Automata, Probability, and Recursion

Mihalis Yannakakis

Department of Computer Science, Columbia University

Abstract. We discuss work on the modeling and analysis of systems with probabilistic and recursive features. Recursive Markov chains extend ordinary finite state Markov chains with the ability to invoke other Markov chains in a potentially recursive manner. The equivalent model of Probabilistic Pushdown Automata extends ordinary pushdown automata with probabilistic actions. Both of these are natural abstract models for probabilistic programs with procedures, and related systems. They generalize other classical well-studied stochastic models, e.g. Stochastic Context-free Grammars and (Multi-type) Branching Processes, that arise in a variety of areas. More generally, Recursive Markov Decision Processes and Recursive Stochastic Games can be used to model recursive systems that have both probabilistic and nonprobabilistic, controllable actions. In recent years there has been substantial work on the algorithmic analysis of these models, regarding basic questions of termination, reachability, and analysis of the properties of their executions. In this talk we will present some of the basic theory, algorithmic methods, results, and challenges.

In recent years there has been a lot of work on the modeling and analysis of systems that have both probabilistic and recursive features. In the talk we will present an overview of some of this work. In this paper we will give a brief, informal introduction to the models, on the type of questions about them that are investigated, and pointers to the literature.

Markov chains are a useful, standard model for representing the behavior of probabilistic systems in a broad variety of domains. *Recursive Markov Chains* extend ordinary finite state Markov chains with a recursive feature [23]. They can be viewed alternatively also as a probabilistic extension of Recursive State Machines (RSM) [4]. Informally, a Recursive Markov Chain (RMC for short) consists of a collection of finite-state component Markov chains that can call each other in a potentially recursive manner, like procedures. Figure 1 shows an example RMC $A = (A_1, A_2)$, consisting of two component Markov chains A_1, A_2. Each component has a set of *entry* nodes and a set of *exit* nodes where execution starts and terminates respectively; for example A_1 has one entry node en and two exit nodes ex_1, ex_2. In addition, each component has a set of other nodes and a set of *boxes*, where each box is mapped to some component and represents a recursive call to that component; for example A_1 has a box b_1 representing a recursive call to A_2. A box has a set of *call ports* and *return ports* corresponding

O.H. Ibarra and B. Ravikumar (Eds.): CIAA 2008, LNCS 5148, pp. 23–32, 2008.

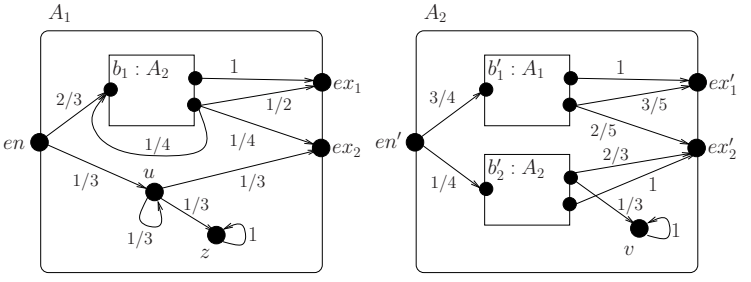

Fig. 1. A sample Recursive Markov Chain

1-1 to the entry and exit nodes of the corresponding component. A transition to a box goes to a specific call port and invokes the corresponding component starting at the corresponding entry node; when (and if) the call terminates at some exit node, then the calling component resumes execution from the corresponding return port of the box. All the transitions are labeled with probabilities as in ordinary Markov chains, summing to 1 for each node, except for call ports and exit nodes that have no transitions; as usual, for computational purposes the probabilities are assumed to be rational numbers.

An RMC A is a succinct finite representation of an underlying (in general) infinite state Markov chain M_A: As an RMC A executes starting from some initial node, at any point in time the process is at a current node of the RMC A and there is a (possibly empty) stack of pending recursive calls (i.e., of boxes); a transition is then selected probabilistically out of the current node, unless the current node is a call port of a box, in which case a new recursive call is initiated, or if the current node is an exit of a component, in which case the component that initiated the last call resumes execution from the appropriate return port of the corresponding box. Note that there is a potentially infinite (countable) number of such 'global' states of the process, because the stack of pending calls may be unbounded, and the stochastic process is a Markov chain M_A on this set of global states.

An RMC in which the calling relation between the components is acyclic is called a *Hierarchical Markov Chain* (HMC), and can be viewed as a probabilistic extension of Hierarchical State Machines [3]. An HMC A represents a finite, but typically exponentially larger, Markov chain M_A. The hierarchical construct is useful to structure and represent compactly large finite Markov chains.

An expressively equivalent model to Recursive Markov Chains is the *Probabilistic Pushdown Automaton* (pPDA) model [19], an extension of pushdown automata with probabilities on the transitions, where the probabilities of all the transitions for each state and top-of-stack symbol sum to 1. The RMC and pPDA models are equivalent in the sense that from a model of one type one can construct efficiently a model of the other type such that the two models represent essentially the same infinite state Markov chain.

Pushdown automata and recursive state machines (without probabilities) were studied for the purpose of analyzing algorithmically the properties (*model checking*) of (abstractions of) programs with procedures [6,17,4]. Similarly, a main motivation for the introduction of RMCs and pPDAs was for the analysis of probabilistic programs with procedures. The procedures correspond to the components of the RMC, the arguments and return values correspond to the entry and exit nodes. The probabilities could arise from randomizing steps or reflect statistical assumptions on the behavior of the program, under which we want to analyze its properties. The properties of interest can range from simple termination and reachability properties, to more complex properties expressed for example by temporal logic or automata specifications. The simplest type of property, and one that plays a central role also for the analysis of more complex properties is termination: Suppose that the RMC starts execution at some node; what is the probability that it will eventually reach a specified exit node (or any exit node), with no pending recursive calls, and terminate? More generally, the nodes and/or edges of a RMC can be labeled from some (finite) alphabet Σ (for example, the letters may correspond to properties satisfied by the states of the program). The executions of the RMC map to words over Σ. A (linear-time) property specification can define the set of desirable (or undesirable) executions by specifying a subset L of finite or infinite words; the question then is, what is the probability that an execution of the RMC starting from some specified initial node (or from some initial distribution) maps to a word in L. Branching-time properties can be similarly specified.

Probability and recursion are fundamental constructs that arise in a variety of contexts, and accordingly several such models have been studied and used in various fields over the years. We discuss next a number of such models.

Branching processes (BP) are an important class of stochastic processes, with applications in various areas such as population genetics, biology and others (see e.g., [34,36,38]). They were introduced first in the single type case by Galton and Watson in the 19th century to study population dynamics, and extended later by Kolmogorov and Sevastyanov to the multi-type case [39]. A branching process models the stochastic evolution of a population of entities of a given (finite) set T of types. For each type $i \in T$, there is a set of probabilistic rules concerning the set of offsprings (their number and types) that an entity of type i produces in the next generation. Starting from an initial population, a branching process evolves from one generation to the next, where in each generation every entity is replaced (independently) by a set of offspring entities chosen probabilistically according to the rules of the type of the entity. There is a well developed mathematical theory of branching processes, see [35] for a comprehensive treatment. Basic quantities of interest in a BP are the *extinction probabilities*: if the process starts with one entity of type i, what is the probability that it will become extinct, i.e. there will be eventually no descendants (these can be used to compute the extinction probability for any initial population). There is a close connection between branching processes and a subclass of RMCs, specifically the class of *1-exit RMCs* where all the components have only 1 exit (denoted *1-RMC*): From

a given finite branching process (i.e., with a finite set of types and rules) we can construct efficiently a 1-RMC such that the extinction probabilities of the types in the BP are equal to termination probabilities of nodes in the 1-RMC [23]. There seems to be a distinct difference in expressiveness and complexity between 1-exit RMCs and multiexit RMCs. The one exit restriction means that when a component terminates, it does not return any information about the call beyond the fact that it terminated. Also, as we'll mention later on, some problems can be solved efficiently for 1-exit RMCs, but we do not know how to solve them with multiple exits (and they may well be intractable.) The 1-exit restriction for RMCs corresponds to a 1-state restriction for pPDAs.

Another intimately connected well-studied model is *Stochastic Context-Free Grammars* (SCFG). These have been studied since the 1970's especially in Natural Language Processing (see e.g. [42]), and have been applied also in other areas such as biological sequence analysis [15,49]. A SCFG is a CFG where every production has an associated probability such that the probabilities of the productions for each nonterminal sum to 1. The SCFG models a stochastic process for generating strings, for example, by a leftmost derivation rule, and gives a probability to each string in the language. From a given SCFG G we can construct efficiently an 1-exit RMC A that is 'equivalent' to G in the sense that the two models represent essentially the same infinite state Markov chain. In particular, the probability of the language of G (i.e., the sum of the probabilities of all the strings in the language, which can be less than 1) is equal to the termination probability of a certain 'initial' node in the 1-RMC A, and the 1-RMC (suitably labeled) and the SCFG induce the same probabilities on the strings of the language. Conversely, it is possible to translate efficiently a given 1-exit RMC to an 'equivalent' branching process or to a SCFG, such that the termination probabilities of the nodes of the RMC are equal to extinction probabilities of the types of the BP or to the probabilities of the languages generated by the nonterminals of the SCFG [23].

Another related model, called *Random walk with back button*, was introduced and studied in [31] as a probabilistic model for web-surfing. It is an extension of a Markov chain with a 'back button' (as in a web browser) that enables the process to trace back its steps. This model corresponds to a proper subclass of 1-RMCs and SCFGs [23].

A class of models, called *Quasi-Birth-Death Processes* (QBD), have been studied for performance analysis in the queuing theory and structured Markov chain community [5,41,45]. A (discrete-time) QBD process is a (countably) infinite state Markov chain whose transition matrix has a certain repeating block structure specified by a constant number of finite matrix blocks. Generalizations of QBDs, called *tree-structured* QBDs and *tree-like* QBDs (which are equivalent to each other [53]), have been also studied; they are an extension of QBDs with an additional tree structure on the states. As shown in [22], (discrete) quasi-birth-death processes are expressively equivalent to probabilistic 1-counter automata, i.e., pPDA where the stack alphabet has only one symbol; tree-structured and tree-like QBDs are equivalent to (unrestricted) pPDA and RMCs.

In all the models we discussed above (RMC, pPDA and their subclasses considered in various fields), all the steps are probabilistic. More generally, some steps of a system/program may be probabilistic while others are not probabilistic but rather are controlled by the system or the environment. Markov Decision Processes (MDP) and Stochastic Games (SG) are standard models for systems that have both probabilistic and nonprobabilistic/controllable aspects (see e.g. [47,32,44]); they have been used in various areas, including in particular in verification as models for probabilistic concurrent systems and for open systems that interact with their environment. Extending these models with a recursive feature gives rise to *Recursive Markov Decision Processes* (RMDP) and *Recursive Stochastic Games* (RSG) [25,28]. A RMDP or RSG is like a RMC except that some of the nodes are probabilistic as in an RMC, i.e., have probabilistic transitions, and some nodes are nonprobabilistic, i.e., their transitions are controlled by the player(s). In an RMDP (like in a MDP) there is only one player that controls all the nonprobabilistic nodes, with the goal of maximizing or minimizing some objective, such as the probability of an event, (for example, termination of the process, or more generally, generation of an execution that satisfies a given property); or there can be a reward (payoff or cost) specified for the individual nodes and/or edges of the RMDP, and the player wants to maximize or minimize the reward (cost) accumulated during the execution. In a game there are two opposing players, one trying to maximize the objective, the other to minimize it. In the general form of a stochastic game (sometimes called *concurrent game*) at each (nonprobabilistic) node, each player has a finite set of possible actions that it can choose from; the players select an action simultaneously and the combination of selected actions determines the transition taken out of the current node. In a simpler form, called *simple* or *turn-based* games, only one player can choose an action (has a 'turn') at each node, i.e. the (nonprobabilistic) nodes are partitioned among the players who control the transitions out of them.

We will touch briefly now on some of the issues, methods, and results on the recursive models. The recursive feature introduces several difficulties that are not present in the nonrecursive case. One difficulty is that the probabilities that we want to compute are typically irrational. Recall that we assumed as usual that the given transitions probabilities of the models are rational. In the case of ordinary Markov chains this implies that the probabilities we want to compute of the usual types of events (including probabilities of general properties expressed for example by automata or temporal logic) are also rational, have polynomially bounded size (number of bits), and they can be computed in polynomial time. This is no more true, even for 1-exit RMCs (and SCFGs and branching processes); for example, the probabilities of termination are typically irrational. Thus the probabilities cannot be computed exactly, and can be only bounded or approximated. We distinguish between qualitative and quantitative questions regarding the desired probabilities. In the *qualitative problem* we want to determine whether a certain probability is 0, 1, or strictly between 0 and 1. For example, does a given a SCFG generate a terminal string with probability 1? Does an execution of a given RMC satisfy a given temporal property

almost surely? In a *quantitative decision problem* we want to determine how a certain probability compares with a given rational bound r, i.e. is it $<, =$ or $> r$? In a *quantitative approximation problem* we want to approximate a desired probability to a specified precision.

The termination probabilities play a central role in the analysis of RMC and pPDA: For each node u of an RMC and each exit node v of the same component as u, let $q(u, v)$ be the probability that the RMC started at u will eventually reach the exit v (with no pending recursive calls) and terminate (the corresponding quantities for pPDA are the probabilities $q(s, Y, t)$ that the pPDA starting from state s with Y on the stack will eventually pop Y and end in state t). The vector q of termination probabilities satisfies a set of equations $x = P(x)$ (one equation for each termination probability) where P is a vector of polynomials with positive rational coefficients. The system may have many solutions; however the vector P defines a monotone mapping from the nonnegative orthant to itself, and has a least fixed point (LFP), i.e., a componentwise least nonnegative solution. The LFP is precisely the vector q of termination probabilities [19,23]. From the equations $x = F(x)$, we can construct a system of polynomial equations and inequalities and use a procedure for the existential theory of the reals [10,48] to solve the qualitative and quantitative termination problems in PSPACE.

For several important subclasses of RMCs more efficient algorithms can be obtained using different methods. For example, the qualitative termination problem can be solved in polynomial time for 1-RMCs (and SCFGs and BPs), as well as for hierarchical Markov chains, using algebraic and combinatorial methods; for RMCs with linear recursion, the probabilities are rational and can be computed exactly in P-time [23]. For back-button processes the probabilities can be approximated in polynomial time using Semidefinite Programming [31]. It is an open question whether the qualitative and quantitative problems can be solved in polynomial time in general; however, this seems unlikely, and there are results indicating that it would require solving at least some hard longstanding open problems. The quantitative decision problem for 1-RMCs and hierarchical Markov chains subsumes the square root sum problem (a 30-year old simple intriguing problem that arises often in geometric computations [33,52]), and a more general problem (called posSLP) that characterizes P-time computability in a RAM model with unit cost rational arithmetic operations [1]; these problems are in PSPACE but are not even known to be in NP. For RMCs with 2 exits, even the qualitative problem (does the RMC terminate with probability 1?) is at least as hard as these problems, and the same holds for the approximation of the termination probabilities with any nontrivial constant error [29,23].

The procedures for the existential theory of reals are impractical. One approach to approximate the LFP of the system $x = P(x)$ is to start with the 0 vector and apply repeatedly P to it; the vector $P^k(0)$ converges to the LFP q as $k \to \infty$, however the convergence is exponentially slow. A faster method that accelerates convergence is to use a decomposed version of Newton's method [23]: after a preprocessing 'cleaning' step, the system is decomposed into strongly connected components (SCC) and Newton is applied bottom up on the DAG of

the SCCs; if we start from the 0 vector, then Newton is well defined (no singularity is encountered) and it converges monotonically to the LFP. This is a more practical approach; experiments with this method are reported in [43,55]. A similar Newton method can be applied more generally to monotone systems of polynomial equations, i.e. systems $x = P(x)$ where P is a vector of polynomials with positive rational coefficients; such a system may not have a fixed point, but if it does then it has a LFP and (decomposed) Newton starting at the 0 vector converges to it [23]. The rate of convergence of the method is investigated in [37,18]. They show that for strongly connected fixed point systems, Newton gains (at least) one bit of precision per iteration after some initial period, and some upper bounds are given for this initial period. In general however (for non-strongly connected systems) there are bad examples that require an exponential number of iterations to achieve a desired precision.

The analysis of more general properties of RMCs and pPDA is studied in [9,19,24,26]. Algorithms and lower bounds are given for linear time properties specified either by automata or by LTL (Linear Temporal Logic) formulas, and for branching time properties. The methods build on the termination analysis of the models as well as on methods for model checking of ordinary (nonrecursive) Markov chains. Results are given both for the general class of RMCs and pPDA, as well as for important subclasses (e.g., 1-RMCs and SCFGs, linear RMCs etc). Quantitative aspects of the executions of pPDA with a reward (cost) structure and checking of properties that involve these quantities are studied in [8,20]; these can be used for example to estimate expected termination time, stack length etc.

Recursive Markov Decision Processes and Stochastic Games are studied in [7,21,25,27,28]. For general multiexit RMDP's and RSGs, the qualitative (and quantitative) termination problems are undecidable, i.e., we cannot determine if a termination probability under optimal play is 1, and we cannot even approximate it [25]. For 1-exit RMDPs and games however, the termination problems (both qualitative and quantitative) are decidable and can be solved in PSPACE (same as for RMCs). These correspond to controlled and game versions of SCFGs and branching processes, for example, optimal control of a branching process to maximize or minimize the probability of extinction. In fact the qualitative problems for 1-RMDPs can be solved in polynomial time [27], and we can also compute the optimal and pessimal expected times to termination [21]. For simple 1-RSGs these problems are in NP$\cap$co-NP and subsume the well-known open problem of Condon [11] of computing the value of simple stochastic games.

Many of the algorithms on probabilistic recursive models have been implemented in a tool called PReMo by Wojtczak and Etessami [55].

We gave in this paper a flavor of some of the recent work on probabilistic recursive models and their analysis. We only mentioned few of the techniques and the results; we refer to the papers for the detailed results. A comprehensive survey paper is being planned with Kousha Etessami [30], and we will defer to that for a thorough exposition.

Acknowledgement. Work partially supported by NSF Grant CCF-0728736.

References

1. Abney, S., McAllester, D., Pereira, F.: Relating probabilistic grammars and automata. In: Proc. 37th Ann. Meeting of Ass. for Comp. Linguistics, pp. 542–549. Morgan Kaufmann, San Francisco (1999)
2. Allender, E., Bürgisser, P., Kjeldgaard-Pedersen, J., Miltersen, P.B.: On the complexity of numerical analysis. In: 21st IEEE Computational Complexity Conference (2006)
3. Alur, R., Yannakakis, M.: Model checking of hierarchical state machines. ACM Trans. Prog. Lang. Sys. 23(3), 273–303 (2001)
4. Alur, R., Benedikt, M., Etessami, K., Godefroid, P., Reps, T.W., Yannakakis, M.: Analysis of recursive state machines. ACM Trans. Progr. Lang. Sys. 27, 786–818 (2005)
5. Bini, D., Latouche, G., Meini, B.: Numerical methods for Structured Markov Chains. Oxford University Press, Oxford (2005)
6. Bouajjani, A., Esparza, J., Maler, O.: Reachability analysis of pushdown automata: Applications to model checking. In: Mazurkiewicz, A., Winkowski, J. (eds.) CONCUR 1997. LNCS, vol. 1243, pp. 135–150. Springer, Heidelberg (1997)
7. Brázdil, T., Brozek, V., Forejt, V., Kučera, A.: Reachability in recursive Markov decision processes. In: Baier, C., Hermanns, H. (eds.) CONCUR 2006. LNCS, vol. 4137, pp. 358–374. Springer, Heidelberg (2006)
8. Brázdil, T., Kučera, A., Esparza, J.: Analysis and prediction of the long-run behavior of probabilistic sequential programs with recursion. In: Proc. of FOCS 2005, pp. 521–530 (2005)
9. Brázdil, T., Kučera, A., Stražovský, O.: Decidability of temporal properties of probabilistic pushdown automata. In: Diekert, V., Durand, B. (eds.) STACS 2005. LNCS, vol. 3404. Springer, Heidelberg (2005)
10. Canny, J.: Some algebraic and geometric computations in PSPACE. In: Proc. of 20th ACM STOC, pp. 460–467 (1988)
11. Condon, A.: The complexity of stochastic games. Inf. & Comp. 96(2), 203–224 (1992)
12. Courcoubetis, C., Yannakakis, M.: The complexity of probabilistic verification. Journal of the ACM 42(4), 857–907 (1995)
13. Courcoubetis, C., Yannakakis, M.: Markov decision processes and regular events. IEEE Trans. on Automatic Control 43(10), 1399–1418 (1998)
14. de Alfaro, L., Majumdar, R.: Quantitative solution of omega-regular games. J. Comp. Sys. Sc. 68(2), 374–397 (2004)
15. Durbin, R., Eddy, S.R., Krogh, A., Mitchison, G.: Biological Sequence Analysis: Probabilistic models of Proteins and Nucleic Acids. Cambridge U. Press (1999)
16. Esparza, J., Gawlitza, T., Kiefer, S., Seidl, H.: Approximative methods for motonone systems of min-max-polynomial equations. In: Proc. 35th ICALP (2008)
17. Esparza, J., Hansel, D., Rossmanith, P., Schwoon, S.: Efficient algorithms for model checking pushdown systems. In: Emerson, E.A., Sistla, A.P. (eds.) CAV 2000. LNCS, vol. 1855, pp. 232–247. Springer, Heidelberg (2000)
18. Esparza, J., Kiefer, S., Luttenberger, M.: Convergence thresholds of Newton's method for monotone polynomial equations. In: Proc. STACS (2008)
19. Esparza, J., Kučera, A., Mayr, R.: Model checking probabilistic pushdown automata. In: Proc. of 19th IEEE LICS 2004 (2004); Full version in Logical Methods in Computer Science 2(1) (2006)

20. Esparza, J., Kučera, A., Mayr, R.: Quantitative analysis of probabilistic pushdown automata: expectations and variances. In: Proc. of 20th IEEE LICS (2005)
21. Etessami, K., Wojtczak, D., Yannakakis, M.: Recursive Stochastic Games with Positive Rewards. In: Proc. 35th ICALP (2008)
22. Etessami, K., Wojtczak, D., Yannakakis, M.: Quasi-birth-death processes, tree-like QBDs, probabilistic 1-counter automata, and pushdown systems (submitted, 2008)
23. Etessami, K., Yannakakis, M.: Recursive Markov chains, stochastic grammars, and monotone systems of non-linear equations. In: Diekert, V., Durand, B. (eds.) STACS 2005. LNCS, vol. 3404, pp. 340–352. Springer, Heidelberg (2005), http://homepages.inf.ed.ac.uk/kousha/bib_index.html
24. Etessami, K., Yannakakis, M.: Algorithmic verification of recursive probabilistic state machines. In: Halbwachs, N., Zuck, L.D. (eds.) TACAS 2005. LNCS, vol. 3440, pp. 253–270. Springer, Heidelberg (2005)
25. Etessami, K., Yannakakis, M.: Recursive Markov Decision Processes and Recursive Stochastic Games. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 891–903. Springer, Heidelberg (2005)
26. Etessami, K., Yannakakis, M.: Checking LTL Properties of Recursive Markov Chains. In: Proc. 2nd Intl. Conf. on Quantitative Evaluation of Systems. IEEE, Los Alamitos (2005)
27. Etessami, K., Yannakakis, M.: Efficient Qualitative Analysis of Classes of Recursive Markov Decision Processes and Simple Stochastic Games. In: Durand, B., Thomas, W. (eds.) STACS 2006. LNCS, vol. 3884, pp. 634–645. Springer, Heidelberg (2006)
28. Etessami, K., Yannakakis, M.: Recursive concurrent stochastic games. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 324–335. Springer, Heidelberg (2006)
29. Etessami, K., Yannakakis, M.: On the complexity of Nash equilibria and other fixed points. In: Proc. of 48th IEEE FOCS (2007)
30. Etessami, K., Yannakakis, M.: Recursive Markov Processes (in preparation, 2008)
31. Fagin, R., Karlin, A., Kleinberg, J., Raghavan, P., Rajagopalan, S., Rubinfeld, R., Sudan, M., Tomkins, A.: Random walks with "back buttons" (extended abstract). In: ACM Symp. on Theory of Computing, pp. 484–493 (2000); Full version in Ann. of App. Prob., 11, pp 810–862 (2001)
32. Filar, J., Vrieze, K.: Competitive Markov Decision Processes. Springer, Heidelberg (1997)
33. Garey, M.R., Graham, R.L., Johnson, D.S.: Some NP-complete geometric problems. In: 8th ACM Symp. on Theory of Computing, pp. 10–22 (1976)
34. Haccou, P., Jagers, P., Vatutin, V.A.: Branching Processes: Variation, Growth, and Extinction of Populations. Cambridge U. Press (2005)
35. Harris, T.E.: The Theory of Branching Processes. Springer, Heidelberg (1963)
36. Jagers, P.: Branching Processes with Biological Applications. Wiley, Chichester (1975)
37. Kiefer, S., Luttenberger, M., Esparza, J.: On the convergence of Newton's method for monotone systems of polynomial equations. In: Proc. 39th Symp. on Theory of Computation (STOC), pp. 217–226 (2007)
38. Kimmel, M., Axelrod, D.E.: Branching processes in biology. Springer, Heidelberg (2002)
39. Kolmogorov, A.N., Sevastyanov, B.A.: The calculation of final probabilities for branching random processes. Dokl. Akad. Nauk SSSR 56, 783–786 (1947) (Russian)
40. Kwiatkowska, M.: Model checking for probability and time: from theory to practice. In: 18th IEEE LICS, pp. 351–360 (2003)

41. Latouche, G., Ramaswami, V.: Introduction to Matrix Analytic Methods in Stochastic Modeling. ASA-SIAM series on statistics and applied probability (1999)
42. Manning, C., Schütze, H.: Foundations of Statistical Natural Language Processing. MIT Press, Cambridge (1999)
43. Nederhof, M.J., Satta, G.: Using Newton's method to compute the partition function of a PCFG (unpublished manuscript, 2006)
44. Neyman, A., Sorin, S. (eds.): Stochastic Games and Applications. Kluwer, Dordrecht (2003)
45. Neuts, M.F.: Stuctured Stochastic Matrices of M/G/1 Type and their applications. Marcel Dekker, New York (1989)
46. Paz, A.: Introduction to Probabilistic Automata. Academic Press, London (1971)
47. Puterman, M.L.: Markov Decision Processes. Wiley, Chichester (1994)
48. Renegar, J.: On the computational complexity and geometry of the first-order theory of the reals, parts I-III. J. Symb. Comp. 13(3), 255–352 (1992)
49. Sakakibara, Y., Brown, M., Hughey, R., Mian, I.S., Sjolander, K., Underwood, R., Haussler, D.: Stochastic context-free grammars for tRNA modeling. Nucleic Acids Research 22(23), 5112–5120 (1994)
50. Sevastyanov, B.A.: The theory of branching processes. Uspehi Mathemat. Nauk 6, 47–99 (1951) (Russian)
51. Shapley, L.S.: Stochastic games. Proc. Nat. Acad. Sci. 39, 1095–1100 (1953)
52. Tiwari, P.: A problem that is easier to solve on the unit-cost algebraic RAM. Journal of Complexity, 393–397 (1992)
53. van Houdt, B., Blondia, C.: Tree structured QBD Markov chains and tree-like QBD processes. Stochastic Models 19(4), 467–482 (2003)
54. Vardi, M.: Automatic verification of probabilistic concurrent finite-state programs. In: Proc. of 26th IEEE FOCS, pp. 327–338 (1985)
55. Wojtczak, D., Etessami, K.: Premo: an analyzer for probabilistic recursive models. In: Grumberg, O., Huth, M. (eds.) TACAS 2007. LNCS, vol. 4424. Springer, Heidelberg (2007), `http://groups.inf.ed.ac.uk/premo/`