Masaryk University
Faculty of Informatics

How To Efficiently Play With Infinitely Many States

Tomáš Brázdil

Habilitation Thesis

2012

Abstract

Formal modeling and verification usually play an important role in system development process. Thus one of the crucial steps in such a process is selection of an appropriate modeling language that would faithfully capture essential features of the system under consideration. Such features often include some kind of randomness as well as interaction between components of the system. Also, many real-world systems operate with potentially unbounded data structures such as stacks and queues. Stochastic games on graphs with infinitely many vertices are a natural abstract model of such potentially infinite state systems with randomness and interaction. From the point of view of modeling and verification, both game-theoretic aspects and related algorithmic issues are of utmost importance. Even though the game theoretic issues can be studied for the abstract class of stochastic games with infinitely many vertices, to deal with algorithmic issues we have to concentrate on classes of finitely generated games.

This thesis surveys the author's contribution to the area of stochastic processes and games with countably infinite number of vertices. After dealing with fundamental game theoretic issues for abstract stochastic games with reachability objectives, we concentrate on a rich class of stochastic games with finitely many control states accompanied by a (first in, last out) stack called *stochastic pushdown games*. Even though most problems concerning these games are undecidable in general, we identify some important subclasses that may be efficiently solved. Then we turn our attention to games with multiple counters called *games on vector addition systems with states*, so far studied only in a non-stochastic variant. We identify an interesting subclass of *consumption games*, which are useful in modeling resource critical systems. Finally, we extend our techniques to deal with planning problems for one processor, where tasks may stochastically generate new tasks, and to solve stabilization problems for a special kind of controlled queueing networks.

The thesis is composed of a collection of papers and an accompanying survey of the most important results. The collection consists of four journal papers and five papers published in proceedings of international conferences. In all papers of the collection, the contribution of the author of this thesis is at least proportional to the number of co-authors. In all cases the author contributed to all activities: discussions and formulation of problems, solution and writing. Particularly, in papers on probabilistic pushdown systems, the contribution of the author is more than the proportion, especially on the side of crucial ideas.

Abstrakt

Formální modelování a verifikace tvoří důležitou součást návrhu mnoha systémů. Jedním z klíčových kroků je tedy volba vhodného modelovacího jazyka. Tato volba je silně závislá na potřebě věrně zachytit klíčové aspekty modelovaného systému. Mezi takové aspekty často patří jistý druh náhodnosti a také interakce jednotlivých částí systému. Mnoho reálných systémů navíc pracuje s potenciálně neomezenými datovými strukturami, jako jsou zásobníky, fronty apod. Stochastické hry na grafech s nekonečně mnoha vrcholy jsou přirozeným abstraktním modelem potenciálně nekonečně stavových systémů, které vykazují prvky náhodnosti a interakce. Z pohledu modelování a verifikace je důležité studovat jednak teoretické aspekty těchto her, jednak jejich algoritmické řešení. Ačkoliv teoretické aspekty mohou být studovány pro abstraktní třídu všech stochastických her s nekonečně mnoha vrcholy, pro algoritmickou analýzu se musíme omezit na různé podtřídy konečně generovaných her.

Tato práce podává přehled autorova přínosu v oblasti stochastických procesů a her s nekonečně mnoha vrcholy. Poté, co shrneme fundamentální teoretické aspekty obecných stochastických her, se budeme věnovat bohaté třídě her s konečně mnoha kontrolními stavy a zásobníkem, které se nazývají *stochastic pushdown games*. Ačkoliv jsou tyto hry v plné obecnosti algoritmicky neřešitelné, identifikujeme několik důležitých podtříd, které mohou být řešeny pomocí efektivních algorimů. Poté se budeme věnovat hrám dvou hráčů s více čítači, které se nazývají *games on vector addition systems with states*. Identifikujeme podtřídu takzvaných *consumption games*, které jsou vhodné pro modelování systémů závislých na několika zdrojích. Později modifikujeme a rozšíříme naše metody na řešení problému plánování úkolů pro jeden procesor v prostředí, kde úkoly mohou náhodně generovat nové úkoly, a nakonec na speciální typ řízených sítí front.

Tato práce se skládá ze souboru článků společně s přehledem nejdůležitějších výsledků. Soubor obsahuje čtyři časopisecké publikace a pět článků publikovaných ve sbornících mezinárodních konferencí. Přínos autora této práce na uvedených článcích odpovídá minimálně poměru danému počtem spoluautorů. Ve všech případech se autor této práce podílel na všech fázích tvorby článku, počínaje diskuzí a stanovením obsahu, přes řešení problémů, až po samotné psaní článku. V případě prací o stochastic pushdown games je podíl autora výraznější, zejména co se týče zásadních myšlenek řešení problémů.

Acknowledgements

First, I would like to thank all my collaborators and students for their strong devotion to our joint work. Special thanks go to Tony Kučera, who remained my exceptional supervisor even after I obtained my PhD. I would like to thank Vojtěch Forejt for collaboration and for reading this text.

I am also very grateful to my wife, Silvie Luisa, for her strong support (for the first time in our lives she attributed some importance to my work), my older daughter, Irenka, for *not* destroying my computer completely and my younger daughter, Alenka, for *not* screaming too loudly.

Contents

Chapter 1

Introduction

Formal verification utilizes mathematical methods to prove that a system satisfies desired properties. This comprises building a formal model of the system using a suitable modeling language. Subsequently, the model is analyzed, using an appropriate verification tool.

The choice of a modeling language depends on the structure and features of the system under consideration. Typical aspects that must be taken into account include randomness and interaction. For example, a system controlling an aircraft should deal with randomness in its environment (a direction and strength of wind) and interaction with other aircraft and control systems on the ground. Such systems can be conveniently modeled using various types of stochastic games and their extensions towards engineering applications. Subsequent analysis of such models may exploit a large reservoir of methods from game theory. Game theory is a rich area of mathematics with many applications in economics, robotics, game playing, verification, etc. (see, e.g., [62, 50]). In this work we concentrate on specific types of games on graphs that are especially useful in verification of systems that interact with their environment.

Another important line of research in formal methods is devoted to infinite-state systems. The study is motivated by the fact that systems with potentially unbounded resources or data structures cannot be faithfully modeled by finite-state systems. For example, recursive programs use (potentially) infinite stacks, queueing systems use unbounded queues, etc. There is a rich theory of discrete infinite-state systems with non-determinism (see e.g. [7]), but the study of infinite-state systems with randomness and interaction is only at the beginning. This text is a compilation of several works on such systems co-authored by the author of this thesis in recent years.

As a formal foundation for our reasoning about infinite-state systems with randomness and interaction we consider turn-based, discrete-time, zero-sum stochastic games, called shortly *stochastic games (SG)*, played on possibly infinite directed graphs (see Figure 1.1 for a simple example of a countable SG).

Each vertex of the graph is owned either by one of the players, Max or Min, or by the random environment. Intuitively, such a game starts by putting a token on some vertex. Subsequently, the token is moved from vertex to vertex either by one of the players or randomly (depending on the owner of the vertex) which gives a sequence of vertices called a *run*. The winning condition is usually specified by a set of winning runs of player Max; the goal of player Max is to maximize the probability of winning, Min minimizes this probability. In most of our work we concentrate on *reachability* objectives where the goal of player Max is to reach a given set of vertices.

We start by investigating SG with reachability objectives from the point of view of classical game theory. That is, we consider existence of a value, optimal strategies, approximately

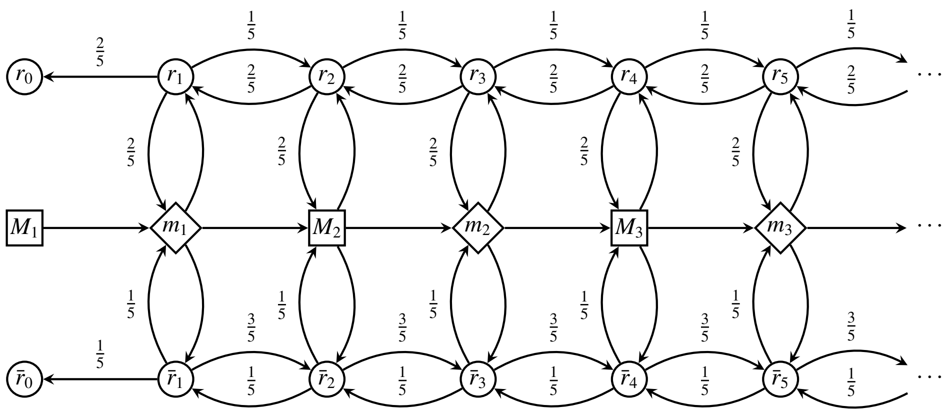


Figure 1.1: A countable SG. The vertices $M_1, M_2, \dots$ belong to Max, $m_1, m_2, \dots$ belong to Min, the remaining vertices $r_0, r_1, \dots$ and $\bar{r}_1, \bar{r}_2, \dots$ are stochastic. Every transition from a stochastic state is labeled with a rational probability of being taken (e.g., the transition from r_1 to m_1 is taken with the probability $\frac{2}{5}$).

optimal (or ε-optimal) strategies, etc. The most interesting fact presented in this part is that most results valid for finite SG break down when we allow infinitely many vertices. In addition, for the abstract class of games with infinitely many vertices, algorithmic problems cannot be solved. As we are mostly interested in effective analysis of systems, we concentrate on subclasses of SG that are finitely generated by various well motivated mechanisms.

Roughly speaking, games considered in most of this work can be seen as finite games enriched with potentially unbounded data structures such as stacks, counters, or queues. More concretely, we consider simple stochastic pushdown games (i.e., games with an unbounded stack), games with multiple counters, and systems for scheduling tasks together with controlled branching queueing networks. Now we briefly outline these extensions, more precise definitions will be provided at appropriate places in later chapters.

We start by adding a (last in, first out) stack to finite SG which yields *stochastic pushdown games (PDA-SG)*, that is games on graphs generated by pushdown automata, a standard model of programs with recursion. A vertex of a PDA-SG, usually called a *configuration*, consists of one out of finitely many control states and some stack contents (i.e., a string of stack symbols from a finite stack alphabet). Transitions of the game may change the control state as well as push and pop symbols onto and from the stack. An owner of each configuration is determined by the control state and the top symbol of the stack.

As most algorithmic issues are undecidable for general PDA-SG, we consider two standard subclasses. The first subclass consists of stateless PDA-SG, called *stochastic BPA games (BPA-SG)*, whose configurations consist just of the stack contents. These games are closely related to controlled versions of branching processes (a model of populations whose individuals may reproduce or die which is very useful in nuclear physics, genomics, computer science, etc.) and stochastic context-free grammars (that are useful in natural language processing, molecular biology, etc.) The second subclass is obtained by restricting PDA-SG to single letter stack alphabet, which basically turns the stack into an unbounded counter, and we get *one-counter stochastic games (OC-SG)*. The counter is useful in modeling energy and other resources, lengths of queues in some special cases of queueing systems, etc. We show that many problems undecidable for PDA-SG become efficiently solvable for both BPA-SG and OC-SG.

By generalizing from one-counter to multiple counter systems, we obtain games on so-called vector addition systems (only non-stochastic variant of these games has been considered so far). A configuration (i.e., a vertex) of such a game consists of one out of finitely many control states and a vector of integer values of counters. In every step, the owner of the current configuration, determined by the control state, chooses one of finitely many available transitions. Each transition changes the control state and the counter values by adding a fixed "displacement" vector of integers to the vector of counter values. We consider these games to be a natural model of resource critical systems where multiple resources may be consumed and reloaded. Thus the objective of one player is to preserve the resources, i.e., to have all counters always positive, the other one strives to reach zero in one of the counters. Our motivation with resource consummation leads us to introduce new symbolic components ω to the displacement vectors whose intuitive meaning is "add an arbitrarily large non-negative integer to the counter". For example, a transition with a displacement vector $(-1, \omega)$ decreases the first counter by one and allows to add an arbitrary amount to the second one. By introducing ω components, we obtain *games on extended vector addition systems with states (eVASS games).* We prove that eVASS games are decidable, but the complexity is extremely high. So later we introduce a special type of eVASS games, called *consumption games*, where displacement vectors cannot contain positive integer components. This intuitively means that resources can be reloaded *only* by means of ω components. Surprisingly, this restriction drastically decreases the complexity of analysis.

All of the above games are completely defined within the framework of SG. In the last chapter we consider two formalisms that do not fall strictly within this framework but still closely resemble (one player) games on graphs with countably many vertices. First, we consider so-called *task systems (TS)*, a simple model of scheduling tasks for execution on one processor where tasks may randomly generate new tasks (computer systems with threads, branch and bound algorithms, etc.) A configuration of such a system consists of a pool of unfinished tasks; in every step a scheduler chooses a task to be executed which may, in turn, randomly generate new tasks to the pool. We are interested in completion time, that is the total number of processed tasks before the pool becomes empty, as well as in minimization of completion space which is the maximal size of the pool during computation (which in turn is the sufficient size of memory needed to store unfinished tasks during computation).

Lastly, we consider a continuous-time version of task systems which allows new tasks to come from the outside, called *controlled branching queueing networks (CBQN).* CBQN fall within the large framework of open queueing networks. From the point of view of queueing theory, CBQN are an extension of standard Jackson's networks with a control mechanism and branching. This means that we allow tasks (or jobs) to generate new tasks according to actions chosen by a scheduler. We consider the problem of stabilization of CBQN that is, roughly speaking, the existence of a scheduler under which the average queue lengths stay finite.

1.1 Papers in the Collection

This text is intended as a standalone survey of the most important results of the papers listed below and thus presents these results using unified notation. Therefore some notation used in this text differs from the corresponding notation in the papers. The following papers are listed in their order of appearance in this survey.

[18] T. Brázdil, V. Brožek, V. Forejt, and A. Kučera. Reachability in recursive Markov decision processes. *Information & Computation*, 206(5):520–537, 2008

This is a full version of paper [17] published at CONCUR 2006.

The author's contribution: 35%, discussions, main ideas of some proofs, some writing.

[21] T. Brázdil, V. Brožek, A. Kučera, and J. Obdržálek. Qualitative reachability in stochastic BPA games. *Information & Computation*, 209(8):1160–1183, 2011

This is a full version of paper [20] published at STACS 2009.

The author's contribution: 40%, discussions, main ideas of some proofs (especially of the hardest proof for the witness strategies), some writing.

[15] T. Brázdil, V. Brožek, K. Etessami, A. Kučera, and D. Wojtczak. One-counter Markov decision processes. In *Proceedings of 21st Annual ACM-SIAM Symp. on Discrete Algorithms (SODA 2010)*, pp. 863–874. SIAM, 2010

The author's contribution: 35%, formulation of the main problem, discussions, the overall strategy of the proof for the non-selective case, some detailed proofs, writing.

[12] T. Brázdil, V. Brožek, and K. Etessami. One-counter stochastic games. In *Proceedings of IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS 2010)*, vol. 8 of *LIPIcs*, pp. 108–119. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2010

The author's contribution: 40%, discussions, main ideas of some crucial proofs, writing.

[13] T. Brázdil, V. Brožek, K. Etessami, and A. Kučera. Approximating the termination value of one-counter MDPs and stochastic games. *Information & Computation*. doi: http://dx.doi.org/ 10.1016/j.ic.2012.01.008. To appear.

This is a full version of paper [14] published at ICALP 2011. It has already been accepted for publication in Information & Computation.

The author's contribution: 35%, discussions, ideas crucial for the main decidability result, some writing.

[27] T. Brázdil, P. Jančar, and A. Kučera. Reachability games on extended vector addition systems with states. In *Proceedings of 37th Internat. Colloq. on Automata, Languages and Programming (ICALP 2010, Part II)*, vol. 6199, pp. 478–489, 2010

The author's contribution: 33%, discussions, some proofs, writing.

[22] T. Brázdil, K. Chatterjee, A. Kučera, and P. Novotný. Efficient controller synthesis for consumption games with multiple resource types. In *Proceedings of 24th International Conference on Computer Aided Verification (CAV 2012)*, vol. 7358 of *LNCS*, pp. 23–38. Springer, 2012

The author's contribution: 30%, discussions, the idea of consumption games, some proofs and algorithms, writing.

[25] T. Brázdil, J. Esparza, S. Kiefer, and M. Luttenberger. Space-efficient scheduling of stochastically generated tasks. *Information & Computation*, 210:87–110, 2012.

This is a full version of paper [24] published at ICALP 2010.

The author's contribution: 25%, discussions, the definition of task systems, some proofs, some writing.

[28] T. Brázdil and S. Kiefer. Stabilization of branching queueing networks. In *Proceedings of 29th Internat. Symp. on Theoretical Aspects of Computer Science (STACS 2012)*, vol. 14 of *LIPIcs*, pp. 507–518. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2012

The author's contribution: 50%, equal collaboration of the co-authors.

1.2 Outline of the Thesis

Chapter 2 This chapter is based on parts of papers [18, 21]. We introduce countable SG with reachability objectives. Then we present fundamental results on determinacy of such games, existence of optimal and winning strategies, etc. All results are presented both for general SG and for subclasses with bounded branching degree. Results of this chapter support most of our reasoning in later chapters where we consider special cases that can be effectively analyzed. In the last part of this chapter we formulate standard algorithmic problems concerning stochastic games on graphs that will occupy the central stage of the rest of this thesis.

Chapter 3 This chapter, based on papers [18, 21, 15, 12, 13], constitutes the core of the thesis. Here the results on algorithmic analysis of PDA-SG with reachability objectives are presented. We start by summarizing existing undecidability results for general PDA-SG. Subsequently, we concentrate on the subclasses of BPA-SG and OC-SG that admit efficient solution for some of the algorithmic problems formulated in Chapter 2. More concretely, Section 3.1 presents results of [18, 21] on qualitative reachability in BPA-SG, and Section 3.2 presents results of [15, 12, 13] on qualitative [15, 12] as well as quantitative [13] reachability in OC-SG.

Chapter 4 This chapter, based on papers [27, 22], deals with eVASS games. As opposed to PDA-SG, these games are non-stochastic but still form a subclass of SG. We consider so-called termination objectives, a special form of reachability, capturing depletion of resources. We present results of [27] about decidability of eVASS games. In Section 4.2, we present results of [22] on consumption games.

Chapter 5 In this chapter, based on papers [25, 28], we present a formalism for modeling and analysis of task systems (TS) and CBQN. Concretely, in Section 5.1, we present the (discrete-time) TS as studied in [25]. In Section 5.2, we treat the (continuous-time) CBQN from [28].

Chapter 2

Countable Stochastic Games

In this section we lay foundations for the rest of this text (except possibly the last chapter) by defining stochastic games with countably many vertices and by studying their fundamental game-theoretic properties. At the end of the chapter we also formulate standard algorithmic problems that will be solved in later chapters for various subclasses of SG.

Syntax: A (countable) *stochastic game (SG)* $\mathcal{G}$ consists of

- a countable set of *vertices* V, partitioned into the vertices $V_\Box$ of player Max, $V_\Diamond$ of player Min, and stochastic vertices V_P,
- a *transition relation* $\rightarrow \subseteq V \times V$ such that for every $v \in V$ there is some u satisfying $v \rightarrow u$,
- a map *Prob* taking each transition $v \rightarrow u$ with $v \in V_P$ to a positive rational number $Prob(v \rightarrow u)$, so that for every $v \in V_P : \sum_{v \rightarrow u} Prob(v \rightarrow u) = 1$.

A SG where $V_\Diamond = \emptyset$ is called a *maximizing Markov Decision Process (maximizing MDP)*, similarly $V_\Box = \emptyset$ defines a *minimizing MDP*. Finally, if $V_\Box = V_\Diamond = \emptyset$ we have a *Markov Chain (MC)*. A SG with $V_P = \emptyset$ is called a *two-player game*.

A *branching degree* of a vertex v is the number of vertices u satisfying $v \rightarrow u$. A game is *finitely-branching* if all vertices have a finite branching degree.

Semantics: The game is initiated by putting a token on some vertex. The token is moved from vertex to vertex by two players, Max and Min, who choose (possibly in random) the next move in the vertices of $V_\Box$ and $V_\Diamond$, respectively. In the vertices of V_P, the outgoing edges are chosen randomly according to the probability distribution *Prob*. Playing this way for infinitely many steps produces a run. Formally, a *run* is an infinite sequence of vertices $\mathbf{w} = v_0 v_1 \cdots$ such that for all $i \geq 1$ we have that $v_{i-1} \rightarrow v_i$. A finite prefix $v_0 \cdots v_k$ of a run is called a *(finite) path*.

Players choose their transitions according to fixed strategies. A *strategy* for player Max is a function σ which to each finite path $w = v_0 \cdots v_k$ (also called a *history* in this context) where $v_k \in V_\Box$, assigns a probability distribution on the set of transitions of the form $v_k \rightarrow u$. In general, strategies may use complete history as well as randomization. We consider special types of strategies obtained by restricting their use of memory and randomization. Namely, a strategy σ is called *deterministic (D)* if for each path w the distribution $\sigma(w)$ assigns probability 1 to some transition. A strategy which is not deterministic is *randomized (R)*. We call σ *memoryless (M)* if $\sigma(w)$ depends only on the last vertex v_k. A strategy which is not memoryless is *history-dependent (H)*. Strategies for Min are defined similarly, just by substituting $V_\Box$ with $V_\Diamond$. Thus, we obtain the MD, MR, HD, and HR strategy classes, where HR are unrestricted strategies and MD are the most restricted memoryless deterministic strategies. In what follows, strategies of Max are denoted by $\sigma, \sigma', \ldots$, strategies of Min by $\pi, \pi', \ldots$ That is, e.g., $\sigma \in MR$ means that σ is an arbitrary memoryless deterministic strategy of player Max.

Objectives, the value and optimal strategies

An *objective* is a measurable [1] set O of runs. Given strategies σ and π of players Max and Min, resp., an initial vertex v, and an objective O, we denote by $\mathbb{P}_v^{\sigma,\pi}(O)$ the probability that a run of O initiated in v is produced when both players follow the strategies σ and π. In a maximizing (or a minimizing) MDP we write just $\mathbb{P}_v^{\sigma}(O)$ (or $\mathbb{P}_v^{\pi}(O)$, resp.)

Player Max strives to maximize the probability of a given objective O, whereas player Min strives to minimize it. [2] Formally, given an objective O and a vertex v, we define the *value in v* by

$$\mathrm{Val}(O,v) \quad := \quad \sup_\sigma \inf_\pi \mathbb{P}_v^{\sigma,\pi}(O) \quad = \quad \inf_\pi \sup_\sigma \mathbb{P}_v^{\sigma,\pi}(O)\,.$$

The latter equality follows from Martin's Blackwell determinacy theorem [59, 58]. Given $\varepsilon \geq 0$, we say that a strategy σ of player Max (or π of player Min) is *ε-optimal in v* if $\inf_\pi \mathbb{P}_v^{\sigma,\pi}(O) \geq \mathrm{Val}(O,v) - \varepsilon$ (or $\sup_\sigma \mathbb{P}_v^{\sigma,\pi}(O) \leq \mathrm{Val}(O,v) + \varepsilon$, respectively). We say that a strategy is *optimal in v* if it is 0-optimal in v.

We are mostly concerned with *reachability objectives* where the goal of player Max is to maximize the probability of reaching a fixed set of vertices T. Denote by *Reach*(T) the set of all runs that eventually visit a vertex of T.

Standard issues: Following classical game theory, we are mostly interested in the following issues:

1. Are there optimal strategies?
2. Are there special types of strategies, such as memoryless and/or deterministic, that are ε-optimal?

Related work and known results

There is of course vast literature on stochastic games and Markov decision processes with *finitely* many vertices (for a survey of classical results see [50, 65], modern results about finite stochastic games in verification and synthesis are surveyed, e.g., in [35, 33]). These games have been studied with many objectives such as mean payoff, total reward, objectives specified by temporal logics, etc. In particular, for stochastic games with finitely many vertices the following is well known:

Fact 2.0.1 ([50])**.** *Consider a SG with finitely many vertices and a reachability objective. Then both players have MD strategies that are optimal in all vertices.*

[1]We assume a standard measurable space of runs generated by basic cylinders, i.e., sets of runs with a fixed prefix. For details see, e.g., [8].

[2]Sometimes the goals of players in stochastic games are specified in a more general way using payoff functions that assign numbers to runs. Given a payoff function f, the goal of Max (Min) is to maximize (minimize) the expected value of f. Our notion of objective corresponds to payoff functions with values in $\{0, 1\}$ which is sufficient for our purposes.

Countable SG are a special case of finitely additive stochastic games studied in [58]. In general, these games were studied with arbitrary Borel objectives, or Borel measurable payoff functions. Determinacy of such games follows from the famous result of Martin, [59]. Such general games do not posses other nice properties, such as existence of optimal strategies and of special types of ε-optimal strategies.

Countable stochastic games with special types of objectives, mostly mean payoff and expected total reward, have also been studied in literature. In particular, [65] contains some sections (e.g., Section 6.10 and Section 8.10) on countable Markov decision processes. Some results of [65] may be translated to reachability objectives, however, they are restricted to Markov decision processes and some issues considered in our work, such as existence of winning strategies, algorithmic problems and optimality for finitely branching games, are not covered.

2.1 Countable SG with Reachability Objectives

One of the main issues addressed in [18, 21], is whether Fact 2.0.1 remains valid for SG with infinitely many vertices. The answer is that, in general, for countable SG with reachability objectives most of Fact 2.0.1 does *not* hold.

Let us fix a SG $\mathcal{G}$, a vertex v of $\mathcal{G}$ and a reachability objective $\mathit{Reach}(T)$. Proposition 4. of [18] implies the following.

(1) An optimal strategy for Max does not necessarily exist in v even though $\mathcal{G}$ is a maximizing MDP where branching degrees of all vertices are bounded by two and $\mathrm{Val}(\mathit{Reach}(T), v) = 1$ (see Figure 2 of [18]).

(2) An optimal strategy for Min does not necessarily exist in v even though $\mathcal{G}$ is a minimizing MDP and $\mathrm{Val}(\mathit{Reach}(T), v) = 0$ (see Figure 1 of [18]).

(3) MR strategies may be strictly weaker than HD strategies even though $\mathcal{G}$ is a minimizing MDP. In fact, there is a minimizing MDP $\mathcal{G}$ in which the following holds:

$$1 \;=\; \inf_{\pi\in MR} \mathbb{P}^{\pi}_{v}(\mathit{Reach}(T)) \;>\; \inf_{\pi\in HD} \mathbb{P}^{\pi}_{v}(\mathit{Reach}(T)) \;=\; 0$$

(see Figure 1 of [18]).

The paper [18] shows that an analogy of (3) does not hold for *maximizing* MDP.

Proposition 2.1.1 ([18], Proposition 4. (4))**.** *If $\mathcal{G}$ is a maximizing MDP, then*

$$\sup_{\sigma\in MD} \mathbb{P}^{\sigma}_{v}(\mathit{Reach}(T)) \quad = \quad \sup_{\sigma\in HR} \mathbb{P}^{\sigma}_{v}(\mathit{Reach}(T))$$

To obtain an analogy of Proposition 2.1.1 for minimizing MDP, [18] proposes to restrict to finitely-branching minimizing MDP, for which even stronger result can be proved.

Proposition 2.1.2 ([18])**.** *If $\mathcal{G}$ is a* finitely-branching *minimizing MDP, Min has a MD strategy which is optimal in v.*

Very recently, the above results for MDP have been generalized to SG as described in the following remark.

Remark 2.1.3. *Results of [29] generalize Proposition 2.1.1 and Proposition 2.1.2 to SG. Concretely, assume that in $\mathcal{G}$ all vertices of Min are finitely-branching. Then*

- *player Min has a MD strategy which is optimal in v,*
- *for player Max we have*

$$\sup_{\sigma \in MD} \inf_{\pi \in HR} \mathbb{P}_v^{\sigma,\pi}(Reach(T)) \quad = \quad \sup_{\sigma \in HR} \inf_{\pi \in HR} \mathbb{P}_v^{\sigma,\pi}(Reach(T))$$

For a more detailed discussion of determinacy in SG see [29].

Quantitative constraints

Although player Max does not necessarily have an optimal strategy in v, we may still ask whether Max has a strategy σ which satisfies a given constraint on the probability of $Reach(T)$, such as $\mathbb{P}_v^{\sigma,\pi}(Reach(T)) \geq \frac{1}{2}$ or $\mathbb{P}_v^{\sigma,\pi}(Reach(T)) > \frac{3}{11}$, against all strategies π of Min.

Generally, given $\sim\, \in \{>, \geq\}$ and $r \in [0,1]$ and an objective O, we say that a strategy σ of player Max is $(O, {\sim} r)$*-winning in* v if $\mathbb{P}_v^{\sigma,\pi}(O) \sim r$ for all π. Similarly, a strategy π of player Min is $(O, {\sim} r)$*-winning in* v if $\mathbb{P}_v^{\sigma,\pi}(O) \not\sim r$ for all σ. The following is a corollary of Theorem 3.3 of [21].

Theorem 2.1.4 ([21], Theorem 3.3)**.** *Assume that $\mathcal{G}$ is finitely-branching and consider the reachability objective $Reach(T)$. Given $\sim\, \in \{>, \geq\}$ and $r \in [0,1]$, one of the players has a $(Reach(T), {\sim} r)$-winning strategy in v.*

Let us remark that combining techniques of [21] and [29] the above proposition may be strengthened to games where only vertices of Min are finitely branching. However, this result has not been published yet.

2.2 Algorithmic Problems for Countable SG

We are mostly interested in algorithmic theory of SG. That is given (a finite representation of) a SG $\mathcal{G}$, a vertex v of $\mathcal{G}$, and an objective O, we want to

1. compute or at least approximate [3] the value $\mathrm{Val}(O, v)$,
2. compute optimal or at least ε-optimal strategies for a given $\varepsilon \geq 0$.

We also consider problems involving constraints: Given $\sim\, \in \{>, \geq\}$ and $r \in [0,1]$,

1. decide whether $\mathrm{Val}(O, v) \sim r$ for a given v,
2. decide whether Max has a $(O, {\sim} r)$-winning strategy in a given vertex and if yes, compute it (or more generally, compute the set of all vertices in which Max has a $(O, {\sim} r)$-winning strategy).

[3]For most SG considered in this text where the stochastic environment is allowed (namely for PDA-SG, BPA-SG, OC-SG and also, in a sense, task systems and CBQN) only approximation of the value is possible since the value may be irrational. In fact, there is a very simple countable Markov chain, that can be implemented using all above models, in which the probability of reaching a distinguished vertex is irrational (see either Figure 1 and Example 2 of [23], or Theorem 3.2 (1) of [47]).

If $r \in \{0, 1\}$, the problems are called *qualitative*, otherwise, they are *quantitative*.

Of course, the above issues cannot be addressed for general games with infinitely many vertices. We need to restrict ourselves to games that possess a finite representation. This is the contents of this thesis, to study various well-motivated restrictions on countable SG that admit effective (and in many cases efficient) algorithmic analysis. For start, let us recall a well known result for finite SG:

Fact 2.2.1 ([37, 39])**.** *Let $\mathcal{G}$ be a SG with finitely many vertices, let v be a vertex of $\mathcal{G}$ and let $Reach(T)$ be a reachability objective. Further, let $\sim \in \{>, \geq\}$ and let $r \in [0, 1]$ be a rational constant. The value* $\mathrm{Val}(Reach(T), v)$ *is rational and the problem whether* $\mathrm{Val}(Reach(T), v) \sim r$ *belongs to* ***NP*** $\cap$ ***co-NP****. If $\mathcal{G}$ is a maximizing or a minimizing MDP with finitely many vertices, the value as well as optimal MD strategies can be computed in polynomial time.*

Let us remark that for games with finitely many vertices the above facts hold for much more complex objectives. However, in this work we are mostly interested in reachability so for other objectives refer, e.g., to [35, 33, 38].

Chapter 3

Stochastic PDA Games

In this chapter we summarize results of the papers [18, 21, 15, 12, 13] concerning stochastic PDA games. We start with general stochastic PDA games and summarize mostly negative results about their algorithmic analysis. Subsequently, we move to classes of stochastic BPA games (Section 3.1 where papers [18, 21] are summarized) and to stochastic OC games (Section 3.2 where papers [15, 12, 13] are summarized).

Syntax: A *stochastic PDA game (PDA-SG)* Δ consists of

- a finite set of *control states* Q and a finite *stack alphabet* Γ; together with a partition of the set $H := Q \times \Gamma$ of *heads* into heads $H_\Box$ of player Max, heads $H_\Diamond$ of player Min, and stochastic heads H_P,
- a finite set of *transition rules* of the form $pX \hookrightarrow q\alpha$, where $p, q \in Q$ are control states, $X \in \Gamma$ is a stack symbol, and $\alpha \in \Gamma^*$ is a (possibly empty) sequence of stack symbols; we assume that for every $pX \in H$ there is at least one transition rule of the form $pX \hookrightarrow q\alpha$,
- a map *Prob* taking each transition rule $pX \hookrightarrow q\alpha$ with $pX \in H_P$ to a positive rational number $Prob(pX \hookrightarrow q\alpha)$, so that for every $pX \in H_P : \sum_{pX \hookrightarrow q\alpha} Prob(pX \hookrightarrow q\alpha) = 1$.

A *configuration* is a pair $p\alpha$ where $p \in Q$ and $\alpha \in \Gamma^*$.

A stochastic PDA game where $H_\Diamond = \emptyset$ is called a *maximizing PDA Markov Decision Process (maximizing PDA-MDP)*, similarly $H_\Box = \emptyset$ defines a *minimizing PDA Markov Decision Process (minimizing PDA-MDP)*. Finally, if $H_\Box = H_\Diamond = \emptyset$ we have a *probabilistic pushdown automaton (pPDA)*.

Semantics: Each PDA-SG induces a countable SG where vertices are configurations and transitions are determined naturally by transition rules:

- $pX\alpha \rightarrow q\beta\alpha$ (here $p, q \in Q$, $X \in \Gamma$, $\alpha, \beta \in \Gamma^*$) iff $pX \hookrightarrow q\beta$
- $p\varepsilon \rightarrow p\varepsilon$ for every $p \in Q$

If $pX \in H_P$, then $Prob(pX\alpha \rightarrow q\beta\alpha) = Prob(pX \hookrightarrow q\beta)$.

A configuration $pX\alpha$ is controlled by player Max (or Min, or the stochastic environment) if the head pX belongs to $H_\Box$ (or to $H_\Diamond$, or to H_P, respectively). Each configuration $p\varepsilon$ is owned by Max.

Intuitively, a run of a PDA-SG starts in some configuration and proceeds as follows: in a configuration of the form $pX\alpha$ a transition rule of the form $pX \hookrightarrow q\beta$ is chosen either by one of the players, or randomly according to *Prob* (depending on the head pX) and then the configuration changes to $q\beta\alpha$.

Objectives: PDA-SG have been considered mainly with reachability objectives. Of course, it is easy to show that reachability of an arbitrary (possibly non-recursive) set of configurations is undecidable, so the reachability problem is usually restricted to regular target sets, i.e., sets of configurations $q\alpha$ accepted by finite state automata with the input alphabet $Q \cup \Gamma$ (that usually read the configuration from right to left). By encoding the finite-state automaton accepting T into the stack symbols (see [43]), reachability of a regular set T can be reduced to so called *termination*, i.e., to reachability of any configuration with empty stack. Denote by *Term* the set of all runs that visit a configuration with empty stack.

Related work and known results

There is vast literature on (non-probabilistic) pushdown automata since it is the fundamental model of recursive programs used in theoretical computer science (see, e.g., [70] for an introduction to PDA, for verification of PDA see, e.g., [11, 43, 31]). Also, (non-stochastic) two player games on PDA have been extensively studied (see, e.g., [72, 2]). Probabilistic pushdown automata have been well researched in recent years (see, e.g., [42, 47]); the author of this thesis contributed to this topic with the papers [30, 26, 16, 19]. For a recent survey of modern results on pPDA see [23].

PDA-SG are semantically equivalent to recursive stochastic games studied by Etessami & Yannakakis in [45]. The following theorem summarizes, in a simplified form, the undecidability results for the termination objectives.

Theorem 3.0.2 ([45]). *Let pX be a head of a maximizing PDA-MDP* Δ. *For every* fixed *rational* ϵ *with* $0 < \epsilon < \frac{1}{2}$ *the following problems are undecidable:*

- *Is* $\mathrm{Val}(\mathit{Term}, pX) \geq 1 - \epsilon$, *or* $\mathrm{Val}(\mathit{Term}, pX) \leq \epsilon$ *?*
- *Assume that one of the following is true: Either* $\mathrm{Val}(\mathit{Term}, pX) = 1$ *and player Max has an optimal strategy, or* $\mathrm{Val}(\mathit{Term}, pX) \leq \epsilon$. *It is an undecidable problem to distinguish the two cases.*

Let pX be a head of a minimizing PDA-MDP Δ. *For every* fixed $r \in (0, 1]$, *the problem whether* $\mathrm{Val}(\mathit{Term}, pX) < r$ *is undecidable.*

Theorem 3.0.2 implies that both the quantitative and qualitative problems with the termination objective are undecidable and the value cannot be effectively approximated even for maximizing PDA-MDP (and thus also for PDA-SG in general).

The only problem for which positive results have been obtained so far is termination with positive probability. The reason is that this problem does not depend on exact transition probabilities and thus standard results for (non-stochastic) two player PDA games apply.

Theorem 3.0.3 ([45]). *Let pX be a head of a PDA-SG* Δ. *The problem whether* $\mathrm{Val}(\mathit{Term}, pX) = 0$ *belongs to* ***EXPTIME***. *This problem is* ***EXPTIME****-hard even in the special case when* $\mathcal{G}$ *is a* minimizing *PDA-MDP. When* $\mathcal{G}$ *is a* maximizing *MDP, the problem whether* $\mathrm{Val}(\mathit{Term}, pX) = 0$ *is decidable in polynomial time.*

Theorem 3.0.2 motivates our study of subclasses of PDA-SG where the reachability problem can be, at least partially, solved.

3.1 Stochastic BPA Games

BPA-SG form a subclass of PDA-SG without control states which means that the definition of BPA-SG can be simplified as follows.

Syntax: A *stochastic BPA game (BPA-SG)* Δ consists of

- a set Γ of (stack) *symbols* partitioned into symbols $\Gamma_\Box$ of player Max, symbols $\Gamma_\Diamond$ of player Min, and stochastic symbols Γ_P,
- a set of *transition rules* of the form $X \hookrightarrow \alpha$, where $X \in \Gamma$ and $\alpha \in \Gamma^*$; we assume that for every $X \in \Gamma$ there is a transition rule of the form $X \hookrightarrow \alpha$,
- a map *Prob* taking each rule $X \hookrightarrow \alpha$ with $X \in \Gamma_P$ to a positive rational number $Prob(X \hookrightarrow \alpha)$, so that for every $X \in \Gamma_P : \sum_{X \hookrightarrow \alpha} Prob(X \hookrightarrow \alpha) = 1$.

A configuration is a string of symbols usually denoted by $\alpha, \beta, \ldots$

Maximizing and minimizing BPA-MDP are defined analogously to maximizing and minizimizing PDA-MDP by removing players Min and Max, respectively. A *probabilistic BPA* is a BPA-SG without the players Max and Min.

Semantics: The dynamics of a BPA-SG follows from the dynamics of PDA-SG, i.e., in a configuration $X\alpha$ the leftmost symbol X is rewritten according to a transition rule chosen either by a player, or randomly. Formally, each BPA-SG induces a countable SG where the vertices are configurations and the transitions are determined as follows: $\varepsilon \to \varepsilon$, and $X\alpha \to \beta\alpha$ iff $X \hookrightarrow \beta$. If $X \in \Gamma_P$, then $Prob(X\alpha \to \beta\alpha) = Prob(X \hookrightarrow \beta)$. Which player controls a given configuration α depends on the leftmost symbol of α (ε is controlled by Max).

One of our aims is to study structure of optimal strategies. In the case of BPA-SG, we identify some special classes of strategies based on their treatment of stack contents. Namely, we say that a strategy is *stackless memoryless (SM)* if its decisions depend only on the top of the stack symbol of the current configuration. Decisions of a *regular* strategy are determined by the state of a fixed finite state automaton with the input alphabet Γ after reading the current stack content (note that stackless memoryless strategies are a very special case of regular strategies).

Objectives: As in the case of PDA-SG, we are mostly concerned with reachability of regular sets of configurations. As opposed to PDA-SG, reachability of a regular set cannot be easily reduced to termination (i.e., reachability of empty stack). However, one may easily prove that reachability of regular sets can be reduced to reachability of *simple* sets of configurations determined by top most symbols of configurations. More precisely, a simple set T determined by a set of heads $\hat{T} \subseteq \Gamma \cup \{\varepsilon\}$ consists of all configurations of the form $\beta\gamma$ where $\beta \in \hat{T}$ and either $\beta \neq \varepsilon$, or $\gamma = \varepsilon$. A finite-state automaton accepting a regular set of configurations T' can be efficiently encoded into stack symbols, which transforms T' to a simple set T. Thus for BPA-SG two problems are considered: the termination problem and reachability of a given simple set.

Recall that by *Term* we denote the set of all runs that visit the configuration with empty stack ε. We denote by $Reach(T)$, where T is a given simple set, the set of all runs that visit a configuration of T.

Related work and known results

Stateless PDA, called basic process algebras (BPA) for historical reasons, are a standard model in the hiearchy of process algebras (see [7]). Probabilistic BPA have been intensively studied in recent years, usually as a subclass of pPDA or recursive Markov chains, due to their close relationship with branching processes and stochastic grammars (for details on this relationship see [48], for a recent survey of the most important results see [23]). They also allow an efficient analysis using non-trivial numerical methods (see, e.g., [48, 23]). BPA-SG are semantically equivalent to one-exit recursive stochastic games studied in [45, 46] where the following theorem is proved.

Theorem 3.1.1 ([45, 46])**.** *Let Δ be a BPA-SG. Both players have SMD strategies that are optimal in every configuration of Δ. Let X be a symbol of Δ, let $\sim \in \{>, \geq\}$ and let $r \in [0,1]$ be a rational constant.*

- *The problem whether $\mathrm{Val}(\mathit{Term}, X) \sim r$ belongs to **PSPACE**. Moreover, the SQUARE-ROOT-SUM* [1] *problem is polynomially reducible to this problem.*
- *The problem whether $\mathrm{Val}(\mathit{Term}, X) = 0$ is solvable in polynomial time, the problem whether $\mathrm{Val}(\mathit{Term}, X) = 1$ belongs to **NP** $\cap$ **co-NP**.*
- *In the special case when Δ is either a maximizing, or a minimizing BPA-MDP, the problem whether $\mathrm{Val}(\mathit{Term}, X) = 0$ and the problem whether $\mathrm{Val}(\mathit{Term}, X) = 1$ are both decidable in polynomial time.*

The papers [45, 46] also present algorithms for computing optimal strategies. The proof of Theorem 3.1.1 relies on expressing the value of the BPA-SG using a system of non-linear min/max equations. Resolving the min/max parts of the equations determines SMD optimal strategies. For details see [45, 46].

3.1.1 The reachability problem for BPA-SG

In [18] we study BPA-MDP and in [21] BPA-SG with reachability objectives. It turns out that reachability is very different from termination. Let us start with existence of optimal strategies.

Let Δ be a BPA-SG, let X be a symbol of Δ and let $\mathit{Reach}(T)$ be a reachability objective where T is a *simple* set. Example 6 of [18] proves the following.

- Player Max does not necessarily have an optimal strategy in X even if Δ is a maximizing BPA-MDP and $\mathrm{Val}(\mathit{Reach}(T), X) = 1$. Moreover, existence of an optimal strategy for Max in X together with $\mathrm{Val}(\mathit{Reach}(T), X) = 1$ do not necessarily imply existence of a SMD optimal strategy for Max in X.
- SMD strategies may be strictly weaker than MD strategies even though Δ is a minimizing BPA-MDP. In fact, there is a minimizing BPA-MDP Δ in which the following holds:

$$\frac{1}{2} = \mathrm{Val}(\mathit{Term}, X) = \inf_{\pi \in MD} \mathbb{P}^{\pi}_{X}(\mathit{Reach}(T)) < \inf_{\pi \in SMD} \mathbb{P}^{\pi}_{X}(\mathit{Reach}(T)) = 1$$

[1]An instance of SQUARE-ROOT-SUM is a tuple of positive integers $a_1, \ldots, a_n, b$, and the question is whether $\sum_{i=1}^{n} \sqrt{a_i} \leq b$. The problem is in **PSPACE**, and the best upper bound currently known is **CH** (counting hierarchy; see Corollary 1.4 in [1]). It is not known whether this bound can be further lowered to some natural Boolean subclass of **PSPACE**, and a progress in answering this question might lead to breakthrough results in complexity theory.

For seemingly fundamental reasons, the value of the game cannot be easily expressed using a system of non-linear equations as for termination. So most problems concerning quantitative reachability are still open for BPA-SG.

In [18, 21] we concentrate on *qualitative* constraints on reachability. That is, we ask whether player Max can reach T with probability equal to one (or greater than zero). The following two theorems summarize the most important "positive" results of both [18, 21].

Given a set of configurations T and ${\sim}r \in \{=1, >0\}$, we denote by $W_{T,\sim r}$ the set of all configurations in which Max has a $(\mathit{Reach}(T), {\sim}r)$-winning strategy.

Theorem 3.1.2 ([18, 21]). *Let Δ be a BPA-SG, let T be a* simple *set of configurations, and let ${\sim}r \in \{=1, >0\}$.*

- *Each set $W_{T,\sim r}$ is regular and, moreover, the number of states of the corresponding automaton is independent of the size of the game.*

  (see Proposition 5.3 and Proposition 6.10 of [21])

- *The membership to $W_{T,=1}$ is in **NP**$\cap$**co-NP** and a finite-state automaton accepting $W_{T,=1}$ is computable in polynomial time with **NP** $\cap$ **co-NP** oracle. In the special case when Δ is either a maximizing, or a minimizing BPA-MDP, the automaton is computable in polynomial time.*

  (see Theorem 6.10 of [21] for BPA-SG and Theorems 8, 9, 10 and 11 of [18] for BPA-MDP)

- *A finite-state automaton accepting $W_{T,>0}$ is computable in polynomial time.*

  (see Theorem 5.3 of [21])

The regular sets of configurations obtained in Theorem 3.1.2 are actually rather simple. They may be described as follows:

- $W_{T,>0} = \mathcal{B}^*\mathcal{A}\Gamma^*$ where $\mathcal{B} = \Gamma \cap W_{T\cup\{\varepsilon\},>0}$ and $\mathcal{A} = \Gamma \cap W_{T,>0}$. Intuitively, in symbols of $\mathcal{B}$, player Max may either force reaching T or termination with positive probability, in symbols of $\mathcal{A}$, player Max may force reaching T with positive probability. The sets $\mathcal{B}$ and $\mathcal{A}$ can be computed using a relatively straightforward fixed-point algorithm.

- $W_{T,=1} = \mathcal{D}^*\mathcal{C}\Gamma$ where $\mathcal{D} = \Gamma \cap W_{T\cup\{\varepsilon\},=1}$ and $\mathcal{C} = \Gamma \cap W_{T,=1}$. Computing the sets $\mathcal{D}$ and $\mathcal{C}$ is the most intricate part of the solution (see [21]).

(Note that the winning sets for Min are complements of the above winning sets for Max due to Theorem 2.1.4 and hence are also effectively regular.)

Theorem 3.1.3 ([18, 21]). *Assume the same as in the previous theorem.*

1. *There is a regular strategy σ for Max and a regular strategy π for Min such that σ is $(\mathit{Reach}(T), =1)$-winning for Max in all configurations of $W_{T,=1}$ and π is $(\mathit{Reach}(T), =1)$-winning for Min in all configurations of $\Gamma^* \setminus W_{T,=1}$. These strategies are computable in polynomial time with **NP** $\cap$ **co-NP** oracle. (For BPA-MDP they are computable in polynomial time.)*

   (see Theorem 6.11 of [21], the claim for BPA-MDP is not explicitly stated in [21] but follows from the proofs)

2. *There is a regular strategy σ for Max and a SMD strategy π for Min such that σ is $(Reach(T), >0)$-winning for Max in all configurations of $W_{T,>0}$, and π is $(Reach(T), >0)$-winning for Min in all configurations of $\Gamma^* \setminus W_{T,>0}$. These strategies are computable in polynomial time.*

   (see Theorem 5.3 of [21])

Remark 3.1.4. *In [18] we consider a bit more general version of reachability objectives, called extended reachability objectives, with safety restrictions on paths reaching T. The main reason for using more general objectives in [18] was to deal with qualitative PCTL objectives in exponential time. However, extended reachability objectives can be easily reduced to the reachability objectives in polynomial time. For details see [18].*

3.2 Stochastic One-Counter Games

Each OC-SG can be seen as a PDA-SG with just one stack symbol. However, for technical reasons we give a bit more general definition allowing negative counter values and use a slightly different notation.

Syntax: A *stochastic one-counter game (OC-SG)* Δ consists of

- a set of control states Q, partitioned into states $Q_\Box$ of player Max, states $Q_\Diamond$ of player Min, and stochastic states Q_P,
- a set of *transition rules* $\delta \subseteq Q \times \{-1, 0, +1\} \times Q$; we assume that for every $p \in Q$ there is at least one transition rule of the form $(p, d, q) \in \delta$,
- a map *Prob* taking each rule $(p, d, q) \in \delta$ with $p \in Q_P$ to a positive rational number $Prob(p, d, q)$, so that for every $p \in Q_P : \sum_{(p,d,q)\in\delta} Prob(p, d, q) = 1$.

A configuration is a pair (p, i) where $p \in Q$ and $i \in \mathbb{Z}$.

Subclasses of maximizing and minimizing OC-MDP are defined as for PDA-MDP by removing players Min and Max, respectively.

Semantics: Each OC-SG induces a countable SG where the vertices are OC-SG configurations and the transitions are determined as follows: $(p, i) \rightarrow (q, i + d)$ iff $(p, d, q) \in \delta$. If $p \in Q_P$, then $Prob((p, i) \rightarrow (q, i + d)) = Prob(p, d, q)$. Which player controls a given configuration (p, i) is determined by the control state p.

Intuitively, a run starts in some configuration. In a configuration of the form (p, i) a transition rule of the form (p, d, q) is either chosen by one of the players, or randomly (depending on the state p) and then the configuration changes to $(q, i + d)$. This defines an infinite run on configurations of the OC-SG.

As for BPA-SG, we identify special classes of strategies for OC-SG based on their treatment of the counter. We say that a strategy is *counterless* if its decisions depend only on the control state of the current configuration. Decisions of a *counter-regular* strategy in a configuration of the form (p, i) are determined by the state of a fixed finite-state automaton with a single letter input alphabet $\{a\}$ after reading the word a^i, i.e., the string of a's of length i. The size of the counter-regular strategy is equal to the size of this automaton.

We also define a special class of strategies with finite memory. Concretely, decisions of a *finite-memory* strategy are determined by the state of a fixed finite-state automaton with the input alphabet Q after reading the sequence of all control states visited in the history. The size of the finite-memory strategy is equal to the size of this automaton.

Objectives: The main goal of [15, 12, 13] is to study termination objectives [2]. As opposed to PDA-SG, we distinguish two ways in which OC-SG may terminate:

- The *selective termination* objective: Given $F \subseteq Q$, we define $\mathit{Term}(F)$ to be the set of all runs that reach a configuration of the form $(q, 0)$ for some $q \in F$ before reaching any configuration with a non-positive counter value.
- The *non-selective termination* objective is defined by $\mathit{Term} := \mathit{Term}(Q)$.

Related work and known results

There is a huge literature on verification of one-counter automata, that is (non-stochastic) one-player OC-SG, see, e.g., [53, 71, 56]. (Non-stochastic) two-player one-counter games have also been studied (see, e.g., [69, 27]). OC Markov chains, i.e., OC-SG without players, have been considered in [44] (under the name quasi-birth-death processes) where efficient approximation algorithms are proposed for computing termination probabilities. Also, as we show, there is a very tight connection between OC-SG and stochastic mean payoff games (see, e.g., [50]). OC-SG can also be seen as a stochastic variant of energy games with one resource (see, e.g., [34, 49]). In particular, our research on OC-SG is partially motivated by modeling energy with the counter.

3.2.1 Non-selective termination for OC-SG

Let us start with existence of optimal strategies for non-selective termination. It turns out that the termination objective is much more intricate for OC-SG than for BPA-SG. Let Δ be an OC-SG, let $v = (p, i)$ be a configuration of Δ and consider the non-selective termination objective *Term*.

- Player Max does not necessarily have an optimal strategy in v even if Δ is a maximizing OC-MDP (see Example A.1 of [13]).
- Even though player Min always has an optimal MD strategy in v, there does not necessarily exist a counterless optimal strategy for Min in v even if Δ is a minimizing OC-MDP. Actually, the structure of optimal strategies for Min may be rather complicated. In particular, as demonstrated by Theorem 3.7 of [6] for a very a special subclass of OC-SG called solvency games [3], Min does not necessarily have a so-called "rich person's" strategy, i.e., a strategy which ignores the precise counter value whenever the value is larger than some threshold. Moreover, no regularity in the choice of transitions w.r.t. the counter value as well as the history has been observed so far.

[2]Note that in the case of OC-SG, reachability of a simple set of configurations can be easily reduced to termination. This is not always true for reachability of *counter-regular* sets which usually calls for some extension of techniques used to solve termination.

[3]As noted in [13], solvency games correspond to OC-MDP where there is only one control state, but there are multiple actions that change the counter value, possibly by more than one per transition, according to a finite-support probability distribution on integers associated with each action. It is not hard to show that these are subsumed by minimizing OC-MDP.

Due to these reasons we concentrate only on

- a qualitative version of the non-selective termination objective,
- an effective *approximation* of the value and on the computation of ε-optimal strategies for a given rational $\varepsilon > 0$.

Solving qualitative non-selective termination for OC-SG

We concentrate only on termination with probability one since termination with positive probability does not depend on exact transition probabilities and thus can be solved using methods for non-stochastic one counter games (see, e.g., [69, 27]). This objective was first considered in [15] but only for maximizing OC-MDP. Later, in [12], the results have been extended to OC-SG. Proposition 15 of [12] summarizes most of the known results [4].

Theorem 3.2.1 ([12], Proposition 15)**.** *Let $v = (p, i)$ be a configuration of an OC-SG Δ.*

1. *If $\mathrm{Val}(\mathit{Term}, v) = 1$, then Max has a deterministic counterless strategy σ which is optimal in v.*
2. *If $\mathrm{Val}(\mathit{Term}, v) < 1$, then Min has a deterministic finite-memory strategy π of size $O(|Q|)$ such that $\sup_\sigma \mathbb{P}_v^{\sigma,\pi}(\mathit{Term}) < 1$.*

*The strategies σ and π are computable in polynomial time using **NP** $\cap$ **co-NP** oracle. If Δ is a maximizing or a minimizing OC-MDP, these strategies are computable in polynomial time.*

The strategy π of Theorem 3.2.1 needs memory in general and is not necessarily optimal. As an immediate corollary to Theorem 3.2.1 and Corollary 16 of [12] we obtain the following.

Corollary 3.2.2. *The problem whether $\mathrm{Val}(\mathit{Term}, v) = 1$ belongs to **NP** $\cap$ **co-NP** and is at least as hard as solving Condon's [37] simple stochastic games [5]. For maximizing or minimizing OC-MDP the problem belongs to **P**.*

Now let us give an outline of the proof of Theorem 3.2.1. We proceed by reduction to other types of objectives. In particular, we consider limit objectives where the goal is to make the counter diverge to infinity and mean payoff objectives where the goal is to control the average change of the counter value. For both limit and mean payoff objectives we provide a complete game theoretic solution that may be of independent interest (see Theorem 3.2.3 below).

Limit and mean payoff objectives. In [12], we show that for large counter values the non-selective termination is in a sense equivalent to the question whether the counter value reaches all negative values, and non-termination is equivalent to divergence to $+\infty$. This intuition is formalized using the following limit objectives.

Given a run $\mathbf{w}$ and $n \geq 1$, we denote by $C^{(n)}(\mathbf{w})$ the value of the counter in the n-th configuration of $\mathbf{w}$.

[4] Note that the complexity bounds for general OC-SG are not explicitly stated in Proposition 15 of [12]. However, they can be easily obtained using similar trick as in the proof of Lemma 6.4 of [21].

[5] This is just a fancy name for the problem of deciding whether the value of a given SG with finitely many vertices is greater than a given rational number.

- The *CoverNeg* objective:

$$\mathit{LimInf}(=-\infty) := \{\mathbf{w} \text{ is a run} \mid \liminf_{n\to\infty} C^{(n)}(\mathbf{w}) = -\infty\}$$

  Intuitively, a run belongs to *LimInf*$(=-\infty)$ iff the counter value eventually gets below any fixed negative value (i.e., the run "covers" all negative numbers).

- The *divergence* objective [6]:

$$\mathit{LimInf}(=+\infty) := \{\mathbf{w} \text{ is a run} \mid \liminf_{n\to\infty} C^{(n)}(\mathbf{w}) = \infty\}$$

  Intuitively, a run belongs to *LimInf*$(=+\infty)$ iff the counter value diverges to infinity.

We show that the divergence objective is equivalent to the positive long-run average change of the counter value (similarly, the CoverNeg objective is closely related to the non-positive long-run average change of the counter value) which gets us closer to the classical mean payoff objectives for finite MDP as studied, e.g., in [65].

- The *positive mean payoff* objective [7]:

$$\mathit{MeanInf}(>0) := \{\mathbf{w} \text{ is a run} \mid \liminf_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} C^{(i)}(\mathbf{w}) - C^{(i-1)}(\mathbf{w}) \ > \ 0\}$$

  Intuitively, this objective is satisfied by runs with positive long-run average change of the counter value.

Observe that the mean payoff objectives do not take into account the exact counter value and thus can be considered as objectives for SG played on control states of the OC-SG. Further, note that our mean payoff objectives differ from the *expected* mean payoff studied in the literature. In our case player Max maximizes the probability that the mean payoff satisfies a given constraint, the classical objective is to maximize the *expectation* of the mean payoff.

Theorem 3.2.3 ([12]). *Let $v = (p, i)$ be a configuration of an OC-SG Δ and let O be an arbitrary limit or mean payoff objective.*

1. *The problem whether* $\mathrm{Val}(O, v) \geq p$ *for a given rational* $p \in [0, 1]$ *belongs to* ***NP*** $\cap$ ***co-NP****. On the other hand, even assuming that* $\mathrm{Val}(O, v) \in \{0, 1\}$*, the problem of deciding whether* $\mathrm{Val}(O, v) = 1$ *is at least as hard as solving the Condon's simple stochastic games. If Δ is an OC-MDP, then* $\mathrm{Val}(O, v)$ *can be computed in polynomial time.*

2. *Both players have deterministic counterless strategies that are optimal in v. If Δ is an OC-MDP, these strategies are computable in polynomial time.*

(For both 1. and 2. see Theorem 2 in the introduction of [12] and the discussion afterwards.)

[6]Both *LimInf*$(=-\infty)$ and *LimInf*$(=+\infty)$ are special cases of the following limit objectives *LimInf*$(\sim \ominus\infty) := \{\mathbf{w} \mid \liminf_{n\to\infty} C^{(n)}(\mathbf{w}) \sim \ominus\infty\}$ and *LimSup*$(\sim \ominus\infty) := \{\mathbf{w} \mid \limsup_{n\to\infty} C^{(n)}(\mathbf{w}) \sim \ominus\infty\}$. Here $\sim \in \{<, >, =\}$ and $\ominus \in \{+, -\}$. However, other combinations of $\sim$ and $\ominus$ are redundant as they might be obtained from *LimInf*$(=-\infty)$ and *LimInf*$(=+\infty)$ by negating signs in transition rules and changing roles of players. (For example, by negating the signs in transition rules, *LimSup*$(=-\infty)$ is "equivalent" to *LimInf*$(=+\infty)$.)

[7]*MeanInf*(>0) is a special case of *MeanInf*$(\sim r) := \{\mathbf{w} \mid \liminf_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} C^{(i)}(\mathbf{w}) - C^{(i-1)}(\mathbf{w}) \ \sim \ r\}$ and *MeanSup*$(\sim r) := \{\mathbf{w} \mid \limsup_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} C^{(i)}(\mathbf{w}) - C^{(i-1)}(\mathbf{w}) \ \sim \ r\}$. Here $\sim \in \{<, >, \leq, \geq\}$ and $r \in \mathbb{R}$. In fact, all objectives except *MeanInf*(>0) are redundant, as they can be obtained from *MeanInf*(>0) by negating signs, subtracting r from the counter updates in transition rules and by changing the roles of players.

Approximating the non-selective termination value for OC-SG

As we noted at the beginning of this section, optimal strategies for player Max do not necessarily exist, and optimal strategies for player Min may be very complicated. Also, as noted before, the value may be irrational. Nevertheless, in [13], we show that the value can be effectively approximated and that ε-optimal strategies can be computed for every rational $\varepsilon > 0$. More precisely, we prove the following.

Theorem 3.2.4 ([13], Theorem 3.1)**.** *There is an algorithm that, given as input: an OC-SG Δ, a configuration $v = (p, i)$ and a (rational) approximation threshold $\varepsilon > 0$, computes a rational number r, such that $|r - \mathrm{Val}(\mathit{Term}, v)| < \varepsilon$ and computes counter-regular deterministic strategies for both players that are ε-optimal in v. If Δ is an OC-MDP, the algorithm runs in exponential time in the size of Δ, and in polynomial time in $log(1/\varepsilon)$ and $log(i)$. If Δ is an OC-SG, the algorithm runs in non-deterministic exponential time in the size of Δ.*

The idea of the proof is to show that there is a threshold t, such that for every configuration (q, i) where $i \geq t$ the probability of reaching zero counter value is either very small, or one. Intuitively, it suffices to observe the long-run average change of the counter value when using an approximately optimal strategy. If it is positive, the probability of reaching zero is very small once we start with a sufficiently large counter value. Non-positive average change means that almost all runs eventually reach zero. This is formally justified using a bit more advanced tools of probability theory such as martingales. There are also several technical problems caused by the rich structure of OC-SG that have to be dealt with.

3.2.2 Selective termination for OC-SG

Selective termination has been considered only in [15] and thus only for maximizing OC-MDP and only in the qualitative version. This objective is even more difficult to deal with than the non-selective one.

Let $v = (q, i)$ be a configuration of an OC-MDP Δ. Even if Δ is a maximizing MDP and $\mathrm{Val}(\mathit{Term}(F), v) = 1$, player Max does not necessarily have an optimal strategy in v (as opposed to the non-selective case where Max has a deterministic counterless optimal strategy whenever the value is one). This implies that in the selective case the problem whether $\mathrm{Val}(\mathit{Term}(F), v) = 1$ does *not* coincide with existence of a $(\mathit{Term}(F), {=}1)$-winning strategy in v.

Decidability of whether $\mathrm{Val}(\mathit{Term}(F), v) = 1$ for a given v is still open even for OC-MDP (in fact, this problem for OC-MDP has been shown to be **BH**-hard in [15] which has been further improved to **PSPACE**-hardness in [52]).

Concerning existence of $(\mathit{Term}(F), {=}1)$-winning strategies, [15] proves the following.

Theorem 3.2.5 ([15], Theorem 4.2)**.** *Let Δ be a maximizing OC-MDP and let $\mathit{Term}(F)$ be a selective termination objective. Denote by $W_{F,=1}$ the set of all configurations in which Max has a $(\mathit{Term}(F), {=}1)$-winning strategy.*

1. *The set $W_{F,=1}$ is regular* [8] *and the automaton accepting $W_{F,=1}$ is computable in exponential time.*

2. *There is a counter-regular strategy for Max, computable in exponential time, which is optimal in all configurations of $W_{F,=1}$.*

[8]Regularity of a set of configurations is defined analogously to the counter-regular strategies, i.e., there must be a finite-state automaton with a one letter input alphabet for each control state p which accepts a^i iff (p, i) belongs to the set.

The proof of the above theorem is based on a reduction to the non-selective case. Intuitively, the regularity of $W_{F,=1}$ follows from the periodicity of $W_{F,=1}$, that is from the fact that for each control state p there are $k, \ell \in \mathbb{N}$, computable in exponential time, such that for all $i, j \geq k$ satisfying $i \pmod{\ell} = j \pmod{\ell}$ we have $(q, i) \in W_{F,=1}$ iff $(q, j) \in W_{F,=1}$. Reduction to the non-selective termination then, roughly speaking, proceeds by encoding the regular structure of the counter into the control states. (Note that combining this idea, formally developed in [15] with results of [12] on the non-selective termination for OC-SG, one may obtain a yet unpublished solution to OC-SG.)

The paper [15] also provides the following lower bound.

Theorem 3.2.6 ([15], Theorem 4.3)**.** *The membership to $W_{F,=1}$ is* ***PSPACE****-hard.*

Chapter 4

Games with Multiple Counters

This chapter presents results of two papers [27, 22]. In Section 4.1, we introduce eVASS games and then, in Section 4.1.1, outline the most important results of [27] about their algorithmic analysis. Later, in Section 4.2, we present consumption games, a subclass of eVASS games studied in [22], and show that they can be efficiently analyzed.

4.1 eVASS Games

Syntax: A *k-dimensional eVASS game* consists of

- a finite set of *control states* Q, together with a partition into the states $Q_\Box$ of player Max and states $Q_\Diamond$ of player Min,
- a finite set of *transitions* T, and two functions $\alpha : T \to Q$ and $\beta : T \to Q$ mapping each transition to its source and target state, respectively; we assume that every state is a source of at least one transition,
- a *transition displacement* mapping $\delta : T \to (\mathbb{Z} \cup \{\omega\})^k$ which maps each transition to its displacement vector.

A *configuration* is an element of $Q \times \mathbb{N}^k$. We write $p\vec{v}$ to denote a generic configuration where $p \in Q$ and $\vec{v} \in \mathbb{N}^k$. (In what follows we denote by $\vec{v}_\ell$ the ℓ-th component of a given vector $\vec{v}$.)

A *k-dimensional VASS game* is a k-dimensional eVASS game where ω components are not allowed in displacement vectors.

Semantics: A k-dimensional eVASS game induces a two-player game whose states are configurations of $Q \times \mathbb{N}^k$, partitioned into sets $Q_\Box \times \mathbb{N}^k$ and $Q_\Diamond \times \mathbb{N}^k$ controlled by players Max and Min, respectively. The transition relation of the two-player game is naturally induced by transitions of the eVASS game: $p\vec{v} \to q\vec{u}$ iff there is $t \in T$ such that

- $p = \alpha(t)$ and $q = \beta(t)$
- for every $1 \leq \ell \leq k$ we have that $\vec{u}_\ell - \vec{v}_\ell$ is either an (arbitrary) non-negative integer, or equal to $\delta(t)_\ell$ depending on whether $\delta(t)_\ell = \omega$, or not.

Intuitively, a run of such a game starts in some configuration. In a configuration of the form $p\vec{v}$, the player controlling $p\vec{v}$ chooses

- a transition $t \in T$ satisfying $\alpha(t) = p$,
- a vector $\vec{d}$ such that for every $1 \leq \ell \leq k$ we have $\vec{v}_\ell + \vec{d}_\ell \geq 0$ and either $\vec{d}_\ell \geq 0$, or $\vec{d}_\ell = \delta(t)_\ell$ depending on whether $\delta(t)_\ell = \omega$, or not.

Then the configuration changes to $q\vec{u}$ where $q = \beta(t)$ and $\vec{u} = \vec{v} + \vec{d}$. This produces an infinite run on configurations of $Q \times \mathbb{N}^k$.

Objectives: eVASS games generate two-player games that are technically a special case of simple stochastic games studied in Chapter 2. So the objectives remain to be sets of runs. Given an objective O, we say that a strategy of player Max (or of player Min) is *O-winning* if player Max (or player Min) has a strategy which induces a run of O no matter what player Min (or player Max, resp.) does.

The objective which we consider for eVASS games, and which is intuitively described in the introduction, is in fact a dual of a multicounter generalization of the termination objective. More precisely, the objective of Max is to *avoid* configurations with zero in the counters, which is a kind of a *safety* objective. As for the termination of one-counter games, we consider two variants, the selective as well as the non-selective one.

- The *selective zero-safety* objective: Given $F \subseteq Q$ define $\mathit{SafeObj}(F)$ to be the set of all runs that *never* reach a configuration of the form $q\vec{v}$ where $q \in F$ and $\vec{v}_i = 0$ for some $1 \leq i \leq k$.
- The *non-selective zero-safety* objective is defined by $\mathit{SafeObj} := \mathit{SafeObj}(Q)$; that is $\mathit{SafeObj}$ contains all runs where no counter is decreased to zero.

Note that if $\mathit{SafeObj}(F)$ is the objective of player Max, the objective of player Min is a variant of selective termination, that is Min strives to reach a configuration with zero in some counter and with the control state in F.

We denote by $\mathit{Win}_\Box(F)$ and $\mathit{Win}_\Diamond(F)$ the sets of configurations of $Q \times \mathbb{N}^k$ in which players Max and Min, respectively, have $\mathit{SafeObj}(F)$-winning strategies (we omit the argument F in the non-selective case).

Related work and known results

One-player VASS games correspond to Petri nets, an extensively studied and widely applied modeling formalism (see, e.g., [64, 67]). A solution of one-player VASS games with zero-safety objectives belongs among classical results [66, 68]. VASS games can also be seen as a special case of monotonic games considered in [3] where it is shown that monotonic games are undecidable (as opposed to eVASS games).

VASS games are also closely related to multi-energy games (see [34, 49], both papers have been published after [27]). Some problems for the multi-energy games can be reduced to related problems for eVASS games. (For a more extensive survey of more loosely related abstract formalisms see the "related work" parts of [27, 22].)

4.1.1 Solving eVASS games with zero-safety objectives

In this subsection we present the main results of [27]. We start with undecidability of the selective case and then concentrate on complexity of the non-selective one.

In [27], the type of the transition displacement mapping δ is restricted [1] to $\delta : T \to \{-1, 0, 1, \omega\}^k$. Note that this restriction does not affect decidability of studied problems as any update $d \in \mathbb{Z}$ may be simulated by $|d|$ individual steps changing the counter one by one. However, some complexity results may be altered if binary encoded transition displacement mappings are considered. In the rest of Section 4.1.1 we assume that $\delta : T \to \{-1, 0, 1, \omega\}^k$.

In order to be able to decide a winner of a game, we need the following determinacy of eVASS games which follows from the Martin's determinacy theorem [59].

Proposition 4.1.1. $Q \times \mathbb{N}^k = Win_\Box(F) \cup Win_\Diamond(F)$

Undecidability of selective zero-safety

The following undecidability result is presented in Section 3 of [27] (as Proposition 4) using a relatively straightforward reduction from the halting problem for two-counter machines.

Proposition 4.1.2 ([27], Proposition 4)**.** *The problem of deciding the winner in* 2*-dimensional VASS games with the* selective *zero-safety objectives is undecidable. For* 3*-dimensional eVASS games, the same problem is highly undecidable (i.e., beyond the arithmetical hierarchy).*

Complexity of non-selective zero-safety

Let us concentrate on the non-selective zero-safety objective. Note that most of our motivation with resource consumption is preserved and, in addition, we obtain games that can be effectively solved.

Observe that now the set $Win_\Box$ is upward closed, i.e., given $p\vec{v}$ in $Win_\Box$ we have $p\vec{u} \in Win_\Box$ for every $\vec{u} \geq \vec{v}$. Similarly, $Win_\Diamond$ is downward closed. It follows from Dickson's lemma [41] that $Win_\Box$ can be finitely represented by the finite set of its minimal elements denoted by $Min_\Box$. The main result of [27], proved as Theorem 10, is the following.

Theorem 4.1.3 ([27], Theorem 10)**.** *Given a k-dimensional eVASS game, the set $Min_\Box$ is computable in* $(k-1)$*-exponential time.*

Very rough idea of the proof is following: First, we show that if *all* counter values are larger than some threshold (exponential in the size of the game), then the winning strategies do not depend on the counters, i.e., for $p\vec{v} \in Win_\Box$ there is a winning strategy for Max in $p\vec{v}$ whose decisions depend only on control states. The same holds for $p\vec{v} \in Win_\Diamond$ and player Min. Subsequently, we consider configurations in which some of the counters may be "small", i.e., below the threshold. The idea is to encode the values of the "small" counters into control states and thus reduce to the previous problem. This is, of course, very rough idea, there are several obstacles to overcome. In particular, the "small" counters should be added gradually, the proof proceeds by induction on the number of counters, which together with the exponential threshold leads to the $(k-1)$-exponential upper bound.

As a corollary of Theorem 4.1.3 we obtain that 2-dimensional eVASS games are solvable in (singly) exponential time. This result has been recently improved in [32], using rather involved arguments, to the following form.

Theorem 4.1.4 ([32])**.** *Given a* 2*-dimensional eVASS game, the set $Min_\Box$ is computable in polynomial time.*

[1] We define eVASS games more generally in this survey to obtain the consumption games as a special case.

4.2 Consumption Games

A *k-dimensional consumption game* [2] is a k-dimensional eVASS game where the transition displacement mapping δ is of the type $\delta : T \to \left(\mathbb{Z}^{\leq 0} \cup \{\omega\}\right)^k$, here $\mathbb{Z}^{\leq 0}$ is the set of all *non-positive* whole numbers.

In other words, in consumption games the counters (in this context called *resources*) can only be increased (we say *reloaded*) using the ω updates. Intuitively, this models systems where resources are consumed but sometimes can be immediately reloaded to maximum (such as gas in cars). See [22] for description of applications and concrete examples.

Objectives: For consumption games we consider two objectives. The first one is precisely the non-selective zero-safety objective for eVASS games. The other one, the cover objective, is motivated by minimization of sufficient battery capacities (or capacities of other resource reservoirs).

- The *zero-safety* objective: Define *SafeObj* to be the set of all runs that never reach a configuration of the form $q\vec{v}$, where $\vec{v}_i = 0$ for some $1 \leq i \leq k$.
- The *cover* objective: Define *CoverObj* to be the set of all runs **w** of *SafeObj* satisfying the following condition: Assuming that $q\vec{u}$ is the initial configuration of **w**, every configuration $p\vec{v}$ visited by **w** satisfies $\vec{v} \leq \vec{u}$.

  (Intuitively, we start with resources loaded to the maximal capacity. Players may use ω to increase the corresponding counter only up to the initial value.)

Given a control state p, we denote by *Safe*(p) (or *Cover*(p)) the set of all vectors $\vec{v}$ such that in $p\vec{v}$ player Max has a *SafeObj*-winning strategy (or *CoverObj*-winning strategy, resp.) Concerning the sets *Safe*(p) and *Cover*(p) we deal with the following problems:

1. *Emptiness:* Decide whether the set is non-empty.
2. *Membership:* Decide whether a given vector $\vec{v}$ belongs to the set. Further, decide whether $\vec{v}$ is a minimal vector of the set.
3. *Minimal vectors:* Compute all minimal elements of the set.

Theorem 4.2.1 ([22]). *Let p be a control state of a k-dimensional consumption game with n control states. Denote by ℓ the maximal $|\delta(t)_i|$ over all transitions t and all $1 \leq i \leq k$ such that $\delta(t)_i \in \mathbb{Z}^{\leq 0}$ (i.e., ℓ is the largest decrease of a counter possible in one step).*

1. *The emptiness problem for Safe(p) is **co-NP**-complete and solvable in $O(k! \cdot n^{k+1})$ time.*
2. *Both the membership problem, i.e., the problem whether $\vec{v} \in$ Safe(p), as well as the problem whether $\vec{v}$ is a minimal vector of Safe(p) are **PSPACE**-hard and solvable in time $|\vec{v}| \cdot (k \cdot \ell \cdot n)^{O(k)}$ where $|\vec{v}|$ is the encoding size of $\vec{v}$.*
3. *The set of minimal vectors of Safe(p) is computable in time $(k \cdot \ell \cdot n)^{O(k)}$.*

(For 1. see Theorem 5. of [22]. For 2. and 3. see Theorem 10 of [22].)

[2]The definition of consumption games used in [22] is syntactically different from the one presented in this text but the two definitions are fully equivalent (in a well-defined sense) and there are linear-time translations between them.

Theorem 4.2.2 ([22])**.** *Assume the same as in the previous theorem.*

1. *The emptiness problem for Cover(p) is* ***co-NP****-complete and solvable in $O(k! \cdot n^{k+1})$ time.*
2. *The membership problem, i.e., the problem whether $\vec{v} \in$ Cover(p), is* ***PSPACE****-hard and solvable in time $O(\Lambda^2 \cdot n^2)$ where $\Lambda = \Pi_{i=1}^{k} \vec{v}_i$. The problem whether $\vec{v}$ is a minimal element of Cover(p) is* ***PSPACE****-hard and solvable in time $O(k \cdot \Lambda^2 \cdot n^2)$.*
3. *The set of minimal vectors of Cover(p) is computable in time $(k \cdot \ell \cdot n)^{O(k \cdot k!)}$.*

(For 1. see Theorem 5. of [22]. For 2. and 3. see Theorem 12 of [22].)

Note that for a fixed dimension k all the above problems are solvable in polynomial time. This means a radical improvement against the known complexity bounds for general eVASS games presented in Theorem 4.1.3. Also, in [22] we provide incremental algorithms for solving some special cases of consumption games. These algorithms may stop much earlier than predicted by the theoretical time bound. These special cases are one-player games and so-called *decreasing* games in which every resource must be consumed (i.e., decreased) on every cycle in control states of the game. The decreasing games are still well motivated as most resources are steadily consumed and it is usually not possible to avoid consummation by cycling among some states. This observation together with the fast incremental algorithms further improves applicability of our results.

Chapter 5

Task Systems and CBQN

In this chapter we present results of two papers, [25, 28]. We start, in Section 5.1, with the (discrete-time) task systems of [25]. Then, in Section 5.2, we move on to the (continuous-time) controlled branching queueing networks of [28].

5.1 Scheduling of Stochastically Generated Tasks

As noted in Section 1, we consider systems of tasks where every task can stochastically generate a set of new subtasks. Tasks can be of different types and each type has a fixed probability distribution on (types of) newly generated subtasks.

Syntax: A *task system (TS)* consists of

- a finite set Γ of *task types*,
- a finite set of *transition rules* of the form $X \hookrightarrow \beta$, where $X \in \Gamma$ and β is a multiset [1] of task types from Γ containing at most two elements; we assume that for every X there is at least one rule of the form $X \hookrightarrow \beta$,
- a map *Prob* taking each transition rule $X \hookrightarrow \beta$ to a positive rational number $\mathit{Prob}(X \hookrightarrow \beta)$, so that for every $X \in \Gamma : \sum_{X \hookrightarrow \beta} \mathit{Prob}(X \hookrightarrow \beta) = 1$,
- an initial task type X_0.

Example 5.1.1. *Consider a task system with two types of tasks X, Y, here X is initial, and the following transition rules:*

$$X \overset{0.2}{\hookrightarrow} \langle X, X \rangle, \quad X \overset{0.3}{\hookrightarrow} \langle X, Y \rangle, \quad X \overset{0.5}{\hookrightarrow} \emptyset, \quad Y \overset{0.7}{\hookrightarrow} \langle X \rangle, \quad Y \overset{0.3}{\hookrightarrow} \langle Y \rangle$$

Here, e.g., a task of the type X may generate two new tasks of the type X (with prob. 0.2), or one task X and one Y (with prob. 0.3), or no other task (with prob. 0.5).

[1] In what follows, we denote by $\langle X_1, \ldots, X_n \rangle$ the (finite) multiset containing precisely $X_1, \ldots, X_n$. Note that the order of elements does not play any role. For example, the multiset containing two A's and one B can be described, e.g., by $\langle A, A, B \rangle$ as well as by $\langle A, B, A \rangle$.

Semantics

Intuitively, tasks are executed by one processor. In every step, a processor selects a task from a pool of unprocessed tasks, processes it, and puts the newly generated tasks (if any) back into the pool. The pool initially contains only one task of the initial type X_0, and the next task to be processed is selected by a *scheduler*. In [25], we consider two types of scheduling for task systems, *online* and *offline*.

Online schedulers make their decisions possibly based on the history of the computation, similarly to strategies in stochastic games. In particular, the scheduler does not have any information about the future (except the fixed probabilities of transition rules).

On the other hand, offline schedulers have a complete information not only about the history but also about the future. That is, every offline scheduler knows how all stochastic choices will be resolved. Note that the capability to predict the future behavior is not completely impractical, the scheduler may, for example, inspect the source code of tasks to determine which subtasks will be generated. Also, an optimal offline scheduler may be used as a yardstick for measuring performance of online schedulers.

Offline Scheduling: Executions of a task system are modeled as *family trees* (or execution trees). Intuitively, a family tree is a finite binary tree whose nodes are tasks; a node is labeled with the type of its task. The initial task is the root, children of a task are the tasks generated by it. Given a family tree, a scheduler decides in what order the tree should be traversed.

We formally define a *family tree t* to be a pair (N, L) where $N \subseteq \{0,1\}^*$ is a finite binary tree (i.e., a finite prefix-closed set of words over $\{0,1\}$) and $L : N \to \Gamma$ is a labeling such that the root $\epsilon \in N$ is labeled with the initial task type X_0, i.e., $L(\epsilon) = X_0$, and every node $w \in N$ satisfies one of the following conditions:

- w is a leaf and $L(w) \hookrightarrow \emptyset$,
- or w has a unique child $w0$ and $L(w) \hookrightarrow \langle L(w0) \rangle$,
- or w has two children $w0$, $w1$ and $L(w) \hookrightarrow \langle L(w0), L(w1) \rangle$.

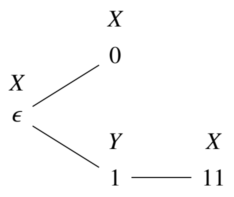
  (Here we assume a fixed ordering on the elements of $\langle L(w0), L(w1) \rangle$.)

Note that every node of a given family tree t is associated with some transition rule which generates its children. We denote by $\mathbb{P}(t)$ the product of the probabilities of all these transition rules.

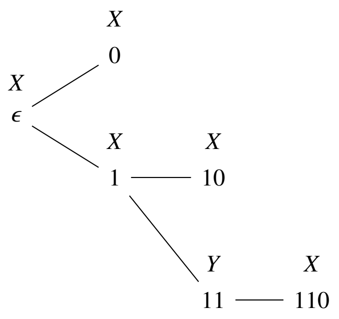
In general, the sum of $\mathbb{P}(t)$ over all trees t does not have to be one. So, in what follows we *assume* that this sum *is* one, and hence that $\mathbb{P}$ is a probability distribution on the family trees. This assumption is equivalent to demanding that the initial task is finished (together with all its subtasks) with probability one. For technical reasons we also assume that all task types are reachable from the initial task type. Proposition 1 of [25] shows that both these conditions can be decided in polynomial time.

Example 5.1.2. *To illustrate the concept of family trees, consider the task system from Example 5.1.1 which induces, e.g., the following family trees t_1 and t_2:*

t_1:



t_2:



Consider the tree t_1. The root is associated with $X \hookrightarrow \langle X, Y\rangle$, the leaf 0 *with $X \hookrightarrow \emptyset$, the node* 1 *with $Y \hookrightarrow X$ and the leaf* 11 *with $X \hookrightarrow \emptyset$. Thus, the probability $\mathbb{P}(t_1)$ is* $0.3 \cdot 0.5 \cdot 0.7 \cdot 0.5$. *The probability $\mathbb{P}(t_2)$ is equal to* $0.2 \cdot 0.5 \cdot 0.3 \cdot 0.5 \cdot 0.7 \cdot 0.5$.

A scheduler now prescribes in which order the family tree should be traversed. Formally, given a tree t, its *derivation* is a sequence $s_1 \Rightarrow s_2 \Rightarrow \cdots \Rightarrow s_k$ where each s_i is a set of nodes of the tree t defined as follows:

- $s_1 = \{\epsilon\}$ contains just the root of t and $s_k = \emptyset$,
- for every $1 \leq i < k$, there is a node $l_i \in s_i$ such that $s_{i+1} = (s_i \setminus \{l_i\}) \cup S$ where S is the set of all children of l_i.

  Elements of s_i are the tasks waiting to be processed in the i-th step. The node l_i is the *task processed in the i-th step*.

An (offline) scheduler σ assigns to a family tree t a derivation $\sigma(t)$ of t. We denote by $\sigma(t)[i]$ the task processed in the i-th step of $\sigma(t)$.

For example, the tree t_2 in Example 5.1.2 allows the following derivation: $\{\epsilon\}, \{0,1\}, \{0,10,11\}, \{10,11\}, \{10,110\}, \{10\}, \emptyset$. That is, in the second step, the task 1 of the type X is processed and generates two new tasks: 10 of type X and 11 of type Y. In the third step, the task 0 is processed and generates no new tasks.

Online scheduling: In [25], the online schedulers are introduced as a special class of offline schedulers (see the beginning of Section 4 of [25]).

A scheduler σ is *online* if for all pairs of family trees $t = (N, L)$ and $t' = (N', L')$ the following holds: If for some i we have

- $\sigma(t) = (s_1 \Rightarrow \cdots \Rightarrow s_i \Rightarrow s_{i+1} \Rightarrow \cdots \Rightarrow s_k)$
- $\sigma(t') = (s_1 \Rightarrow \cdots \Rightarrow s_i \Rightarrow s'_{i+1} \Rightarrow \cdots \Rightarrow s'_k)$

where every node $n \in s_1 \cup \cdots \cup s_i$ is labeled with the same task type in both t and t', i.e., $L(n) = L'(n)$, then $\sigma(t)[i] = \sigma(t')[i]$, i.e., the task processed in the i-th step of $\sigma(t)$ is the same as the task processed in the i-th step of $\sigma(t')$.

This intuitively means that whenever σ traverses a completely similar part of two family trees (that is the same nodes with the same labels), then σ chooses the same task to be executed next. In yet another words, this definition simply says that decisions of the online scheduler are determined only by the part of the family tree which has already been traversed, that is, loosely speaking, the past of the computation.

Objectives: Efficiency of schedulers is usually measured in terms of two values: completion time and completion space. First let us consider the completion time. We define a random variable T which for a given family tree returns the length of its derivation (note that all derivations of a given tree have the same length).

Due to a close relationship between probabilistic BPA (i.e., BPA-SG without players) and TS with so-called *depth-first* schedulers (i.e., schedulers that process tasks in the last in, first out order), the variable T can be analyzed using a large collection of tools developed for the probabilistic BPA. In particular, using methods described in [23], one may decide in polynomial time whether the expected value $\mathbb{E}(T)$ is finite or infinite. In the former case the TS is called *subcritical* and in the latter *critical*. These notions play a crucial role in the analysis of the completion space (see below). The paper [23] also develops tools for quantitative analysis of T such as algorithms approximating $\mathbb{E}(T)$ and the cumulative distribution function of T.

However, as T does not depend on scheduling, the paper [25] concentrates on the *completion space*, i.e., the maximal number of unfinished tasks concurrently waiting for execution. Given a derivation $(s_1 \Rightarrow s_2 \Rightarrow \ldots \Rightarrow s_k)$ of a family tree t, we define the *size* of the derivation to be $\max\{|s_1|, \ldots, |s_k|\}$ where each $|s_i|$ is the size of the set of nodes s_i. For every scheduler σ, we define a random variable S^σ which given a family tree t returns the size of the derivation $\sigma(t)$. We define a random variable S^{op} which given a family tree t returns $\min_\sigma S^\sigma(t)$, that is the minimal size of a derivation of t. Roughly speaking, the offline scheduling is concerned with S^{op}, i.e., with minimizing the completion space over all schedulers. The online scheduling considers only S^σ where σ ranges over online schedulers.

We are concerned with the following questions:

- Are the expected values $\mathbb{E}(S^{op})$ and $\mathbb{E}(S^\sigma)$ finite or infinite?
- How large are the "tail" probabilities $\mathbb{P}(S^{op} \geq k)$ and $\mathbb{P}(S^\sigma \geq k)$? Are they exponentially bounded?

Related work and known results

As noted above, task systems with a special type of depth-first schedulers are equivalent to probabilistic BPA. They are also closely related to the classical model of multi-type *branching processes* (or Galton-Watson processes) that have been studied for more than one hundred years (see, e.g., [54, 4]). The main difference between the branching processes and our task systems is that in branching processes all "tasks" in the pool are performed simultaneously in parallel as opposed to task systems in which just one task is performed (branching processes, of course, have a completely different motivation, the "tasks" are individuals of some species that either die, or reproduce in every step).

The problem of scheduling tasks for a multiprocessor but without stochastic branching has been extensively studied (see, e.g., [63, 9]; we provide more references in the introduction of [25]). An extension of our stochastic task systems towards multiprocessor planning has not been considered yet. This direction seems to be very interesting; in particular, in the multiprocessor case, planning may affect not only space but also time.

5.1.1 Offline Scheduling

Let us consider the variable S^{op} returning the minimal completion space for every family tree. Surprisingly, there is a close connection between the probability distribution of S^{op} and the Newton's method for approximating a zero of a differentiable function.

Consider a function $\vec{f} : \mathbb{R}^\Gamma \to \mathbb{R}^\Gamma$ defined by

$$\vec{f}_X(\vec{v}) \quad := \quad \sum_{X \overset{p}{\hookrightarrow} \langle Y,Z \rangle} p \cdot \vec{v}_Y \cdot \vec{v}_Z + \sum_{X \overset{p}{\hookrightarrow} \langle Y \rangle} p \cdot \vec{v}_Y + \sum_{X \overset{p}{\hookrightarrow} \emptyset} p$$

The function $\vec{f}$ has the least fixed-point $\vec{\mu}$ which plays a fundamental role in analysis of probabilistic recursive systems [2]. It has recently been shown in [48, 57] that $\vec{\mu}$ can be computed by applying the Newton's method to $\vec{f}(\vec{v}) - \vec{v}$. Concretely, $\vec{\mu} = \lim_{k\to\infty} \vec{v}^{(k)}$ where

$$\vec{v}^{(0)} = \vec{0} \qquad \text{and} \qquad \vec{v}^{(k+1)} = \vec{v}^{(k)} + \left(I - J\vec{f}(\vec{v}^{(k)})\right)^{-1} \cdot \left(\vec{f}(\vec{v}^{(k)}) - \vec{v}^{(k)}\right)$$

Here $J\vec{f}(\vec{v}^{(k)})$ is the Jacobian matrix of partial derivatives of $\vec{f}$ evaluated at $\vec{v}^{(k)}$, I is the identity matrix and $\vec{0} = (0, \ldots, 0)$. Surprisingly, in [25] we prove the following.

Theorem 5.1.3 ([25], Theorem 1 and Corollary 2)**.** *Assuming that $\mu = (1, \ldots, 1)$ (i.e., that all tasks are finished with probability one), for the initial task type X_0 and every $k \geq 0$,*

$$\mathbb{P}(S^{op} \leq k) \quad = \quad \vec{v}^{(k)}_{X_0}$$

Moreover, there are real numbers $c > 0$ and $0 < d < 1$ such that for all $k \geq 0$,

$$\mathbb{P}(S^{op} \geq k) \quad \leq \quad c \cdot d^k$$

which implies that $\mathbb{E}(S^{op}) = \sum_{k=1}^{\infty} k \cdot \mathbb{P}(S^{op} = k) = \sum_{k=1}^{\infty} \mathbb{P}(S^{op} \geq k) < \infty$

The last inequality is in striking contrast with online scheduling where the expectation can be infinite no matter what the online scheduler is doing.

5.1.2 Online Scheduling

Now let us concentrate on online schedulers. Note that even though all tasks are finished under an online scheduler σ with probability one, the expected value of S^σ may still be infinite.

In [25], we characterize a finiteness of $\mathbb{E}(S^\sigma)$ using the concept of criticality. Recall that a TS is critical if the expected time to finish the initial task (together with all its subtasks) is infinite, i.e., $\mathbb{E}(T) = \infty$. Otherwise, it is subcritical. Theorem 6 of [25] shows that subcriticality characterizes finiteness of the expected completion space.

Theorem 5.1.4 ([25], Theorem 6)**.** *If the system is critical, then $\mathbb{E}(S^\sigma) = \infty$ for all online schedulers σ. If the system is subcritical, then $\mathbb{E}(S^\sigma) < \infty$ for all online schedulers σ (since also the expected time to finish the initial task is finite).*

Now let us concentrate on the probability distribution $\mathbb{P}(S^\sigma \geq k)$. In [25], we derive exponential bounds on $\mathbb{P}(S^\sigma \geq k)$ for *subcritical* task systems under a mild technical restriction of *compactness* (for definition see Section 4 of [25]). As noted in [25], every task system can be compactified in such a way that for every online scheduler in the compactified system there is an online scheduler in the original one with nearly the same distribution on the completion space (see Proposition 4. of [25]). The following is a simplified version of Theorem 2 and Proposition 5 from [25].

[2] One may prove that $\vec{\mu}_X$ is precisely the probability with which a task of the type X is ever finished. Due to our assumptions we have that $\vec{\mu} = (1, \ldots, 1)$.

Theorem 5.1.5 ([25], Theorem 2 and Proposition 5)**.** *Assuming that the task system is subcritical (and compact), there exist rational numbers $c, d \in (1, \infty)$ and $c', d' \in (0, \infty)$, computable in polynomial time, such that for all online schedulers σ and all $k \geq 1$ the following holds*

$$\frac{c'}{c^{k+2} - 1} \quad \leq \quad \mathbb{P}(S^{\sigma} \geq k) \quad \leq \quad \frac{d'}{d^{k} - 1} \tag{5.1}$$

Note that the bounds of Theorem 5.1.5 hold over all online schedulers. The paper [25] provides sharper bounds for a subclass of so-called *light-first* schedulers. Intuitively, a light-first scheduler always picks tasks with the least expected completion time. Note that in general such schedulers might not be optimal for completion space but it seems to be a good heuristics (at least better bounds on $\mathbb{P}(S^{\sigma} \geq k)$ can be derived as shown in Theorem 4 of [25]).

We also identify a subclass of *continuing* task systems [3] for which the supremum of all d's satisfying the inequality (5.1) of Theorem 5.1.5 can be efficiently approximated (up to a given error tolerance).

[3]A task X is continuing if there are no rules of the form $X \hookrightarrow \langle Y \rangle$ and, moreover, for every rule of the form $X \hookrightarrow \langle Y, Z \rangle$ we have that either $Y = X$, or $Z = X$.

5.2 Controlled Branching Queueing Networks

In [28] we consider a mix of our task systems and Jackson's queueing networks called *controlled branching queueing networks* (CBQN). For background on queueing networks see, e.g., [10]. Very briefly, a queueing network is a network of interconnected servers that process tasks (usually called *jobs* in this context). Each server has a queue in which jobs are waiting to be processed. Once a job is processed by a server, it is either finished, or sent to a randomly chosen (according to a fixed probability distribution) server for further processing. In *open* queueing networks, new jobs may also come to the network from the outside. The network works in real-time, in a basic version the rates of incoming jobs as well as the rates in which servers perform the jobs are fixed. One of the most important questions concerning queueing networks is the question of stability which intuitively means that servers manage to process their jobs and no queue explodes (see below for a more precise definition).

As oppossed to the standard model of queueing networks, CBQN allow a limited amount of external control and branching of jobs, i.e., we allow jobs to stochastically generate new jobs. As opposed to task systems, CBQN allow external jobs to enter the system and jobs are processed in real-time.

Syntax: A *controlled branching queueing network (CBQN) with $n \in \mathbb{N}$ queues* consists of

- an arrival rate of jobs μ_0,
- an arrival production function $Prob_0 : \mathbb{N}^n \to [0,1]$; we assume that $Prob_0$ is non-zero only on a finite set $R_0 \subseteq \mathbb{N}^n$ and that $\sum_{\vec{v} \in R_0} Prob_0(\vec{v}) = 1$,
- a set of n queues, denoted by numbers $i = 1, \ldots, n$, that contain jobs waiting to be processed by servers, each with a *queue rate* $\mu_i \in (0, \infty)$,
- a finite set of actions Ξ_i for every queue $i \in \{1, \ldots, n\}$ [4],
- a production function $Prob_i(\xi_i) : \mathbb{N}^n \to [0,1]$ for every queue $i = 1, \ldots, n$ and every action $\xi_i \in \Xi_i$; we assume that $Prob_i(\xi_i)$ is non-zero only on a finite set $R_i(\xi_i) \subseteq \mathbb{N}^n$ and that $\sum_{\vec{v} \in R_i(\xi_i)} Prob_i(\xi_i)(\vec{v}) = 1$.

A configuration is a vector $\vec{x} \in \mathbb{N}^n$ where each $\vec{x}_i$ specifies the number of jobs waiting in the queue i.

Semantics: A run starts in the configuration $\vec{0} = (0, \ldots, 0)$ where all queues are empty. At any point of time, one of the following events may take place: either jobs arrive to the network from the outside, or a job from some queue is processed by the corresponding server. Both the inter-arrival as well as processing times of servers are exponentially distributed. The rate of arrivals is μ_0 and the rate of processing jobs from the queue i is μ_i.

The arrival of jobs from the outside technically means that new jobs are generated to the queues according to $Prob_0$. More precisely, if the arrival event takes place, a vector $\vec{v}$ is randomly chosen from R_0 (with prob. $Prob_0(\vec{v})$) and then $\vec{v}_j$ jobs are added to the queue j for every $j \in \{1, \ldots, n\}$.

[4]In [28] the set of actions is denoted by Σ_i and the individual actions by σ_i. However, in this text the notation is changed due to collisions with the notation for schedulers from previous chapters.

Whenever a job from queue i is processed by a server, new jobs are randomly generated to the queues according to $Prob_i(\xi_i)$ where ξ_i is the current action selected by a *scheduler*. More precisely, a vector $\vec{v}$ is randomly chosen from $R_i(\xi_i)$ (with prob. $Prob_i(\xi_i)(\vec{v})$) and then $\vec{v}_j$ jobs are added to the queue j for every $j \in \{1, \dots, n\}$.

We consider randomized history-dependent (late) schedulers that choose actions randomly, according to an arbitrary distribution which may possibly depend on the complete history of the run, whenever either an arrival, or a processing event occurs.

Formally, a network induces a *continuous time Markov decision process* whose state space is $\mathbb{N}^n$ and the action set is $\Xi = \Xi_1 \times \cdots \times \Xi_n$, for precise definitions see [28].

Example 5.2.1. *Consider a CBQN with two queues* $1, 2$*. Define* $\mu_0 := 1$*. This means that, on average, new jobs arrive to the network once per a time unit.*

Define $R_0 = \{(1,2),(0,2)\}$ *and define* $Prob_0$ *to assign* $Prob_0(1,2) = 0.4$ *and* $Prob_0(0,2) = 0.6$*. Then every time new jobs come to the network, it is either the case that three jobs arrived, one to the queue* 1 *and two to* 2 *(with prob.* 0.4*), or only two jobs arrived, both to the queue* 2 *(with prob.* 0.6*).*

Define $\mu_1 = 2$ *and* $\mu_2 = 4$*. This implies that, on average, two jobs from the queue* 1 *and four jobs from the queue* 2 *are processed per time unit.*

Define Ξ_1 *to be* $\{\xi, \xi'\}$*. This means that a scheduler controlling the network may choose (anytime) from two "modes" of the queue* 1*.*

Define $R_1(\xi) = \{(1,0),(0,1)\}$ *and* $R_1(\xi') = \{(1,1)\}$*. Further, define* $Prob_1(\xi)(1,0) = 0.3$ *and* $Prob_1(\xi)(0,1) = 0.7$*, and* $Prob_1(\xi')(1,1) = 1$*. Now whenever a job from the queue* 1 *is processed and the action* ξ *is currently chosen by the scheduler, then either one new job is generated to the queue* 1 *(with prob.* 0.3*), or one new job goes to the queue* 1 *(with prob.* 0.7*). If the action* ξ' *is chosen, then one new job goes to the queue* 1 *and one to the queue* 2*.*

Objectives: As noted at the beginning of this section, one of the fundamental issues in queueing theory (and in the general theory of stochastic processes) is *stochastic stability* of a given system. There exist various formal definitions of stability for stochastic systems (for discussion see, e.g., Section 3 in [61]).

We use the following notion of ergodicity to define stable CBQN. A scheduler is *ergodic* if almost all runs leaving the configuration $\vec{0}$ return back to $\vec{0}$ and the expected return time is finite. More precisely, fixing a scheduler σ, we denote by $\vec{x}(t) \in \mathbb{N}^n$ the current configuration in time t (i.e., the vector of sizes of all queues in time t) and define a random variable R by

$$R \quad := \quad \inf\{t > 0 \mid \vec{x}(t) = \vec{0},\ \exists t' < t : \vec{x}(t') \neq \vec{0}\}$$

The scheduler σ is ergodic if $\mathbb{E}(R) < \infty$. One may easily show that then the expected total size of the queues between the first two visits to $\vec{0}$ is finite as well, i.e., the network does not explode. Our goal is to compute an ergodic scheduler σ.

One may also be interested in the (limit) distribution of the size of the queues. Given $\vec{x} \in \mathbb{N}^n$, $\mathbb{P}(\vec{x}(t) = \vec{x})$ is the probability that the configuration of the CBQN is $\vec{x}$ in time t under the fixed scheduler σ. If the limit $\pi(\vec{x}) := \lim_{t\to\infty} \mathbb{P}(\vec{x}(t) = \vec{x})$ exists for every $\vec{x} \in \mathbb{N}^n$ and $\sum_{\vec{x}\in\mathbb{N}^n} \pi(\vec{x}) = 1$, then $\pi : \mathbb{N}^n \to [0,1]$ is called the *stationary distribution*. Note that the stationary distribution π may not exist in general. We show that if there is an ergodic scheduler, then there is also an ergodic scheduler with the stationary distribution.

Related work and known results

As pointed out above, there is a huge amount of literature on fundamentals of queueing networks (see, e.g., [10, 36]) and also their applications in computer science (see, e.g., [73, 40, 74]). A modern version of the original paper by Jackson (first published in 1963) is [55].

Controlled queueing networks have also been studied. Some authors propose to control just the inflow of jobs (e.g., [5]), others, such as [60, 51], propose to control assignment of jobs to servers. The latter work influenced our choice of the control model where branching of jobs and also the assignment to servers is controlled by the scheduler.

Of course, CBQN are closely related to BPA-SG (via their relationship with task systems) and hence also other models of branching systems. For a more detailed description of related work see the Related work paragraph of Section 1. of [28].

5.2.1 Stabilization of CBQN in polynomial time

We show that if there is an ergodic scheduler, then there is a static randomized ergodic scheduler. A randomized scheduler is *static* if its distribution on actions is fixed and does not change in time. The main result of [28] is the following

Theorem 5.2.2. *Let $\mathcal{N}$ be a CBQN with n queues. It is decidable in polynomial time whether there exists an (arbitrary) ergodic scheduler for $\mathcal{N}$. If it exists, one can compute, in polynomial time, a* static randomized *ergodic scheduler for $\mathcal{N}$ with a stationary distribution π such that there exists an exponential moment of the total number of waiting jobs, i.e., there is $\delta > 0$ such that the following sum is finite $\sum_{\vec{x}\in\mathbb{N}^n} \exp(\delta\|\vec{x}\|)\pi(\vec{x})$. (Here $\|\vec{x}\| = \sum_{i=1}^{n} |\vec{x}_i|$ is the 1-norm of $\vec{x}$.)*

To prove Theorem 5.2.2 we generalize the concept of *traffic equations* (see, e.g., [28]) from the theory of Jackson networks (that is CBQN without branching). Intuitively, the traffic equations express the fact that the inflow of jobs to a given queue must be equal to the outflow. Remarkably, the traffic equations characterize the stability of the Jackson network. More precisely, a Jackson network is ergodic if and only if there is a solution of the traffic equations whose components are strictly smaller than the rates of the corresponding queues (we call such a solution *deficient*). We show how to extend the traffic equations so that they characterize the stability of CBQN.

In [28] we start with *uncontrolled* branching networks and add control later on. So first, we set up the traffic equations for uncontrolled branching networks and show that if there is a deficient solution of these equations, then the network is ergodic. This result is of independent interest and requires the construction of a suitable *Lyapunov function*. Then we generalize the traffic equations to controlled branching networks (we obtain a traffic linear program) and show that any ergodic scheduler determines a deficient solution. This solution naturally induces a static scheduler, which, when fixed, determines an uncontrolled network with deficiently solvable traffic equations.

Bibliography

[1] E. Allender, P. Bürgisser, J. Kjeldgaard-Pedersen, and P.B. Miltersen. On the complexity of numerical analysis. *SIAM Journal of Computing*, 38:1987–2006, 2008.

[2] R. Alur, S. La Torre, and P. Madhusudan. Modular strategies for recursive game graphs. *Theor. Comput. Sci.*, 354(2):230–249, 2006.

[3] S. Anderson and G. Bruns. The formalization and analysis of a communications protocol. *Formal Aspects of Computing*, 6:92–112, 1994.

[4] K.B. Athreya and P.E. Ney. *Branching Processes*. Springer, 1972.

[5] A. Azaron and S.M.T. Fatemi Ghomi. Optimal control of service rates and arrivals in Jackson networks. *European Journal of Operational Research*, 147(1):17–31, 2003.

[6] N. Berger, N. Kapur, L. Schulman, and V. Vazirani. Solvency games. In *Proceedings of IARCS Annual Conf. on Foundations of Software Technology and Theoretical Computer Science (FST & TCS 2008)*, vol. 2 of *LIPIcs*, pp. 61–72. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2008.

[7] J.A. Bergstra, A. Ponse, and S.A. Smolka, editors. *Handbook of Process Algebra*. Elsevier, 2001.

[8] P. Billingsley. *Probability and Measure*. Wiley, 1995.

[9] R. Blumofe and C. Leiserson. Scheduling multithreaded computations by work stealing. *JACM*, 46(5):720–748, September 1999.

[10] G. Bolch, S. Greiner, H. de Meer, and K. Trivedi. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*. Wiley, 2006.

[11] A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: application to model checking. In *Proceedings of 8th International Conference on Concurrency Theory (CONCUR 1997)*, vol. 1243 of *LNCS*, pp. 135–150. Springer, 1997.

[12] T. Brázdil, V. Brožek, and K. Etessami. One-counter stochastic games. In *Proceedings of IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS 2010)*, vol. 8 of *LIPIcs*, pp. 108–119. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2010.

[13] T. Brázdil, V. Brožek, K. Etessami, and A. Kučera. Approximating the termination value of one-counter MDPs and stochastic games. *Information & Computation*. doi: http://dx.doi.org/10.1016/j.ic.2012.01.008. To appear.

[14] T. Brázdil, V. Brožek, K. Etessami, and A. Kučera. Approximating the termination value of one-counter MDPs and stochastic games. In *Proceedings of 38th Internat. Colloq. on Automata, Languages and Programming (ICALP 2011, Part II)*, vol. 6756 of *LNCS*, pp. 332–343. Springer, 2011.

[15] T. Brázdil, V. Brožek, K. Etessami, A. Kučera, and D. Wojtczak. One-counter Markov decision processes. In *Proceedings of 21st Annual ACM-SIAM Symp. on Discrete Algorithms (SODA 2010)*, pp. 863–874. SIAM, 2010.

[16] T. Brázdil, V. Brožek, and V. Forejt. Branching-time model-checking of probabilistic pushdown automata. *Electr. Notes Theor. Comput. Sci.*, 239:73–83, 2009.

[17] T. Brázdil, V. Brožek, V. Forejt, and A. Kučera. Reachability in recursive Markov decision processes. In *Proceedings of 17th Internat. Conf. on Concurrency Theory (CONCUR 2006)*, vol. 4137 of *LNCS*, pp. 358–374. Springer, 2006.

[18] T. Brázdil, V. Brožek, V. Forejt, and A. Kučera. Reachability in recursive Markov decision processes. *Information & Computation*, 206(5):520–537, 2008.

[19] T. Brázdil, V. Brožek, J. Holeček, and A. Kučera. Discounted properties of probabilistic pushdown automata. In *Proceedings of 15th Internat. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2008)*, vol. 5330 of *LNCS*, pp. 230–242. Springer, 2008.

[20] T. Brázdil, V. Brožek, A. Kučera, and J. Obdržálek. Qualitative reachability in stochastic BPA games. In *Proceedings of 26th Internat. Symp. on Theoretical Aspects of Computer Science (STACS 2009)*, vol. 3, pp. 207–218. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2009.

[21] T. Brázdil, V. Brožek, A. Kučera, and J. Obdržálek. Qualitative reachability in stochastic BPA games. *Information & Computation*, 209(8):1160–1183, 2011.

[22] T. Brázdil, K. Chatterjee, A. Kučera, and P. Novotný. Efficient controller synthesis for consumption games with multiple resource types. In *Proceedings of 24th International Conference on Computer Aided Verification (CAV 2012)*, vol. 7358 of *LNCS*, pp. 23–38. Springer, 2012.

[23] T. Brázdil, J. Esparza, S. Kiefer, and A. Kučera. Analyzing probabilistic pushdown automata. *Formal Methods in System Design*, 2012. doi: http://dx.doi.org/10.1007/s10703-012-0166-0. To appear.

[24] T. Brázdil, J. Esparza, S. Kiefer, and M. Luttenberger. Space-efficient scheduling of stochastically generated tasks. In *Proceedings of 37th Internat. Colloq. on Automata, Languages and Programming (ICALP 2010, Part II)*, vol. 6199, pp. 539–550, 2010.

[25] T. Brázdil, J. Esparza, S. Kiefer, and M. Luttenberger. Space-efficient scheduling of stochastically generated tasks. *Information & Computation*, 210:87–110, 2012.

[26] T. Brázdil, J. Esparza, and A. Kučera. Analysis and prediction of the long-run behavior of probabilistic sequential programs with recursion. In *Proceedings of 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005)*, pp. 521–530. IEEE, 2005.

[27] T. Brázdil, P. Jančar, and A. Kučera. Reachability games on extended vector addition systems with states. In *Proceedings of 37th Internat. Colloq. on Automata, Languages and Programming (ICALP 2010, Part II)*, vol. 6199, pp. 478–489, 2010.

[28] T. Brázdil and S. Kiefer. Stabilization of branching queueing networks. In *Proceedings of 29th Internat. Symp. on Theoretical Aspects of Computer Science (STACS 2012)*, vol. 14 of *LIPIcs*, pp. 507–518. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2012.

[29] T. Brázdil, A. Kučera, and Petr Novotný. Determinacy in stochastic games with unbounded payoff functions. In *Proceedings of 8th Internat. Doctoral Workshop on Mathematical and Engineering Methods in Computer Science (MEMICS 2012)*. Springer. To appear.

[30] T. Brázdil, A. Kučera, and O. Stražovský. On the decidability of temporal properties of probabilistic pushdown automata. In *Proceedings of 22nd Annual Symp. on Theoretical Aspects of Computer Science (STACS 2005)*, vol. 3404 of *LNCS*, pp. 145–157. Springer, 2005.

[31] O. Burkart and B. Steffen. Model checking for context-free processes. In *Proceedings of 3rd International Conference on Concurrency Theory (CONCUR 1992)*, vol. 630 of *LNCS*, pp. 123–137. Springer, 1992.

[32] J. Chaloupka. Z-reachability problem for games on 2-dimen-sional vector addition systems with states is in P. In *Proceedings of 4th Internat. Workshop on Reachability Problems (RP 2010)*, vol. 6227 of *LNCS*, pp. 104–119. Springer, 2010.

[33] K. Chatterjee, L. Doyen, and T. Henzinger. A survey of stochastic games with limsup and liminf objectives. In *Proceedings of 36th Internat. Colloq. on Automata, Languages and Programming (ICALP 2009)*, vol. 5556 of *LNCS*, pp. 1–15. Springer, 2009.

[34] K. Chatterjee, L. Doyen, T. Henzinger, and J.-F. Raskin. Generalized mean-payoff and energy games. In *Proceedings of IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS 2010)*, vol. 8 of *LIPIcs*, pp. 505–516. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010.

[35] K. Chatterjee and T. Henzinger. A survey of stochastic - regular games. *J. Comput. Syst. Sci.*, 78(2):394–413, 2012.

[36] H. Chen and D. Yao. *Fundamentals of Queueing Networks*. Springer, 2001.

[37] A. Condon. The complexity of stochastic games. *Information and Computation*, 96(2):203–224, 1992.

[38] C. Courcoubetis and M. Yannakakis. Markov decision processes and regular events. In *Proceedings of 17th Internat. Colloq. on Automata, Languages and Programming (ICALP 1990)*, vol. 443 of *LNCS*, pp. 336–349. Springer, 1990.

[39] C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of the Association for Computing Machinery*, 42(4):857–907, 1995.

[40] J. Daigle. *Queueing Theory with Applications to Packet Telecommunication*. Springer, 2010.

[41] L. E. Dickson. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *American Journal of Mathematics*, 35(4), 1913.

[42] J. Esparza, A. Kučera, and R. Mayr. Model-checking probabilistic pushdown automata. *Logical Methods in Computer Science*, 2(1:2):1–31, 2006.

[43] J. Esparza, A. Kučera, and S. Schwoon. Model-checking LTL with regular valuations for pushdown systems. *Information & Computation*, 186(2):355–376, 2003.

[44] K. Etessami, D. Wojtczak, and M. Yannakakis. Quasi-birth-death processes, tree-like QBDs, probabilistic 1-counter automata, and pushdown systems. *Performance Evaluation*, 67(9):837–857, 2010.

[45] K. Etessami and M. Yannakakis. Recursive Markov decision processes and recursive stochastic games. In *Proceedings of 32nd Internat. Colloq. on Automata, Languages and Programming (ICALP 2005)*, vol. 3580 of *LNCS*, pp. 891–903. Springer, 2005.

[46] K. Etessami and M. Yannakakis. Efficient qualitative analysis of classes of recursive Markov decision processes and simple stochastic games. In *Proceedings of 23rd Annual Symp. on Theoretical Aspects of Computer Science (STACS 2006)*, vol. 3884 of *LNCS*, pp. 634–645. Springer, 2006.

[47] K. Etessami and M. Yannakakis. Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations. *Journal of the Association for Computing Machinery*, 56, 2009.

[48] K. Etessami and M. Yannakakis. Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations. *Journal of the ACM*, 56(1), 2009.

[49] U. Fahrenberg, L. Juhl, K. G. Larsen, and J. Srba. Energy games in multiweighted automata. In *Proceedings of 8th Internat. Colloq. on Theoretical Aspects of Computing (ICTAC 2011)*, vol. 6916 of *LNCS*, pp. 95–115. Springer, 2011.

[50] J. Filar and K. Vrieze. *Competitive Markov Decision Processes*. Springer, 1996.

[51] K.D. Glazebrook and J. Niño-Mora. A linear programming approach to stability, optimisation and performance analysis for Markovian multiclass queueing networks. *Annals of Operations Research*, 92:1–18, 1999.

[52] S. Göller and M. Lohrey. Branching-time model checking of one-counter processes. In *Proceedings of 27th International Symposium on Theoretical Aspects of Computer Science (STACS 2010)*, vol. 5 of *LIPIcs*, pp. 405–416. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010.

[53] S. Göller, R. Mayr, and A.W. To. On the computational complexity of verifying one-counter processes. In *Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science (LICS 2009)*, pp. 235–244. IEEE, 2009.

[54] T.E. Harris. *The Theory of Branching Processes*. Springer, 1963.

[55] J. Jackson. Jobshop-like queueing systems. *Manage. Sci.*, 50(12 Supplement):1796–1802, 2004.

[56] P. Jančar. Decidability of bisimilarity for one-counter processes. *Information and Computation*, 158(1):1–17, 2000.

[57] S. Kiefer, M. Luttenberger, and J. Esparza. On the convergence of Newton's method for monotone systems of polynomial equations. In *Proceedings of 39th Annual ACM Symp. on Theory of Computing (STOC 2007)*, pp. 217–226. ACM, 2007.

[58] A. Maitra and W. Sudderth. Finitely additive stochastic games with Borel measurable payoffs. *International Journal of Game Theory*, 27:257–267, 1998.

[59] D.A. Martin. The determinacy of Blackwell games. *Journal of Symbolic Logic*, 63(4):1565–1581, 1998.

[60] S. Meyn. *Control Techniques for Complex Networks*. Cambridge University Press, 2008.

[61] S. Meyn and R.L. Tweedie. *Markov Chains and Stochastic Stability*. Cambridge University Press, 2009.

[62] R. Myerson. *Game Theory: Analysis of Conflict*. Harvard University Press, 1991.

[63] G. Narlikar and G. Blelloch. Space-efficient scheduling of nested parallelism. *ACM Trans. Program. Lang. Syst.*, 21(1):138–173, January 1999.

[64] J.L. Peterson. *Petri Net Theory and the Modelling of Systems*. Prentice-Hall, 1981.

[65] M.L. Puterman. *Markov Decision Processes*. Wiley, 1994.

[66] C. Rackoff. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, 6:223–231, 1978.

[67] W. Reisig. *Petri Nets—An Introduction*. Springer, 1985.

[68] L.E. Rosier and H.-C. Yen. A multiparameter analysis of the boundedness problem for vector addition systems. *Journal of Computer and System Sciences*, 32:105–135, 1986.

[69] O. Serre. Parity games played on transition graphs of one-counter processes. In *Proceedings of 9th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS 2006)*, vol. 3921 of *LNCS*, pp. 337–351. Springer, 2006.

[70] M. Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 2012.

[71] A.W. To. Model checking FO(R) over one-counter processes and beyond. In *Proceedings of 18th Annual Conference of the EACSL on Computer Science Logic (CSL 2009)*, vol. 5771 of *LNCS*, pp. 485–499. Springer, 2009.

[72] I. Walukiewicz. Pushdown processes: Games and model-checking. *Information and Computation*, 164(2):234–263, 2001.

[73] W. Yue, Y. Takahashi, and H. Takagi. *Advances in Queueing Theory and Network Applications*. Springer, 2010.

[74] H. Zisgen, I. Meents, B.R. Wheeler, and T. Hanschke. A queueing network based system to model capacity and cycle time for semiconductor fabrication. In *Proceedings of 40th Winter Simulation Conference (WSC 2008)*, pp. 2067–2074. WSC, 2008.