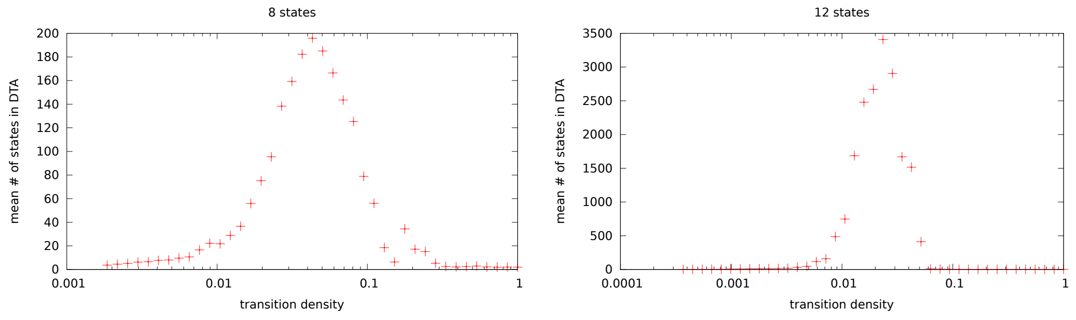

Figure 1: Graphs plotting the mean size of the equivalent DTA obtained by determinization over the density for NTA of size 8 and 12.

the problem; the only difference is the size of the resulting DTA, which is generally larger in setting (B). Also, minimization does not change the location of the peak, which means that hard instances for determinization are also hard instances for minimization. In addition, we also performed experiments that confirmed the similar result by [2] for finite-state string automata using the string automata toolkit FSM<2.0> of [6].

References

[1] Walter S. Brainerd. The minimalization of tree automata. *Inform. and Control*, 13(5):484–491, 1968.

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

[3] John Doner. Tree acceptors and some of their applications. *J. Comput. System Sci.*, 4(5):406–451, 1970.

[4] Ferenc Gécseg and Magnus Steinby. *Tree Automata.* Akadémiai Kiadó, Budapest, 1984.

[5] Ferenc Gécseg and Magnus Steinby. Tree languages. In *Handbook of Formal Languages*, volume 3, chapter 1, pages 1–68. Springer, 1997.

[6] Thomas Hanneforth. *fsm2* - a scripting language interpreter for manipulating weighted finite-state automata. In *Proc. FSMNLP*, volume 6062 of LNCS, pages 13–30. Springer, 2010.

[7] Ted Leslie. Efficient approaches to subset construction. Technical report, University of Waterloo, Canada, 1995.

[8] Jonathan May and Kevin Knight. Tiburon: A weighted tree automata toolkit. In *Proc. CIAA*, volume 4094 of LNCS, pages 102–113. Springer, 2006.

[9] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proc. COLING-ACL*, pages 433–440. Association for Computational Linguistics, 2006.

[10] Lynette van Zijl. *Generalized Nondeterminism and the Succinct Representation of Regular Languages.* PhD thesis, Stellenbosch University, South Africa, 1997.

[11] Sheng Yu. Regular languages. In *Handbook of Formal Languages*, volume 1, chapter 2, pages 41–110. Springer, 1997.