Efficiently Computing the Density of Regular Languages

Manuel Bodirsky[1], Tobias Gärtner[2], Timo von Oertzen[2], and Jan Schwinghammer[3]

[1] Humboldt-Universität zu Berlin, Germany
[2] Universität des Saarlandes, Germany
[3] University of Sussex, UK

Abstract. A regular language L is called *dense* if the fraction f_m of words of length m over some fixed signature that are contained in L tends to one if m tends to infinity. We present an algorithm that computes the number of accumulation points of (f_m) in polynomial time, if the regular language L is given by a finite deterministic automaton, and can then also efficiently check whether L is dense. Deciding whether the least accumulation point of (f_m) is greater than a given rational number, however, is coNP-complete. If the regular language is given by a *non-deterministic* automaton, checking whether L is dense becomes PSPACE-hard. We will formulate these problems as convergence problems of partially observable Markov chains, and reduce them to combinatorial problems for periodic sequences of rational numbers.

1 Introduction

In computational logics, the complexity of *almost-sure validity* became a fundamental question to logical formalisms, besides e.g. the complexity of membership test and validity. Grandjean [9] showed that almost-sure validity for first-order logic in the finite is PSPACE-complete, whereas validity in the finite is undecidable, by Trakhtenbrot's theorem.

If we are considering formal languages, the corresponding concept to almost-sure validity is the limit behavior of the density of a language. The density of a language L over the alphabet Σ is the sequence $(f_m)_{m=0}^{\infty}$ of the fractions of words of length m in the language, $f_m =_{\mathsf{df}} \frac{|L \cap \Sigma^m|}{|\Sigma|^m}$. The density of regular languages has already been studied in [2], and the methodology to analyze it using formal power series is standard by now [11]. It is known that $(f_m)_m$ has finitely many rational accumulation points [2]. However, to the best of our knowledge the algorithmic complexity of e.g. computing

[1] Supported by the Deutsche Forschungsgemeinschaft (DFG) within the European graduate program 'Combinatorics, Geometry, and Computation' (No. GRK 588/2).

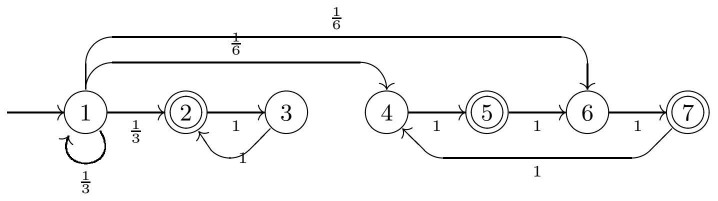

Fig. 1. Periodic and reducible Markov chain where the probability that the system is in the set $\{2, 5, 7\}$ converges to $\frac{1}{2}$.

the number of accumulation points of (f_m) has not yet been discussed. We show that the computation of $\liminf f_m$ is coNP-hard, whereas $\lim f_m$ can be computed in time $O(n^3)$, if n is the size of the deterministic automaton accepting L. If the language is given by a nondeterministic automaton, the problem to decide whether the accepted language is dense becomes PSPACE-hard.

Partially observable Markov chains. The density problem for a deterministic automaton can be translated into a convergence problem for Markov chains. We view f_m as the probability that a word of length m, chosen uniformly at random, leads to an accepting state in the given finite deterministic automaton. The automaton can then be considered as the state space of a finite Markov chain, having transitions with probability $\frac{1}{|\Sigma|}$ for every labeled edge in the finite deterministic automaton. The accepting states of the automaton are the so-called set of observable states in the Markov chain. We are interested in the probability that the system is in the far future in this *observable* set of states.

The specification and verification of long-run average properties of probabilistic systems was recently studied by de Alfaro [6]. De Alfaro also presents efficient algorithms for model checking these long-time average properties using stable-state distributions of Markov chains. However, in general the Markov chain for the automaton is not *aperiodic*, and we are not interested in the *average* behaviour, but rather in the probability of a property of the system at some specific time point in the far future. There might be a limit probability, even though there is no stable-state distribution (see Fig. 1).

We show that computing the minimal or maximal accumulation point of (f_m) is coNP-complete. This means that we cannot expect to find an efficient algorithm that computes the minimum probability that a

$$\begin{array}{l}
\;\;1\,1\,3\,0\,2\,2\,\ldots \\
+\,2\,4\,4\,6\,6\,3\,3\,5\,5\,7\,\ldots \\
+\,6\,4\,2\,3\,1\,4\,5\,3\,1\,2\,5\,3\,4\,2\,0\,\ldots \\
\hline
=9\,9\,9\,9\,9\,9\,9\,9\,9\,9\,9\,9\,9\,9\,9\,\ldots
\end{array}$$

Fig. 2. A visualization of a sum of periodic sequences with period one.

system has a certain property after a long run. However, it is possible to compute all the accumulation points in time polynomial in their number. In particular, if a property has a limit probability, i.e. if there is only one accumulation point, we will present an efficient algorithm to determine its value. Another problem will be the computation of the *number* of accumulation points.

Periodic Sequences of Rational Numbers. We will reduce the probabilistic problems above to equivalent combinatorial problems for periodic sequences. A *periodic sequence* over some field X is an infinite sequence $(\alpha[m])_{m=0}^{\infty}$ of elements in X such that there is an integer $p > 0$ so that $\alpha[m] = \alpha[m + p]$ for all $m \geq 0$. The least such integer is called the *period* $|\alpha|$ of α. If we add two sequences α and β componentwise, the result is obviously again a periodic sequence, and the period is at most $\mathsf{lcm}(|\alpha|, |\beta|)$. But sometimes a set of periodic sequences adds up to a sequence with a shorter period. Consider for example the sequences in Figure 2. The largest possible length of the sum of the sequences is $\mathsf{lcm}(6, 10, 15) = 30$, but in fact their sum has period one. We will investigate how to compute the sum of a set of periodic sequences without evaluating a possibly exponential number of entries in the sum.

2 Preliminaries

The long run behavior of the density of a deterministic finite automaton can be seen as a probabilistic process. If the regular language is given by a nondeterministic automaton, we first have to determinize the automaton, which might lead to an exponential blow-up of the size of the automaton. In fact, in this case the problem whether the language has a limit density becomes hard for PSPACE.

Proposition 1. *The problem to decide for a given nondeterministic finite automaton whether it accepts a dense language is PSPACE-hard.*

Proof. (sketch) We can adapt the classical proof showing that the non-universality problem for nondeterministic finite automata is PSPACE-hard [1]: Let M be a Turing Machine M accepting a language in polynomial space and let x an input. We can encode machine configurations (i.e., tape contents, state and head position) as words w, and computations of M as words $\#w_1\# \cdots \#w_k\#$. We can construct a (nondeterministic) automaton of size polynomial in M and x that rejects exactly the words $\#w_1\# \cdots \#w_k\#v$ where $w_1, \ldots, w_k$ represents an accepting computation of M on x and v is any word.

Clearly if there is such an accepting computation of M on x and n is the length of its representation, then the density of the language accepted by the automaton is at most $1 - 1/n < 1$. Conversely, this language is dense (in fact, universal) if x is not accepted by M. $\square$

To deal with the deterministic case, we recall in this section some notions common in the Markov chain literature.

Definition 1. *A* partially-observable Markov chain (POM) *can be described by a tuple* (V, A, s_0, O) *consisting of a finite set* V *of states and a function* $A : V^2 \to [0, 1]$ *specifying* transition probabilities[1] *, i.e. we have that* $\sum_{u \in V} A(u, v) = 1$ *for all* $v \in V$*. The* $|V|$*-dimensional vector* s_0 *is called the* initial distribution*. The set* $O \subseteq V$ *denotes the set of* observable states*.*

If we identify V with $\{1, \ldots, |V|\}$, the transition function A can be seen as a $|V| \times |V|$-matrix of rational numbers. This matrix $A = (a_{ij})_{i,j \in V}$ determines a directed weighted quasi graph, the *transition graph*, where there is an edge from v to u if $a_{uv} \neq 0$. We will freely use graph theoretic notions, and call a POM *strongly connected* (or *irreducible*), if its transition graph is. Strongly connected components of the transition graph with no outgoing edges are called *terminal components*. For simplicity we will identify a POM with its transition matrix or its transition graph, if initial distribution and labeling are clear from the context.

The *periodicity* of a strongly connected component in a POM is the greatest common divisor of the length of all the cycles in the underlying transition graph. The periodicity of a POM is the least common multiple

[1] Note that we assume that the transition probabilities are real numbers. When analyzing the running time of algorithms dealing with POMs we will only count the number of additions and multiplications of field elements that we have to perform. For the application to densities of regular languages it suffices to represent the probabilities by rational numbers, and thus we will separately mention how to deal with rational numbers.

of the periodicities of its terminal components. A POM is called *aperiodic* if its periodicity is one. An aperiodic and irreducible POM is called *ergodic*. We can draw POMs as graphs like in Figure 1 on page 2, where we can see a transition graph of periodicity four.

A *distribution* s is a $|V|$-dimensional vector of numbers from $[0,1]$. We denote the i-th component of this vector by $(s)_i$. A *run* of a POM is an (infinite) sequence $(s_m)_{m=0}^{\infty}$ of distributions, where s_0 is the initial distribution, and s_i is defined to be As_{i-1}, for $i \geq 1$. A *stable-state* distribution s is a distribution such that $As = s$.

For POMs we are not interested in stable-state distributions, but in the long-run behaviour of the sequence of probabilities $(f_m)_{m=0}^{\infty}$ that the system is in the set of observable states, where $f_m := \sum_{v \in O}(s_m)_v$. These are the problems for POMs we are investigating:

1. Check whether (f_m) converges.
2. Determine the *minimal* accumulation point of (f_m).
3. Determine the *number* of accumulation points of (f_m).
4. Determine the accumulation points of (f_m).

Convergence of aperiodic and irreducible Markov chains reduces to finding a stable state distribution of the Markov chain (see e.g. [3]):

Theorem 1 (Basic Limit Theorem). *Let A be an aperiodic and irreducible POM. Then $\lim_{m\to\infty} A^m s$ exists for all initial distributions s, and is independent of s.*

Moreover, we can efficiently find this limit distribution by finding the eigenvector to the eigenvalue 1 of A, i.e. solving a linear equation system. Since the POMs considered in this paper are in general neither aperiodic nor strongly connected, we cannot apply this theorem directly.

3 Reducing the problem

In this section show how to reduce the convergence problems mentioned in Section 2 to combinatorial problems of periodic sequences. The facts of this section are all essentially known [7], but we state them to emphasize their algorithmic aspects.

Let (V, A, s, O) be a POM, and suppose we are interested in the sequence of probabilities f_m that the system at time point m is in the set of observed states O. Only states in terminal components can contribute to the value of the accumulation points of f_m (see [7]), since the probability that the POM is in any other state converges to zero.

```
COMPUTE-LIMIT$(B, s)$
1  Compute terminal components $B_1, \ldots, B_q$.
2  for $v : v \notin B_1 \cup \cdots \cup B_q$
3  do reduce self loops at $v$
4       reduce edges to $v$.
5  for $i = 0, \ldots, q$
6  do $\lambda_i \leftarrow \sum_{v \in V,\, u \in B_i} (s)_v\ A_{vu}$
7       $b_i \leftarrow$ Eigenvector to eigenvalue 1 of
8         the transition matrix of $B_i$.
9  return the vector $\sum_{i=1}^{q} \lambda_i \bar{b}_i$ .
```

Fig. 3. Computing the density of an aperiodic POM. This procedure is used by the algorithm in Figure 4. The sub-procedures *reduce solf loops* and *reduce edges*, and correctness proofs can be found at [4].

```
COMPUTE-PERIOD$(A, s)$
1  Compute subchains $A_1, \ldots, A_p$
2   induced by the terminal components
3   of periodicity $l_1, \ldots, l_p$.
4  for $i = 0, \ldots, p$;   $j = 0, \ldots, l_i - 1$
5  do $\alpha_i[j] \leftarrow \sum_{v \in O}$(COMPUTE-LIMIT$(A^{l_i}, A^j s))_v$.
6  return $\alpha_1, \ldots, \alpha_p$.
```

Fig. 4. The reduction of the convergence problems to period problems, calling the procedure COMPUTE-LIMIT of Figure 3. Periodicities of directed graphs are easy to compute (see, e.g., [10]).

The main idea is to analyze each terminal component separately, computing the periodic contribution of every terminal component to the probabilities f_m. To this end we introduce the notion of a *subchain* of a POM: Given a set of states S, we replace all the outgoing edges of states that do not have a path into S by a self-loop.

Definition 2. *Let (V, A, s, O) be a POM, and $S \subseteq V$ be a set of states. Then we will define the* subchain *(V, B, s, O) of A induced by S. The transition function of B is defined for all $u, v \in V$ by*

$$B(u,v) := \begin{cases} A(u,v) & \textit{if there is a path in } A \textit{ from } u \textit{ into } S \\ 1 & \textit{if } u = v,\ \textit{and there is no path in } A \textit{ from } u \textit{ into } S \\ 0 & \textit{otherwise} \end{cases}$$

Obviously, an induced subchain is a POMas well. Let $l_1, \ldots, l_p$ be the periodicity of the subchains $A_1, \ldots, A_p$ induced by the terminal components. Thus the periodicity of the POM is $l := \mathsf{lcm}(l_1, \ldots, l_p)$.

Now for each of these subchains A_i and for $0 \leq j < l_i$, we define

$$\alpha_i[j] := \lim_{m\to\infty} \sum_{v\in O} (A_i^{ml_i+j} s)_v \tag{1}$$

As we will see in the next proposition, these limits exist and can be computed. The proposition states that the global accumulation points can be computed using the periodic contributions of the subchains induced by the terminal components:

Proposition 2. *Let A be a POM, and $A_1, \ldots, A_p$ the subchains induced by the terminal components $S_1, \ldots, S_p$. Let l_i denote the periodicity of A_i, and $l := \mathsf{lcm}(l_1, \ldots, l_p)$ the periodicity of A. Then for every $v \in S_i$ and every initial distribution s the following limits exists and can be computed, and we have:*

$$\lim_{m\to\infty} (A^{lm} s)_v = \lim_{m\to\infty} (A_i^{l_i m} s)_v \tag{2}$$

In particular, $\lim_{m\to\infty} f_{ml+j} = \sum_{1\leq i\leq p} \alpha_i[j]$. A proof and an algorithm for computing the α_i can be found in the full version of the paper at [4].

4 Periodic Sequences of Field Elements

In the previous sections we saw how to compute certain characteristic periodic sequences of rational numbers that describe the long run behaviour of a POM with respect to the sequence of probabilities (f_m) that the system is in an observed state. If we want to know whether this probability converges, it suffices to check whether all accumulation points are equal. In this section we present a polynomial time algorithm that avoids to check in a brute-force way exponentially many different entries in the periodic sequence of the sum.

A sequence $(\alpha[i])_{i=0}^{\infty}$ of elements over some set X is *periodic* if there is an integer $p > 0$ so that $\alpha[m] = \alpha[m+p]$ for all $m \geq 0$. The least such integer is called the *period* $|\alpha|$ of α. Here we are interested in algorithmic problems for periodic sequences over real (or rational) numbers, and thus we assume that X is a field. In Section 2 we asked several questions concerning the convergence of POMs. By the reduction of the previous section they correspond to the following problems for periodic sequences of integers $\alpha_1, \ldots, \alpha_k$, where a periodic sequence α is given by a finite sequence $\alpha[0], \ldots, \alpha[|\alpha|-1]$ of elements in X:
Let β be the sequence defined by $\beta[j] := \sum_{i=1}^{k} \alpha_i[j]$;

1. Check whether $|\beta| = 1$.
2. Determine the minimal element $\mathsf{min}\{\beta[i] \mid 0 \leq i < |\beta|\}$ of the periodic sequence β.
3. Determine the length $|\beta|$ of the sequence β.
4. Determine the entries of the periodic sequence β.

These problems are in fact polynomial time equivalent to the corresponding problems for POMs: Assume we are given a set of periodic integer sequences. It is then easy to specify a POM and an observation set such that the respective convergence problem leads to the corresponding period sum problem.

For Problem 4, we have to measure the complexity of an algorithm in both n as above and $m := |\beta|$, because in this case the size of the output β itself might be exponential. The second problem turns out to be hard:

Proposition 3. *The problem to determine whether the minimal element in the sum of given periodic sequences is greater or equal than a given value k is coNP-complete.*

This can be proven by reduction of the complement of the NP-complete problem *simultaneous inequalities* [8], which stays hard even if the numbers of the instance are represented in unary (by inspection of the NP-hardness proof given in [12]). A proof can be found in the full version of the paper available at [4]. In the next section we will show that there is an efficient algorithm for Problem 1, 3 and 4.

5 An Efficient Algorithm for Periods over the Rationals

The main idea of the algorithms for the period sum problems presented here is to represent a periodic sequence α as the power series $p_\alpha(X)$ defined by $\sum_{i=1}^{\infty} \alpha[i-1]X^{-i}$. Let $l := |\alpha|$; it is easy to verify the following closed representation of this power series:

$$p_\alpha(X) = \frac{\sum_{i=0}^{l-1} \alpha[i]X^{l-i-1}}{X^l - 1}.$$

Given any fraction of polynomials such that the denominator divides $X^l - 1$, this fraction can be expanded to such a representation of a periodic sequence. The sequence can then be determined by dividing the numerator by the denominator with a polynomial division; this can easily be verified for a denominator $X^l - 1$ and must therefore hold for any divisor of $X^l - 1$, because the result of a polynomial division is invariant under expansion and cancelation of the fraction.

For the period problems we are given the sequences $\alpha_1, \dots, \alpha_k$, and want to analyze their sum. Adding up the fractions $p_{\alpha_1}(X), \dots, p_{\alpha_k}(X)$ yields a fraction representation $\frac{u}{v}$ of the sum of the sequences. We would like to compute the potentially exponential period of $\frac{u}{v}$ without actually computing the entries of the periodic sequence. For the denominator v, we first compute the least common multiple of the denominators of all summands; since these are of the form $X^l - 1$, their zeroes are exactly all l-th roots of unity. We can therefore represent v by the list of its roots of unity. The entries of the list are stored as fractions $\frac{p_j}{q_j}$ such that $z_j = \exp(\frac{p_j}{q_j} 2\pi i)$ is the j-th root of unity in the list. For each z_j in the list, we test whether u also evaluates to 0 at z_j. If so, z_j can be canceled, and we eliminate it from the list.

We would like to compute the period of the resulting representation, i.e. the minimal m such that the fraction has denominator $X^m - 1$. Since every remaining z_j in the list is an l-th root for every multiple l of q_j, it suffices to find the minimal m that is a multiple of every q_j. Thus the period m is the smallest common multiple of the q_j.

If we are given periodic sequences of rational numbers, testing whether the numerator is zero at a root of unity can be done numerically. We first calculate the greatest common divisor of the numerator and the denominator, which is guaranteed to have only roots of unity as zeros. The minimal distance between two roots of unity is 2π times the distance of their representing fraction, which is limited by the inverse of the input size, and we can therefore test whether the polynomial contains a certain root of unity with a linear number of bits of precision. To approximate the values at the roots of unity up to n bits we need $O(n^2 \log n)$ time.

To actually compute the entries of the sequence sum (Problem 4) we perform the division of $\frac{u}{v}$ step by step, and stop after m steps.

Proposition 4. *Let $n := |\alpha_1| + \dots + |\alpha_k|$ be the size of a set of periodic integer sequences, and $m := |\alpha_1 + \dots + \alpha_k|$ the length of their sum. Then the problem to calculate m is in $O(n^2 \log n \log\log n)$. Computing the entries of the sum takes $O(n^2 \log n \log\log n + m)$ operations. In particular, we can check in $O(n^2 \log n \log\log n)$ whether the sum has period one.*

Proof. The dominating step with respect to the input size n is the reduction of the at most n fractions to higher terms before adding them: Assuming an $n \log n \log\log n$ multiplication algorithm (see e.g. [5]), the algorithm runs within $O(n^2 \log n \log\log n)$. For large m, the performance of the division is the bottleneck, requiring m operations. $\square$

6 Conclusion

We reduced the problem to determine the limit behaviour of regular languages to convergence problems for partially observable Markov chains. In this more general setting we reduced the problem to combinatorial period problems over fields that can be solved efficiently using power series representations. It is possible to efficiently compute the potentially exponential number of accumulation points of the density of a regular language given by a deterministic automaton. Moreover, we presented an algorithm that computes the accumulation points in time polynomial in the input and output. The overall running time of the algorithms for the tractable cases is dominated by the reduction to period problems over rational numbers, which involves the solution of a linear equation system.

If the language is given by a nondeterministic algorithm, we proved that the density problem is PSPACE-hard; we do not know whether it is PSPACE-complete. We would also like to know the computational complexity of checking whether a context free language, given by a generating grammar, is dense, i.e., its density converges to one.

References

1. A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and analysis of algorithms.* Addison-Wesley, 1974.
2. J. Berstel. Sur la densité asymptotique de langages formels. *ICALP*, pages 345–358, 1972.
3. R. N. Bhattacharya and E. C. Waymire. *Stochastic processes with applications.* Wiley Series in Probability and Mathematical Statistics, New York, 1990.
4. M. Bodirsky, T. Gärtner, T. von Oertzen, and J. Schwinghammer. Efficiently computing the density of regular languages. Full version, available under `http://www.informatik.hu-berlin.de/~bodirsky/publications`.
5. P. Bürgisser, M. Clausen, and M. Shokrollahi. *Algebraic Complexity Theory.* Springer Verlag, 1997.
6. L. de Alfaro. How to specify and verify the long-run average behavior of probabilistic systems. In *Proc. 13th IEEE Symp. on Logic in Computer Science*, IEEE Computer Society Press, 1998.
7. F. R. Gantmacher. *The Theory of Matrices.* Chelsea Pub. Co., 1977.
8. M. Garey and D. Johnson. *A Guide to NP-completeness.* CSLI Press, 1978.
9. E. Grandjean. Complexity of the first-order theory of almost all finite structures. *Information and Control*, 57:180–204, 1983.
10. K. Mehlhorn and S. Näher. *LEDA. A platform for combinatorial and geometric computing.* Cambridge University Press, Cambridge, 1999.
11. A. Salomaa and M. Soittola. *Automata-Theoretic Aspects of Formal Power Series.* Springer-Verlag, 1978.
12. L. J. Stockmeyer and A. Meyer. Word problems requiring exponential time. In *Proc. 5th Ann. ACM Sypm. on Theory of Computing*, number 1–9 in Association of Computing Machinery, 1972.