Quantitative Formal Methods Meets Algorithmic Game Theory

Thesis submitted for the degree of Doctor of Philosophy

Guy Avni

Submitted to the Senate of the Hebrew University of Jerusalem

February, 2016

This work was carried out under the supervision of Orna Kupferman.

Acknowledgements

First and foremost, I would like to thank my advisor Orna Kupferman. It has been one hell of a journey. At the end of my MSc studies, when I was deliberating on which path to choose, she promised the Phd path "would be fun". This is all the convincing I needed, and I have to say that she kept her promise. The last time I wrote an acknowledgement section and thanked her, I mentioned she taught me how to write so that my writing was "reasonable" at the time. Looking back, my view then was very superficial. What I referred to as "teaching writing" were merely English syntax rules that I didn't know. The teaching is much deeper. So deep, that I don't want to embarrass myself again and thank her on specific points. I feel Orna has let me be her apprentice for these years and I thank her for teaching me her craft. In addition to her guidance in the academia, I also see her as a role model for life outside the academia, and thank her for the lessons there as well.

The work on this thesis had a few key moments. One of the crucial ones started out as a standard mail, which I sent Orna. It involved some work that I ran into and thought it was connected to what we were doing at the time. She replied that it was nice, not related, but might be related to what she was doing with Tami. She asked me if I wanted to join on the grounds that "It's a lot of fun". She didn't get that one wrong either. It was a pleasure working with Tami Tamir, Orna's identical twin sister. Tami guided me throughout the work on algorithmic game theory and gave the complement to Orna's guidance. Without her, this thesis would not have been the same. I thank her greatly.

Also, I had the pleasure of working with Thomas Henzinger, and I'm looking forward to working together in the future.

My accomplice throughout the past few years has been Shaull Almagor. We shared an office, worked together, shared beds in conferences, and many drinks and dinners (some ending better while others ending worse). Shaull helped make the past few years enjoyable and I thank him greatly. The second ongoing member of my office was Jonathan Moshieff. Apart from making it a nice place, he was a goto guy for many topics ranging from combinatorics to managing bank accounts.

Life outside the academia was a roller coaster in these past few years. Holding my hand through it is my wife Efrat. I hope I can ever even out with all she has given in order for my thesis to succeed. The high points of the ride are inhabited by my two daughters Tamar and Ella, who I thank for my second childhood. I also thank my slightly extended family; my parents and brothers for their sincere caring and giving throughout these years. An acknowledgement section is nothing without a special thanks to my dog Bigi with whom I had fruitful discussion and debates in our long nightly walks. To close, I thank my *gariin* mates for always being there.

Abstract

This thesis lies on the boundary between two fields: formal methods and algorithmic game theory (AGT, for short). We adapt ideas from formal methods in AGT and back. The goal in formal methods is to formally reason about systems. So, the bread and butter of this field is the study of specifications and of ongoing computations. AGT is a rapidly evolving field that takes a computational approach to game theory. One of its subfields on which we focus, studies the stability of different classes of games.

We start by applying the concept of a rich specification in *network formation games*, which constitute a well studied class of games in AGT. A network is modeled by a directed graph, and each player has a source and target vertex, which she wishes to connect. The strategies of each player are the simple paths connecting her source to the target. Each edge has a cost and the players that use the edge split the cost evenly among them. The players' objective in the traditional game are reachability.

We study an extension of network formation games in which the edges in the network are labeled by letters from some alphabet and the players' objectives are given by a regular language over the same alphabet. In particular, the richer objectives are such that the paths selected by the players need not be simple, thus a player may traverse some transitions several times. We refer to this extension as *automata formation games*. Edge costs are shared by the players with the share being proportional to the number of times the edge is traversed. We study the stability of automata formation games with respect to standard measures in AGT: existence of equilibria, equilibrium efficiency, and we study computational problems for these games.

Network formation games are a special case of *cost-sharing games*. Our automata formation games give rise to an extension of cost-sharing games in which the players' strategies consist of multisets of resources rather than sets of resources. We also study *multiset congestion games* in which a greater load on a resource has a negative effect on cost. We refer to the union of these two games as *multiset resource-allocation games*, and we analyze the stability of this class of games and compare with the games that it extends.

For the direction from AGT to formal methods, we analyze a problem in formal methods as a resource-allocation game. Synthesis is the automated construction of a system from its specification. In real life, hardware and software systems are rarely constructed from scratch. Rather, a system is typically constructed from a library of components. Lustig and Vardi formalized this intuition and studied a setting in which a designer searches for a *design*, which is a recipe to glue the components from the library together to form a *compositional system*. We extend on Lustig and

Vardi's setting in two directions. We add costs to the components in the library, and we consider a setting in which multiple designers use the same library of components. The cost of a designer's design is affected by the choices of the other designers. For example, if the components are processors and the cost models running time, then designers would prefer to use processors with lower load in order to increase their system's performance. A multiset resource allocation game arises from this setting.

We return to the direction from formal methods to AGT. Ongoing games are common in formal methods and we apply ideas from such games to resource-allocation games. We relax two of the main assumptions of these games – the players choose their strategies in one shot and the players move simultaneously. We introduce and study *dynamic resource-allocation games*, which allow the players to choose resources in an iterative and non-concurrent manner. The definitions of our game are similar to the definitions of ongoing games in formal methods, while we ask stability questions that originate from AGT.

Traditional formal methods are Boolean in nature: a system either satisfies its specification or not. On the other hand, game theory is quantitative in nature: each player has a value in an outcome of a game, which she wishes to increase. We join a growing effort in recent years to lift traditional formal methods to reason about quality. The *automata-theoretic* approach uses the theory of automata as a unifying paradigm for system specification and verification. In the quantitative setting, *weighted finite automata* (WFAs, for short) are an important class of automata to model specifications as well as systems. We study several problems for WFAs. The containment problem for WFAs is undecidable while it is a problem of great practical importance. We suggest a heuristic approach to bypass its undecidability that is based on adapting *abstraction* and *simulation*, which are well known in the Boolean setting, to the quantitative setting. Next, we study a setting of partially-specified quantitative systems, which is again well studied in the Boolean setting. We model partial systems as WFAs with missing weights, and we study the problem of completing a partial WFA such that it satisfies some given restrictions. Finally, we introduce and study the problem of replacing nondeterminism in WFAs with probabilistic transitions. We refer to this process as *stochastization* of WFAs, and we study the problem of find a stochastization that does not alter "too much" the weighted language of a given WFA.

Letter Indicating the Contribution to Each Chapter

In all four chapters, I was the principal researcher. The first and third chapters are joint work with Orna Kupferman and Tami Tamir. Tami is an expert in algorithmic game theory. She guided me in these topics throughout these two works. The second chapter is joint work with Orna Kupferman. The final chapter is joint work with Orna Kupferman and Thomas Henzinger. Thomas is an expert in multi-agent games. He helped guide me throughout this work.

Contents

1 Introduction **10**

1.1 Game Theory 10

1.2 Quantitative Formal Methods 21

2 Network-Formation Games with Regular Objectives **32**

3 Synthesis from Component Libraries with Costs **56**

4 Congestion Games with Multisets of Resources and Applications in Synthesis **102**

5 Dynamic Congestion Games **135**

6 Discussion **165**

1 Introduction

The goal in formal methods is to formally reason about systems. Either proving formally that a system satisfies its specification, synthesizing a system from a given specification, or reasoning about other aspects of systems. Traditional formal methods are Boolean in nature: a system either satisfies its specification or not. In recent years, there is a growing effort to lift traditional formal methods to reason about quality. So, for example, rather than asking whether the system satisfies the specification, one might ask how well the system satisfies the specification. The origin of the quantitative aspect can be in the system [24], the specification [3, 4], or both [21].

In addition to the questions that arise when lifting Boolean questions to the quantitative setting, other questions arise that may not have been considered at all in the Boolean setting. For example, a specification can assign values to computations. It makes sense to find a system that approximates the specification. Namely, it assigns to every computation a value that is close to the one the specification assigns to it. Such an approximation is less natural in the Boolean setting. We return to approximation in the second part of the thesis.

In the first part of the thesis we focuses on the last type of questions, and in particular, we study such questions on games.

1.1 Game Theory

We start with an example. As we mentioned above, a designer can have a specification that assigns different values to systems. But a system rarely stands on its own. Typically, a designer only designs one component of the system. The different components either interact with each other, or compete for global resources. In the later case, the value given by the specification can be the time it takes for it to run. So, the value is affected by how much competition there is on the resources the system uses. The designers are selfish. They only care about increasing the value of their own component. So, a game arrises, which we formalize throughout this section.

Many problems in formal methods are solved using games (c.f., [7]). Typically, the games that are studied are two-player games; one player takes the role of the system and plays against an adversary whose goal is to show that the system is faulty. These games are zero-sum games; the system wins iff it can satisfy the specification no matter how the adversary behaves. Also, many systems under consideration in formal methods are reactive and non-terminating, thus the games are typically ongoing games of infinite duration.

Recall the game that we described between several designers who design components that compete for resources. It has a different flavor to it from the games that are typically studied in formal methods. It is played between several designers, so it is a multi-player game. The systems have values and the goal of the designers is to maximizes their system's value, so the game is not zero-sum. Finally, the game "ends" when the designers choose a system, so it is a "one-round game" rather than ongoing. The question we ask is also different. We are not interested in a "winning strategy" for the system as in formal methods. Rather we want a "stable outcome": one in which the designers have no incentive to alter their chosen designs. The flavor of this game as well as the questions asked are studied in *algorithmic game theory* (AGT, for short). AGT is a rapidly growing field [45] that lies in the intersection between several fields including computer science and economics.

We introduce the questions asked in AGT in more detail, while using a running example of *network formation games* (NFGs, for short) [6], which constitutes a well studied class of games in AGT. A network is modeled by a directed graph, and each player has a source and target vertex, which she wishes to connect. The strategies of each player are the simple paths connecting her source to the target. Each edge has a cost and the players that use the edge, split the cost evenly between them. The game is "one round" in the sense that the players select a path in one shot. A *profile* is a vector of strategies (paths), one for each player.

We present the classic questions on NFGs. These questions in different variants will accompany us throughout this thesis. (*i*) Existence of an equilibrium. Recall that players are selfish. So, not all profiles are *stable* in the sense that players might benefit from changing their strategy. In NFGs and in most games we consider, the notion of stability that is considered is a *Nash equilibrium* (NE, for short)[1]. An NE is a profile in which no player can benefit from a unilateral deviation. We ask whether each instance of the game has a profile of pure strategies that constitutes an NE. (*ii*) An analysis of *equilibrium inefficiency*. It is well known that decentralized decision-making may lead to solutions that are sub-optimal from the point of view of society as a whole. The cost of a profile is the sum of players' costs in it. We quantify the inefficiency incurred due to selfish behavior according to the *price of anarchy* (PoA) [37] and *price of stability* (PoS) [6] measures. In both measures we compare the cost of stable profiles against the *social optimum* profile (SO, for short), which is the cheapest profile and is not necessarily stable. The PoA is the worst-case inefficiency of a Nash equilibrium (that is, the ratio between the cost of the worst NE and the SO). The PoS is the best-case inefficiency of a Nash equilibrium (that is,

[1]Throughout this thesis, we concentrate on *pure* strategies rather than considering *mixed* strategies, which allow choosing a probabilistic distribution on pure strategies. This is also the choice in the vast majority of works in AGT on NFGs as well as the other games we consider.

the ratio between the cost of the best NE and the SO). (*iii*) We study computational questions that vary slightly according to the game under consideration. First, the *best-response* problem; given strategies for the players $1, \ldots, k-1$, find the optimal strategy for Player k. Second, depending on the answer to question (i), we study the existence of NE or the complexity of finding one. In some cases we also study the complexity of finding the SO.

This thesis lies is the meeting point of formal methods, and in particular quantitative formal methods, with AGT. We present several works that adapt ideas from formal methods in AGT and back.

Automata formation games The players' objectives in NFGs are can be thought of as reachability; a player's goal is to *reach* her destination. We extend network-formation games to a setting in which the players have richer objectives. This involves two changes of the underlying setting: First, the edges in the network are labeled by letters from a designated alphabet. For example, the alphabet letters model the security level of an edge or its bandwidth. Second, the objective of each player is specified by a *language* over this alphabet. Each player has a regular language and she should select a path labeled by a word in her objective language. For example, a player's language might restrict to paths traversing high security links, or, if the game models a delivery service, a player's language can require multiple visits to a certain location.

If we view the network as a *nondeterministic weighted finite automaton* (WFA) $\mathcal{A}$, which we discuss in length later on, then the set of strategies for a player with objective L is the set of accepting runs of $\mathcal{A}$ on some word in L. Accordingly, we refer to our extension as *automaton-formation games* (AFGs, for short). Unlike traditional NFGs, the runs selected by the players need not be simple, thus a player's path may traverse some edges several times. Edge costs are still shared by the players, but now the share is split proportionally to the number of times the edge is traversed. This latter issue is the main technical difference between AFGs and NFGs, and as we shall see, it is very significant.

We study questions $(i) - (iii)$ above for AFGs and compare the answers with these in NFGs. The answer for question (i) in NFGs is positive; every NFG is guaranteed to have an NE. In fact, NFGs are *potential games*, which have an even stronger property; every sequence of improving moves of the players converges to an NE. When the improving moves in the sequence are the best possible, the sequence is often called a *best-response dynamics*. On the other hand, we show that even very restrictive fragments of AFGs are not guaranteed to have an NE. Recall that the network can be viewed as a WFA $\mathcal{A}$. We consider the following classes of WFAs: (1) *all-accepting*, in which all the states of $\mathcal{A}$ are accepting, thus its language is

prefix closed (2) *uniform costs*, in which all edges have the same cost, and (3) *single letter*, in which $\mathcal{A}$ is over a single-letter alphabet. We consider the following classes of specifications: (1) *single word*, where the language of each player is a single word, and (2) *symmetric*, where all players have the same objective. We refer to AFGs that are all-accepting, uniform costs, single letter, and single word as *weak* AFGs, and we show that weak AFGs are not guaranteed to have an NE. Maybe even more surprising, we show that symmetric instances of AFGs are not guaranteed to have an NE.

Regarding question (*ii*), of equilibrium inefficiency, we show that while the PoA in AFGs agrees with the one in classical NFGs and is equal to the number of players, the PoS also equals the number of players, again already for the very restricted weak instances. This is in contrast with classical NFGs, where the PoS tends to *log* the number of players. We do manage to find structural restrictions on the network with which the social optimum is an NE, thus we have PoS$= 1$ for these instances.

Finally, we address (*iii*), namely computational problems on AFGs. We show that for some restricted instances, finding the SO can be done efficiently, while for other restricted instances, the complexity agrees with the NP-completeness of classical NFGs. The best-response problem is NP-complete, while it is in P for NFGs. We show that deciding the existence of NE is Σ^2_P-complete for AFGs. This problem is not studied for NFGs as they are guaranteed to have an NE.

These results have been described in [14, 13] and in Chapter 2 of the thesis.

Multiset resource allocation games NFGs an be viewed as a special case of *cost-sharing games* (SGs, for short). Such a game is played on a set of resources. A player's possible strategies is a collection of sets of resources. As in NFGs, each resource has a cost that is split among the player that use it. In the cost-sharing game that corresponds to an NFG, the resources are the edges of the graph and the strategies of a player are the sets of edges that correspond to the simple paths that connect her two vertices.

We view AFGs as cost-sharing games. Again, a strategy that corresponds to a path consists of the edges that the path traverses. Recall that paths in AFGs need not be simple. So, a resource may appear several times in a strategy in the corresponding cost-sharing game, making it a *multiset*. Thus, AFGs are a special case of *multiset cost-sharing games*.

Cost-sharing games model settings in which resources have costs, which are split between the players using them. In such cases, the *load* on the resource, namely the total number it is used, has a positive effect. However, in many settings, the load on the resources has a negative effect. For example, returning to the network game, the network can model a map of roads. A higher load on a road implies a traffic

jam that produces a higher cost for the players using it. We refer to such games as *congestion games* (CGs, for short) [49]

Formally, a CG is similar to a cost-sharing game only that instead of resource costs, each resource e has a *latency function* of the form $\ell_e : \mathbb{N} \to \mathbb{R}$, where $\ell_e(\gamma)$ is the cost of a single use of e when the load on it is γ. So, if Player i uses e n_i times, she pays $n_i \cdot \ell_e(\gamma)$ for e. Cost-sharing games can be thought of as a special case of CGs in which resource e has a cost c_e and the latency function is $\ell_e(\gamma) = c_e/\gamma$. Note that if Player i uses e n_i times, then she pays $c_e \cdot n_i/\gamma$, which is the proportional sharing rule we studied in the previous section. For convenience, we make the distinction between cost-sharing and congestion games, and refer to their union as *resource allocation games* (RAGs, for short). We introduce and study multiset RAGs.

Our results in terms of existence of NE and equilibrium inefficiency for AFGs carry over to multiset cost-sharing games. We study these two questions for multiset congestion games (MCGs, for short). In terms of NE existence, the answer depends on the latency functions. For affine latency functions, i.e., functions of the form $a \cdot x + b$, we show good news; affine MCGs are potential games and are guaranteed to have an NE. On the other hand, already for quadratic latency functions, there are symmetric instances with no NE. We study the equilibrium inefficiency for affine MCGs. We show that the PoA is PoA $= 1 + \phi$, where $\phi \approx 1.618$ is the golden ratio, and the PoS is between 2 and 1.631. Again, much stabler than in cost-sharing games.

In order to put our results in context, we compare our results with these known for *weighted* congestion games (WCGs, for short) [42]. These are congestion games in which each Player i has a *weight* $w_i \in \mathbb{N}$, and his contribution to the load of the resources she uses as well as her payments are multiplied by w_i. WCGs can be viewed as a special case of MCGs, where each resource in a strategy for Player i repeats w_i times.

We summarize the comparison between the classes of games in Table 1 below. Our upper bounds for MCGs match the known upper bounds for WCGs. Whenever our lower bounds match the ones of WCGs, they are given with much simpler instances.

	∃NE	PoA	PoS
Congestion Games	Yes	2.5[29]	≈ 1.577[29, 25]
WCGs	Affine	$1 + \phi$[17]	$\approx 1.577 \leqslant$ PoS $\leqslant 2$[20]
MCGs	Affine	$1 + \phi$	$1.631 \leqslant$ PoS $\leqslant 2$

Table 1: A comparison between congestion games, WCGs, and MCGs.

These results have been described in [15] and in Chapter 4 of the thesis.

Synthesis from component libraries The results above can be seen as an adaption of ideas from formal methods to AGT – strategies that are a multiset of resources arise when we transition from a network with reachability objectives to an automaton in which paths need not be simple. Here, we go in the other direction and apply ideas from AGT to formal methods. Specifically, we show an application of MCGs in the problem of *synthesis from component libraries*, which is a formalization of the game we mentioned in the beginning of the introduction.

A central problem in formal methods is synthesis [47], namely the automated construction of a system from its specification. In real life, hardware and software systems are rarely constructed from scratch. Rather, a system is typically constructed from a library of components by *gluing* components from a library (allowing multiple uses)[40]. For example, when designing an internet browser, a designer does not implement the TCP protocol but uses existing implementations as black boxes.

We follow the definitions of [40]. A *design* is a recipe to glue the components together. The components are *black boxes*, so the design sees only the exit state through which the component completes its computation and relinquishes control. Based on this information, the design decides which component gets control next, and so on. Given a design $\mathcal{D}$ and a library of components $\mathcal{L}$ we can compose the components according to the design to construct the *compositional* system $\mathcal{A}_{\mathcal{L},\mathcal{D}}$, which is a concrete system. The input to the synthesis from components problem is a library $\mathcal{L}$ and a specification $\mathcal{S}$. The goal is to find a design $\mathcal{D}$ such that $\mathcal{A}_{\mathcal{L},\mathcal{D}}$ meets the specification $\mathcal{S}$. We then refer to $\mathcal{D}$ as a *correct* design with respect to $\mathcal{S}$.

We study synthesis from component libraries with costs in the *closed* and *open* settings. In both settings, the specification is given by means of a deterministic automaton $\mathcal{S}$ on finite words (DFA). In the closed setting, the specification is a regular language over some alphabet Σ and the library consists of box-DFAs (that is, DFAs with exit states) over Σ. The compositional system here is a DFA. Correctness means that the language over Σ of the composition equals the language of $\mathcal{S}$. In the open setting, the specification $\mathcal{S}$ is over sets I and O of input and output signals, and the library consists of box-I/O-transducers. The compositional system here is a transducer over I and O. Correctness here means that the interaction of the composition defined by $\mathcal{D}$ with all input sequences generates a computation over $I \cup O$ that is in the language of $\mathcal{S}$.

We extend on Lustig and Vardi's setting in two aspects. First, we assume that components have a costs. The cost of a component models its quality and is paid each time the component is used. For example, the cost of a component can be the number of states in the component, so the total cost of design is the number of states in the compositional system. It makes sense to find a cheaper system as this is a

system with less states that is assumed to be simpler. We study the the problem of finding a correct design as well as the problem of finding a cheapest correct design in the closed and open settings. In the closed setting, finding a correct design can be done in polynomial time, and finding a cheapest correct design is NP-complete. In the open setting, the design problem is EXPTIME-complete while finding a cheapest correct design is NEXPTIME-complete.

The public cost is a cost that is affected by the choices of the other designers. For example, it can be the price that needs to be paid in order to design the component. Then, the designers who use the component share this price.

First, in order to capture a wide set of scenarios in practice, we associate with each component in the library two types of costs: a *private cost* and a *public cost*. The private cost models quality. It concerns the performance of the component and

We continue to our second extension of Lustig and Vardi's work. In their work, the library of components can be seen as if it is used by a single user. However, component libraries are typically used by multiple users simultaneously. The *quality* cost above can be thought of as a private cost. In the setting of multiple users, it makes sense to consider a *public* cost that is affected by the choices of the users. We assume that each component has a latency function as in RAGs. We distinguish between positive and negative effects for load. In the cost-sharing setting, the users who use a component share the price of constructing the component. On the other hand, and maybe more reasonable is the congestion case. Components can be seen as processors, then a higher load means a decrease in performance.

This setting gives rise to a RAG, which we refer to as a *component library game* (CLG, for short). A CLG is given by a shared library $\mathcal{L}$ and a specification $\mathcal{S}_i$ for each player. The resources in the corresponding RAG are the components. Player i's strategies are the correct designs with respect to $\mathcal{S}_i$. Note that a correct design $\mathcal{D}_i$ corresponds to a multiset of components, namely these components that the design uses.

We show that our good and bad news in terms of NE existence and equilibrium inefficiency carry over from multiset RAGs to CLGs. We study computational problems for CLGs and we focus on closed systems. Our results for cost-sharing and congestion games coincide. We show that finding a best-response is NP-complete and deciding the existence of NE is Σ_P^2-complete.

These results have been described in [11, 15] and in Chapters 3 and 4 of the thesis.

Cost-sharing scheduling games We study a restrictive class of multiset cost sharing games in which the players' multiset include a single resource. In this setting, players can be thought of as jobs and resources as machines. Each job

has a set of machines that can process it, and each such machine has a different processing time for the job. Each machine has an activation cost that needs to be covered by the jobs assigned to it. Jobs assigned to a particular machine share its cost proportionally to the load they generate.

Again, we study the three questions above. We study both unilateral and coordinated deviations, distinguishing between instance having unit or arbitrary machine-activation costs. Our results are detailed in Table 2, where k is the number of jobs and m is the number of machines.

Activation costs	Processing times	Pure Nash Equilibrium			Strong Equilibrium		
		$\exists$	PoA	PoS	$\exists$	SPoA	SPoS
Unit	arbitrary	yes	$\min\{m, k\}$	1	no	$\min\{m, \frac{k}{2} + \frac{1}{2}\}$	$\min\{\frac{m}{2}, \frac{k}{4} + \frac{1}{2}\}$
	machine-indp.	yes	$\min\{m, k\}$	1	yes	$\min\{\frac{m}{2}, \frac{k}{4} + \frac{1}{2}\}$	$\min\{\frac{m}{2}, \frac{k}{4} + \frac{1}{2}\}$
Arbitrary	arbitrary	no†	k	k	no	k	k
	machine-indp.	yes	$k^{\ddagger}$	k	yes$^{\mathcal{S}}$	k	k

Table 2: Summary of our results. (†) Deciding whether a PNE exists is NP-complete. (‡) Adopted to our model from [6]. ($\mathcal{S}$) Extension of [51].

These results have been described in [16] and do not appear in the thesis due to lack of space.

Dynamic resource allocation games A key feature of RAGs is that the players choose their strategies *in one shot* and *concurrently*. That is, a strategy for a player is a subset of the resources – chosen as a whole, and the players choose their strategies simultaneously. In many settings, however, resource sharing proceeds in a different way. First, in many settings, the choices of the players are made resource by resource as the game evolves. For example, when the network in an NFG models a map of roads and players are drivers choosing routes, it makes sense to allow each driver not to commit to a full route in the beginning of the game but rather to choose one road (edge) at each junction (vertex), gradually composing the full route according to the congestion observed. Second, in many settings, the choices of the players are not made concurrently. We describe an example of such a setting, which is inspired by [36]. We return to the example in which a network models a map of roads. Driving on a road takes time. Assume it takes a duration of one unit of time to complete an edge. Also, assume the players start driving on their paths at different times in $[0, 1]$. So, the players who start at time t will choose edges at times $t + 1, t + 2, \ldots$. Specifically, the set of players that start driving at time t choose concurrently. Moreover, they are aware of the choices of players that started driving at every time $t' \neq t$. With respect to these players, their choices are sequential.

We introduce and study *dynamic resource allocation games*, which allow the players to choose resources in an iterative and non-concurrent manner. A dynamic RAG is given by a pair $\mathcal{G} = \langle G, \nu \rangle$, where G is a k-player RAG and $\nu : \{1, \ldots, k\} \to \{1, \ldots, k\}$ is a *scheduler*. A dynamic RAG proceeds in *phases*. In each phase, each player chooses one resource. A phase is partitioned into *turns*, and the scheduler dictates which players proceed in each turn. Formally, Player i moves at turn j if $\nu(i) = j$. Note that the scheduler may assign the same turn to several players, in which case they choose a resource simultaneously in a phase. Once all turns have been taken, a phase is concluded and a new phase begins. There are two "extreme" schedulers: (1) A *sequential* scheduler assigns different turns to all players, i.e., ν is a permutation, reflecting the fact that the players make their choices sequentially, one player in each turn. (2) A *concurrent* scheduler assigns the same turn to all the player; i.e., $\nu(i) = 1$ for all $i \in \{1, \ldots, k\}$, reflecting the fact that all players proceed concurrently in the first (and only) turn in each phase. A *strategy* for a player in a dynamic RAG maps the history of choices made by the players so far (that is, the choices of all players in earlier phases as well as the choices of players that proceed in earlier turns in the current phase) and returns his next choice. A player finishes playing once the resources he has chosen satisfy the objective. The game terminates once all players finish playing. A strategy profile in the game is a vector of strategies – one for each player. The outcome of a profile is an assignment of a set of resources to each player. The cost of each players in a profile is induced by the costs of the resources in his set, which depends on their load and latency functions as in usual RAG.

We adjust the questions we asked on RAGs to the dynamic setting. We start with existence of equilibrium. We note that while the definition of NE applies to all games, in particular to dynamic ones, an NE is less suited in this case as it might include "uncredible threats". A more appropriate notion of equilibria *subgame perfect equilibrium* (SPE, for short) [50]. A strategy profile is a SPE if it represents an NE of every subgame of the original game. Informally, in an SPE the players must take into an account their observation of the game before making a choice.

We study the existence of SPE in different classes of dynamic RAGs. In addition to the classification of RAGs to cost-sharing games and congestion games, we also classify them by type of their strategies. We study *singleton* RAGs in which the player's actions consist of singletons of resources, and *symmetric* RAGs in which the players' actions are the same. We show that singleton symmetric congestion games are guaranteed to have an SPE. We show that this class is maximal by showing a singleton congestion game as well as a symmetric congestion with no SPE. We show that cost-sharing games are not guaranteed to have an SPE as well. Here, however we show that singleton cost-sharing games are guaranteed to have an SPE. We find

two of these results surprising. First, that an SPE is not guaranteed to exist, and second, that while in the simultaneous setting cost-sharing games are less stable than congestion games, the "order of stability" is not carried over to the dynamic setting.

Next, we study the inefficiency of equilibrium for the two classes of games that are guaranteed to have an SPE. We show that it coincides with the simultaneous setting in both cases. Finally, we study computational problems for dynamic RAGs. We show that deciding the existence of SPE is PSPACE-complete. We also study the problem of finding a schedule that admits an SPE under given constraints on the order the players move, and show that this problem is also PSPACE-complete.

These results are described in Chapter 5.

Repairing multi-player concurrent games The last work we present is more formal methods in nature and has a somewhat different flavor from the previous works. As mentioned above, synthesis is the automated construction of systems from their specifications [47]. Here, we take a different approach to synthesizing systems from components than the one we present above. We assume that the components interact, where each component has its own objective. Thus, we consider a game in which each component is modeled by a player.

The game that arrises from this setting is a multiplayer concurrent ongoing game [23]. Such a game is played by moving a token on a directed graph. At each state, players select concurrently an action, and the next position of the token is decided according to the the vector of actions they choose. So, an outcome of the game is an infinite sequence of states. Each player has an ω-regular objective that specifies which infinite paths meet his objective. Thus, the specification is a Boolean specification; a path either satisfies the specification or not. We list two examples of objectives. In *reachability* objectives each Player i has a set of states S_i. An infinite path π satisfies his objective if π crosses a state in S_i. In *Büchi* objectives each Player i again has a set of states S_i. An infinite path π satisfies the objective if it crosses S_i infinitely often. Note that a path might satisfy some of the players objectives while refuting others. Thus, the game is not a zero-sum game.

It is easy to find instances of such games with no NE. We introduce and study *repair* of multi-player games. We consider a setting with an authority (the designer) that aims to stabilize the interaction among the components and to increase the social welfare. In standard reactive synthesis [47], there are various ways to cope with situations when a specification is not realizable. Obviously, the specification has to be weakened, and this can be done either by adding assumptions on the behavior of the environment, fairness included, or by giving up some of the requirements on the system [28, 39]. In our setting, where the goal is to obtain stability, and the

game is not zero-sum, a repair may both weaken and strengthen the specifications, which, in our main model, is modeled by modifications to the winning conditions.

The input to the *specification-repair problem* (SR problem, for short) is a game along with a *cost function*, describing the cost of each repair. For example, in Büchi games the cost function specifies, for each vertex v and player i, the cost of making v accepting for Player i and the cost of making v rejecting. The cost may be 0, reflecting the fact that v is accepting or rejecting in the original specification of Player i, or it may be ∞, reflecting the fact that the original classification of v is a feature of the specification that the designer is not allowed to modify. We consider some useful classes of cost functions, like *uniform costs* – where all assignments cost 1, except for one that has cost 0 and stands for the original classification of the vertex, or *don't-care costs* – where several assignments have cost 0, reflecting a don't-care original classification, and all other assignments have cost ∞.

The goal of the designer is to suggest a repair to the winning conditions with which the game has an NE. One way to quantify the quality of a repair is its cost, and indeed the problem also gets as input a bound on the budget that can be used in the repairs. Another way, which has to do with the social welfare, considers the specifications that are satisfied in the obtained NE. Specifically, in the *rewarded specification-repair problem* (RSR problem, for short), the input also includes a *reward function* that maps subsets of specifications to rewards. When the suggested repair leads to an NE with a set W of "winners", i.e., players whose objective is satisfied, the designer gets a reward that corresponds to the specifications of the players in W. The quality of a solution then refers both to the budget it requires and to its reward.

Studying the SR and RSR problems, we distinguish between several classes, characterized by the type of winning conditions, cost functions, and reward functions. From a complexity point of view, we also distinguish between the case where the number of players is arbitrary and the one where it is constant. The problem of deciding whether an NE exists is known to be NP-complete with an arbitrary number of players for most common ω-regular objectives, excluding Büchi where the complexity is polynomial [23]. It is not too hard to lift the NP lower bound to the SR and RSR problems. The main challenge is the Büchi case, where one should find the cases where the polynomial complexity of deciding whether an NE exists can be lifted to the SR and RSR problems, and the cases where the need to find a repair shifts the complexity of the problem to NP. We show that the polynomial complexity can be maintained for don't-care costs, but the other settings are NP-complete. We then check whether fixing the number of players can reduce the complexity of the SR and RSR problems, either by analyzing the complexity of the algorithms for an arbitrary number of players, or by introducing new algorithms. We show that

in many cases, we can solve the problem in polynomial time, mainly thanks to the fact that it is possible to go over all possible subsets of players in search for a subset that can win in an NE.

These results have been described in [1] and do not appear in the thesis due to lack of space.

1.2 Quantitative Formal Methods

Game theory is quantitative in nature: every player has a value in an outcome of a game, which she tries to increase. In the previous section we added aspects from formal methods to game theory. Namely, we went from reachability objectives in network formation games to richer objectives, we considered dynamics in resource allocation games, and we considered an application of games to synthesis, which is a problem in formal methods. In the second part, we take a dual approach. We add aspects from game theory to traditional formal methods, and we focus in the addition of quantitative outcomes.

Weighted automata The *automata-theoretic* approach uses the theory of automata as a unifying paradigm for system specification and verification [52, 53]. By viewing computations as words (over the alphabet of possible assignments to variables of the system), we can view both the system and its specification as languages. Questions like satisfiability of specifications or their satisfaction can then be reduced to questions about automata and their languages.

Traditional automata accept or reject their input, and are therefore Boolean. In recent years, there is growing need and interest in quantitative reasoning. *Weighted finite automaton* (WFA, for short) map words to numerical values. Technically, every transition in a weighted automaton $\mathcal{A}$ has a value, and the value of a run is the sum of the costs of the transitions. The value that $\mathcal{A}$ assigns to a finite word w, denoted $val(\mathcal{A}, w)$, is either the value of the most expensive or cheapest accepting run of $\mathcal{A}$ on w, depending on the application[2].

The rich structure of weighted automata makes them intriguing mathematical objects. Fundamental problems that have been solved decades ago for Boolean automata are still open or known to be undecidable in the weighted setting [44]. For example, while in the Boolean setting, nondeterminism does not add to the expressive power of the automata, not all weighted automata can be determinized, and the problem of deciding whether a given nondeterministic weighted automaton

[2]In general, weighted automata may be defined with respect to all semirings. For our applications here, we consider WFAs over $\mathbb{Q}$, with the addition of the semi-ring being max or min and its multiplication being $+$.

can be determinized is still open, in the sense we do not even know whether it is decidable.

A problem of great interest in the context of automata is the *containment* problem. In the Boolean setting, the containment problem asks, given two automata $\mathcal{A}$ and $\mathcal{B}$, whether all the words in Σ^* that are accepted by $\mathcal{A}$ are also accepted by $\mathcal{B}$. In the weighted setting, the "goal" of words is not just to get accepted, but also to do so with a maximal value. Accordingly, the containment problem for WFAs asks, given two WFAs $\mathcal{A}$ and $\mathcal{B}$, whether every word accepted by $\mathcal{A}$ is also accepted by $\mathcal{B}$, and its value in $\mathcal{A}$ is less than or equal to its value in $\mathcal{B}$. We then say that $\mathcal{B}$ contains $\mathcal{A}$, denoted $\mathcal{A} \subseteq \mathcal{B}$. In the Boolean setting, the containment problem is PSPACE-complete [41]. In the weighted setting, the problem is in general undecidable [2, 38]. The problem is indeed of great interest: In the automata-theoretic approach to reasoning about systems and their specifications, containment amounts to correctness of systems with respect to their specifications. The same motivation applies for weighted systems, with the specifications being quantitative [27].

Making weighted containment feasible We suggest here a heuristic approach to bypass the undecidability of the weighed containment problem. Even in the Boolean setting, where the containment problem is decidable, its PSPACE complexity is an obstacle in practice and researchers have suggested two orthogonal methods for coping with it. One is to replace containment by a pre-order that is easier to check, with the leading such pre-order being the *simulation* preorder [43]. A second method, useful also in other paradigms for reasoning about the huge, and possibly infinite, state space of systems is *abstraction* [19, 30]. Essentially, in abstraction we hide some of the information about the system.

We apply both techniques to the weighted case. First, we extend the simulation preorder of the Boolean setting to WFAs. For two WFAs $\mathcal{A}$ and $\mathcal{B}$, we denote by $\mathcal{A} \leqslant \mathcal{B}$ the fact that $\mathcal{A}$ simulates $\mathcal{B}$. We show that $\mathcal{A} \leqslant \mathcal{B}$ implies weighted language containment. Also, we show that deciding whether $\mathcal{A}$ simulates $\mathcal{B}$ can be done in NP $\cap$ coNP.

We then extend abstraction to the weighted case. Here, we assume that the given WFAs $\mathcal{A}$ and $\mathcal{B}$ are equipped with abstraction functions α and β, respectively. Using these functions we construct over-approximation $\mathcal{A}^\alpha_\uparrow$ and $\mathcal{B}^\beta_\uparrow$ and under approximations $\mathcal{A}^\alpha_\downarrow$ and $\mathcal{B}^\beta_\downarrow$, of $\mathcal{A}$ and $\mathcal{B}$, respectively. It is not hard to see that if $\mathcal{A}^\alpha_\uparrow \subseteq \mathcal{B}^\beta_\downarrow$, then $\mathcal{A} \subseteq \mathcal{B}$, and that if $\mathcal{A}^\alpha_\downarrow \not\subseteq \mathcal{B}^\beta_\uparrow$, then $\mathcal{A} \not\subseteq \mathcal{B}$. We show that the above is valid not just of containment but also for our weighted-simulation relation. This gives rise to the following heuristics. We start by checking $\mathcal{A}^\alpha_\uparrow \leqslant \mathcal{B}^\beta_\downarrow$ and $\mathcal{A}^\alpha_\downarrow \not\leqslant \mathcal{B}^\beta_\uparrow$, for some (typically coarse) initial abstraction functions α and β. If we are lucky and one of them holds, we are done. Otherwise, we use information from the decision

procedure to refine the abstractions.

These results have been described in [8] and do not appear in the thesis due to lack of space.

When does abstraction help? While on the topic of abstraction, we make a short detour to Boolean formal methods. The biggest advantage of abstraction of DFAs is that it reduces the state space. One of its disadvantages is that it increases the nondeterminism. In particular, an abstraction of a DFA need not be deterministic. The fact abstraction does not preserve determinism is a serious drawback as determinisism makes most algorithms simpler and it is even crucial in some settings.

We ask whether, given the need to determinize an abstract automaton, abstraction still leads to smaller automata. Formally, consider a deterministic finite automaton (DFA, for short) $\mathcal{A}$, and let $\mathcal{A}_\alpha$ be a nondeterministic finite automaton obtained from $\mathcal{A}$ by applying an abstraction function α. Let $\mathcal{D}_\alpha$ be the minimal DFA equivalent to $\mathcal{A}_\alpha$. We ask whether $\mathcal{D}_\alpha$ is smaller than $\mathcal{A}$. If so, we say that α is helpful.

We show that, surprisingly, abstractions are not always helpful. In fact, we show a family of DFAs and abstraction functions for them for which the abstract automata are exponentially bigger than the original automata. We also study the problem of deciding whether a given abstraction function is helpful for a given DFA and show that it is PSPACE-complete.

These results have been described in [9] and do not appear in the thesis due to lack of space.

Parameterized weighted containment In addition to verification, the automata-theoretic approach in the Boolean setting has proven useful also in reasoning about *partially-specified* systems and specifications, where some components are not known or hidden. Partially-specified systems are used mainly in *stepwise design*: One starts with a system with "holes" and iteratively completes them in a way that satisfies some specification [31, 32]. From the other direction, partially-specified specifications are used for system exploration. A primary example is *query checking*: [26], the specification contains variables, and the goal is to find a maximal assignment to the variables with which the explored system satisfies the specification.

We study partial specified systems and specifications in the quantitative setting. We introduce and study *parameterized weighted containment* (PWC, for short): given three WFAs $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$, with $\mathcal{B}$ being partial, the goal is to find an assignment to the missing costs in $\mathcal{B}$ so that we end up with $\mathcal{B}'$ for which $\mathcal{A} \subseteq \mathcal{B}' \subseteq \mathcal{C}$. We also consider a one-bound version of the problem, where only $\mathcal{A}$ or only $\mathcal{C}$ are given

in addition to $\mathcal{B}$, and the goal is to find a minimal assignment with which $\mathcal{A} \subseteq \mathcal{B}'$ or, respectively, a maximal one with which $\mathcal{B}' \subseteq \mathcal{C}$.

Since weighted containment is undecidable, we restrict the problem in two aspects. First, we study the PWC problem where all three WFAs are deterministic. We show that the problem can be solved in polynomial time and the solution is based on strong mathematical tools. We describe a convex polytope $P \subseteq \mathbb{R}^k$ that includes exactly all the legal assignments for the missing costs in $\mathcal{B}$. The polytope P is defined using exponentially many constraints, so reasoning about it naively would give an exponential time algorithm. Fortunately, we are able to represent the constraints in a compact manner using a *separation oracle*, and use the results of [33, 34, 46] to reason about the polytope efficiently.

Our second restriction of the problem is to replace weighted containment with weighted simulation, which we describe above and is decidable in NP$\cap$coNP. We argue that the one-bound problem is not interesting, as a minimal/maximal solution need not exist. For the two bound problem, we show that the problem is NP-complete. Given the computational difficulty of handling nondeterministic WFAs in general, we view these results as good news

These results have been described in [10] and do not appear in the thesis due to lack of space.

Stochastization of weighted automata *Probabilistic automata* (PFAs, for short) where introduced by Rabin in the 60s [48]. The idea is to replace nondeterminism with probability. Each transition in a PFA has a probability, the probability of a run is the product of the probabilities of the transitions it traverses, and the "value" of a word is the probability of the accepting runs on it. Thus, it is a number in $[0, 1]$.

We combine the probabilistic ideas in PFAs with the quantitative ideas of WFAs to obtain a *probabilistic weighted finite automaton* (PWFA, for short). There, each transition has two weights, which we refer to as the *cost* and the *probability*[3]. The weight that the PWFA assigns to a word is then the expected cost of the runs on it. That is, as in WFAs, the cost of each run is the sum the costs of the transitions along the run, and as in PFAs, the contribution of each run to the weight of a word depends on both its cost and probability. While PFAs have been extensively studied (e.g., [18]), we are only aware of [35] in which PWFAs were considered.

We introduce and study *stochastization* of WFAs. Given a WFA $\mathcal{A}$, stochastization turns it into a PWFA $\mathcal{A}'$ by labeling its transitions with probabilities. Recall that in a WFA, the weight of a word is the minimal weight of a run on it. Stochasti-

[3]For technical reasons we assume here that no run of a WFA gets stuck and all runs are accepting.

zation of a WFA $\mathcal{A}$ results in a PWFA $\mathcal{A}'$ with the same set of runs, and the weight of a word is the expected cost of these runs. Accordingly, the weight of a word in $\mathcal{A}'$ can only increase with respect to its weight in $\mathcal{A}$. Hence, we seek stochastizations in which $\mathcal{A}'$ *α-approximates* $\mathcal{A}$ for the minimal possible factor $\alpha \geqslant 1$. That is, the value of every word in $\mathcal{A}'$ is at most α times its value in $\mathcal{A}$.

We describe one of the motivatations for stochastization of WFAs. In [5], the authors describe a framework for using WFAs in order to reason about *online algorithms*. An online algorithm can be viewed as a reactive system: at each round, the environment issues a request, and the algorithm should process it. The sequence of requests is not known in advance, and the goal of the algorithm is to minimize the overall cost of processing the sequence. The most interesting question about an online algorithm refers to its *competitive ratio*: the worst-case (with respect to all input sequences) ratio between the cost of the algorithm and the cost of an optimal solution – one that may be given by an *offline* algorithm, which knows the input sequence in advance. An online algorithm that achieves a competitive ratio α is said to be *α-competitive*.

The framework in [5] models optimization problems by WFAs, relates the "unbounded look ahead" of the optimal offline algorithm with nondeterminism, and relates the "no look ahead" of online algorithms with determinism. So, given a WFA $\mathcal{A}$ that models an online optimization problem, the authors study the problem of finding a determinization $\mathcal{D}$ of $\mathcal{A}$ that maintains the same states as the WFA and only *prunes* transitions. Such a determinization corresponds to an online algorithm. They show that if $\mathcal{D}$ α-approximates $\mathcal{A}$, then the online algorithm that corresponds to $\mathcal{D}$ is α-competitive.

We broaden the framework by considering *randomized* online algorithms, namely ones that may toss coins in order to choose their actions. Indeed, it is well known that many online algorithms that use randomized strategies achieve a better competitive ratio [22]. Similar to the above, given a WFA $\mathcal{A}$, we search for an α-stochastization $\mathcal{P}$ of $\mathcal{A}$. A randomized online algorithm that corresponds to $\mathcal{P}$ is then α-competitive.

Given a WFA $\mathcal{A}$ and a factor $\alpha \geqslant 1$, the *approximated stochastization problem* (AS problem, for short) is to decide whether there is a stochastization of $\mathcal{A}$ that α-approximates it. We study the AS problem and show that it is in general undecidable. Special tractable cases include two types of restrictions. First, restrictions on α: we show that when $\alpha = 1$, the problem coincides with determinization by pruning of WFAs, which can be solved in polynomial time [5]. Then, restrictions on the structure of the WFA: we define the class of constant-ambiguous WFAs, namely WFAs whose degree of nondeterminism is a constant, and show that the AS problem for them is in PSPACE. On the other hand, the AS problem is NP-hard already for 7-ambiguous WFAs, namely WFAs that have at most 7 runs on each word. Even

more restricted are tree-like WFAs, for which the problem can be solved in polynomial time, and so is the problem of finding a minimal approximation factor α. We show that these restricted classes are still expressive enough to model interesting optimization problems.

These results have been described in [12] and do not appear in the thesis due to lack of space.

References

[1] S. Almagor, G. Avni, and O. Kupferman. Repairing multi-player games. In *26th International Conference on Concurrency Theory, CONCUR 2015, Madrid, Spain, September 1.4, 2015*, pages 325–339, 2015.

[2] S. Almagor, U. Boker, and O. Kupferman. What's decidable about weighted automata? In *9th Int. Symp. on Automated Technology for Verification and Analysis*, volume 6996 of *Lecture Notes in Computer Science*, pages 482–491. Springer, 2011.

[3] S. Almagor, U. Boker, and O. Kupferman. Formalizing and reasoning about quality. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*, pages 15–27, 2013.

[4] S. Almagor, U. Boker, and O. Kupferman. Discounting in LTL. In *Tools and Algorithms for the Construction and Analysis of Systems - 20th International Conference, TACAS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014. Proceedings*, pages 424–439, 2014.

[5] B. Aminof, O. Kupferman, and R. Lampert. Reasoning about online algorithms with weighted automata. *ACM Transactions on Algorithms*, 6(2), 2010.

[6] E. Anshelevich, A. Dasgupta, J. Kleinberg, E. Tardos, T. Wexler, and T. Roughgarden. The price of stability for network design with fair cost allocation. *SIAM J. Comput.*, 38(4):1602–1623, 2008.

[7] K.R. Apt and E. Grädel. *Lectures in Game Theory for Computer Scientists*. Cambridge University Press, 2011.

[8] G. Avni and O. Kupferman. Making weighted containment feasible: A heuristic based on simulation and abstraction. In *Proc. 23rd Int. Conf. on Concurrency Theory*, volume 7454, pages 84–99, 2012.

[9] G. Avni and O. Kupferman. When does abstraction help? *Information Processing Letters*, 113:901–905, 2013.

[10] G. Avni and O. Kupferman. Parameterized weighted containment. *ACM Trans. Comput. Log.*, 16(1):6:1–6:25, 2014.

[11] G. Avni and O. Kupferman. Synthesis from component libraries with costs. In *Proc. 25th Int. Conf. on Concurrency Theory*, pages 156–172, 2014.

[12] G. Avni and O. Kupferman. Stochastization of weighted automata. In *Mathematical Foundations of Computer Science 2015 - 40th International Symposium, MFCS 2015, Milan, Italy, August 24-28, 2015, Proceedings, Part I*, pages 89–102, 2015.

[13] G. Avni, O. Kupferman, and T. Tamir. From reachability to temporal specifications in cost-sharing games. In *Automated Reasoning - 7th International Joint Conference, IJCAR 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 19-22, 2014. Proceedings*, pages 1–15, 2014.

[14] G. Avni, O. Kupferman, and T. Tamir. Network-formation games with regular objectives. In *Proc. 17th Int. Conf. on Foundations of Software Science and Computation Structures*, volume 8412 of *Lecture Notes in Computer Science*, pages 119–133. Springer, 2014.

[15] G. Avni, O. Kupferman, and T. Tamir. Congestion games with multisets of resources and applications in synthesis. In *Proc. 35th Conf. on Foundations of Software Technology and Theoretical Computer Science*, pages 365–379, 2015.

[16] G. Avni and T. Tamir. Cost-sharing scheduling games on restricted unrelated machines. In *Algorithmic Game Theory - 8th International Symposium, SAGT 2015, Saarbrücken, Germany, September 28-30, 2015, Proceedings*, pages 69–81, 2015.

[17] B. Awerbuch, Y. Azar, and A. Epstein. The price of routing unsplittable flow. *SIAM J. Comput.*, 42(1):160–177, 2013.

[18] P. Azaria. *Introduction to Probabilistic Automata (Computer Science and Applied Mathematics)*. Academic Press, Inc., Orlando, FL, USA, 1971.

[19] T. Ball, E. Bounimova, B. Cook, V. Levin, J. Lichtenberg, C. McGarvey, B. Ondrusek, S.K. Rajamani, and A. Ustuner. Thorough static analysis of device drivers. In *EuroSys*, 2006.

[20] V. Bilò. A unifying tool for bounding the quality of non-cooperative solutions in weighted congestion games. In *WAOA*, pages 215–228, 2012.

[21] U. Boker, K. Chatterjee, T. A. Henzinger, and O. Kupferman. Temporal specifications with accumulative values. *ACM Trans. Comput. Log.*, 15(4):27:1–27:25, 2014.

[22] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[23] P. Bouyer, R. Brenguier, and N. Markey. Nash equilibria for reachability objectives in multi-player timed games. In *CONCUR 2010 - Concurrency Theory, 21th International Conference, CONCUR 2010, Paris, France, August 31-September 3, 2010. Proceedings*, pages 192–206, 2010.

[24] P. Bouyer, U. Fahrenberg, K. Larsen, N. Markey, and J. Srba. Infinite runs in weighted timed automata with energy constraints. In *FORMATS*, pages 33–47, 2008.

[25] I. Caragiannis, M. Flammini, C. Kaklamanis, P. Kanellopoulos, and L. Moscardelli. Tight bounds for selfish and greedy load balancing. *Algorithmica*, 61(3):606–637, 2011.

[26] W. Chan. Temporal-logic queries. In *Proc. 12th Int. Conf. on Computer Aided Verification*, volume 1855 of *Lecture Notes in Computer Science*, pages 450–463. Springer, 2000.

[27] K. Chatterjee, L. Doyen, and T. Henzinger. Quantative languages. In *Proc. 17th Annual Conf. of the European Association for Computer Science Logic*, pages 385–400, 2008.

[28] K. Chatterjee, T. Henzinger, and B. Jobstmann. Environment assumptions for synthesis. In *Proc. 19th Int. Conf. on Concurrency Theory*, volume 5201 of *Lecture Notes in Computer Science*, pages 147–161. Springer, 2008.

[29] G. Christodoulou and E. Koutsoupias. On the price of anarchy and stability of correlated equilibria of linear congestion games. In *ESA*, pages 59–70, 2005.

[30] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for the static analysis of programs by construction or approximation of fixpoints. In *Proc. 4th ACM Symp. on Principles of Programming Languages*, pages 238–252. ACM, 1977.

[31] E.W. Dijkstra. *A Discipline of Programming*. Prentice-Hall, 1976.

[32] L. Fix, N. Francez, and O. Grumberg. Program composition and modular verification. In *Proc. 18th Int. Colloq. on Automata, Languages, and Programming*, pages 93–114, 1991.

[33] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[34] R. Karp and C. Papadimitriou. On linear characterizations of combinatorial optimization problems. In *Proc. 21st IEEE Symp. on Foundations of Computer Science*, pages 1–9, 1980.

[35] S. Kiefer, A. S. Murawski, J. Ouaknine, B. Wachter, and J. Worrell. On the complexity of equivalence and minimisation for q-weighted automata. *Logical Methods in Computer Science*, 9(1), 2013.

[36] R. Koch and M. Skutella. Nash equilibria and the price of anarchy for flows over time. *Theory Comput. Syst.*, 49(1):71–97, 2011.

[37] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. *Computer Science Review*, 3(2):65–69, 2009.

[38] D. Krob. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. *International Journal of Algebra and Computation*, 4(3):405–425, 1994.

[39] W. Li, L. Dworkin, and S. A. Seshia. Mining assumptions for synthesis. In *9th IEEE/ACM International Conference on Formal Methods and Models for Codesign, MEMOCODE 2011, Cambridge, UK, 11-13 July, 2011*, pages 43–50, 2011.

[40] Y. Lustig and M.Y. Vardi. Synthesis from component libraries. *STTT*, 15(5-6):603–618, 2013.

[41] A.R. Meyer and L.J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential time. In *Proc. 13th IEEE Symp. on Switching and Automata Theory*, pages 125–129, 1972.

[42] I. Milchtaich. Congestion games with player-specific payoff functions. *Games and Economic Behavior*, 13(1):111 – 124, 1996.

[43] R. Milner. An algebraic definition of simulation between programs. In *Proc. 2nd Int. Joint Conf. on Artificial Intelligence*, pages 481–489. British Computer Society, 1971.

[44] M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.

[45] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.

[46] M.W. Padberg and M. R. Rao. *The Russian Method and Integer Programming*. Working paper series (Salomon Brothers Center for the Study of Financial Institutions). Salomon Brothers Center for the Study of Financial Institutions, 1980.

[47] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. 16th ACM Symp. on Principles of Programming Languages*, pages 179–190, 1989.

[48] M. O. Rabin. Probabilistic automata. *Information and Control*, 6:230–245, 1963.

[49] R.W. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2:65–67, 1973.

[50] R. Selten. Spieltheoretische behandlung eines oligopolmodells mit nachfrageträgheit. *Zeitschrift für die gesamte Staatswissenschaft*, 121, 1965.

[51] V. Syrgkanis. The complexity of equilibria in cost sharing games. In *Proc. of the 6th International Conference on Internet and Network Economics*, WINE'10, pages 366–377, 2010.

[52] W. Thomas. Automata on infinite objects. *Handbook of Theoretical Computer Science*, pages 133–191, 1990.

[53] M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.

Network-Formation Games with Regular Objectives*

Guy Avni[†] *Orna Kupferman*[‡] *Tami Tamir*[§]

Abstract

Classical network-formation games are played on a directed graph. Players have reachability objectives, and each player has to select a path satisfying his objective. Edges are associated with costs, and when several players use the same edge, they evenly share its cost. The theoretical and practical aspects of network-formation games have been extensively studied and are well understood. We introduce and study *network-formation games with regular objectives*. In our setting, the edges are labeled by alphabet letters and the objective of each player is a regular language over the alphabet of labels, given by means of an automaton or a temporal-logic formula. Thus, beyond reachability properties, a player may restrict attention to paths that satisfy certain properties, referring, for example, to the providers of the traversed edges, the actions associated with them, their quality of service, security, etc.

Unlike the case of network-formation games with reachability objectives, here the paths selected by the players need not be simple, thus a player may traverse some transitions several times. Edge costs are shared by the players with the share being proportional to the number of times the transition is traversed. We study the existence of a pure Nash equilibrium (NE), convergence of best-response-dynamics, the complexity of finding the social optimum, and the inefficiency of a NE compared to a social-optimum solution. We examine several classes of networks (for example, networks with uniform edge costs, or alphabet of size 1) and several classes of regular objectives. We show that many properties of classical network-formation games are no longer valid in our game. In particular, a pure NE might not exist and the Price of Stability equals the number of players (as opposed to logarithmic in the number of players in the classic setting, where a pure NE always exists). In light of these results, we also present special cases for which the resulting game is more stable.

*Published is the proceedings of the 17th Foundations of Software Science and Computation Structures, LNCS 8412, pages 119–133, Springer, 2014, and in the proceedings of the 7th International Joint Conference, LNCS 8562, pages 1–15, Springer, 2014. A full version was submitted.

[†]School of Computer Science and Engineering, The Hebrew University, Jerusalem, Israel

[‡]School of Computer Science and Engineering, The Hebrew University, Jerusalem, Israel

[§]School of Computer Science, The Interdisciplinary Center, Herzliya, Israel

1 Introduction

Network design and formation is a fundamental well-studied challenge that involves many interesting combinatorial optimization problems. In practice, network design is often conducted by multiple strategic users whose individual costs are affected by the decisions made by others. Early works on network design focus on analyzing the efficiency and fairness properties associated with different sharing rules (e.g., [27, 35]). Following the emergence of the Internet, there has been an explosion of studies employing game-theoretic analysis to explore Internet applications, such as routing in computer networks and network formation [21, 1, 16, 2]. In network-formation games (for a survey, see [40]), the network is modeled by a weighted graph. The weight of an edge indicates the cost of activating the transition it models, which is independent of the number of times the edge is used. Players have reachability objectives, each given by sets of possible source and target nodes. Players share the cost of edges used in order to fulfill their objectives. Since the costs are positive, the runs traversed by the players are simple. Under the common Shapley cost-sharing mechanism, the cost of an edge is shared evenly by the players that use it.

The players are selfish agents who attempt to minimize their own costs, rather than to optimize some global objective. In network-design settings, this would mean that the players selfishly select a path instead of being assigned one by a central authority. The focus in game theory is on the *stable* outcomes of a given setting, or the *equilibrium* points. A Nash equilibrium (NE) is a profile of the players' strategies such that no player can decrease his cost by an unilateral deviation from his current strategy, that is, assuming that the strategies of the other players do not change.[1]

Reachability objectives enable the players to specify possible sources and targets. Often, however, it is desirable to refer also to other properties of the selected paths. For example, in a *communication* setting, edges may belong to different providers, and a user may like to specify requirements like "all edges are operated by the same provider" or "no edge operated by AT&T is followed by an edge operated by Verizon". Edges may also have different quality or security levels (e.g., "noisy channel", "high-bandwidth channel", or "encrypted channel"), and again, users may like to specify their preferences with respect to these properties. In *planning* or in *production systems*, nodes of the network correspond to configurations, and edges correspond to the application of actions. The objectives of the players are sequences of actions that fulfill a certain plan, which is often more involved than just reachability [25]; for example "once the arm is up, do not put it down until the block is placed".

[1] Throughout this paper, we concentrate on pure strategies and pure deviations, as is the case for the vast literature on cost-sharing games.

The challenge of reasoning about behaviors has been extensively studied in the context of formal verification. While early research concerned the input-output relations of terminating programs, current research focuses on on-going behaviors of reactive systems [26]. The interaction between the components of a reactive system correspond to a multi-agent game, and indeed in recent years we see an exciting transfer of concepts and ideas between the areas of game theory and formal verification: logics for specifying multi-agent systems [3, 13], studies of equilibria in games that correspond to the synthesis problem [12, 11, 20], an extension of mechanism design to on-going behaviors [30], studies of non-zero-sum games in formal methods [14, 10], and more.

In this paper we extend network-formation games to a setting in which the players can specify regular objectives. This involves two changes of the underlying setting: First, the edges in the network are labeled by letters from a designated alphabet. Second, the objective of each player is specified by a *language* over this alphabet. Each player should select a path labeled by a word in his objective language. Thus, if we view the network as a *nondeterministic weighted finite automaton* (WFA) $\mathcal{A}$, then the set of strategies for a player with objective L is the set of accepting runs of $\mathcal{A}$ on some word in L. Accordingly, we refer to our extension as *automaton-formation games*. As in classical network-formation games, players share the cost of edges they use. Unlike the classical game, the runs selected by the players need not be simple, thus a player may traverse some edges several times. Edge costs are shared by the players, with the share being proportional to the number of times the edge is traversed. This latter issue is the main technical difference between automaton-formation and network-formation games, and as we shall see, it is very significant.

Many variants of cost-sharing games have been studied. A generalization of the network-formation game of [2] in which players are weighted and a player's share in an edge cost is proportional to its weight is considered in [15], where it is shown that the weighted game does not necessarily have a pure NE. In congestion games, sharing of a resource increases its cost. Studied variants of congestion games include settings in which players' payments depend on the resource they choose to use, the set of players using this resource, or both [34, 31, 32, 23]. In some of these variants a NE is guaranteed to exist while in others it is not.

All the variants above are different from automaton-formation games, where a player needs to select a *multiset* of resources (namely, the edges he is going to traverse) rather than a set without repetitions. In the context of formal methods, an appealing application of such games is that of *synthesis from components*, where the resources are components from a library, and agents need to synthesize their objectives using the components, possibly by a repeated use of some components.

In some settings, the components have construction costs (e.g., the money paid to the designer of the component), in which case the corresponding multiset game is a cost-sharing game [6], and our results here can be generalized to apply for this settings. In other settings, the components have congestion effects (e.g., the components are CPUs, and the more players that use them, the slower the performance is), in which case the corresponding game is a multiset congestion game [9].

We study the theoretical and practical aspects of automaton-formation games. In addition to the general game, we consider classes of instances that have to do with the network, the specifications, or to their combination. Recall that the network can be viewed as a WFA $\mathcal{A}$. We consider the following classes of WFAs: (1) *all-accepting*, in which all the states of $\mathcal{A}$ are accepting, thus its language is prefix closed (2) *uniform costs*, in which all edges have the same cost, and (3) *single letter*, in which $\mathcal{A}$ is over a single-letter alphabet. We consider the following classes of specifications: (1) *single word*, where the language of each player is a single word, (2) *symmetric*, where all players have the same objective. We also consider classes of instances that are intersections of the above classes.

Each of the restricted classes we consider corresponds to a real-life variant of the general setting. Let us elaborate below on single-letter instances. The language of an automaton over a single letter $\{a\}$ induces a subset of $\mathbb{N}$, namely the numbers $k \in \mathbb{N}$ such that the automaton accepts a^k. Accordingly, single-letter instances correspond to settings in which a player specifies possible lengths of paths. Several communication protocols are based on the fact that a message must pass a pre-defined length before reaching its destination. This includes *onion routing*, where the message is encrypted in layers [38], or *proof-of-work* protocols that are used to deter denial of service attacks and other service abuses such as spam (e.g., [19]).

We provide a complete picture of the following questions for various instances (for formal definitions, see Section 2): (*i*) Existence of a *pure Nash equilibrium*. That is, whether each instance of the game has a profile of pure strategies that constitutes a NE. As we show, unlike the case of classical network design games, a pure NE might not exist in general automaton-formation games and even in very restricted instances of it. (*ii*) The complexity of finding the *social optimum* (SO). The SO is a profile that minimizes the total cost of the edges used by all players; thus the one obtained when the players obey some centralized authority. We show that for some restricted instances finding the SO can be done efficiently, while for other restricted instances, the complexity agrees with the NP-completeness of classical network-formation games. (*iii*) An analysis of *equilibrium inefficiency*. It is well known that decentralized decision-making may lead to solutions that are sub-optimal from the point of view of society as a whole. We quantify the inefficiency incurred due to selfish behavior according to the *price of anarchy* (PoA) [29, 36] and *price of stability*

(PoS) [2] measures. The PoA is the worst-case inefficiency of a Nash equilibrium (that is, the ratio between the worst NE and the SO). The PoS is the best-case inefficiency of a Nash equilibrium (that is, the ratio between the best NE and the SO). We show that while the PoA in automaton-formation games agrees with the one in classical network-formation games and is equal to the number of players, the PoS also equals the number of players, again already in very restricted instances. This is in contrast with classical network-formation games, where the PoS tends to *log* the number of players. Thus, the fact that players may choose to use edges several times significantly increases the challenge of finding a stable solution as well as the inefficiency incurred due to selfish behavior. We find this as the most technically challenging result of this work. We do manage to find structural restrictions on the network with which the social optimum is a NE.

The technical challenge of our setting is demonstrated in the seemingly easy instance in which all players have the same objective. Such *symmetric* instances are known to be the simplest to handle in all cost-sharing and congestion games studied so far. Specifically, in network-formation games, the social optimum in symmetric instances is also a NE and the PoS is 1. Moreover, in some games [22], computing a NE is PLS-complete in general, but solvable in polynomial time for symmetric instances. Indeed, once all players have the same objective, it is not conceivable that a player would want to deviate from the social-optimum solution, where each of the k players pays $\frac{1}{k}$ of the cost of the optimal solution. We show that, surprisingly, symmetric instances in AF-games are not simple at all. Specifically, a NE is not guaranteed to exist in the general case, and in single-letter networks, the social optimum might not be a NE, and the PoS is at least $\frac{k}{k-1}$. In particular, for symmetric two-player AF games, we have that $PoS = PoA = 2$. We also show that the PoA equals the number of players already for very restricted instances.

2 Preliminaries

2.1 Automaton-formation games

A *nondeterministic finite weighted automaton* on finite words (WFA, for short) is a tuple $\mathcal{A} = \langle \Sigma, Q, \Delta, q_0, F, c \rangle$, where Σ is an alphabet, Q is a set of states, $\Delta \subseteq Q \times \Sigma \times Q$ is a transition relation, $q_0 \in Q$ is an initial state, $F \subseteq Q$ is a set of accepting states, and $c : \Delta \to \mathbb{R}$ is a function that maps each transition to the cost of its formation [33]. A *run* of $\mathcal{A}$ on a word $w = w_1, \ldots, w_n \in \Sigma^*$ is a sequence of states $\pi = \pi^0, \pi^1, \ldots, \pi^n$ such that $\pi^0 = q_0$ and for every $0 \leqslant i < n$ we have $\Delta(\pi^i, w_{i+1}, \pi^{i+1})$. The run π is *accepting* iff $\pi^n \in F$. The *length* of π is n, whereas its size, denoted $|\pi|$, is the number of different transitions in it. Note that $|\pi| \leqslant n$.

An *automaton-formation game* (AF game, for short) between k selfish players is a pair $\langle \mathcal{A}, O \rangle$, where $\mathcal{A}$ is a WFA over some alphabet Σ and O is a k-tuple of regular languages over Σ. Thus, the objective of Player i is a regular language L_i, and he needs to choose a word $w_i \in L_i$ and an accepting run of $\mathcal{A}$ on w_i in a way that minimizes his payments. The cost of each transition is shared by the players that use it in their selected runs, where the share of a player in the cost of a transition e is proportional to the number of times e is used by the player. Formally, The set of strategies for Player i is $\mathcal{S}_i = \{\pi : \pi \text{ is an accepting run of } \mathcal{A} \text{ on some word in } L_i\}$. We assume that $\mathcal{S}_i$ is not empty. We refer to the set $\mathcal{S} = \mathcal{S}_1 \times \ldots \times \mathcal{S}_k$ as the set of *profiles* of the game.

Consider a profile $P = \langle \pi_1, \pi_2, \ldots, \pi_k \rangle$. We refer to π_i as a sequence of transitions. Let $\pi_i = e_i^1, \ldots, e_i^{\ell_i}$, and let $\eta_P : \Delta \to \mathbb{N}$ be a function that maps each transition in Δ to the number of times it is traversed by all the strategies in P, taking into an account several traversals in a single strategy. Denote by $\eta_i(e)$ the number of times e is traversed in π_i, that is, $\eta_i(e) = |\{1 \leqslant j \leqslant \ell_i : e_i^j = e\}|$. Then, $\eta_P(e) = \sum_{i=1\ldots k} \eta_i(e)$. The *cost of player i in the profile P* is

$$cost_i(P) = \sum_{e \in \pi_i} \frac{\eta_i(e)}{\eta_P(e)} c(e). \tag{1}$$

For example, consider the WFA $\mathcal{A}$ depicted in Figure 1. The label $e_1 : a, 1$ on the transition from q_0 to q_1 indicates that this transition, which we refer to as e_1, traverses the letter a and its cost is 1. We consider a games between two players. Player 1's objective is the language is $L_1 = \{ab^i : \ i \geqslant 2\}$ and Player 2's language is $\{ab, ba\}$. Thus, $\mathcal{S}_1 = \{\{e_1, e_2, e_2\}, \{e_1, e_2, e_2, e_2\}, \ldots\}$ and $\mathcal{S}_2 = \{\{e_3, e_4\}, \{e_1, e_2\}\}$. Consider the profile $P = \langle \{e_1, e_2, e_2\}, \{e_3, e_4\} \rangle$, the strategies in P are disjoint, and we have $cost_1(P) = 2 + 2 = 4, cost_2(P) = 1 + 3 = 4$. For the profile $P' = \langle \{e_1, e_2, e_2\}, \{e_1, e_2\} \rangle$, it holds that $\eta_1(e_1) = \eta_2(e_1)$ and $\eta_1(e_2) = 2 \cdot \eta_2(e_2)$. Therefore, $cost_1(P') = \frac{1}{2} + 2 = 2\frac{1}{2}$ and $cost_2(P') = \frac{1}{2} + 1 = 1\frac{1}{2}$.

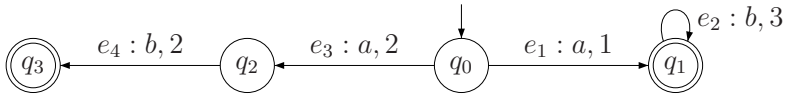


Figure 1: An example of a WFA.

We consider the following instances of AF games. Let $G = \langle \mathcal{A}, O \rangle$. We start with instances obtained by imposing restrictions on the WFA $\mathcal{A}$. In *one-letter* instances, $\mathcal{A}$ is over a singleton alphabet, i.e., $|\Sigma| = 1$. When depicting such WFAs, we omit the letters on the transitions. In *all-accepting* instances, all the states in $\mathcal{A}$ are accepting; i.e., $F = Q$. In *uniform-costs* instances, all the transitions in the WFA have the same cost, which we normalize to 1. Formally, for every $e \in \Delta$, we have $c(e) = 1$. We

continue to restrictions on the objectives in O. In *single-word* instances, each of the languages in O consists of a single word. In *symmetric* instances, the languages in O coicide, thus the players all have the same objective. We also consider combinations on the restrictions. In particular, we say that $\langle \mathcal{A}, O \rangle$ is *weak* if it is one-letter, all states are accepting, costs are uniform, and objectives are single words. Weak instances are simple indeed – each player only specifies a length of a path he should patrol, ending anywhere in the WFA, where the cost of all transitions is the same. As we shall see, many of our hardness results and lower bounds hold already for the class of weak instances.

2.2 Nash equilibrium, social optimum, and equilibrium inefficiency

For a profile P, a strategy π_i for Player i, and a strategy π, let $P[\pi_i \leftarrow \pi]$ denote the profile obtained from P by replacing the strategy for Player i by π. A profile $P \in \mathcal{S}$ is a *pure Nash equilibrium* (NE) if no player i can benefit from unilaterally deviating from his run in P to another run; i.e., for every player i and every run $\pi \in \mathcal{S}_i$ it holds that $cost_i(P[\pi_i \leftarrow \pi]) \geqslant cost_i(P)$. In our example, the profile P is not a NE, since Player 2 can reduce his payments by deviating to profile P'.

The (social) cost of a profile P, denoted $cost(P)$, is the sum of costs of the players in P. Thus, $cost(P) = \sum_{1 \leqslant i \leqslant k} cost_i(P)$. Equivalently, if we view P as a set of transitions, with $e \in P$ iff there is $\pi \in P$ for which $e \in \pi$, then $cost(P) = \sum_{e \in P} c(e)$. We denote by OPT the cost of an optimal solution; i.e., $OPT = \min_{P \in \mathcal{S}} cost(P)$. It is well known that decentralized decision-making may lead to sub-optimal solutions from the point of view of society as a whole. We quantify the inefficiency incurred due to self-interested behavior according to the *price of anarchy* (PoA) [29, 36] and *price of stability* (PoS) [2] measures. The PoA is the worst-case inefficiency of a Nash equilibrium, while the PoS measures the best-case inefficiency of a Nash equilibrium. Formally,

Definition 2.1 *Let $\mathcal{G}$ be a family of games, and let $G \in \mathcal{G}$ be a game in $\mathcal{G}$. Let $\Upsilon(G)$ be the set of Nash equilibria of the game G. Assume that $\Upsilon(G) \neq \varnothing$.*

- *The* price of anarchy *of G is the ratio between the* maximal *cost of a NE and the social optimum of G. That is, $PoA(G) = \max_{P \in \Upsilon(G)} cost(P)/OPT(G)$. The* price of anarchy *of the family of games $\mathcal{G}$ is $PoA(\mathcal{G}) = sup_{G \in \mathcal{G}} PoA(G)$.*

- *The* price of stability *of G is the ratio between the* minimal *cost of a NE and the social optimum of G. That is, $PoS(G) = \min_{P \in \Upsilon(G)} cost(P)/OPT(G)$. The* price of stability *of the family of games $\mathcal{G}$ is $PoS(\mathcal{G}) = sup_{G \in \mathcal{G}} PoS(G)$.*

Uniform sharing rule: A different cost-sharing rule that could be adopted for automaton-formation games is the uniform sharing rule, according to which the cost of a transition e is equally shared by the players that traverse e, independent of the number of times e is traversed by each player. Formally, let $\kappa_P(e)$ be the number of runs that use the transition e at least once in a profile P. Then, the cost of including a transition e at least once in a run is $c(e)/\kappa_P(e)$. This sharing rule induces a potential game, where the potential function is identical to the one used in the analysis of the classical network design game [2]. Specifically, let $\Phi(P) = \sum_{e \in E} c(e) \cdot H(\kappa_P(e))$, where $H_0 = 0$, and $H_k = 1 + 1/2 + \ldots + 1/k$. Then, $\Phi(P)$ is a potential function whose value reduces with every improving step of a player, thus a pure NE exists and BRD is guaranteed to converge. The similarity with classical network-formation games makes the study of this setting straightforward. Thus, throughout this paper we only consider the proportional sharing rule as defined in (1) above.

3 Properties of Automaton-Formation Games

In this section we study the theoretical properties of AF games: existence of NE and equilibrium inefficiency. We show that AF games need not have a pure Nash equilibrium. This holds already in the very restricted class of weak instances, and is in contrast with network-formation games. There, BRD converges and a pure NE always exists [2]. We then analyze the PoS in AF games and show that there too, the situation is significantly less stable than in network-formation games.

Theorem 3.1 *Automaton-formation games need not have a pure NE. This holds already for the class of weak instances.*

Proof: Consider the WFA $\mathcal{A}$ depicted in Figure 2 and consider a game with $k = 2$ players. The language of each player consists of a single word. Recall that in one-letter instances we care only about the lengths of the objective words. Let these be ℓ_1 and ℓ_2, with $\ell_1 \gg \ell_2 \gg 0$ that are multiples of 12. For example, $\ell_1 = 30000, \ell_2 = 300$. Let C_3 and C_4 denote the cycles of length 3 and 4 in $\mathcal{A}$, respectively. Let D_3 denote the path of length 3 from q_0 to q_1. Every run of $\mathcal{A}$ consists of some repetitions of these cycles possibly with one pass on D_3.

We claim that no pure NE exists in this instance. Since we consider long runs, the fact that the last cycle might be partial is ignored in the calculations below.

[2]Best-response-dynamics (BRD) is a local-search method where in each step some player is chosen and plays his best-response strategy, given that the strategies of the other players do not change.

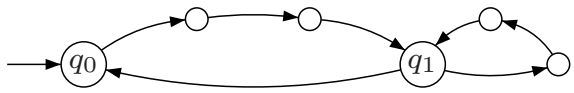


Figure 2: A weak instance of AF games with no NE.

We first show that the only candidate runs for Player 1 that might be part of a NE profile are $\pi_1 = (C_4)^{\frac{\ell_1}{4}}$ and $\pi'_1 = D_3 \cdot (C_3)^{\frac{\ell_1}{3}-1}$. If Player 1 uses both C_3 and C_4 multiple times, then, given that $\ell_1 \gg \ell_2$, he must almost fully pay for at least one of these cycles, thus, deviating to the run that repeats this fully-paid cycle is beneficial.

When Player 1 plays π_1, Player 2's best response is $\pi_2 = (C_4)^{\frac{\ell_2}{4}}$. In the profile $\langle \pi_1, \pi_2 \rangle$, Player 1 pays almost all the cost of C_4, so the players' costs are $(4-\varepsilon, \varepsilon)$. This is not a NE. Indeed, since $\ell_2 \gg 0$, then by deviating to π'_1, the share of Player 1 in D_3 reduces to almost 0, and the players' costs in $\langle \pi'_1, \pi_2 \rangle$, are $(3+\varepsilon, 4-\varepsilon)$. This profile is not a NE as Player 2's best response is $\pi'_2 = D_3 \cdot (C_3)^{\frac{\ell_2}{3}-1}$. Indeed, in the profile $\langle \pi'_1, \pi'_2 \rangle$, the players' costs are $(4.5-\varepsilon, 1.5+\varepsilon)$ as they share the cost of D_3 and Player 1 pays almost all the cost of C_3. This is not a NE either, as Player 1 would deviate to the profile $\langle \pi_1, \pi'_2 \rangle$, in which the players' costs are $(4-\varepsilon, 3+\varepsilon)$. The latter is still not a NE, as Player 2 would head back to $\langle \pi_1, \pi_2 \rangle$. We conclude that no NE exists in this game. □

The fact a pure NE may not exist is a significant difference between standard cost-sharing games and AF games. The bad news do not end here and extend to equilibrium inefficiency. We first note that the cost of any NE is at most k times the social optimum (as otherwise, some player pays more than the cost of the SO and can benefit from migrating to his strategy in the SO). Thus, it holds that $PoS \leqslant PoA \leqslant k$. The following theorem shows that this is tight already for highly restricted instances.

Theorem 3.2 *The PoS in* AF *games equals the number of players. This holds already for the class of weak instances.*

Proof: We show that for every $k, \delta > 0$ there exists a simple game with k players for which the PoS is more than $k - \delta$. Given k and δ, let r be an integer such that $r > \max\{k, \frac{k-1}{\delta} - 1\}$. Consider the WFA $\mathcal{A}$ depicted in Figure 3. Let $L = \langle \ell_1, \ell_2, \ldots, \ell_k \rangle$ for $\ell_2 = \ldots = \ell_k$ and $\ell_1 \gg \ell_2 \gg 0$ denote the lengths of the objective words. Thus, Player 1 has an 'extra-long word' and the other $k-1$ players have words of the same, long, length. Let C_r and C_{r+1} denote, respectively, the cycles of length r and $r+1$ to the right of q_0. Let D_r denote the path of length

r from q_0 to q_1, and let D_{kr} denote the 'lasso' consisting of the kr-path and the single-edge loop to the left of q_0.

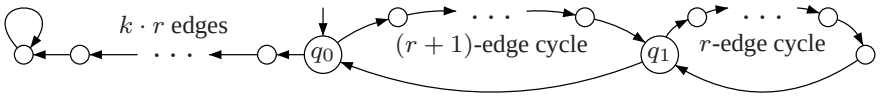


Figure 3: A weak instance of AF games for which $PoS = k$.

The social optimum of this game is to buy C_{r+1}. Its cost is $r + 1$. However, as we show, the profile P in which all players use D_{kr} is the only NE in this game. We first show that P is a NE. In this profile, Player 1 pays $r + 1 - \varepsilon$ and each other player pays $r + \varepsilon/(k-1)$. No player will deviate to a run that includes edges from the right side of $\mathcal{A}$. Next, we show that P is the only NE of this game: Every run on the right side of $\mathcal{A}$ consists of some repetitions of C_{r+1} and C_r, possibly with one traversal of D_r. Since we consider long runs, the fact that the last cycle might be partial is ignored in the calculations below.

In the social optimum profile, Player 1 pays $r + 1 - \varepsilon$ and each of the other players pays $\varepsilon/(k-1)$. The social optimum is not a NE as Player 1 would deviate to $D_r \cdot C_r^*$ and will reduce his cost to $r + \varepsilon'$. The other players, in turn, will also deviate to $D_r \cdot C_r^*$. In the profile in which they are all selecting a run of the form $D_r \cdot C_r^*$, Player 1 pays $r + r/k - \varepsilon > r + 1$ and prefers to return to C_{r+1}^*. The other players will join him sequentially, until the non-stable social optimum is reached. Thus, no NE that uses the right part of $\mathcal{A}$ exists. Finally, it is easy to see that no run that involves edges from both the left and right sides of $\mathcal{A}$ or includes both C_{r+1} and C_r can be part of a NE.

The cost of the NE profile is $kr+1$ and the PoS is therefore $\frac{kr+1}{r+1} = k - \frac{k-1}{r+1} > k - \delta$. □

4 Computational Complexity Issues in AF Games

In this section we study the computational complexity of three problems: finding the cost of the social optimum, finding the best-response of a player, and deciding the existence of a NE. Recall that the social optimum (SO) is a profile that minimizes the total cost the players pay. It is well-known that finding the social optimum in a network-formation game is NP-complete. We show that this hardness is carried over to simple instances of AF games. On the positive side, we identify non-trivial classes of instances, for which it is possible to compute the SO efficiently. The other issue we consider is the complexity of finding the best strategy of a single player, given the current profile, namely, the best-response of a player. In network-formation games,

computing the best-response reduces to a shortest-path problem, which can be solved efficiently. We show that in AF games, the problem is NP-complete. Finally, recall that AF games are not guaranteed to have a NE. We study the problem of deciding, given an AF game, whether it has a NE. We term this problem ∃NE. We show that the ∃NE problem is Σ^2_P-complete.

We start with the problem of finding the value of the social optimum.

Theorem 4.1 *Finding the value of the social optimum in* AF *games is NP-complete. Moreover, finding the social optimum is NP-complete already in single-worded instances that are also uniform-cost and are either single-lettered or all-accepting.*

Proof: We start with membership in NP. Given a WFA $\mathcal{A}$ with objectives $w_1, \ldots, w_k$ and value $c \in \mathbb{R}$, we can guess a witness profile P and check whether it satisfies $cost(P) \leqslant c$ in polynomial time. For proving hardness, we show a reduction from the *Set-Cover* (SC) problem. Consider an input $\langle U, S, m\rangle$ to SC. Recall that U is a set of elements, $S = \{C_1, \ldots, C_z\} \subseteq 2^U$ is a collection of subsets of elements of U, and $m \in \mathbb{N}$. Then, $\langle U, S, m\rangle$ is in SC iff there is a subset S' of S of size at most m that covers U. That is, $|S'| \leqslant m$ and $\bigcup_{C \in S'} C = U$.

Given an input $\langle U, S, m\rangle$ to SC, we construct a uniform-cost single-letter WFA $\mathcal{A}$ and a vector of k integers, where the i-th integer corresponds to the length of the (single) word in L_i. We fix a value y, such that $\langle U, S, m\rangle$ in SC iff the SO value of the game played on $\mathcal{A}$ with the objectives in $\{L_i\}$ is y. We construct $\mathcal{A} = \langle \{a\}, Q, q_0, \Delta, \{q_{acc}\}, c\rangle$ as follows (see an example in the left of Figure 4). The set Q includes the initial and accepting states, a state for every set in S, and intermediate states required for the disjoint runs defined below. Without loss of generality, we assume that $U = \{1, \ldots, k\}$. Consider an element $i \in U$. For every $C \in S$ such that $i \in C$, there is a disjoint run of length i from C to q_{acc}. Also, for every $C \in S$, there is a transition $\langle q_0, C\rangle$ in Δ. The cost of all transitions in Δ is 1. For every $1 \leqslant i \leqslant k$, the length of the word in $|L_i|$ is $i + 1$. We define $w = m + (1 + 2 + \ldots + k)$. The size of $\mathcal{A}$ is clearly polynomial in $|U|$ and $|S|$.

Figure 4: The WFAs produced by the reduction for $U = \{1, 2, 3\}$ and $S = \{\{1, 2\}, \{2, 3\}\}$.

The construction for uniform-cost all-accepting instances is very similar (see an example in the right of Figure 4). Let $z = \lceil \log(n) \rceil$ and $\Sigma = \{0, 1\}$. For $C \in S$ and $i \in C$, we have a z-length path from C to q_{acc} that is labeled with the binary

representation of $i-1$ (padded with preceding zeros if needed). The label on all transitions from q_0 to the S states is 0. For $1 \leqslant i \leqslant k$, the word for Player i is a single 0 letter followed by the binary representation of $i-1$. The size of $\mathcal{A}$ is clearly polynomial in $|U|$ and $|S|$.

We claim that there exists a set-cover of size m iff $OPT \leqslant m+(1+2+\ldots+k)$ for the uniform-cost single-letter instance and $OPT \leqslant m+k \cdot z$ for the uniform-cost all-accepting instance. We prove the claim for the uniform-cost single-letter instance. The proof for the uniform-cost all-accepting instance is very similar. For the first direction, let $S' = \{s_{i_1}, \ldots, s_{i_m}\}$ be a set cover. We show a profile $P = \{\pi_1, \ldots, \pi_k\}$ such that $cost(P) \leqslant m + (1 + 2 + \ldots + k)$. Recall that the input length for Player i is $i + 1$. Since S' is a set cover, there is a set $s \in S'$ with $i \in s$. We define the run π_i to proceed from q_0 to s and from there to q_{acc} on a run of length i. Clearly, the runs $\pi_1, \ldots, \pi_n$ are all legal-accepting runs. Moreover, the runs use m transitions from $\{q_0\} \times S \subseteq E$. Thus, $cost(P) \leqslant m + (1 + 2 + \ldots + k)$, implying $OPT \leqslant m + (1 + 2 + \ldots + k)$, and we are done.

For the second direction, assume $OPT = m'+(1+2+\ldots+k) \leqslant m+(1+2+\ldots+k)$, we prove that there is a set cover of size m'. Let $S^* = \langle \pi_1, \ldots, \pi_k \rangle$. Thus, $OPT = cost(S^*) = m'$. Let $S' \subset S$ be such that $s \in S'$ iff the transition $\langle q_0, s \rangle$ is used in one of the runs in S^*. Note that the run of every player consists of a transition (q_0, s) followed by a disjoint run of length i to q_{acc}. Therefore, $OPT = m'+(1+2+\ldots+k)$, and, $|S'| = m' \leqslant m$. We claim that S' is a set cover. For every $i \in U$, the first transition in the run is a transition $\langle q_0, s \rangle$ for some $s \in S$, as otherwise, player i can not proceed to q_{acc} along a run of length i. By our definition of S' we have $s \in S'$, thus $i \in U$ is covered. □

The hardness results in Theorem 4.1 for single-word specification use one of two properties: either there is more than one letter, or not all states are accepting. We show that finding the SO in instances that have both properties can be done efficiently, even for specifications with arbitrary number of words.

For a language L_i over $\Sigma = \{a\}$, let $short(i) = \min_j\{a^j \in L_i\}$ denote the length of the shortest word in L_i. For a set O of languages over $\Sigma = \{a\}$, let $\ell_{max}(O) = \max_i short(i)$ denote the length of the longest shortest word in O. Clearly, any solution, in particular the social optimum, must include a run of length $\ell_{max}(O)$. Thus the cost of the social optimum is at least the cost of the cheapest run of length $\ell_{max}(O)$. Moreover, since the WFA is single-letter and all-accepting, the other players can choose runs that are prefixes of this cheapest run, and no additional transitions should be acquired. We show that finding the cheapest such run can be done efficiently.

Theorem 4.2 *The cost of the social optimum in a single-letter all-accepting in-*

stance $\langle \mathcal{A}, O \rangle$ *is the cost of the cheapest run of length* $\ell_{max}(O)$*. Moreover, this cost can be found in polynomial time.*

Proof: Clearly, any solution, in particular the social optimum, must include a run of length $\ell_{max}(O)$. Thus the cost of the social optimum is at least the cost of the cheapest run of length $\ell_{max}(O)$. Moreover, since there are no target vertices, the other players can be assigned runs that are prefixes of the cheapest run, and no additional transitions should be acquired.

We claim that finding the cheapest such run can be done efficiently. Recall that q_0 is the initial state in $\mathcal{A}$, and let $|Q| = n$. We view $\mathcal{A}$ as a weighted-directed graph $G = \langle V, E, c \rangle$, where the vertices V are the states Q, there is an edge $e \in E$ between two vertices if there is a transition between the two corresponding states, and the cost of the edges is the same as the cost of the transition in $\mathcal{A}$. For $0 \leqslant i \leqslant n$, let $d_i : V \times V \to \mathbb{Q}^+$ be the function that, given two vertices $u, v \in V$, returns the value of the cheapest pah of length i from u to v, and ∞ if no such path exists. Note that there is no requirement that the path is simple, and indeed we may traverse cycles in order to accommodate i transitions. The function $d : V \times V \to \mathbb{Q}^+$, returns the value of the cheapest path of any length between two given vertices. Given two vertices $u, v \in V$, computing $d(u, v)$ can be done using Dijkstra's algorithm, and, given an index $i \in \mathbb{N}$, it is possible to compute $d_i(u, v)$ by a slight variation of the Bellman-Ford algorithm.

We distinguish between two cases. If $\ell_{max} > 2n - 2$, we claim that the value of the social optimum is $min\{d(q_0, v) + d(v, v) : v \in V\}$. If $\ell_{max} \leqslant 2n-2$, then we claim that the value of the social optimum is the minimum value of $d_i(q_0, v) + d_j(v, v)$, where $v \in V$, $0 \leqslant i \leqslant \ell_{max}$, $0 \leqslant j \leqslant \ell_{max} - i$, and if $j = 0$, then $i = \ell_{max}$.

We start with the first case. Assume $\ell_{max} > 2n - 2$. Let $ALG = min\{d(q_0, v) + d(v, v) : v \in V\}$. Recall that S^* is the social optimum profile, and $OPT = cost(S^*)$. For the first direction, we claim that $ALG \leqslant OPT$. Let π be a run in S^* of length ℓ_{max}, where we assume π is a sequence of transitions. Clearly, $OPT \geqslant cost(\pi)$. Since ALG takes the minimum over all vertices, it suffices to prove that $cost(\pi) \geqslant d(q_0, v) + d(v, v)$ for some $v \in V$. We view π as a path in the graph G, and we claim that π contains a sub-path that starts in q_0 and ends in v and a sub-path that is a cycle from v to itself, for some $v \in V$. Thus, $OPT \geqslant cost(\pi) \geqslant cost(x) + cost(y) \geqslant d(q_0, v) + d(v, v) \geqslant ALG$. We continue to prove the claim. Since $\ell_{max} > n$, there is a vertex v that appears twice in π. We split π into two paths, at the first appearance of v. That is, $\pi = x \cdot y'$, where x is a path that ends in v and v does not appear in x again. Note that if $v = q_0$, then $\pi = y'$. Since π is a legal run, it starts in q_0, and we have that x is a path from q_0 to v. We continue to prove that there is a cycle y from v to itself that is contained in y'. Indeed, since v appears at least twice in π,

and since y' is a sequence of transitions that starts in v, we have that v appears in y' at least twice, and we are done.

We continue to prove that $ALG \geqslant OPT$. Let $v \in V$ be the vertex that attains the minimum in $min\{d(q_0, v) + d(v, v) : v \in V\}$. Let $\tau = \tau_1 \cdot \tau_2$ be a run such that τ_1 is a simple path from q_0 to v with $cost(\tau_1) = d(q_0, v)$ and τ_2 is a simple cycle from v to itself with $cost(\tau_2) = d(v, v)$. We claim that $cost(\tau) \geqslant OPT$. Since τ_1 and τ_2 are simple, we have $|\tau_1| \leqslant n - 1$ and $|\tau_2| \leqslant n - 1$. Thus, $|\tau| < 2n - 2$. We extend τ to a path of length ℓ_{max} by traversing the loop τ_2 many times. Clearly, τ is a legal run of the automaton $\mathcal{A}$ on a word of length ℓ_{max}. Consider the profile S in which the players choose runs that are prefixes of τ'. Since the only transitions used in S are those in τ, we have $cost(S) = cost(\tau)$. Since S^* is the social optimum, we have $ALG = cost(S) \geqslant cost(S^*) = OPT$, and we are done.

The case in which $\ell_{max} \leqslant 2n - 2$ is proven in a similar manner. □

We turn to prove the hardness of finding the best-response of a player. Our proof is valid already for a single player that needs to select a strategy on a WFA that is not used by other players (one-player game).

Theorem 4.3 *Finding the best-response of a player in* AF *games is NP-complete.*

Proof: We start with membership in NP. Given a WFA $\mathcal{A}$ with objectives $L_1, \ldots, L_k$ and value $c \in \mathbb{R}$, we can guess a witness profile P and check whether it satisfies $cost(P) \leqslant c$ in polynomial time.

For proving hardness, we show a reduction from the *Set-Cover* (SC) problem. Consider an input $\langle U, S, m\rangle$ to SC. Recall that $U = \{1, \ldots, n\}$ is a set of elements, $S = \{C_1, \ldots, C_z\} \subseteq 2^U$ is a collection of subsets of elements of U, and $m \in \mathbb{N}$. Then, $\langle U, S, m\rangle$ is in SC iff there is a subset S' of S of size at most m that covers U. That is, $|S'| \leqslant m$ and $\bigcup_{C \in S'} C = U$.

Given an input $\langle U, S, m\rangle$ to SC, we construct a game $\langle \mathcal{A}, O\rangle$ such that $\langle U, S, m\rangle$ is in SC iff the SO in the game is at most l. The game is a one-player game. We start by describing the specification L of the player. The alphabet of L is $S \cup U$ and it is given by the regular expression $(C_1 + \ldots + C_m) \cdot 1 \cdot (C_1 + \ldots + C_m) \cdot 2 \cdot \ldots \cdot (C_1 + \ldots + C_m) \cdot n$. The WFA $\mathcal{A}$ is over the alphabet $S \cup U$. There is a single initial state q_{in} and a state for every set in S. For $1 \leqslant i \leqslant z$, there is a C_i-labeled transition from q_{in} to the state C_i, and for every $j \in C_i$, there is a j-labeled transition from the state C_i back to q_{in}. The first type of transitions cost 1 and the second cost 0 (for an example see Figure 5).

We prove the correctness of the reduction: For the first direction, assume there is a set cover of at most l. Consider the word w in which, for every $1 \leqslant j \leqslant n$, the letter that precedes j is $C_i \in S$ such that C_i is in the set cover. Clearly, $w \in L$ and

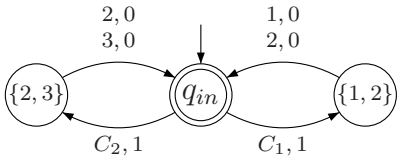


Figure 5: The WFA produced by the reduction for $U = \{1, 2, 3\}$ and $S = \{\{1, 2\}, \{2, 3\}\}$.

since it uses at most l letters from S, the profile in which the player chooses it, costs at most l. Thus, the SO is also at most l. For the other direction, assume the SO is attained in a profile with the word $w \in L$. It is not hard to see that the letters from S that appear in w form a set cover of size at most l. □

We turn to study the problem of deciding whether a NE exists. We show that ∃NE is complete for Σ_2^P – the second level of the polynomial hierarchy. Namely, decision problems solvable in polynomial time by a nondeterministic Turing machine augmented by an *oracle* for an NP-complete problem. An oracle for a computational problem is a black box that is able to produce a solution for any instance of the problem in a single operation. Thus, for every problem $P \in \Sigma_2^P$ there is a machine such that for every $x \in P$ there is a polynomial accepting computation (with polynomial many queries to the oracle). As co-NP is the dual complexity class of NP, the dual complexity class of Σ_2^P is Π_2^P. Thus, a problem P is Σ_2^P-complete iff its complement $\overline{P}$ is Π_2^P-complete.

The upper bound is easy: guess a profile, and use k calls to an oracle for the best-response problem to verify that no player can benefit from deviating. For the lower bound, we alter the reduction in [6], for the ∃NE problem in a similar game. The reduction is from the complement of the *min-max vertex cover* problem, which is known to Σ_2^P-complete [28].

Theorem 4.4 *The problem of deciding whether an* AF *has a NE is* Σ_P^2*-complete.*

5 Tractable Instances of AF Games

In the example in Theorem 3.1, Player 1 deviates from a run on the shortest (and cheapest) possible path to a run that uses a longer path. By doing so, most of the cost of the original path, which is a prefix of the new path and accounts to most of its cost, goes to Player 2. We consider *semi-weak* games in which the WFA is uniform-cost, all-accepting, and single-letter, but the objectives need not be a single word. We identify a property of such games that prevents this type of deviation and which guarantees that the social optimum a NE. Thus, we identify a family of AF games in which a NE exists, finding the SO is easy, and the PoS is 1.

Definition 5.1 *Consider a semi-weak game $\langle \mathcal{A}, O \rangle$. A* lasso *is a path $u \cdot v$, where u is a simple path that starts from the initial state and v is a simple cycle. A lasso ν is minimal in $\mathcal{A}$ if $\mathcal{A}$ does not have shorter lassos. Note that for minimal lassos $u \cdot v$, we have that $u \cap v = \varnothing$. We say that $\mathcal{A}$ is* resistant *if it has no cycles or there is a minimal lasso $\nu = u \cdot v$ such that for every other lasso ν' we have $|u \backslash \nu'| + |v| \leqslant |\nu' \backslash \nu|$.*

Consider a resistant weak game $\langle \mathcal{A}, O \rangle$. In order to prove that the social optimum is a NE, we proceed as follows. Let ν be the lasso that is the witness for the resistance of $\mathcal{A}$. We show that the profile S^* in which all players choose runs that use only the lasso ν or a prefix of it, is a NE. The proof is technical and we go over all the possible types of deviations for a player and use the weak properties of the network along with its resistance. By Theorem 4.2, the cost of the profile is the SO. Hence the following.

Theorem 5.1 *For resistant semi-weak games, the social optimum is a NE.*

Proof: Consider a resistant semi-weak game $\langle \mathcal{A}, O \rangle$, thus $\mathcal{A}$ has no cycles or there is a minimal lasso in $\mathcal{A}$ that satisfies the resistance requirements. Recall that by Theorem 4.2, the social optimum is the profile S^* in which all players use prefixes of the cheapest run of length $\ell_{max}(O)$. Formally, let $\ell_1 \geqslant \ell_2 \geqslant \ldots \geqslant \ell_k$, where for $1 \leqslant i \leqslant k$, ℓ_i be the minimal length of a word in L_i. That is, $\ell_1 = \ell_{max}(O)$. Then, $S^* = \langle \pi_1, \ldots, \pi_k \rangle$, where for $1 \leqslant i \leqslant k$, the run π_i is of length ℓ_i and π_1 uses the lasso that is the witness for resistance, or an acyclic path if the lasso's length is larger than ℓ_1.

We claim that S^* is a NE. Assume otherwise, thus there are $1 \leqslant i \leqslant k$ and π'_i such that $cost_i(S^*) > cost_i(S^*[i \leftarrow \pi'_i])$. Assume Wlog that $|\pi'_i| = \ell_i$ as otherwise Player i can deviate to a prefix of length ℓ_i of π'_i and only improve his payment. We use S' to refer to $S^*[i \leftarrow \pi'_i]$. For $1 \leqslant j \leqslant k$, let ν_j be the set of transitions that are used in π_j. Similarly, let ν'_i be the transitions used in π'_i. Note that $\nu_1, \ldots, \nu_k, \nu'_i$ are paths of transitions.

We distinguish between four cases. In the first case, both ν_i and ν'_i are simple paths. First, note that every transition in $\nu_i \cap \nu'_i$ costs the same for Player i in both profiles. Next, we claim that every transition in $\nu'_i \backslash \nu_i$ costs at least as much as any transition in $\nu'_i \backslash \nu_i$. Indeed, since all players use prefixes of ν_1, the sharing along the path monotonically decreases. That is, assuming $\nu_i = t_1, \ldots, t_n$, then for $1 \leqslant j \leqslant n-1$, in S^*, the number of players using transition t_j is at least that of t_{j+1}. Since $\mathcal{A}$ has uniform transition costs, the claim follows. Finally, since the runs are simple, the sizes of $\nu_i \backslash \nu'_i$ and $\nu'_i \backslash \nu_i$ are equal. Thus, $cost_i(S^*) \leqslant cost_i(S')$, and we reach a contradiction to the fact that Player i deviates.

In the second case, ν_i is simple and ν'_i is lasso. Thus, $|\nu'_i| \leqslant |\nu_i|$. If $|\nu'_i| = |\nu_i|$, we return to the previous case. We assume $|\nu'_i| < |\nu_i|$, and show that we reach a

contradiction to our assumption that $\mathcal{A}$ is resistant. Recall that $|\nu_1| \geqslant |\nu_i|$. If π_1 uses a lasso, then ν_i' is a shorter lasso, contradicting the minimality of the witness lasso for resistance. If π_1 does not use a lasso, then we reach a contradiction to our assumption that the witness lasso has length greater than ℓ_1.

In the third case, ν_i is a lasso and ν_i' is simple. Thus, $\nu_i = \nu_1$. Consider a transition $e \in \nu_i$. Let x_e and x'_e be the number of times Player i uses e in π_i and π'_i, respectively. Thus, $x_e > 0$ and $x'_e \leqslant 1$. Let y_e be the number of times the other players use e in S^* and also in S' as none of them alter their strategy. Consider a transition $e \in \nu_i$ having $x'_e = 1$. That is, Player i reduces his number of uses of transition e from x_e to 1. Since the number of times Player i uses a transition in π'_i is at most 1, there are $(x_e - 1)$ transitions that are not used by Player i in π_i and are used once in π'_i. Since $\nu_i = \nu_1$, these transitions are all in $\nu'_i \backslash \nu_i$ and Player i pays 1 for each of them. Consider a transition $e \in \nu_i$. Let $cost_i^e(S^*)$ and $cost_i^e(S')$ be the cost Player i pays for transition e in profiles S^* and S', respectively. If $x'_e = 1$, then by the above

$$cost_i^e(S^*) - cost_i^e(S') = \frac{x_e}{y_e + x_e} - \big(\frac{1}{y_e + 1} + (x_e - 1)\big) =$$

$$= \frac{x_e y_e + x_e + y_e^2 - y_e x_e^2 - x_e^2 - y_e^2 x_e}{(y_e + x_e) \cdot (y_e + 1)} \leqslant 0$$

Similarly, if $x'_e = 0$, then the change in cost incurred by e is:

$$cost_i^e(S^*) - cost_i^e(S') = \frac{x_e}{y_e + x_e} - x_e \leqslant 0$$

Since $cost_i(S^*) - cost_i(S') = \sum_{e \in \Delta} cost_i^e(S^*) - cost_i^e(S')$, we have $cost_i(S^*) - cost_i(S') \leqslant 0$, and thus $cost_i(S^*) \leqslant cost_i(S')$, which is a contradiction to the fact that Player i deviates.

We continue to the final case in which both ν_i and ν_i' are lassos. As in the previous case, $\nu_i = \nu_1$. Recall that the lasso ν_1 is the lasso that is the witness for the resistance of $\mathcal{A}$. We show that the lasso ν_i' violates our requirement for ν_1 and thus we reach a contradiction. Let $\nu_1 = u \cdot v$, where u is a simple path from the initial state and v is a simple cycle. Thus,

$$cost_i(S^*) = cost_i(S^*, u) + cost_i(S^*, v) \leqslant cost_i(S^*, u \cap \nu'_i) + |u \backslash \nu'_i| + |v|.$$

Also,

$$cost_i(S') = cost_i(S', u \cap \nu'_i) + cost_i(S', \nu'_i \cap v) + |\nu'_i \backslash \nu_i| \geqslant cost_i(S^*, u \cap \nu'_i) + |\nu'_i \backslash \nu_i|.$$

Subtracting both inequalities we get:

$$cost_i(S^*) - cost_i(S') \leqslant |u \backslash \nu'_i| + |v| - |\nu'_i \backslash \nu_i|.$$

Since $cost_i(S^*) - cost_i(S') > 0$, we get:

$$|\nu_i' \backslash \nu_i| > |u \backslash \nu_i'| + |v|,$$

which is a contradiction to the resistance of $\mathcal{A}$, and we are done. □

A corollary of Theorem 5.1 is the following:

Corollary 5.2 *For resistant semi-weak games, we have PoS= 1.*

We note that resistance can be defined also in WFAs with non-uniform costs, with $cost(\nu)$ replacing $|\nu|$. Resistance, however, is not sufficient in the *slightly* stronger model where the WFA is single-letter and all-accepting but not uniform-cost. Indeed, given k, we show a such a game in which the PoS is kx, for a parameter x that can be arbitrarily close to 1. Consider the WFA A in Figure 5. Note that $\mathcal{A}$ has a single lasso and is thus a resistant WFA. The parameter ℓ_1 is a function of x, and the players' objectives are single words of lengths $\ell_1 \gg \ell_2 \gg \ldots \gg \ell_k \gg 0$. Similar to the proof of Theorem 3.2, there is only one NE in the game, which is when all players choose the left chain. The social optimum is attained when all players use the self-loop, and thus for a game in this family, $PoS = \frac{k \cdot x}{1}$. Since x tends to 1, we have $PoS = k$ for resistant all-accepting single-letter games.

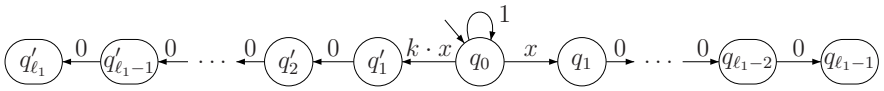


Figure 6: A resistant all-accepting single-letter game in which the PoS tend to k.

6 Surprises in Symmetric Instances

In this section we consider the class of symmetric instances, where all players share the same objective. That is, there exists a language L, such that for all $1 \leqslant i \leqslant k$, we have $L_i = L$. In such instances it is tempting to believe that the social optimum is also a NE, as all players evenly share the cost of the solution that optimizes their common objective. While this is indeed the case in all known symmetric games, we show that, surprisingly, this is not valid for AF-games, in fact already for the class of one-letter, all accepting, unit-cost and single-word instances.

Moreover, we start by showing that a NE need not exist in general symmetric instances.

Theorem 6.1 *Symmetric instances of* AF*-games need not have a pure NE.*

Proof: Consider a WFA $\mathcal{A}$ consisting of a single accepting state with two self loops, labeled $(a, 1)$ and $(b, \frac{5}{14} - \epsilon)$. Let n_1 and n_2 be such that $0 \ll n_2 \ll n_1$. We define $L = a^6 + ab^{n_1} + aab^{n_2} + aaab$. We denote the 4 strategies available to each of the players by A, B, C, and D, with $A = (6, 0)$ indicating 6 uses of the a transition and 0 uses of the b transition, $B = (1, n_1)$, $C = (2, n_2)$, and $D = (3, 1)$.

In order to show that there is no NE, we only have to show that the four profiles in which the players follow the same strategy are not a NE. Indeed, it is easy to see that for every other profile, one of the players would deviate to one of these profiles. Now, in profile $\langle A, A\rangle$ both players pay $\frac{1}{2}$ as they split the cost of the a-transition evenly. This is not a NE as Player 1 (or, symmetrically, Player 2) would deviate to $\langle B, A\rangle$, where he pays $\frac{1}{7}$ for the a-transition and the full price of the b-transition, which is $\frac{5}{14} - \epsilon$, thus he pays $\frac{1}{2} - \epsilon$.

In profile $\langle B, B\rangle$, both players pay $\frac{1}{2}$ for the a-transition plus $\frac{5}{2 \cdot 14} - \epsilon$ for the b-transition, which sums to $0.678 - \epsilon$. This is not a NE, as Player 1 would deviate to $\langle C, B\rangle$, where he pays $\frac{2}{3}$ for the a-transition and, as $n_2 \ll n_1$, only ϵ for the b-transition.

In profile $\langle C, C\rangle$, again both players pay $0.678 - \epsilon$. By deviating to $\langle D, C\rangle$, Player 1 reduces his payment to $\frac{3}{5} + \epsilon$. Finally, in profile $\langle D, D\rangle$, both players pay $0.678 - \epsilon$ and when deviating to $\langle A, D\rangle$, Player 1 reduces his payment to $\frac{6}{9}$. □

We turn to study the equilibrium inefficiency, starting with the PoA. It is easy to see that in symmetric AF games, we have $PoA = k$. This bound is achieved, as in the classic network-formation game, by a network with two parallel edges labeled by a and having costs k and 1. The players all have the same specification $L = \{a\}$. The profile in which all players select the expensive path is a NE. We show that $PoA = k$ is achieved even for weak symmetric instances.

Theorem 6.2 *The PoA equals the number of players, already for weak symmetric instances.*

Proof: We show a lower bound of k. The example is a generalization of the PoA in cost sharing games [2]. For k players, consider the weak instance depicted in Figure 6, where all players have the length k. Intuitively, the social optimum is attained when all players use the loop $\langle q_0, q_0\rangle$ and thus $OPT = 1$. The worst NE is when all players use the run $q_0 q_1 \dots q_k$, and its cost is clearly k. Formally, there are two NEs in the game:

- The cheap NE is when all players use the loop $\langle q_0, q_0\rangle$. This is indeed a NE because if a player deviates, he must buy at least the transition $\langle q_0, q_1\rangle$. Thus, he pays at least 1, which is higher than $\frac{1}{k}$, which is what he pays when all players use the loop.

- The expensive NE is when all players use the run $q_0, q_1, \ldots, q_k$. This is a NE because a player has two options to deviate. Either to the run that uses only the loop, which costs 1, or to a run that uses the loop and some prefix of $q_0, q_1, \ldots, q_k$, which costs at least $1 + \frac{1}{k}$. Since he currently pays 1, he has no intention of deviating to either runs.

Since the cheap NE costs 1 and the expensive one costs k, we get $PoA = k$. □

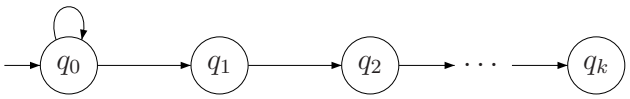


Figure 7: The WFA $\mathcal{A}$ for which a symmetric game with $|L| = 1$ achieves $PoA = k$.

We now turn to the PoS analysis. We first demonstrate the anomaly of having $PoS > 1$ with the two-player game appearing in Figure 8. All the states in the WFA $\mathcal{A}$ are accepting, and the objectives of both players is a single long word. The social optimum is when both players traverse the loop q_0, q_1, q_0. Its cost is $2 + \epsilon$, so each player pays $1 + \frac{\epsilon}{2}$. This, however, is not a NE, as Player 1 (or, symmetrically, Player 2) prefers to deviate to the run $q_0, q_1, q_1, q_1, \ldots$, where he pays the cost of the loop q_1, q_1 and his share in the transition from q_0 to q_1. We can choose the length of the objective word and ϵ so that this share is smaller than $\frac{\epsilon}{2}$, justifying his deviation. Note that the new situation is not a NE either, as Player 2, who now pays 2, is going to join Player 1, resulting in an unfortunate NE in which both players pay 1.5.

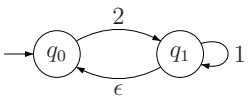


Figure 8: The WFA $\mathcal{A}$ for which the SO in a symmetric game is not a NE.

It is not hard to extend the example from Figure 8 to $k > 2$ players by changing the 2-valued transition to k, and adjusting ϵ and the lengths of the players accordingly. The social optimum and the only NE are as in the two-player example. Thus, the PoS in the resulting game is $1 + \frac{1}{k}$.

A higher lower bound of $1 + \frac{1}{k-1}$ is shown in the following theorem. Although both bounds tend to 1 as k grows to infinity, this bound is clearly stronger. Also, for $k = 2$, the bound $PoS = 1 + \frac{1}{k-1} = 2$ is tight. We conjecture that $\frac{k}{k-1}$ is tight for every $k > 1$.

Theorem 6.3 *In a symmetric k-player game, the PoS is at least $\frac{k}{k-1}$.*

Proof: For $k \geqslant 2$, we describe a family of symmetric games for which the PoS tends to $\frac{k}{k-1}$. For $n \geqslant 1$, the game $G_{\epsilon,n}$ uses the WFA that is depicted in Figure 9. Note that this is a one-letter instance in which all states are accepting. The players have an identical specification, consisting of a single word w of length $\ell \gg 0$. We choose ℓ and $\epsilon = \epsilon_0 > \ldots > \epsilon_{n-1}$ as follows. Let $C_0, \ldots, C_n$ denote, respectively, the cycles with costs $(k^n + \epsilon_0), (k^{n-1} + \epsilon_1), \ldots, (k + \epsilon_{n-1}), 1$. Let $r_0, \ldots, r_n$ be lasso-runs on w that end in $C_0, \ldots, C_n$, respectively. Consider $0 \leqslant i \leqslant n-1$ and let P_i be the profile in which all players choose the run r_i. We choose ℓ and ϵ_i so that Player 1 benefits from deviating from P_i to the run r_{i+1}, thus P_i is not a NE. Note that by deviating from r_i to r_{i+1}, Player 1 pays the same amount for the path leading to C_i. However, his share of the loop C_i decreases drastically as he uses the k^{n-i}-valued transition only once whereas the other players use it close to ℓ times. On the other hand, he now buys the loop C_{i+1} by himself. Thus, the change in his payment change is $\frac{1}{k} \cdot (k^{n-i} + \epsilon_i) - (\epsilon' + k^{n-(i+1)} + \epsilon_{i+1})$. We choose ϵ_{i+1} and ℓ so that $\frac{\epsilon_i}{k} > \epsilon' + \epsilon_{i+1}$, thus the deviation is beneficial.

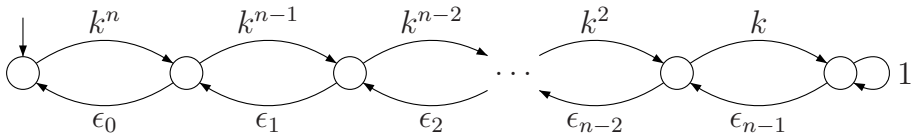


Figure 9: The network of the identical-specification game $G_{\epsilon,n}$, in which PoS tends to $\frac{k}{k-1}$.

We claim that the only NE is when all players use the run r_n. Indeed, it is not hard to see that every profile in which a player selects a run that is not from $r_0, \ldots, r_n$ cannot be a NE. Also, a profile in which two players select runs r_i and r_j, for $1 \leqslant i < j \leqslant n$, cannot be a NE as the player using r_i can decreases his payment by joining the other player in r_j. Finally, by our selection of $\epsilon_1, \ldots, \epsilon_n$, and ℓ, every profile in which all the players choose the run r_i, for $0 \leqslant i \leqslant n-1$, is not a NE.

Clearly, the social optimum is attained when all players choose the run r_0, and its cost is $k^n + \epsilon$. Since the cost of the only NE in the game is $\sum_{0 \leqslant i \leqslant n} k^{n-i}$, the PoS in this family of games tends to $\frac{k}{k-1}$ as n grows to infinity and ϵ to 0. □

Finally, we note that our hardness result in Theorem 4.3 implies that finding the social optimum in a symmetric AF-game is NP-complete. Indeed, since the social optimum is the cheapest run on some word in L, finding the best-response in a one-player game is equivalent to finding the social optimum in a symmetric game. This is contrast with other cost-sharing and congestion game (e.g. [22], where the social optimum in symmetric games can be computed using a reduction to max-flow).

Acknowledgments. We thank Michal Feldman, Noam Nisan, and Michael Schapira

for helpful discussions.

References

[1] S. Albers, S. Elits, E. Even-Dar, Y. Mansour, and L. Roditty. On Nash Equilibria for a Network Creation Game. In *Proc. 17th SODA*, pages 89-98, 2006.

[2] E. Anshelevich, A. Dasgupta, J. Kleinberg, É. Tardos, T. Wexler, and T. Roughgarden. The Price of Stability for Network Design with Fair Cost Allocation. *SIAM J. Comput.* 38(4): 1602–1623, 2008.

[3] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, 2002.

[4] B. Aminof, O. Kupferman, and R. Lampert, Reasoning about online algorithms with weighted automata, *ACM Transactions on Algorithms*, 6(2), 2010.

[5] B. Alpern and F.B. Schneider. Recognizing safety and liveness. *Distributed computing*, 2:117–126, 1987.

[6] G. Avni and O. Kupferman, Synthesis from Component Libraries with Costs, In *Proc. 25th CONCUR*, pages 156–172, 2014.

[7] G. Avni, O. Kupferman, T. Tamir, Network-Formation Games with Regular Objectives In *Proc. 17th FoSSaCS*, pages 119–133, 2014.

[8] G. Avni, O. Kupferman, T. Tamir, From Reachability to Temporal Specifications in Cost-Sharing Games, In *Proc. 7th IJCAR*, pages 1–15, 2014.

[9] G. Avni, O. Kupferman, T. Tamir, Congestion Games with Multisets of Resources and Applications in Synthesis, *Submitted*.

[10] T. Brihaye, V. Bruyère, J. De Pril, and H. Gimbert. On subgame perfection in quantitative reachability games. *Logical Methods in Computer Science*, 9(1), 2012.

[11] K. Chatterjee. Nash equilibrium for upward-closed objectives. In *Proc. 15th CSL*, LNCS 4207, pages 271–286. Springer, 2006.

[12] K. Chatterjee, T. A. Henzinger, and M. Jurdzinski. Games with secure equilibria. *Theoretical Computer Science*, 365(1-2):67–82, 2006.

[13] K. Chatterjee, T. A. Henzinger, and N. Piterman. Strategy logic. In *Proc. 18th CONCUR*, pages 59–73, 2007.

[14] K. Chatterjee, R. Majumdar, and M. Jurdzinski. On Nash equilibria in stochastic games. In *Proc. 13th CSL*, LNCS 3210, pages 26–40. Springer, 2004.

[15] H. Chen and T. Roughgarden. Network Design with Weighted Players, *Theory of Computing Systems*, 45(2), 302–324, 2009.

[16] J. R. Correa, A. S. Schulz, and N. E. Stier-Moses. Selfish Routing in Capacitated Networks. *Mathematics of Operations Research* 29: 961–976, 2004.

[17] N. Daniele, F. Guinchiglia, and M.Y. Vardi. Improved automata generation for linear temporal logic. In *Proc. 11th CAV*, LNCS 1633, pages 249–260. Springer, 1999.

[18] M. Droste, W. Kuich, and H. Vogler (eds.), Handbook of Weighted Automata, Springer, 2009.

[19] C. Dwork and M. Naor. Pricing via Processing or Combatting Junk Mail, In *Proc. 12th CRYPTO*, pages 139–177, 1992.

[20] D. Fisman, O. Kupferman, and Y. Lustig. Rational synthesis. In *Proc. 16th TACAS*, LNCS 6015, pages 190–204. Springer, 2010.

[21] A. Fabrikant, A. Luthra, E. Maneva, C. Papadimitriou, and S. Shenker. On a network creation game. In *Proc. 22nd PODC*, pages 347-351, 2003.

[22] A. Fabrikant, C. Papadimitriou, and K. Talwarl, The Complexity of Pure Nash Equilibria, *Proc. 36th STOC*, pages 604–612, 2004.

[23] M. Feldman and T. Tamir. Conflicting Congestion Effects in Resource Allocation Games. Journal of *Operations Research* 60(3), pages 529–540, 2012.

[24] P. von Falkenhausen and T. Harks. Optimal Cost Sharing Protocols for Scheduling Games. In *Proc. 12th EC*, pages 285-294, 2011.

[25] G. de Giacomo and M. Y. Vardi. Automata-Theoretic Approach to Planning for Temporally Extended Goals, In *European Conferences on Planning*, pages 226–238, 1999.

[26] D. Harel and A. Pnueli. On the development of reactive systems. In *Logics and Models of Concurrent Systems*, volume F-13 of *NATO Advanced Summer Institutes*, pages 477–498. Springer, 1985.

[27] S. Herzog, S. Shenker, and D. Estrin. Sharing the "Cost" of Multicast Trees: An Axiomatic Analysis. *IEEE/ACM Transactions on Networking*, 1997.

[28] K-I. Ko and C-L. Lin. On the complexity of min-max optimization problems and their approximation. In *Minimax and Applications*, volume 4 of *Nonconvex Optimization and Its Applications*, pages 219–239. Springer, 1995.

[29] E. Koutsoupias and C. Papadimitriou. Worst-case Equilibria. *Computer Science Review*,3(2): 65–69, 2009.

[30] O. Kupferman and T. Tamir. Coping with selfish on-going behaviors. *Information and Computation*, 210:1–12, 2012.

[31] M. Mavronicolas, I. Milchtaich, B. Monien, and K. Tiemann. Congestion Games with Player-specific Constants. In *Proc 32nd MFCS*, pp. 633–644, 2007.

[32] I. Milchtaich. Weighted Congestion Games With Separable Preferences. *Games and Economic Behavior*, 67, 750-757, 2009.

[33] M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.

[34] D. Monderer and L. Shapley. Potential Games. *Games and Economic Behavior*, 14:124–143, 1996.

[35] H. Moulin and S. Shenker. Strategyproof Sharing of Submodular Costs: Budget Balance Versus Efficiency. *Journal of Economic Theory*, 18: 511–533, 2001.

[36] C. Papadimitriou. Algorithms, Games, and the Internet. In *Proc 33rd STOC*, pages 749–753, 2001.

[37] R. Paes Leme, V. Syrgkanis, E. Tardos. The curse of simultaneity. *Innovations in Theoretical Computer Science (ITCS), pages 60-67, 2012.*

[38] M. G. Reed, P. F. Syverson, and D. M. Goldschlag. Anonymous Connections and Onion Routing IEEE J. on Selected Areas in Communication*, Issue on Copyright and Privacy Protection, 1998.*

[39] R. W. Rosenthal. A Class of Games Possessing Pure-Strategy Nash Equilibria. International Journal of Game Theory*, 2: 65–67, 1973.*

[40] E. Tardos and T. Wexler. Network Formation Games and the Potential Function Method, In Algorithmic Game Theory*, Cambridge University Press, 2007.*

[41] B. Vöcking. In N. Nisan, T. Roughgarden, E. Tardos and V. Vazirani, eds., Algorithmic Game Theory. Chapter 20: Selfish Load Balancing*. Cambridge University Press, 2007.*

Synthesis from Component Libraries with Costs*

Guy Avni† *Orna Kupferman*‡

Abstract

Synthesis is the automated construction of a system from its specification. In real life, hardware and software systems are rarely constructed from scratch. Rather, a system is typically constructed from a library of components. Lustig and Vardi formalized this intuition and studied LTL synthesis from component libraries. In real life, designers seek optimal systems. In this paper we add optimality considerations to the setting. We distinguish between quality considerations (for example, size – the smaller a system is, the better it is), and pricing (for example, the payment to the company who manufactured the component). We study the problem of designing systems with minimal quality-cost and price. A key point is that while the quality cost is individual – the choices of a designer are independent of choices made by other designers that use the same library, pricing gives rise to a resource-allocation game – designers that use the same component share its price, with the share being proportional to the number of uses (a component can be used several times in a design). We study both closed and open settings, and in both we solve the problem of finding an optimal design. In a setting with multiple designers, we also study the game-theoretic problems of the induced resource-allocation game.

1 Introduction

Synthesis is the automated construction of a system from its specification. The classical approach to synthesis is to extract a system from a proof that the specification is satisfiable. In the late 1980s, researchers realized that the classical approach to synthesis is well suited to *closed* systems, but not to *open* (also called *reactive*) systems [1, 28]. A reactive system interacts with its environment, and a correct system

*Published in the proceedings of the 25th Concurrency theory, LNCS 8704, pages 156–172, Springer, 2013. A full version was submitted.

†School of Computer Science and Engineering, The Hebrew University, Israel

‡School of Computer Science and Engineering, The Hebrew University, Israel

should have a *strategy* to satisfy the specification with respect to all environments. It turns out that the existence of such a strategy is stronger than satisfiability, and is termed *reliability*.

In spite of the rich theory developed for synthesis, in both the closed and open settings, little of this theory has been reduced to practice. This is in contrast with verification algorithms, which are extensively applied in practice. We distinguish between algorithmic and conceptual reasons for the little impact of synthesis in practice. The algorithmic reasons include the high complexity of the synthesis problem (PSPACE-complete in the closed setting [32] and 2EXPTIME-complete in the open setting [28], for specifications in LTL) as well as the intricacy of the algorithms in the open setting – the traditional approach involves determinization of automata on infinite words [31] and a solution of parity games [22].

We find the argument about the algorithmic challenge less compelling. First, experience with verification shows that even nonelementary algorithms can be practical, since the worst-case complexity does not arise often. For example, while the model-checking problem for specifications in second-order logic has nonelementary complexity, the model-checking tool MONA [16] successfully verifies many specifications given in second-order logic. Furthermore, in some sense, synthesis is not harder than verification: the complexity of synthesis is given with respect to the specification only, whereas the complexity of verification is given with respect to the specification and the system, which is typically much larger than the specification. About the intercity of the algorithms, in the last decade we have seen quite many alternatives to the traditional approach – Safraless algorithms that avoid determinization and parity games, and reduce synthesis to problems that are simpler and are amenable to optimizations and symbolic implementations [19, 25, 26].

The arguments about the conceptual and methodological reasons are more compelling. We see here three main challenges, relevant in both the closed and open settings. First, unlike verification, where a specification can be decomposed into subspecifications, each can be checked independently, in synthesis the starting point is one comprehensive specification. This inability to decompose or evolve the specification is related to the second challenge. In practice, we rarely construct systems from scratch or from one comprehensive specification. Rather, systems are constructed from existing components. This is true for both hardware systems, where we see IP cores or design libraries, and software systems, where web APIs and libraries of functions and objects are common. Third, while in verification we only automate the check of the system, automating its design is by far more risky and unpredictable – there are typically many ways to satisfy a satisfiable or realizable specification, and designers will be willing to give up manual design only if they can count on the automated synthesis tool to construct systems of comparable quality. Traditional

synthesis algorithms do not attempt to address the quality issue.

In this paper we continue earlier efforts to cope with the above conceptual challenges. Our contribution extends both the setting and the results of earlier work. The realization that design of systems proceeds by composition of underlying components is not new to the verification community. For example, [20] proposed a framework for component-based modelling that uses an abstract layered model of components, and [14] initiated a series of works on interface theories for component-based design, possibly with a reuse of components in a library [15]. The need to consider components is more evident in the context of software, where, for example, recursion is possible, so components have to be equipped with mechanisms for call and return [4]. The setting and technical details, however, are different from these in the synthesis problem we consider here. The closer to our work here is [27], which studied LTL synthesis from reusable component libraries. Lustig and Vardi studied two notions of component composition. In the first notion, termed data-flow composition, components are cascaded so that the outputs of one component are fed to other components. In the second notion, termed control-flow composition, the composition is flat and control flows among the different components. The second notion, which turns out to be the decidable one [27], is particularly suitable for modelling web-service orchestration, where users are typically offered services and interact with different parties [3].

Let us turn now to the quality issue. Traditional formal methods are based on a Boolean satisfaction notion: a system satisfies, or not, a given specification. The richness of today's systems, however, calls for specification formalisms that are *multi-valued*. The multi-valued setting arises directly in probabilistic and weighted systems and arises indirectly in applications where multi-valued satisfaction is used in order to model quantitative properties of the system like its size, security level, or quality. Reasoning about quantitative properties of systems is an active area of research in recent years, yielding quantitative specification formalisms and algorithms [13, 18, 12, 2, 11]. In quantitative reasoning, the Boolean satisfaction notion is refined and one can talk about the cost, or reward, of using a system, or, in our component-based setting, the cost of using a component from the library.

In order to capture a wide set of scenarios in practice, we associate with each component in the library two costs: a *quality cost* and a *construction cost*. The quality cost, as describes above, concerns the performance of the component and is paid each time the component is used. The construction cost is the cost of adding the component to the library. Thus, a design that uses a component pays its construction cost once. When several designs use the same component, they share its construction cost. This corresponds to real-life scenarios, where users pay, for example, for web-services, and indeed their price is influenced by the market

demand.

In [5], the authors study the problem of synthesizing a *hierarchical* system from a library of components that satisfies a specification while attempting to find a succinct system. They assume that rather than one specification, the input is a sequence of specifications $\phi_1, \ldots, \phi_m$ that attempt to guide the synthesis. The construction is then incremental. At step i, a component that satisfies ϕ_i is added to the library. The component C_m is then output as the final system.

We study synthesis from component libraries with costs in the closed and open settings. In both settings, the specification is given by means of a deterministic automaton $\mathcal{S}$ on finite words (DFA).[1] In the closed setting, the specification is a regular language over some alphabet Σ and the library consists of box-DFAs (that is, DFAs with exit states) over Σ. In the open setting, the specification $\mathcal{S}$ is over sets I and O of input and output signals, and the library consists of box-I/O-transducers. The boxes are black, in the sense that a design that uses components from the library does not see Σ (or $I \cup O$) nor it sees the behavior inside the components. Rather, the mode of operation is as in the control-flow composition of [27]: the design gives control to one of the components in the library. It then sees only the exit state through which the component completes its computation and relinquishes control. Based on this information, the design decides which component gets control next, and so on.

In more technical details, the synthesis problem gets as input the specification $\mathcal{S}$ as well as a library $\mathcal{L}$ of components $\mathcal{B}_1, \ldots, \mathcal{B}_n$. The goal is to return a correct design – a transducer $\mathcal{D}$ that reads the exit states of the components and outputs the next component to gain control. In the closed setting, correctness means that the language over Σ that is generated by the composition defined by $\mathcal{D}$ is equal to the language of $\mathcal{S}$. In the open setting, correctness means that the interaction of the composition defined by $\mathcal{D}$ with all input sequences generates a computation over $I \cup O$ that is in the language of $\mathcal{S}$.

We first study the problem without cost and reduce it to the solution of a two-player safety game $\mathcal{G}_{\mathcal{L},\mathcal{S}}$. In the closed setting, the game is of full information and the problem can be solved in polynomial time. In the open setting, the flexibility that the design have in responding to different input sequences introduces partial information to the game, and the problem is EXPTIME-complete. We note that in [27], where the open setting was studied and the specification is given by means of an LTL formula, the complexity is 2EXPTIME-complete, thus one could have

[1] It is possible to extend our results to specifications in LTL. We prefer to work with deterministic automata, as this setting isolates the complexity and technical challenges of the design problem and avoids the domination of the doubly-exponential complexity of going from LTL to deterministic automata.

expected our complexity to be only polynomial. We prove, however, hardness in EXPTIME, showing that it is not just the need to transfer the LTL formula to a deterministic formalism that leads to the high complexity.

We then turn to integrate cost to the story. As explained above, there are two types of costs associated with each component $\mathcal{B}_i$ in $\mathcal{L}$. The first type, quality cost, can be studied for each design in isolation. We show that even there, the combinatorial setting is not simple. While for the closed setting an optimal design can be induced from a memoryless strategy of the designer in the game $\mathcal{G}_{\mathcal{L},\mathcal{S}}$, making the problem of finding an optimal design NP-complete, seeking designs of optimal cost may require sophisticated compositions in the open setting. In particular, we show that optimal designs may be exponentially larger than other correct designs[2], and that an optimal design may not be induced by a memoryless strategy in $\mathcal{G}_{\mathcal{L},\mathcal{S}}$. We are still able to bound the size of an optimal transducer by the size of $\mathcal{G}_{\mathcal{L},\mathcal{S}}$, and show that the optimal synthesis problem is NEXPTIME-complete.

The second type of cost, namely construction cost, depends not only on choices made by the designer, but also on choices made by designers of other specifications that use the library. Indeed, recall that the construction cost of a component is shared by designers that use this component, with the share being proportional to the number of uses (a component can be used several times in a design). Hence, the setting gives rise to a *resource-allocation game* [30, 17]. Unlike traditional resource-allocation games, where players' strategies are sets of resources, here each strategy is a multiset – the components a designer needs. As has been the case in [8], the setting of multisets makes the game less stable. We show that the game is not guaranteed to have a *Nash Equilibrium* (NE), and that the problem of deciding whether an NE exists is Σ_2^P-complete. We then turn to the more algorithmic related problems and show that the problems of finding an optimal design given the choices of the other designers (a.k.a. the *best-response* problem, in algorithmic game theory) and of finding designs that minimize the total cost for all specifications (a.k.a. the *social optimum*) are both NP-complete.

Recently, in [9], the setting of synthesis from component libraries by multiple users has been considered also for the setting in which the costs of the components have *congestion* effects rather than *cost-sharing* as we study here. For example, components might model processors and cost can model performance. When many users use the same component, congestion increases and performance decreases.

While the cost model we describe above is suited for some settings, e.g., in cases where the goal is to minimize the number of states in the system, in other settings a *computation-based* cost model is more appropriate. For example, in a system that

[2]Recall that "optimal" here refers to the quality-cost function.

issues grants upon requests, a goal of the designer can be to design a system that minimizes the waiting time for a grant once a request is received. A standard model for reasoning about such costs of computations is *lattice automata* [24]. Such an automaton assigns to each word a value which is an element of some lattice.

We study the closed synthesis problem from component libraries where the specification is given by a deterministic lattice automaton (LDFA, for short) and the components are box LDFAs. Thus, our goal is to compose the components in the library to construct an LDFA that is equivalent to the specification LDFA, where equivalence means that the two automata assign the same values to all words. We are able to show that the problem can be solved in polynomial time using a similar idea to that in Boolean setting. Our solution introduces a new type of LDFAs, which compensate for the lack of a canonical minimal LDFA [21] and might be of independent interest.

2 Preliminaries

Automata, transducers, and boxes A *deterministic finite automaton* (DFA, for short) is a tuple $\mathcal{A} = \langle \Sigma, Q, \delta, q_0, F \rangle$, where Σ is an alphabet, Q is a set of states, $\delta : Q \times \Sigma \to Q$ is a partial transition function, $q_0 \in Q$ is an initial states, and $F \subseteq Q$ is a set of accepting states. We extend δ to words in an expected way, thus $\delta^* : Q \times \Sigma^* \to Q$ is such that for $q \in Q$, we have $\delta^*(q, \epsilon) = q$ and for $w \in \Sigma^*$ and $\sigma \in \Sigma$, we have $\delta^*(q, w \cdot \sigma) = \delta(\delta^*(q, w), \sigma)$. When $q = q_0$, we sometimes omit it, thus $\delta^*(w)$ is the state that $\mathcal{A}$ reaches after reading w. We assume that all states are reachable from q_0, thus for every $q \in Q$ there exists a word $w \in \Sigma^*$ such that $\delta^*(w) = q$. We refer to the *size* of $\mathcal{A}$, denoted $|\mathcal{A}|$, as the number of its states.

The *run* of $\mathcal{A}$ on a word $w = w_1, \ldots w_n \in \Sigma^*$ is the sequence of states $r = r_0, r_1, \ldots, r_n$ such that $r_0 = q_0$ and for every $0 \leqslant i \leqslant n-1$ we have $r_{i+1} = \delta(r_i, w_{i+1})$. The run r is accepting iff $r_n \in F$. The *language* of $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of words $w \in \Sigma^*$ such that the run of $\mathcal{A}$ on w is accepting, or, equivalently, $\delta^*(w) \in F$. For $q \in Q$, we denote by $L(\mathcal{A}^q)$ the language of the DFA that is the same as $\mathcal{A}$ only with initial state q. Note that since $\mathcal{A}$ is deterministic and δ is partial, there is at most one run of $\mathcal{A}$ on each word.

A *transducer* models an interaction between a system and its environment. It is similar to a DFA except that in addition to Σ, which is referred to as the input alphabet, denoted Σ_I, there is an output alphabet, denoted Σ_O, and rather than being classified to accepting or rejecting, each state is labeled by a letter from Σ_O[3]. Formally, a transducer is a tuple $\mathcal{T} = \langle \Sigma_I, \Sigma_O, Q, q_0, \delta, \nu \rangle$, where Σ_I is an input alphabet, Σ_O is an output alphabet, Q, q_0, and $\delta : Q \times \Sigma_I \to Q$ are as in a DFA,

[3]These transducers are sometimes referred to as *Moore machines*.

and $\nu : Q \to \Sigma_O$ is an output function. We require $\mathcal{T}$ to be *receptive*. That is, δ is complete, so for every input word $w \in \Sigma_I^*$, there is a run of $\mathcal{T}$ on w. Consider an input word $w = w_1, \ldots, w_n \in \Sigma_I^*$. Let $r = r_0, \ldots, r_n$ be the run of $\mathcal{T}$ on w. The *computation of* $\mathcal{T}$ *in* w is then $\sigma_1, \ldots, \sigma_n \in (\Sigma_I \times \Sigma_O)^*$, where for $1 \leqslant i \leqslant n$, we have $\sigma_i = \langle w_i, \nu(r_{i-1})\rangle$. We define the language of $\mathcal{T}$, denoted $L(\mathcal{T})$, as the set of all its computations. For a specification $L \subseteq (\Sigma_I \times \Sigma_O)^*$, we say that $\mathcal{T}$ *realizes* L iff $L(\mathcal{T}) \subseteq L$. Thus, no matter what the input sequence is, the interaction of $\mathcal{T}$ with the environment generates a computation that satisfies the specification. For two words $u \in \Sigma_I^*$ and $v \in \Sigma_O^*$ of length n we define the *product* of the two words, denoted $u \oplus v$, as $w = w_1 \ldots w_n \in (\Sigma_I \times \Sigma_O)^*$, where, for $1 \leqslant i \leqslant n$, we have $w_i = \langle u_i, v_i\rangle$.

By adding *exit states* to DFAs and transducers, we can view them as components from which we can compose systems. Formally, we consider two types of components. Closed components are modeled by *box-DFAs* and open components are modeled by *box-transducers*. A box-DFA augments a DFA by a set of exit states. Thus, a box-DFA is a tuple $\langle \Sigma, Q, \delta, q_0, F, E\rangle$, where $E \subseteq Q$ is a nonempty set of exit states. There are no outgoing transitions from an exit state. Also, the initial state cannot be an exit state and exit states are not accepting. Thus, $q_0 \notin E$ and $F \cap E = \varnothing$. Box-transducers are defined similarly, and their exit states are not labeled, thus $\nu : Q \backslash E \to \Sigma_O$.

Component libraries A *component library* is a collection of boxes $\mathcal{L} = \{\mathcal{B}_1, \ldots, \mathcal{B}_n\}$. We say that $\mathcal{L}$ is a *closed library* if the boxes are box-DFAs, and is an *open library* if the boxes are box-transducers. Let $[n] = \{1, \ldots, n\}$. In the first case, for $i \in [n]$, let $\mathcal{B}_i = \langle \Sigma, C_i, \delta_i, c_i^0, F_i, E_i\rangle$. In the second case, $\mathcal{B}_i = \langle \Sigma_I, \Sigma_O, C_i, \delta_i, c_i^0, \nu_i, E_i\rangle$. Note that all boxes in $\mathcal{L}$ share the same alphabet (input and output alphabet, in the case of transducers). We assume that the states of the components are disjoint, thus for every $i \neq j \in [n]$, we have $C_i \cap C_j = \varnothing$. We use the following abbreviations $\mathcal{C} = \bigcup_{i\in[n]} C_i$, $\mathcal{C}_0 = \bigcup_{i\in[n]} \{c_i^0\}$, $\mathcal{F} = \bigcup_{i\in[n]} F_i$, and $\mathcal{E} = \bigcup_{i\in[n]} E_i$. We define the *size* of $\mathcal{L}$ as $|\mathcal{C}|$.

We start by describing the intuition for composition of closed libraries. A *design* is a recipe to compose the components of a library $\mathcal{L}$ (allowing multiple uses) into a DFA. A run of the design on a word starts in an initial state of one of the components in $\mathcal{L}$. We say that this component has the initial *control*. When a component is in control, the run uses its states, follows its transition function, and if the run ends, it is accepting iff it ends in one of the components' accepting states. A component relinquishes control when the run reaches one of its exit states. It is then the design's duty to assign control to the next component, which gains control through its initial state.

Formally, a design is a transducer $\mathcal{D}$ with input alphabet $\mathcal{E}$ and output alphabet

$[n]$. We can think of $\mathcal{D}$ as running beside the components. When a component reaches an exit state e, then $\mathcal{D}$ reads the input letter e, proceeds to its next state, and outputs the index of the component to gain control next. Note that $\mathcal{D}$ does not read the alphabet Σ and has no information about the states that the component visits. It only sees which exit state has been reached.

Consider a design $\mathcal{D} = \langle \mathcal{E}, [n], D, \delta, d^0, \nu \rangle$ and a closed library $\mathcal{L}$. We formalize the behavior of $\mathcal{D}$ by means of the *composition DFA* $\mathcal{A}_{\mathcal{L},\mathcal{D}}$ that simulates the run of $\mathcal{D}$ along with the runs of the box-DFAs. Formally, $\mathcal{A}_{\mathcal{L},\mathcal{D}} = \langle \Sigma, Q_{\mathcal{L},\mathcal{D}}, \delta_{\mathcal{L},\mathcal{D}}, q^0_{\mathcal{L},\mathcal{D}}, F_{\mathcal{L},\mathcal{D}} \rangle$ is defined as follows. The set of states $Q_{\mathcal{L},\mathcal{D}} \subseteq (\mathcal{C} \backslash \mathcal{E}) \times D$ consists of pairs of a *component state* from $\mathcal{C}$ and an *design state* from S. The component states are consistent with ν, thus $Q_{\mathcal{L},\mathcal{D}} = \bigcup_{i \in [n]} (C_i \backslash E_i) \times \{q : \nu(q) = i\}$. In exit states, the composition immediately moves to the initial state of the next component, which is why the component states of $\mathcal{A}_{\mathcal{L},\mathcal{D}}$ do not include $\mathcal{E}$. Consider a state $\langle c, q \rangle \in Q_{\mathcal{L},\mathcal{D}}$ and a letter $\sigma \in \Sigma$. Let $i \in [n]$ be such that $c \in C_i$. When a run of $\mathcal{A}_{\mathcal{L},\mathcal{D}}$ reaches the state $\langle c, q \rangle$, the component $\mathcal{B}_i$ is in control. Recall that c is not an exit state. Let $c' = \delta_i(c, \sigma)$. If $c' \notin E_i$, then $\mathcal{B}_i$ does not relinquish control after reading σ and $\delta_{\mathcal{L},\mathcal{D}}(\langle c, q \rangle, \sigma) = \langle c', q \rangle$. If $c' \in E_i$, then $\mathcal{B}_i$ relinquishes control through c', and it is the design's task to choose the next component to gain control. Let $q' = \delta(q, c')$ and let $j = \nu(q')$. Then, $\mathcal{B}_j$ is the next component to gain control (possibly $j = i$). Accordingly, we advance $\mathcal{D}$ to q' and continue to the initial state of $\mathcal{B}_j$. Formally, $\delta_{\mathcal{L},\mathcal{D}}(\langle c, q \rangle, \sigma) = \langle c^0_j, q' \rangle$. (Recall that $c^0_j \notin E_j$, so the new state is in $Q_{\mathcal{L},\mathcal{D}}$.) Note also that a visit in c' is skipped. The component that gains initial control is chosen according to $\nu(d^0)$. Thus, $q^0_{\mathcal{L},\mathcal{D}} = \langle c^0_j, d^0 \rangle$, where $j = \nu(d^0)$. Finally, the accepting states of $\mathcal{A}_{\mathcal{L},\mathcal{D}}$ are these in which the component state is accepting, thus $F_{\mathcal{L},\mathcal{D}} = \mathcal{F} \times D$.

The definition of a composition for an open library is similar. There, the composition is a transducer $\mathcal{T}_{\mathcal{L},\mathcal{D}} = \langle \Sigma_I, \Sigma_O, Q_{\mathcal{L},\mathcal{D}}, \delta_{\mathcal{L},\mathcal{D}}, q^0_{\mathcal{L},\mathcal{D}}, \nu_{\mathcal{L},\mathcal{D}} \rangle$, where $Q_{\mathcal{L},\mathcal{D}}$, $q^0_{\mathcal{L},\mathcal{D}}$, and $\delta_{\mathcal{L},\mathcal{D}}$ are as in the closed setting, except that $\delta_{\mathcal{L},\mathcal{D}}$ reads letters in Σ_I, and $\nu_{\mathcal{L},\mathcal{D}}(\langle c, q \rangle) = \nu_i(c)$, for $i \in [n]$ such that $c \in C_i$.

Consider a closed-library $\mathcal{L}$, a design $\mathcal{D}$, and the run r of $\mathcal{A}_{\mathcal{L},\mathcal{D}}$ on $w = w_0 \cdots w_l$. We partition w according to positions in which control is transferred among components. Equivalently, positions in which r skips visits in exit states. Thus, $w = y_0 \cdots y_k$ is such that for all $0 \leqslant i < k$, we have that $y_i \in \Sigma^+$ and the composition $\mathcal{A}_{\mathcal{L},\mathcal{D}}$ takes a transfer transition exactly when it reads the last letter of y_i. An exception is y_k, which may be empty (this happens when r ends upon entering the last component to gain control). We then say that w is *suffix-less*. The definitions in the open setting are similar.

3 The Design Problem

The *design problem* gets as input a component library $\mathcal{L}$ and a specification that is given by means of a DFA $\mathcal{S}$. The problem is to decide whether there exists a correct design for $\mathcal{S}$ using the components in $\mathcal{L}$. In the closed setting, a design $\mathcal{D}$ is correct if $L(\mathcal{A}_{\mathcal{L},\mathcal{D}}) = L(\mathcal{S})$. In the open setting, $\mathcal{D}$ is correct if the transducer $\mathcal{T}_{\mathcal{L},\mathcal{D}}$ realizes $\mathcal{S}$. Our solution to the design problem reduces it to the problem of finding the winner in a turn-based two-player game, defined below.

A *turn-based two-player game* is played on an arena $\langle V, \Delta, V_0, \alpha \rangle$, where $V = V_1 \cup V_2$ is a set of vertices that are partitioned between Player 1 and Player 2, $\Delta \subseteq V \times V$ is a set of directed edges, $V_0 \subseteq V$ is a set of initial vertices, and α is an objective for Player 1, specifying a subset of V^ω. We consider here *safety games*, where $\alpha \subseteq V$ is a set of vertices that are *safe* for Player 1. The game is played as follows. Initially, Player 1 places a token on one of the vertices in V_0. Assume the token is placed on a vertex $v \in V$ at the beginning of a round. The player that owns v is the player that moves the token to the next vertex, where the legal vertices to continue to are $\{v' \in V : \langle v, v' \rangle \in \Delta\}$. The outcome of the game is a *play* $\pi \in V^\omega$. The play is winning for Player 1 if for every $i \geqslant 1$, we have $\pi_i \in \alpha$. Otherwise, Player 2 wins.

A *strategy* for Player i, for $i \in \{1, 2\}$, is a recipe that, given a prefix of a play, tells the player what his next move should be. Thus, it is a function $f_i : V^* \cdot V_i \to V$ such that for every play $\pi \cdot v \in V^*$ with $v \in V_i$, we have $\langle v, f_i(\pi \cdot v) \rangle \in \Delta$. Since Player 1 moves first, we require that $f_1(\epsilon)$ is defined and is in V_0. For strategies f_1 and f_2 for players 1 and 2 respectively, the play $out(f_1, f_2) \in V^\omega$ is the unique play that is the outcome the game when the players follow their strategies. A strategy f_i for Player i is *memoryless* if it depends only in the current vertex, thus it is a function $f_i : V_i \to V$.

A strategy is *winning* for a player if by using it he wins against every strategy of the other player. Formally, a strategy f_1 is winning for Player 1 iff for every strategy f_2 for Player 2, Player 1 wins the play $out(f_1, f_2)$. The definition for Player 2 is dual. It is well known that safety games are *determined*, namely, exactly one player has a winning strategy, and admits *memoryless* strategies, namely, Player i has a winning strategy iff he has a memoryless winning strategy. Deciding the winner of a safety game can done in linear time.

Solving the design problem We describe the intuition of our solution for the design problems. Given a library $\mathcal{L}$ and a specification $\mathcal{S}$ we construct a safety game $\mathcal{G}_{\mathcal{L},\mathcal{S}}$ such that Player 1 wins $\mathcal{G}_{\mathcal{L},\mathcal{S}}$ iff there is a correct design for $\mathcal{S}$ using the components in $\mathcal{L}$. Intuitively, Player 1's goal is to construct a correct design, thus he

chooses the components to gain control. Player 2 challenges the design that Player 1 chooses, thus he chooses a word (over Σ in the closed setting and over $\Sigma_I \times \Sigma_O$ in the open setting) and wins if his word is a witness for the incorrectness of Player 1's design.

Closed designs The input to the closed-design problem is a closed-library $\mathcal{L}$ and a DFA $\mathcal{S}$ over the alphabet Σ. The goal is to find a correct design $\mathcal{D}$. Recall that $\mathcal{D}$ is correct if the DFA $\mathcal{A}_{\mathcal{L},\mathcal{D}}$ that is constructed from $\mathcal{L}$ using $\mathcal{D}$ satisfies $L(\mathcal{A}_{\mathcal{L},\mathcal{D}}) = L(\mathcal{S})$. We assume that $\mathcal{S}$ is the minimal DFA for the language $L(\mathcal{S})$.

Theorem 3.1 *The closed-design problem can be solved in polynomial time.*

Proof: Given a closed-library $\mathcal{L}$ and a DFA $\mathcal{S} = \langle \Sigma, S, \delta_{\mathcal{S}}, s^0, F_{\mathcal{S}} \rangle$, we describe a safety game $\mathcal{G}_{\mathcal{L},\mathcal{S}}$ such that Player 1 wins $\mathcal{G}_{\mathcal{L},\mathcal{S}}$ iff there is a design of $\mathcal{S}$ using components from $\mathcal{L}$. Recall that $\mathcal{L}$ consists of box-DFAs $\mathcal{B}_i = \langle \Sigma, C_i, \delta_i, c_i^0, F_i, E_i \rangle$, for $i \in [n]$, and that we use $\mathcal{C}$, $\mathcal{C}_0$, $\mathcal{E}$, and $\mathcal{F}$ to denote the union of all states, initial states, exit states, and accepting states in all the components of $\mathcal{L}$. The number of vertices in $\mathcal{G}_{\mathcal{L},\mathcal{S}}$ is $|(\mathcal{C}_0 \cup \mathcal{E}) \times S|$ and it can be constructed in polynomial time. Since solving safety games can be done in linear time, the theorem follows.

We define $\mathcal{G}_{\mathcal{L},\mathcal{S}} = \langle V, E, V_0, \alpha \rangle$. First, $V = (\mathcal{C}_0 \cup \mathcal{E}) \times S$ and $V_0 = \mathcal{C}_0 \times \{s^0\}$. Recall that Player 1 moves when it is time to decide the next (or first) component to gain control. Accordingly, $V_1 = \mathcal{E} \times S$. Also, Player 2 challenges the design suggested by Player 1 and chooses the word that is processed in a component that gains control, so $V_2 = \mathcal{C}_0 \times S$.

Consider a vertex $\langle e, s \rangle \in V_1$. Player 1 selects the next component to gain control. This component gains control through its initial state. Accordingly, E contains edges $\langle \langle e, s \rangle, \langle c_i^0, s \rangle \rangle$, for every $i \in [n]$. Note that since no letter is read when control is passed, we do not advance the state in $\mathcal{S}$. Consider a vertex $v = \langle c_i^0, s \rangle \in V_2$. Player 2 selects the word that is read in the component $\mathcal{B}_i$, or equivalently, he selects the exit state from which $\mathcal{B}_i$ relinquishes control. Thus, E contains an edge $\langle \langle c_i^0, s \rangle, \langle e, s' \rangle \rangle$ iff there exists a word $u \in \Sigma^*$ such that $\delta_i^*(u) = e$ and $\delta_{\mathcal{S}}^*(s, u) = s'$.

We now turn to define the winning condition. All the vertices in V_1 are in α. A vertex $v \in V_2$ is not in α if it is possible to extend the word traversed for reaching v to a witness for the incorrectness of $\mathcal{D}$. Accordingly, a vertex $\langle c_i^0, s \rangle$ is not in α if one of the following holds. First ("the suffix witness"), there is a finite word that is read inside the current component and witnesses the incorrectness. Formally, there is $u \in \Sigma^*$ such that $\delta_i^*(u) \in F_i$ and $\delta_{\mathcal{S}}^*(s, u) \notin F_{\mathcal{S}}$, or $\delta_i^*(u) \in C_i \backslash (F_i \cup E_i)$ and $\delta_{\mathcal{S}}^*(s, u) \in F_{\mathcal{S}}$. Second ("the infix witness"), there are two words that reach the same exit state of the current component yet the behavior of $\mathcal{S}$ along them is different. Formally, there exist words $u, u' \in \Sigma^*$ such that $\delta_i^*(u) = \delta_i^*(u') \in E_i$ and

$\delta^*_{\mathcal{S}}(s,u) \neq \delta^*_{\mathcal{S}}(s,u')$. Intuitively, the minimality of $\mathcal{S}$ enables us to extend either u or u' to an incorrectness witness. Given $\mathcal{L}$ and $\mathcal{S}$, the game $\mathcal{G}_{\mathcal{L},\mathcal{S}}$ can be constructed in polynomial time.

We claim that there is a correct design $\mathcal{D}$ iff Player 1 wins $\mathcal{G}_{\mathcal{L},\mathcal{S}}$. Assume first that there is a correct design $\mathcal{D} = \langle \mathcal{E}, [n], D, \delta, d^0, \nu \rangle$, thus $L(\mathcal{A}_{\mathcal{L},\mathcal{D}}) = L(\mathcal{S})$. We construct a winning strategy $f_{\mathcal{D}}$ for Player 1. The strategy $f_{\mathcal{D}}$ proceeds like $\mathcal{D}$. First, $f_{\mathcal{D}}(\epsilon) = \langle c^0_i, s^0 \rangle$, with $i = \nu(d^0)$. Then, for a finite play π, let $\langle e_0, s_0 \rangle, \langle e_1, s_1 \rangle \ldots \langle e_m, s_m \rangle$ be its projection on V_1. Thus, $e_0, \ldots, e_m \in \mathcal{E}$ and $s_0, \ldots, s_m \in S$. We define $f_{\mathcal{D}}(\pi) = \langle c^0_i, s_m \rangle$, for $i = \nu(\delta^*_{\mathcal{D}}(e_0, \ldots, e_m))$.

We claim that $f_{\mathcal{D}}$ is a winning strategy. Assume towards contradiction that there is a Player 2 strategy f_2 that wins against $f_{\mathcal{D}}$. Let $\pi = out(f_{\mathcal{D}}, f_2)$, let $\langle c^0_i, s \rangle$ be its first vertex that is not in α, and let w be the word in Σ^* that Player 2 follows along the prefix of π that reaches $\langle c^0_i, s \rangle$. Finally, let $d \in D$ be such that $\langle c^0_i, d \rangle$ is the state that $\mathcal{A}_{\mathcal{L},\mathcal{D}}$ reaches when it reads w. Since we define $f_{\mathcal{D}}$ to agree with $\mathcal{D}$, then the component state of this state in $\mathcal{A}_{\mathcal{L},\mathcal{D}}$ is indeed c^0_i.

We distinguish between two cases. First, if $\langle c^0_i, s \rangle$ exits α because of a suffix witness, let $u \in \Sigma^*$ be such that $\delta^*_i(u) \in F_i$ and $\delta^*_{\mathcal{S}}(s,u) \notin F_{\mathcal{S}}$. (The case where $\delta^*_i(u) \notin (F_i \cup E_i)$ and $\delta^*_{\mathcal{S}}(s,u) \in F_{\mathcal{S}}$ is similar). Let $c = \delta^*_i(u)$. The run of $\mathcal{A}_{\mathcal{L},\mathcal{D}}$ on $w \cdot u$ ends in the state $\langle \delta^*_i(u), s \rangle$. Since $\delta^*_i(u) \in F_i$, we have $w \cdot u \in L(\mathcal{A}_{\mathcal{L},\mathcal{D}})$. By the definition of E, we have that $\delta^*_{\mathcal{S}}(w) = s$. Since $\delta^*_{\mathcal{S}}(s,u) \notin F_{\mathcal{S}}$, we have $w \cdot u \notin L(\mathcal{S})$. Thus, $L(\mathcal{A}_{\mathcal{L},\mathcal{D}}) \neq L(\mathcal{S})$, and we reach a contradiction to the correctness of $\mathcal{D}$.

In the second case, of an infix witness, there exist words $u, u' \in \Sigma^*$ such that $\delta^*_i(u) = \delta^*_i(u') \in E_i$ and $\delta^*_{\mathcal{S}}(s,u) = p \neq p' = \delta^*_{\mathcal{S}}(s,u')$. Since $\mathcal{S}$ is a minimal DFA for $L(\mathcal{S})$, we have $L(\mathcal{S}^p) \neq L(\mathcal{S}^{p'})$. Thus, wlog, there is a word $z \in L(\mathcal{S}^p) \backslash L(\mathcal{S}^{p'})$. Recall that $\delta^*_{\mathcal{L},\mathcal{D}}(w) = \langle c^0_i, d \rangle$. Let $d' = \delta_{\mathcal{D}}(d,e)$ and $j = \nu(d')$. Since $\delta^*_{\mathcal{L},\mathcal{D}}(w \cdot u) = \delta^*_{\mathcal{L},\mathcal{D}}(w \cdot u') = \langle c^0_j, d' \rangle$, we have $\delta^*_{\mathcal{L},\mathcal{D}}(w \cdot u \cdot z) = \delta^*_{\mathcal{L},\mathcal{D}}(w \cdot u' \cdot z)$. Thus, $w \cdot u \cdot z \in L(\mathcal{A}_{\mathcal{L},\mathcal{D}})$ iff $w \cdot u' \cdot z \in L(\mathcal{A}_{\mathcal{L},\mathcal{D}})$. However, $w \cdot u \cdot z \in L(\mathcal{S})$ and $w \cdot u' \cdot z \notin L(\mathcal{S})$. Thus, we reach a contradiction to the correctness of $\mathcal{D}$, and we are done.

Assume now that Player 1 wins the game $\mathcal{G}_{\mathcal{L},\mathcal{S}}$. Let f be a memoryless winning strategy for Player 1. We construct a correct design $\mathcal{D}_f$ from f. Note that all the successors of a vertex in V_1 are in V_2. Thus, $f : V_1 \to V_2$. We define $\mathcal{D}_f = \langle \mathcal{E}, [n], D, \delta, s^0, \nu \rangle$ as follows. First, $D = V_2 = \mathcal{C}_0 \times S$. Consider a state $v = \langle c^0_i, s \rangle \in V_2 \cap \alpha$. Recall that c^0_i is the initial state of the component $\mathcal{B}_i$. Since $v \in \alpha$, the lack of an infix witness implies that for every exit state $e \in E_i$ there is exactly one state $s' \in S$ such that $\langle \langle c^0_i, s \rangle, \langle e, s' \rangle \rangle \in E$. We define $\delta(v,e) = f(\langle e, s' \rangle)$. Note that if $v \notin \alpha$ or $e \notin E_i$, then we can define $\delta(v,e)$ arbitrarily. The labeling function ν is defined as expected, with $\nu(\langle c^0_i, s \rangle) = i$.

We prove that $\mathcal{D}_f$ is a correct design. Assume towards contradiction that there is a word $w \in L(\mathcal{A}_{\mathcal{L},\mathcal{D}_f}) \backslash L(\mathcal{S})$. The case where $w \in L(\mathcal{S}) \backslash L(\mathcal{A}_{\mathcal{L},\mathcal{D}_f})$ is similar. Consider

the run r of $\mathcal{A}_{\mathcal{L},\mathcal{D}_f}$ on w. Let $\mathcal{B}_{i_1},\ldots,\mathcal{B}_{i_m} \in \mathcal{L}^*$ be sequence of components that r traverses and $e_{i_1},\ldots,e_{i_{m-1}} \in \mathcal{E}^*$ be the corresponding exit states. Let $y_1,\ldots,y_m$ be the partition of w according to $\mathcal{D}$. Thus, for $1 \leqslant j < m$, we have $y_i \in \Sigma^+$ and $\delta^*_{i_j}(y_i) = e_{i_j} \in E_{i_j}$, and $y_m \in \Sigma^*$. Since $w \in L(\mathcal{A}_{\mathcal{L},\mathcal{D}_f})$, we have $\delta^*_{i_m}(w_m) \in F_{i_m}$. Note that the word $y_1 \cdots y_{m-1} \in \Sigma^*$ is suffix-less, thus $\delta^*_{\mathcal{L},\mathcal{D}_f}(y_1 \cdot \ldots \cdot y_{m-1}) = \langle c^0_{i_m}, d\rangle$ for some $d \in D$ with $\nu(d) = i_m$. Since we defined $\mathcal{D}_f$ to agree with f on the components that gain control, the finite play π that is the outcome of the game when Player 1 plays f and Player 2 chooses the exit states $e_{i_1},\ldots,e_{i_{m-1}}$ reaches the Player 2 vertex $v = \langle c^0_{i_m}, s\rangle \in V_2$, for $s \in S$ such that $\delta^*_{\mathcal{S}}(y_1 \cdots y_{m-1}) = s$. We claim that $v \notin \alpha$. Indeed, $\delta^*_i(y_m) \in F_i$ and since $w \notin L(\mathcal{S})$, we have $y_m \notin L(\mathcal{S}^s)$. Thus, π is a winning play for Player 2, contradicting our assumption that f is a winning strategy, and we are done. □

Open designs We continue to study the open setting. Recall that there, the input is a DFA $\mathcal{S}$ over the alphabet $\Sigma_I \times \Sigma_O$ and an open library $\mathcal{L}$. The goal is to find a correct design $\mathcal{D}$ or return that no such design exists, where $\mathcal{D}$ is correct if the composition transducer $\mathcal{T}_{\mathcal{L},\mathcal{D}}$ realizes $L(\mathcal{S})$.

Lustig and Vardi [27] studied the design problem in a setting in which the specification is given by means of an LTL formula. They showed that the problem is 2EXPTIME-complete. Given an LTL formula one can construct a deterministic parity automaton that recognizes the language of words that satisfy the formula. The size of the automaton is doubly-exponential in the size of the formula. Thus, one might guess that the design problem in a setting in which the specification is given by means of a DFA would be solvable in polynomial time. We show that this is not the case and that the problem is EXPTIME-complete. As in [27], our upper bound is based on the ability to "summarize" the activity inside the components. Starting with an LTL formula, the solution in [27] has to combine the complexity involved in the translation of the LTL formula into an automaton with the complexity of finding a design, which is done by going throughout a universal word automaton that is expanded to a tree automaton. Starting with a deterministic automaton, our solution directly uses games. The interesting contribution, however, is the lower bound, showing that problem is EXPTIME-hard even when the specification is given by means of a deterministic automaton. We start with the upper bound.

Theorem 3.2 *The open-design problem is in EXPTIME.*

Proof: Given an open-library $\mathcal{L}$ and a DFA $\mathcal{S} = \langle \Sigma_I \times \Sigma_O, S, \delta_{\mathcal{S}}, s^0, F_{\mathcal{S}}\rangle$, we describe a safety game $\mathcal{G}_{\mathcal{L},\mathcal{S}}$ such that Player 1 wins $\mathcal{G}_{\mathcal{L},\mathcal{S}}$ iff there is a design for $\mathcal{S}$ using components from $\mathcal{L}$. The number of vertices in $\mathcal{G}_{\mathcal{L},\mathcal{S}}$ is exponential in S and

$\mathcal{C}$. Since solving safety games can be done in linear time, membership in EXPTIME follows.

We define $\mathcal{G}_{\mathcal{L},\mathcal{S}} = \langle V, E, V_0, \alpha\rangle$ as follows.[4] Recall that $\mathcal{C}$, $\mathcal{C}_0$, $\mathcal{E}$, and $\mathcal{F}$ are the union of all states, initial states, exit states, and accepting states in all the components of $\mathcal{L}$. We define $V = (\mathcal{C}_0 \cup \mathcal{E}) \times 2^S$ with $V_1 = \mathcal{E} \times 2^S$ and $V_2 = \mathcal{C}_0 \times 2^S$. Also, $V_0 = \mathcal{C}_0 \times \{\{s^0\}\}$. As in the closed-setting, Player 1 selects the components that gain control, thus for a vertex $\langle e, T\rangle \in V_1$ we have $\langle\langle e, T\rangle, \langle c_i^0, T\rangle\rangle \in E$, for every $c_i^0 \in \mathcal{C}_0$. Player 2 selects the word that is processed in the component, or equivalently, the exit state from which it relinquishes control, thus for a vertex $v = \langle c_i^0, T\rangle \in V_2$ we have $\langle\langle c_i^0, T\rangle, \langle e, T'\rangle\rangle \in E$ iff for every $s' \in T'$ there is a state $s \in T$ and a word $u \in \Sigma_I^*$ such that $\delta_i^*(u) = e$ and, assuming $w \in (\Sigma_I \times \Sigma_O)^*$ is the computation of $\mathcal{B}_i$ that corresponds to u, we have $\delta_{\mathcal{S}}^*(s, w) = s'$. Note that for $c_i^0 \in \mathcal{C}_0$, $T \in 2^S$, and $e \in \mathcal{E}$, there is at most one, nonempty, subset $T' \in 2^S$ such that $\langle\langle c_i^0, T\rangle, \langle e, T'\rangle\rangle \in E$. The set of vertices that are loosing for Player 1 consists of states $\langle c_i^0, T\rangle \in V_2$ from which Player 2 can generate a suffix-witness to the incorrectness of the design. Formally, $\langle c_i^0, T\rangle \in \alpha$ iff there exists $u \in L(\mathcal{B}_i)$ and $s \in T$ such that $u \notin L(\mathcal{S}^s)$.

We claim that there is a correct design $\mathcal{D}$ iff Player 1 wins $\mathcal{G}_{\mathcal{L},\mathcal{S}}$. For the first direction, consider a correct design $\mathcal{D}$, thus $L(\mathcal{T}_{\mathcal{L},\mathcal{D}}) \subseteq L(\mathcal{S})$. We construct a winning Player 1 strategy $f_{\mathcal{D}}$. Recall that a design reads exit states and outputs components. Further recall that assuming the game does not end, a Player 2 move is a choice of an exit state. We define $f_{\mathcal{D}}$ so that it responds to Player 2's choice the same way $\mathcal{D}$ responds. We define $f_{\mathcal{D}}(\epsilon) = \langle c_i^0, \{\{s^0\}\}\rangle$ for $i = \nu(d^0)$, thus $f_{\mathcal{D}}$ and $\mathcal{D}$ assign initial control to the same component. Consider a finite play π and let $\langle e_1, T_1\rangle, \ldots, \langle e_m, T_m\rangle$ be the projection of π on V_2, thus $e_1, \ldots, e_m \in \mathcal{E}$ and $T_1, \ldots, T_m \in 2^S$. We define $f_{\mathcal{D}}(\pi) = \langle c_i^0, T_m\rangle$ where $\nu(\delta_{\mathcal{D}}^*(e_1, \ldots, e_m)) = i$.

We claim that $f_{\mathcal{D}}$ is a winning strategy. Assume towards contradiction that there is a Player 2 strategy f_w that wins against $f_{\mathcal{D}}$. Let $\pi = \langle c_{i_1}^0, \{s^0\}\rangle, \langle e_{i_1}, T_1\rangle, \langle c_{i_2}^0, T_1\rangle, \langle e_{i_2}, T_2\rangle, \ldots, \langle e_{i_{m-1}}, T_m\rangle, \langle c_{i_m}^0, T_m\rangle$ be the finite losing prefix of $out(f_{\mathcal{D}}, f_w)$, thus $\langle c_{i_m}^0, T_m\rangle \notin \alpha$. Since $\langle c_{i_m}^0, T_m\rangle \notin \alpha$ there is a state $s_m \in T_m$ and a word $w_m \in L(\mathcal{B}_i)$ such that $w_m \notin L(\mathcal{S}^{s_m})$. It is not hard to see that there are words $w_1, \ldots, w_{m-1} \in (\Sigma_I \times \Sigma_O)^*$ and states $s_1, \ldots, s_{m-1} \in S$ such that for $1 \leqslant j \leqslant m-1$ we have $w_j \in L(\mathcal{B}_{i_j})$, $\delta_{i_j}^*(c_{i_j}^0, w_j) = e_{i_j}$, $\delta_{\mathcal{S}}^*(w_1) = s_1$, and $\delta_{\mathcal{S}}^*(s_j, w_j) = s_{j+1}$. It is not hard to see that since we defined $f_{\mathcal{D}}$ to agree with $\mathcal{D}$, the components that gain control in the run of $\mathcal{T}_{\mathcal{L},\mathcal{D}}$ on $w = w_1 \cdot \ldots w_m$ are $\mathcal{B}_{i_1}, \ldots, \mathcal{B}_{i_m}$, thus it is possible to prove by induction that $w \in L(\mathcal{T}_{\mathcal{L},\mathcal{D}})$. Moreover, the run of $\mathcal{S}$ on w is not accepting,

[4] A different way to construct $\mathcal{G}_{\mathcal{L},\mathcal{S}}$ would be to go through a *partial-information* game (see Theorem 3.3) with vertices in $\mathcal{C} \times S$, where Player 1 cannot distinguish between vertices $\langle e, d\rangle$ and $\langle e, d'\rangle$ for $e \in \mathcal{E}$ and $d, d' \in S$. The game $\mathcal{S}$ is the corresponding game. We describe it directly, which also shows that the exponential dependency is only in S.

thus $w \in L(\mathcal{T}_{\mathcal{L},\mathcal{D}}) \backslash L(\mathcal{S})$, and we reach a contradiction to the correctness of $\mathcal{D}$.

We continue to the second direction. Assume Player 1 wins the game $\mathcal{G}_{\mathcal{L},\mathcal{S}}$. Thus, he has a memoryless winning strategy $f_{\mathcal{D}}$ from which we construct a design $\mathcal{D}$. Intuitively, in $\mathcal{D}$, we skip exit states and proceed according to $f_{\mathcal{D}}$. Thus, the states of $\mathcal{D}$ are $V_2 = \mathcal{C}_0 \times 2^S$. Consider a state $\langle c_i^0, T\rangle \in V_2$, where recall that c_i^0 is the initial state of the component $\mathcal{B}_i \in \mathcal{L}$, and an exit state $e \in E_i$ of $\mathcal{B}_i$. Recall that there is a unique subset $T' \in 2^S$ such that $\langle\langle c_i^0, T\rangle, \langle e, T'\rangle\rangle \in E$. We define $\delta_{\mathcal{D}}(\langle c_i^0, T\rangle, e) = f_{\mathcal{D}}(\langle e, T'\rangle)$.

We claim that $\mathcal{D}$ is a correct design. Assume towards contradiction that there is a word $w \in L(\mathcal{T}_{\mathcal{L},\mathcal{D}}) \backslash L(\mathcal{S})$. Consider the run r of $\mathcal{T}_{\mathcal{L},\mathcal{D}}$ on w. Let $\mathcal{B}_{i_1}, \ldots, \mathcal{B}_{i_m} \in \mathcal{L}^*$ be sequence of components that r traverses. Let $w_1, \ldots, w_m$ be the partition of w according to $\mathcal{D}$. That is, for $1 \leqslant j \leqslant m$, the subword w_j is induced while r is in component $\mathcal{B}_{i_j}$ and $\delta_{i_j}^*(w_j) \in E_{i_j}$. Let $\pi_w = e_{i_1}, \ldots, e_{i_{m-1}} \in \mathcal{E}^*$ be the exit states that r visits. Note that since B_{i_m} gains control last the word $w_1 \cdot \ldots \cdot w_{m-1}$ is suffix-less, thus $\delta_{\mathcal{L},\mathcal{D}}^*(w_1 \cdot \ldots \cdot w_{m-1}) = \langle c_{i_m}^0, d\rangle$ for some $d \in D$ having $\nu(d) = i_m$. Moreover, $w_m \in L(\mathcal{B}_{i_m})$. Since we defined $\mathcal{D}$ to agree with $f_{\mathcal{D}}$ on the components that gain control, the finite play π that is the outcome of the game when Player 1 plays $f_{\mathcal{D}}$ and Player 2 chooses the exit states $e_{i_1}, \ldots, e_{i_{m-1}}$ reaches the Player 2 vertex $v = \langle c_{i_m}^0, T\rangle \in V_2$, for some $T \in 2^S$. We claim that $v \notin \alpha$. Indeed, the definition of E implies that there is a vertex $s \in T$ such that $\delta_{\mathcal{S}}^*(w_1 \cdot \ldots \cdot w_{m-1}) = s$. Since $w \notin L(\mathcal{S})$, we have $w_m \notin L(\mathcal{S}^s)$ and $w_m \in L(\mathcal{B}_{i_m})$. Thus, π is a winning play for Player 2, contradicting our assumption that $f_{\mathcal{D}}$ is a winning strategy, and we are done. □

We continue to study the lower bound.

Theorem 3.3 *The open-design problem is EXPTIME-hard.*

Proof: We describe a reduction from the problem of deciding whether Player 1 has a winning strategy in a *partial-information safety game*, known to be EXPTIME-complete [10].

Partial-information games (PI games, for short) are a variant of the *full-information* games (FI games, for short) defined above in which Player 1 has imperfect information [29]. The vertices, which we refer to as *locations*, denoted L, are partitioned into *observations*, denoted $\mathcal{O}$. Player 1 is unaware of the location on which the token is placed and is only aware of the observation it is in. Accordingly, In his turn, Player 1 cannot select the next location to move the token to. Instead, the edges in the game are labeled with actions, denoted Γ. In each round, Player 1 selects an action and Player 2 resolves nondeterminism and chooses the next location the token moves to. Initially, the token is placed on $l^0 \in L$. The set of labeled edges in

the game is $\Lambda \subseteq L \times \Gamma \times L$, and the safety objective α is given with respect to the observations, thus $\alpha \subseteq \mathcal{O}$.

Formally, a PI game is played on an arena $\langle \Gamma, L, \mathcal{O}, l_0, \Upsilon, \alpha \rangle$, where Γ is a set of actions, L is a set of locations, $\mathcal{O} \subseteq 2^L$ is a set of observations that form a partition of L (that is, $\mathcal{O} = \{L_1, \ldots, L_k\}$ where for all $i \neq j \in [k]$, we have that $L_i \cap L_j = \varnothing$, and $\bigcup_{i \in [k]} L_i = L$), $l_0 \in L$ is an initial location, $\Upsilon \subseteq L \times \Gamma \times L$ are labeled edges, and α is a winning condition defined with respect to $\mathcal{O}$. In particular, in safety games $\alpha \subseteq \mathcal{O}$ are safe observations for Player 1. We require that for every location $l \in L$ and action $a \in \Gamma$, there is a location $l' \in L$ such that $\langle l, a, l' \rangle \in \Upsilon$.

The game proceeds similarly to FI games. At the beginning of the game, a token is placed on the initial location l_0. Assume the token is on a location $l \in L$. Player 1 moves first and selects an action $a \in \Gamma$. Player 2 resolves non-determinism and selects a location $l' \in L$ such that $\langle l, a, l' \rangle \in \Upsilon$. Since Player 1 only observes the member in $\mathcal{O}$ in which l is, a strategy for Player 1 is a function $f_1 : (\mathcal{O} \cdot \Gamma)^* \cdot \mathcal{O} \to \Gamma$. Since Player 2 has complete information, a strategy for him is a function $f_2 : L^+ \cdot \Gamma \to L$ such that for a play $\pi = l_0, \ldots, l_m$ and an action $a \in \Gamma$, we have $\langle l_m, a, f_2(\pi, a) \rangle \in \Upsilon$. The definition of winning strategies are as in the full-information setting.

Consider a PI safety game $\mathcal{G} = \langle \Gamma, L, \mathcal{O}, l^0, \Upsilon, \alpha \rangle$. We construct a library $\mathcal{L}$ and a DFA $\mathcal{S}$ such that there is a correct design for $\mathcal{S}$ using the components of $\mathcal{L}$ iff Player 1 wins $\mathcal{G}$. Recall that a design reads an exit state and outputs the component that gets control next. Also, a Player 1 strategy in $\mathcal{G}$ reads observations and outputs actions. Accordingly, the library $\mathcal{L}$ consists of box-transducers $\mathcal{B}_a$, one for every action $a \in \Gamma$. The exit states of the components correspond to the observations in $\mathcal{O}$. That is, when a component exits through an observation $L_i \in \mathcal{O}$, the design decides which component $\mathcal{B}_a \in \mathcal{L}$ gains control, which corresponds to a Player 1 strategy that chooses the action $a \in \Gamma$ from the observation L_i.

Next, we define words to correspond to Player 2 strategies. Recall that Player 2 resolves nondeterminism in $\mathcal{G}$. That is, when the token is in location $l \in L$ and Player 1 selects an action $a \in \Gamma$, Player 2 selects an edge $\langle l, a, l' \rangle \in \Upsilon$ and the token moves to the location l'. Accordingly, $\Sigma_I = \Upsilon$, thus a word in Σ_I^* is a sequence of edges. We define the specification $\mathcal{S}$ so that a word witnesses the incorrectness of the design only if it corresponds to a *loosing path* in $\mathcal{G}$, where $\langle l_0, a_1, l'_1 \rangle, \langle l_1, a_2, l'_2 \rangle, \langle l_2, a_2, l'_3 \rangle, \ldots, \langle l_{m-1}, a_{m-1}, l'_m \rangle \in \Upsilon^*$ is a loosing path if for all $1 \leqslant i \leqslant m-1$ we have that $l'_i = l_i$ and $l'_m \notin \alpha$.

Finally, we define the components so that a correct design corresponds to a winning Player 1 strategy. For $a \in \Gamma$, the component $\mathcal{B}_a \in \mathcal{L}$ can *process* every edge that is labeled with a. When reading such an edge it relinquishes control, and when reading an edge that is labeled with $b \neq a$, the component enters a sink which is intuitively a rejecting sink. Thus, in order to avoid processing a word $w = t_1 \ldots t_m$

that corresponds to a loosing path, a design must assign control to some component $\mathcal{B}_a$ with $a \neq a_i$, after reading the input prefix $t_1 \dots t_{i-1}$, for $1 \leqslant i \leqslant m$.

Formally, for $a \in \Gamma$, we define the box-transducer $\mathcal{B}_a = \langle \Upsilon, \{\top, \bot\}, \{c_a^0, c_a^{rej}\} \cup \mathcal{O}, c_a^0, \delta_a, \nu_a, \mathcal{O} \rangle$, where $\nu_a(c_a^0) = \top$, $\nu_a(c_a^{rej}) = \bot$, and δ_a is defined as follows. Consider an edge $\langle l, a, l' \rangle \in \Upsilon$. We define $\delta_a(c_a^0, \langle l, a, l' \rangle) = P$, for the observation P with $l' \in P$. For an edge $\langle l, b, l' \rangle \in \Upsilon$, with $b \neq a$, we define $\delta_a(c_a^0, \langle l, b, l' \rangle) = c_a^{rej}$. The state c_a^{rej} is a sink, thus $\delta_a(c_a^{rej}, \sigma) = c_a^{rej}$ for all $\sigma \in \Sigma_I$. Note that the component $\mathcal{B}_a$ relinquishes control and outputs $\top$ when it reads a transition labeled a. Otherwise, it gets stuck in the rejecting sink.

The specification is given by the DFA $\mathcal{S} = \langle \Sigma_I \times \Sigma_O, S, \delta_{\mathcal{S}}, l^0, F_{\mathcal{S}} \rangle$, where $S = L \cup \{s_{acc}\}$, the accepting states $F_{\mathcal{S}}$ are the states that do not belong to observations in α, thus $F_{\mathcal{S}} = S \setminus \bigcup_{P \in \alpha} P$, and we describe $\delta_{\mathcal{S}}$ in the following. Consider a location $l \in L$. For every edge $t = \langle l, a, l' \rangle \in \Upsilon$ we define $\delta_{\mathcal{S}}(l, \langle t, \top \rangle) = l'$. For every other letter $\sigma \in \Sigma_I \times \Sigma_O$, we define $\delta_{\mathcal{S}}(l, \sigma) = s_{acc}$.

Note that a word $w \in (\Sigma_I \times \Sigma_O)^*$ is not in $L(\mathcal{S})$ if it is of the form $w = w_1 \oplus w_2 \in (\Sigma_I \times \Sigma_O)^*$ for $w_2 \in \top^*$ and $w_1 \in \Sigma_I^*$ corresponds to a *loosing path* in $\mathcal{G}$. That is, $w = t_1 \dots t_m$, where $t_1, \dots, t_m \in \Upsilon$, $t_1 = \langle l^0, a_1, l_1 \rangle$, for $1 \leqslant i < m$, the target location of the transition t_i is the source location of t_{i+1}, and assuming $t_m = \langle t_{m-1}, a_m, t_m \rangle$, we have $t_m \notin \alpha$. Recall that a component $\mathcal{B}_a \in \mathcal{L}$ relinquishes control after reading an edge that the action $a \in \Gamma$ participates in. When reading an edge that a does not participate in, $\mathcal{B}_a$ enters a sink and outputs $\bot$. Thus, in order to avoid processing the word w as in the above, a design must assign control to some component $\mathcal{B}_a$ with $a \neq a_i$, after reading the input prefix $t_1 \dots t_{i-1}$, for $1 \leqslant i \leqslant m$.

We claim that Player 1 wins $\mathcal{G}$ iff there is a correct design $\mathcal{D}$ for $\mathcal{S}$ using the components in $\mathcal{L}$. For the first direction, consider a Player 1 winning strategy f_1. We construct a design $\mathcal{D}$ inductively. Let $a_1 = f_1(P_0)$, where $P_0 \in \mathcal{O}$ is such that $l^0 \in P_0$. The first component to gain control is $\mathcal{B}_{a_1}$. Consider a run on some word, and let $\mathcal{B}_{a_1}, P_1, \mathcal{B}_{a_2}, P_2, \dots \mathcal{B}_{a_m} P_m$ be the sequence of components and corresponding exit states that the run visits. Recall that $P_1, \dots, P_m$ are observations in $\mathcal{O}$. Let $\pi = P_0, a_1, P_1, \dots, P_m$, where $l^0 \in P_0$. Note that π might not correspond to a path in $\mathcal{G}$, in which case $\mathcal{D}$ assigns control to an arbitrary component. Otherwise, assuming $f_1(\pi) = a_{m+1}$, $\mathcal{D}$ assigns control to $\mathcal{B}_{a_{m+1}}$.

We claim that $\mathcal{D}$ is a correct design. Assume towards contradiction that there is a word $w \in L(\mathcal{T}_{\mathcal{L},\mathcal{D}}) \setminus L(\mathcal{S})$. Since $w \notin L(\mathcal{S})$, its projection on Σ_O is in $\top^*$ and the projection of w on Σ_I is a loosing path $t_1, \dots, t_m$. For $1 \leqslant i \leqslant m$, let $t_i = \langle l_{i-1}, a_i, l_i \rangle$, where $l_0 = l^0$. Let r be the run of $\mathcal{T}_{\mathcal{L},\mathcal{D}}$ on w. Since the projection of w on Σ_O is a sequence of $\top$'s, the sequence of components and exit states that r passes is $\mathcal{B}_{a_1}, P_1, \dots, \mathcal{B}_{a_m}, P_m$. Indeed, if for $1 \leqslant i \leqslant m$ and $b \neq a_i$, the component $\mathcal{B}_b \in \mathcal{L}$ is in control when a_i is read, the component outputs $\bot$. Let π be the path

that corresponds to $t_1, \ldots, t_m$, thus $\pi = l_0, \ldots, l_m$, where $l_0 = l^0$. Consider the Player 2 strategy f_2 that selects the edges $t_1, \ldots, t_m$. Since by our definition of the components in $\mathcal{L}$, for $1 \leqslant i \leqslant m$, we have $l_i \in P_i$, we can prove by induction that π is a prefix of $out(f_1, f_2)$. Since $l_m \notin \alpha$, the strategy f_2 is winning against f_1, which is a contradiction to the fact that it is a winning strategy, and we are done.

For the second direction, consider a correct design $\mathcal{D}$. We define a Player 1 strategy $f_{\mathcal{D}}$ inductively as follows. We abuse notation and refer to the labeling function ν of $\mathcal{D}$ as a function from its states D to $\mathcal{L}$. Assume the first component that $\mathcal{D}$ assigns control to is $\mathcal{B}_{a_1} \in \mathcal{L}$. For the observation P_0 such that $l^0 \in P_0$, we define $f_1(P_0) = a_1$. Consider a play $\pi = l_0, l_1, \ldots, l_m$, where $l_0 = l^0$. Let $\tau = P_0, a_1, P_1, \ldots, a_m, P_m$ such that for $1 \leqslant i \leqslant m$ we have $l_i \in P_i$. Let $a_1, \ldots, a_m \in \Gamma$ such that, for $1 \leqslant i \leqslant m$ we have $\langle l_{i-1}, a_i, l_i \rangle \in \Upsilon$. Let $\delta^*_{\mathcal{D}}(P_1 \ldots P_m) = d$ and let $\nu(d) = \mathcal{B}_{a_m} \in \mathcal{L}$. We define $f_{\mathcal{D}}(\tau) = a_m$.

We claim that $f_{\mathcal{D}}$ is a winning strategy. Assume towards contradiction that there is a Player 2 strategy f_2 and a finite prefix π of $out(f_{\mathcal{D}}, f_2)$ that is winning for Player 2. Let $\tau = t_1, \ldots, t_m$ be the edges that π traverses. Consider the run of $\mathcal{T}_{\mathcal{L},\mathcal{D}}$ on τ. We can prove by induction that by our definition of $f_{\mathcal{D}}$, for $1 \leqslant i \leqslant m$, when the letter $t_i = \langle l_{i-1}, a_i, l_i \rangle$ is read, the component $\mathcal{B}_{a_i}$ is in control. Thus, the output of $\mathcal{T}_{\mathcal{L},\mathcal{D}}$ when reading the word τ is a sequence w of $|\tau|$ $\top$'s. Since τ corresponds to a loosing path, we have $\tau \oplus w \in L(\mathcal{T}_{\mathcal{L},\mathcal{D}}) \backslash L(\mathcal{S})$, which is a contradiction to the correctness of $\mathcal{D}$, and we are done. □

4 Libraries with Costs

Given a library and a specification, there are possibly many, in fact infinitely many, designs that are solutions to the design problem. As a trivial example, assume that $L(\mathcal{S}) = a^*$ and that the library contains a component $\mathcal{B}$ that traverses the letter a (that is, $\mathcal{B}$ consists of an accepting initial state that has an a-transition to an exit state). An optimal design for $\mathcal{S}$ uses $\mathcal{B}$ once: it has a single state with a self loop in which $\mathcal{B}$ is called. Other designs can use $\mathcal{B}$ arbitrarily many times. When we wrote "optimal" above, we assumed that the smaller the design is, the better it is. In this section we would like to formalize the notion of optimality and add to the composition picture different costs that components in the libraries may have.

In order to capture a wide set of scenarios in practice, we associate with each component in $\mathcal{L}$ two costs: a *construction cost* and a *quality cost*. The costs are given by the functions *c-cost*, *q-cost* : $\mathcal{L} \to \mathbb{R}^+ \cup \{0\}$, respectively. The construction cost of a component is the cost of adding it to the library. Thus, a design that uses a component pays its construction cost once, and (as would be the case in Section 6),

when several designs use a component, they share its construction cost. The quality cost measures the performance of the component, and involves, for example, its number of states. Accordingly, a design pays the quality cost of a component every time it uses it, and the fact the component is used by other designs is not important.[5]

Formally, consider a library $\mathcal{L} = \{\mathcal{B}_1, \ldots, \mathcal{B}_n\}$ and a design $\mathcal{D} = \langle [n], E, D, d^0, \delta, \nu \rangle$. The number of times $\mathcal{D}$ uses a component B_i is $nused(\mathcal{D}, \mathcal{B}_i) = |\{d \in D : \nu(d) = i\}|$. The set of components that are used in $\mathcal{D}$, is $used(\mathcal{D}) = \{\mathcal{B}_i : nused(\mathcal{D}, \mathcal{B}_i) \geqslant 1\}$. The cost of a design is then $cost(\mathcal{D}) = \sum_{\mathcal{B} \in used(\mathcal{D})} c\text{-}cost(\mathcal{B}) + nused(\mathcal{D}, \mathcal{B}) \cdot q\text{-}cost(\mathcal{B})$.

We state the problem of finding the cheapest design as a decision problem. For a specification DFA $\mathcal{S}$, a library $\mathcal{L}$, and a threshold μ, we say that an input $\langle \mathcal{S}, \mathcal{L}, \mu \rangle$ is in BCD (standing for "bounded cost design") iff there exists a correct design $\mathcal{D}$ such that $cost(\mathcal{D}) \leqslant \mu$. In this section we study the BCD problem in a setting with a single user. Thus, decisions are independent of other users of the library, which, recall, may influence the construction cost.

In section 3, we reduced the design problem to the problem of the solution of a safety game. In particular, we showed how a winning strategy in the game induces a correct design. Note that while we know that safety games admits memoryless strategies, there is no guarantee that memoryless strategies are guaranteed to lead to optimal designs. We first study this point and show that, surprisingly, while memoryless strategies are sufficient for obtaining an optimal design in the closed setting, this is not the case in the open setting. The source of the difference is the fact that the language of a design in the open setting may be strictly contained in the language of the specification. The approximation may enable the user to generate a design that is more complex and is still cheaper in terms of cost. This is related to the fact that over approximating the language of a DFA may result in exponentially bigger DFAs [6]. We are still able to bound the size of the cheapest design by the size of the game.

4.1 On the optimality and non-optimality of memoryless strategies

Consider a closed library $\mathcal{L}$ and a DFA $\mathcal{S}$. Recall that a correct design for $\mathcal{S}$ from components in $\mathcal{L}$ is induced by a winning strategy of Player 1 in the game $\mathcal{G}_{\mathcal{L},\mathcal{S}}$ (see Theorem 3.1). If the winning strategy is not memoryless, we can trim it to

[5] One might consider a different quality-cost model, which takes into an account the cost of *computations*. The cost of a design is then the maximal or expected cost of its computations. Such a cost model is appropriate for measures like the running time or other complexity measures. We take here a global approach, which is appropriate for measures like the number of states or security level.

a memoryless one and obtain a design whose state space is a subset of the design induced by the original strategy. Since the design has no flexibility with respect to the language of $\mathcal{S}$, we cannot do better. Hence the following lemma.

Lemma 4.1 *Consider a closed library $\mathcal{L}$ and a DFA $\mathcal{S}$. For every $\mu \geqslant 0$, if there is a correct design $\mathcal{D}$ with $cost(\mathcal{D}) \leqslant \mu$, then there is a correct design $\mathcal{D}'$ induced by a memoryless strategy for Player 1 in $\mathcal{G}_{\mathcal{L},\mathcal{S}}$ such that $cost(\mathcal{D}') \leqslant \mu$.*

Proof: Consider a correct design $\mathcal{D}$ with $cost(\mathcal{D}) \leqslant \mu$. Let $f_{\mathcal{D}}$ be a winning Player 1 strategy in the safety game $\mathcal{G}_{\mathcal{L},\mathcal{S}}$ as constructed in the proof of Theorem 3.1. Note that while we know that safety games admit memoryless strategies, there is no guarantee that memoryless strategies are guaranteed to lead to minimal-cost designs. Thus, $f_{\mathcal{D}}$ need not be a memoryless strategy. We construct a Player 1 memoryless strategy $f'_{\mathcal{D}}$ by *trimming* $f_{\mathcal{D}}$ in every *memoryfull* vertex. Then, we construct a design $\mathcal{D}'$ from $f'_{\mathcal{D}}$ and show that it costs no more than $\mathcal{D}$.

In order to prove the claim formally, we need a few definitions. Consider a Player 1 vertex $v = \langle e, s\rangle \in V_1$. We define $adj(f_{\mathcal{D}}, v) \in V$ to be the vertices $f_{\mathcal{D}}$ continues to from v. That is, $u \in adj(f_{\mathcal{D}}, v)$ if there is a play π_u that ends in v and is an outcome of the game when Player 1 plays $f_{\mathcal{D}}$ and Player 2 plays some strategy, and $f_{\mathcal{D}}(\pi_u) = u$. We refer to π_u as a *witness* play for $u \in adj(f_{\mathcal{D}}, v)$. Note that there can be many witness plays for a vertex in $adj(f_{\mathcal{D}}, v)$. Further note that if $adj(f_{\mathcal{D}}, v)$ is a singleton, then v is a *memoryless* vertex with respect to $f_{\mathcal{D}}$. Consider a vertex $u = \langle c_i^0, s\rangle \in adj(f_{\mathcal{D}}, v)$. Intuitively, since we define $f_{\mathcal{D}}$ to assign control as $\mathcal{D}$, and since $f_{\mathcal{D}}$ continues to u after π_u, there must be a state $d_u \in D$ with $\nu(d_u) = i$. Thus, d_u is a *witness* state for the fact that $f_{\mathcal{D}}(\pi_u) = u$. Again, there can be several witness states for a vertex in $adj(f_{\mathcal{D}}, v)$. Formally, let π_u be a witness play for u and let $w_u \in \Sigma^*$ be the word that is induced by the Player 2 choices in π_u. That is, $\delta^*_{\mathcal{S}}(w_u) = s$, where recall that $v = \langle e, s\rangle$. Moreover, let $e_1, \ldots, e_m$ be the exit states that are traversed in π_u, and let $d_u = \delta^*_{\mathcal{D}}(e_1 \ldots e_m)$. Since $f_{\mathcal{D}}$ assigns control to the same components as $\mathcal{D}$, the run of $\mathcal{A}_{\mathcal{L},\mathcal{D}}$ on w_u ends in the state $\langle c_i^0, d_u\rangle$. We refer to w_u and d_u as a *witness* word and state for u, respectively.

We define $f'_{\mathcal{D}}$ as follows. For every $v \in V_1$, we choose a vertex $u \in adj(f_{\mathcal{D}}, v)$ arbitrarily and define $f'_{\mathcal{D}}(v) = u$. It is not hard to prove that $f'_{\mathcal{D}}$ is a winning strategy. Let $\mathcal{D}'$ be the design that corresponds to $f'_{\mathcal{D}}$ and is constructed as in the proof of Theorem 3.1. Since $f'_{\mathcal{D}}$ is winning, $\mathcal{D}'$ is a correct design. We claim that $cost(\mathcal{D}') \leqslant cost(\mathcal{D})$. Recall that the states of $\mathcal{D}'$ are the Player 2 vertices in $\mathcal{G}_{\mathcal{L},\mathcal{S}}$. Further recall that in the construction of $\mathcal{D}'$ we skip Player 1 vertices and proceed according to $f'_{\mathcal{D}}$. Consider a reachable Player 2 vertex $\langle c_i^0, s\rangle$. We denote by ν' the labeling function of $\mathcal{D}'$. Recall that c_i^0 is the initial state of the component $\mathcal{B}_i \in \mathcal{L}$ and $\nu'(\langle c_i^0, s\rangle) = i$. First, we claim that if a component is used in $\mathcal{D}'$ then it is used

in $\mathcal{D}$, thus $used(\mathcal{D}') \subseteq used(\mathcal{D})$. Indeed, if $\mathcal{B}_i \in used(\mathcal{D}')$, there is a state in $\mathcal{D}'$ that corresponds to a reachable Player 2 vertex $v = \langle c_i^0, s \rangle$, for some $s \in S$. Let s_v be a witness state of v. Since the labeling of s_v is $\nu(s_v) = i$, we have $\mathcal{B}_i \in used(\mathcal{D})$. We conclude that the sum of construction costs that is incurred by $\mathcal{D}'$ is at most that of $\mathcal{D}$.

We prove that the sum of quality costs incurred by $\mathcal{D}'$ is at most that of $\mathcal{D}$. We prove that for every $\mathcal{B}_i \in used(\mathcal{D}')$ we have $nused(\mathcal{D}', \mathcal{B}_i) \leqslant nused(\mathcal{D}, \mathcal{B}_i)$. For a reachable vertex $v \in V_2$ let $d_v \in D$ be an arbitrary choice of witness state of v. Let w_v be the corresponding witness word. We show that the mapping from a reachable vertex $v \in V_2$ to $d_v \in D$ is a one-to-one mapping. Consider $v, u \in V_2$ with $d_v = d_u$. Let $d = d_v = d_u$. Let $i = \nu(d)$. Since the component-state of v and u is the initial state of $\mathcal{B}_i$, we have $v = \langle c_i^0, s \rangle$ and $u = \langle c_i^0, s' \rangle$. To conclude the proof, we show that $s = s'$, thus $v = u$. We claim that $L(\mathcal{S}^s) = L(\mathcal{S}^{s'})$ and since $\mathcal{S}$ is a minimal DFA, it would follow that $s = s'$. Recall that w_v and w_u are the witness words for v and u, respectively, thus $\delta_{\mathcal{S}}^*(w_v) = s$ and $\delta_{\mathcal{S}}^*(w_{v'}) = s'$. Moreover, since w_v and w_u are the witness words that correspond to the witness states $d_v = d_u = d$, we have $\delta_{\mathcal{L},\mathcal{D}}^*(w_v) = \delta_{\mathcal{L},\mathcal{D}}^*(w_u) = \langle c_i^0, d \rangle$. Consider a word $x \in \Sigma^*$. If $x \in L(\mathcal{S}^s)$, then since $\delta_{\mathcal{S}}^*(w_v) = s$, we have $w_v \cdot x \in L(\mathcal{S})$. Since $\mathcal{D}$ is a correct design, $w_v \cdot x \in L(\mathcal{A}_{\mathcal{L},\mathcal{D}})$. Thus, $\delta_{\mathcal{L},\mathcal{D}}^*(w_v \cdot x)$ is an accepting state. Since $\delta_{\mathcal{L},\mathcal{D}}^*(w_v) = \delta_{\mathcal{L},\mathcal{D}}^*(w_u)$, we have $w_u \cdot x \in L(\mathcal{A}_{\mathcal{L},\mathcal{D}})$ and in turn $w_u \cdot x \in L(\mathcal{S})$. Since $\delta_{\mathcal{S}}^*(w_u) = s'$, we have $u \in L(\mathcal{S}^{s'})$. The other direction is symmetric, and we are done. □

While Lemma 4.1 seems intuitive, it does not hold in the setting of open systems. There, a design has the freedom to generate a language that is a subset of $L(\mathcal{S})$, as long as it stays receptive. This flexibility allows the design to generate a language that need not be related to the structure of the game $\mathcal{G}_{\mathcal{L},\mathcal{S}}$, which may significantly reduce its cost. Formally, we have the following.

Lemma 4.2 *There is an open library $\mathcal{L}$ and a family of DFAs $\mathcal{S}_n$ such that $\mathcal{S}_n$ has a correct design $\mathcal{D}_n$ with cost 1 but every correct design for $\mathcal{S}_n$ that is induced by a memoryless strategy for Player 1 in $\mathcal{G}_{\mathcal{L},\mathcal{S}_n}$ has cost n.*

Proof: We define $\mathcal{S}_n = \langle \Sigma_I \times \Sigma_O, S_n, \delta_{\mathcal{S}_n}, s_0^0, F_{\mathcal{S}_n} \rangle$, where $\Sigma_I = \{\tilde{0}, \tilde{1}, \#\}$, $\Sigma_O = \{0, 1, \_\}$, and S_n, $\delta_{\mathcal{S}_n}$ and $F_{\mathcal{S}_n}$ are as follows. Essentially, after reading a prefix of i #'s, for $1 \leqslant i \leqslant n$, the design should arrange its outputs so that the i-th and $(n+i)$-th letters agree (they are $\tilde{0}$ and 0, or are $\tilde{1}$ and 1). One method to do it is to count the number of #'s and then check the corresponding indices. Another method is to keep track of all the first n input letters and make sure that they repeat. The key idea is that while in the second method we strengthen the specification (agreement is checked with respect to all i's, ignoring the length of the #-prefix), it is still receptive, which is exactly the flexibility that the open setting allows. We define

$\mathcal{S}_n$ and the library $\mathcal{L}$ so that the structure of $\mathcal{G}_{\mathcal{L},\mathcal{S}_n}$ supports the first method, but counting each # has a cost of 1. Consequently, a memoryless strategy of Player 1 in $\mathcal{G}_{\mathcal{L},\mathcal{S}_n}$ induces a design that counts, and is therefore of cost n, whereas an optimal design follows the second method, and since it does not count the number of #'s, its cost is only 1.

We can now describe the DFA $\mathcal{S}_n$ in more detail. It consists of n chains, sharing an accepting sink s_{acc} and a rejecting sink s_{rej}. For $0 \leqslant i \leqslant n-1$, we describe the i-th chain, which is depicted in Figure 1. When describing $\delta_{\mathcal{S}_n}$, for ease of presentation, we sometimes omit the letter in Σ_I or Σ_O and we mean that every letter in the respective alphabet is allowed. For $0 \leqslant i < n-1$, we define $\delta_{\mathcal{S}_n}(s_0^i, \#) = s_0^{i+1}$ and $\delta_{\mathcal{S}_n}(s_0^n, \#) = s_0^n$. Note that words of the form $\#^i a_1 \tilde{1} a_2 b 0$ or $\#^i a_1 \tilde{0} a_2 b 1$ are not in $L(\mathcal{S}_n)$, where if $0 \leqslant i \leqslant n-1$, then $a_1 \in (\tilde{0}+\tilde{1})^i$, $a_2 \in (\tilde{0}+\tilde{1})^{n-i-1}$, and $b \in (0+1)^i$, and if $i > n-1$ then the lengths of a_1, a_2, and b are $n-1$, 0, and $n-1$, respectively. We require that after reading a word in $\#^*(\tilde{0}+\tilde{1})^n$ there is an output of n letters in $\{0,1\}$. Thus, for $n \leqslant j \leqslant n+i+1$, we define $\delta_{\mathcal{S}_n}(s_j^{i,0}, \_) = \delta_{\mathcal{S}_n}(s_j^{i,1}, \_) = s_{rej}$. Also, $\mathcal{S}_n$ accepts every word that has a # after the initial prefix of # letters. Thus, for $1 \leqslant j \leqslant i$, we define $\delta_{\mathcal{S}_n}(s_j^i, \#) = s_{acc}$, and for $i+1 \leqslant j \leqslant n+i+1$ and $t \in \{0,1\}$ we define $\delta_{\mathcal{S}_n}(s_j^{i,t}, \#) = s_{acc}$.

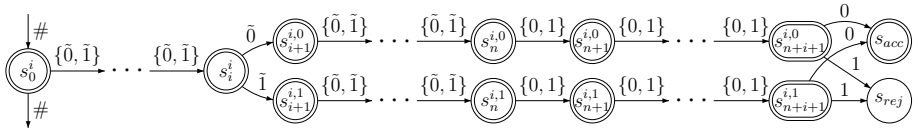


Figure 1: A description of the i-th chain of the specification $\mathcal{S}_n$.

The library $\mathcal{L}$ is depicted in Figure 2. The quality and construction costs of all the components is 0, except for $\mathcal{B}_1$ which has $q\text{-}cost(\mathcal{B}_1) = 1$ and $c\text{-}cost(\mathcal{B}_1) = 0$.

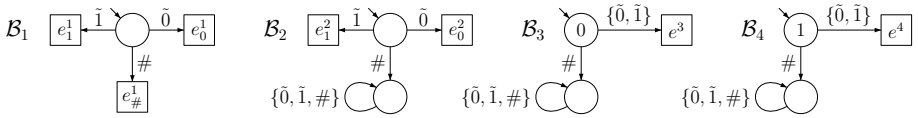


Figure 2: The library $\mathcal{L}$. Exit states are square nodes and the output of a state is written in the node.

We First claim that every correct design must cost at least 1. Indeed, such a design must use $\mathcal{B}_1$ at least once. Otherwise, the output sequence for the input word $\#\tilde{0}^{n+1}$ is a sequence in $\_^*$. Recall that we require that after reading a word in $\#^*(\tilde{0}+\tilde{1})^n$ there is an output of n letters in $\{0,1\}$, thus such a design is incorrect.

We describe a correct design $\mathcal{D}_n$ with $cost(\mathcal{D}_n) = 1$. Intuitively, as explained above, $\mathcal{D}_n$ does not track the number of #'s that are read and can thus use $\mathcal{B}_1$ only once. Instead, $\mathcal{D}_n$ keeps track of all of the n input letters in $\{\tilde{0},\tilde{1}\}$ that follow the

sequence in $\#^*$. Thus, after $\tilde{0}$ or $\tilde{1}$ is read, $\mathcal{B}_2$ gains control for n consecutive times. Then, assuming the word $u \in \{\tilde{0}, \tilde{1}\}^n$ is read in the first phase, control is alternated between $\mathcal{B}_3$ and $\mathcal{B}_4$ so that the word $v \in \{0,1\}^n$ is output, where u and v represent the same vector. Note that $\mathcal{D}_n$ uses the components $\mathcal{B}_2$, $\mathcal{B}_3$, and $\mathcal{B}_4$ exponentially many times in n.

Formally, we define $\mathcal{D}_n = \langle \Sigma_I, \Sigma_O, D_n, \delta_{\mathcal{D}_n}, d^0, \nu \rangle$. We define $\nu(d^0) = 1$ and $\delta_{\mathcal{D}_n}(d^0, e^1_{\#}) = d^0$. When $\mathcal{B}_1$ relinquishes control through e^1_1 or e^1_0 control is passed to $\mathcal{B}_2$ for n consecutive times. During this phase, $\mathcal{D}_n$ *remembers* the vector in $\{\tilde{0}, \tilde{1}\}^n$ that is read, thus $\mathcal{D}_n$ has $2^{\Omega(n)}$ states. Next, the components $\mathcal{B}_3$ and $\mathcal{B}_4$ alternate control for n times as we describe in the following. Consider an input word $w = \#^i \cdot a_1 \dots a_n b$, where $a_1, \dots, a_n \in \{\tilde{0}, \tilde{1}\}^n$ and $b \in \{\tilde{0}, \tilde{1}\}^m$ for $0 \leqslant m < n-1$. After reading w either $\mathcal{B}_3$ or $\mathcal{B}_4$ relinquish control. If $a_{m+1} = 0$, then the next component to gain control is $\mathcal{B}_3$ and otherwise it is $\mathcal{B}_4$. After n alternations of control $\mathcal{B}_4$ is assigned control indefinitely.

Since the initial state of $\mathcal{D}_n$ is the only state that is labeled with $\mathcal{B}_1$, we have $nused(\mathcal{B}_1) = 1$, thus $cost(\mathcal{D}_n) = 1$. We claim that $\mathcal{D}_n$ is a correct design. Indeed, note that $L(\mathcal{T}_{\mathcal{L},\mathcal{D}_n})$ consists of two types of words. The first type are prefixes of words in $\#^* ab0^*$, for $a \in \{\tilde{0}, \tilde{1}\}^n$ and $b \in \{0,1\}^n$ that represent the same vector, thus for $1 \leqslant i \leqslant n$, if $a_i = \tilde{0}$, then $b_i = 0$ and if $a_i = \tilde{1}$, then $b_i = 1$. The second type are words that have a prefix in $\#^*(\tilde{0} + \tilde{1})^+\#$. Clearly, $L(\mathcal{T}_{\mathcal{L},\mathcal{D}_n}) \subseteq L(\mathcal{S}_n)$, and we are done. □

4.2 Solving the BCD problem

Theorem 4.3 *The BCD problem is NP-complete for closed designs.*

Proof: Consider an input $\langle \mathcal{S}, \mathcal{L}, \mu \rangle$ to the BCD problem. By Lemma 4.1, we can restrict the search for a correct design $\mathcal{D}$ with $cost(\mathcal{D}) \leqslant \mu$ to these induced by a memoryless strategy for Player 1 in $\mathcal{G}_{\mathcal{L},\mathcal{S}}$. By the definition of the game $\mathcal{G}_{\mathcal{L},\mathcal{S}}$, such a design has at most $|\mathcal{C}_0 \times S|$ states. Since checking if a design is correct and calculating its cost can be done in polynomial time, membership in NP follows.

For the lower bound, we describe a reduction from SET-COVER. Consider an input $\langle U, T, \mu \rangle$ to SET-COVER, where $U = \{1, \dots, l\}$ is a universe, $T = \{T_1, \dots, T_m\}$ is a set of subsets over U, i.e., $T_i \subseteq U$, for $1 \leqslant i \leqslant m$, and $\mu \in \mathbb{N}$. We construct a closed-library $\mathcal{L}$ and a DFA $\mathcal{S}$ such that there is a design $\mathcal{D}$ that costs at most μ iff there is a set-cover of size at most μ.

We construct the DFA $\mathcal{S}$ over the alphabet U as a chain of $l+1$ states that accepts only the word $12 \dots l$. There are $m+2$ components in $\mathcal{L}$. For $1 \leqslant i \leqslant m$, the component $\mathcal{B}_i$ corresponds to the set T_i. Its initial state is c^0_i and it has an exit

state e^i_j for every element $j \in T_i$. For $j \in T_i$ there is a transition $\delta_i(c^0_i, j) = e^i_j$. The last two components are $\mathcal{B}_{acc}$ and $\mathcal{B}_{rej}$. For consistency we also refer to these components as $\mathcal{B}_{m+1}$ and $\mathcal{B}_{m+2}$, respectively. They consist of a single initial non-exit state that is accepting in $\mathcal{B}_{acc}$ and rejecting in $\mathcal{B}_{rej}$. For $\mathcal{B}_i \in \mathcal{L}$, we define $c\text{-}cost(\mathcal{B}_i) = 1$ and we define $c\text{-}cost(\mathcal{B}_{acc}) = c\text{-}cost(\mathcal{B}_{rej}) = 0$. Finally, the quality cost of the components is 0, thus we define $q\text{-}cost \equiv 0$.

We claim that there is a set cover of size at most μ iff there is a design of cost at most μ. For the first direction, consider a set cover $T' \subseteq T$ with $|T'| \leqslant \mu$. For $j \in U$, let $cover(j)$ be the index such that $j \in T_{cover(j)}$ and $T_{cover(j)} \in T'$. We construct a design $\mathcal{D}$ with $cost(\mathcal{D}) \leqslant \mu$. We define $\mathcal{D}$ so that when $\mathcal{A}_{\mathcal{L},\mathcal{D}}$ reads the word $1 \dots l$, the component that gets j-th control, for $1 \leqslant j \leqslant l$, is $\mathcal{B}_{cover(j)}$. If the j-th input letter is $j' \neq j$, then the next component to gain control is $\mathcal{B}_{rej}$. Finally, the component that gains $l+1$-th control is $\mathcal{B}_{acc}$.

Formally, $\mathcal{D} = \langle \mathcal{E}, [m+2], D, \delta_{\mathcal{D}}, d_1, \nu \rangle$, where $\{d_1, \dots, d_l, d_{acc}, d_{rej}\}$ and we define ν and $\delta_{\mathcal{D}}$ as follows. For $1 \leqslant j \leqslant l$, we define $\nu(d_j) = cover(j)$. For $1 \leqslant i \leqslant m$, recall that the component $\mathcal{B}_i$ has an exit state e^i_j for every $j \in T_i$. For $j < l$ and $1 \leqslant i \leqslant m$, we define $\delta_{\mathcal{D}}(d_j, e^i_j) = d_{j+1}$ and we define $\delta_{\mathcal{D}}(d_l, e^i_l) = d_{acc}$. For $j \neq i$, we define $\delta_{\mathcal{D}}(d_j, e^i_j) = d_{rej}$. It is not hard to see that $L(\mathcal{A}_{\mathcal{L},I}) = \{12 \dots l\}$, and thus $\mathcal{D}$ is a correct design. The components that $\mathcal{D}$ uses are the ones that correspond to the sets in T', thus $\mathcal{D}$ uses at most μ components with construction cost 1, and its cost is at most μ.

For the other direction, assume there is a correct design $\mathcal{D}$ with $cost(\mathcal{D}) \leqslant \mu$. Consider the collection $T' \subseteq T$ of sets that correspond to the components $\{\nu(d) : d \in D \text{ and } \nu(d) \notin \{\mathcal{B}_{acc}, \mathcal{B}_{rej}\}\}$. We claim that T' is a set cover with at most μ sets. Since $cost(\mathcal{D}) \leqslant \mu$, it uses at most μ components that correspond to sets in T, thus $|T'| \leqslant \mu$. We are left to show that T' is a set cover. Consider the run r of $\mathcal{A}_{\mathcal{L},\mathcal{D}}$ on the word $1 \dots l$. Since $\mathcal{D}$ is correct, r is accepting. Specifically it does not get stuck. Since the components we described above relinquish control after reading a single letter, the control is passed $l+1$ times during r. Let $\mathcal{B}_{i_1}, \dots, \mathcal{B}_{i_l}, \mathcal{B}_{i_{l+1}}$ be the sequence of components that gain control. Consider $j \in U$. We claim that j is covered in T'. Since r does not get stuck, there is a transition labeled j in the component $\mathcal{B}_{i_j}$, the j-th component to gain control in r. Thus, $j \in T_{i_j}$, the set in T that corresponds to $\mathcal{B}_{i_j}$. Since $T_{i_j} \in T'$, j is covered, and we are done. □

Remark 4.4 Note that a different attempt to reduce SET-COVER to the BCD problem would be to define the components as in the proof of Theorem 4.3 and define $\mathcal{S}$ so that it accepts one-letter words for each element in U. However, this attempt fails since, intuitively, a design does not know which letter is going to be read. Even if there is a set cover $T' \subseteq T$ and control is assigned to the component $\mathcal{B}_i$

where $T_i \in T'$, a letter $j \in U \backslash T_i$ can be read. Thus, the fact that we use a one-word specification allows a design to expect the next letter that should be read.

We turn to study the open setting, which is significantly harder than the closed one. For the upper bound, we first show that while we cannot restrict attention to designs induced by memoryless strategies, we can still bound the size of optimal designs:

Theorem 4.5 *For an open library $\mathcal{L}$ with ℓ components and a specification $\mathcal{S}$ with n states, a cheapest correct design $\mathcal{D}$ has at most $\binom{n}{n/2} \cdot \ell$ states.*

Proof: Given $\mathcal{S}$ and $\mathcal{L}$, assume towards contradiction that the cheapest smallest design $\mathcal{D}$ for $\mathcal{S}$ using the components in $\mathcal{L}$ has more than $\binom{n}{n/2} \cdot \ell$ states.

Consider a word $w \in L(\mathcal{T}_{\mathcal{L},\mathcal{D}})$. Let $\mathcal{B}_{i_1}, \ldots, \mathcal{B}_{i_m} \in \mathcal{L}$ be the components that are traversed in the run r of $\mathcal{T}_{\mathcal{L},\mathcal{D}}$ that induces w. Let $w = w_1 \cdot \ldots \cdot w_m$, where, for $1 \leqslant j \leqslant m$, the word w_j is induced in the component $\mathcal{B}_{i_j}$. We say that w is suffix-less if $w_m = \epsilon$, thus r ends in the initial state of the last component to gain control. We denote by $\pi_w(\mathcal{D}) = e_{i_1}, \ldots, e_{i_{m-1}} \in \mathcal{E}^*$ the sequence of exit states that r visits.

For a state $d \in D$, we define the set $S_d \subseteq S$ so that $s \in S_d$ iff there exists a suffix-less word $w \in (\Sigma_I \times \Sigma_O)^*$ such that $\delta^*_{\mathcal{S}}(w) = s$ and $\delta^*_{\mathcal{D}}(\pi_w(\mathcal{D})) = d$. Since $\mathcal{D}$ has more than $\binom{n}{n/2} \cdot \ell$ states, there is a component $\mathcal{B}_i \in \mathcal{L}$ such that the set $D' \subseteq \mathcal{L}$ of states that are labeled with $\mathcal{B}_i$ is larger than $\binom{n}{n/2}$. Thus, there must be two states $d, d' \in D'$ that have $S_{d'} \subseteq S_d$. Note that $\nu(d) = \nu(d') = i$.

We construct a new design $\mathcal{D}'$ by merging d' into d. Formally, we define $D' = D \backslash \{d'\}$. If d' is the initial state of $\mathcal{D}$, then we define $d'^0 = d$ and otherwise we define $d'^0 = d^0$. For $t \in D'$ and $e \in \mathcal{E}$, if $\delta_{\mathcal{D}}(t, e) = d'$, then we define $\delta_{\mathcal{D}'}(t, e) = d$, and otherwise we define $\delta_{\mathcal{D}'}(t, e) = \delta_{\mathcal{D}}(t, e)$. Finally, for every $t \in D'$ we define $\nu'(t) = \nu(t)$.

Clearly, for every component $\mathcal{B} \in \mathcal{L}$ we have $nused(\mathcal{D}, \mathcal{B}) \geqslant nused(\mathcal{D}', \mathcal{B})$, thus $cost(\mathcal{D}) \geqslant cost(\mathcal{D}')$. Moreover, $\mathcal{D}'$ has less states than $\mathcal{D}$. Thus, in order complete the contradiction and conclude the proof of the theorem, we prove the following.

Claim 4.6 *We claim that $\mathcal{D}'$ is a correct design, thus $L(\mathcal{T}_{\mathcal{L},\mathcal{D}'}) \subseteq L(\mathcal{S})$.*

In order to prove the claim we prove the following.

Claim 4.7 *For every suffix-less (w.r.t $\mathcal{D}'$) word $x \in L(\mathcal{T}_{\mathcal{L},\mathcal{D}'})$ there is a suffix-less (w.r.t $\mathcal{D}$) word $y \in L(\mathcal{T}_{\mathcal{L},\mathcal{D}})$ such that $\delta^*_{\mathcal{S}}(x) = \delta^*_{\mathcal{S}}(y)$ and $\delta^*_{\mathcal{D}}(\pi_y(\mathcal{D})) = \delta^*_{\mathcal{D}'}(\pi_x(\mathcal{D}'))$.*

Before proving the correctness of Claim 4.7 we show that it implies the correctness of Claim 4.6. Assume towards contradiction that there is a word $w \in$

$L(\mathcal{T}_{\mathcal{L},\mathcal{D}'})\backslash L(\mathcal{S})$. Let r be the run of $\mathcal{T}_{\mathcal{L},\mathcal{D}'}$ that induces w and let $\mathcal{B}_i \in \mathcal{L}$ be the last component to gain control in r. Let $w_0, \ldots, w_m$ be the partition of w with respect to $\mathcal{D}'$ and let $x = w_0 \cdot \ldots \cdot w_{m-1}$. That is, x is the longest suffix-less prefix of w. Note that $\pi_w(\mathcal{D}') = \pi_x(\mathcal{D}')$. By Claim 4.7 there is a suffix-less (w.r.t $\mathcal{D}$) word $y \in L(\mathcal{T}_{\mathcal{L},\mathcal{D}})$ such that $\delta^*_{\mathcal{S}}(x) = \delta^*_{\mathcal{S}}(y)$ and $\delta^*_{\mathcal{D}}(\pi_y(\mathcal{D})) = \delta^*_{\mathcal{D}'}(\pi_x(\mathcal{D}'))$. Since $\delta^*_{\mathcal{D}}(\pi_y(\mathcal{D})) = \delta^*_{\mathcal{D}'}(\pi_x(\mathcal{D}'))$, the last component to gain control in the two runs is $\mathcal{B}_i$. Since both x and y are suffix-free, the runs that induce them end in the initial state of $\mathcal{B}_i$, thus $\delta^*_{\mathcal{L},\mathcal{D}}(y) = \delta^*_{\mathcal{L},\mathcal{D}'}(x)$. Recall that $w = x \cdot w_n \in L(\mathcal{T}_{\mathcal{L},\mathcal{D}'})$. Thus, $y \cdot w_n \in L(\mathcal{T}_{\mathcal{L},\mathcal{D}})$. Since $\delta^*_{\mathcal{S}}(y) = \delta^*_{\mathcal{S}}(x)$ and $w \notin L(\mathcal{S})$, we have $y \cdot w_n \notin L(\mathcal{S})$. Thus, $y \cdot w_n \in L(\mathcal{T}_{\mathcal{L},\mathcal{D}})\backslash L(\mathcal{S})$, and we reach a contradiction to the correctness of $\mathcal{D}$, and we are done.

We continue to prove Claim 4.7. Consider a suffix-less (w.r.t $\mathcal{D}'$) word $x \in L(\mathcal{T}_{\mathcal{L},\mathcal{D}'})$ and let r be the run of $\mathcal{D}'$ on $\pi_x(\mathcal{D}')$, which are the exit states that $\mathcal{T}_{\mathcal{L},\mathcal{D}'}$ visits when inducing x. The proof is by induction on the number of visits of r to $d \in D'$. For the base case, r does not visit d. Thus, r is a run of $\mathcal{D}$ on $\pi_x(\mathcal{D}')$, and we define $y = x$.

Assume the claim is correct for runs with m visits to d and we prove for runs with $m+1$ visits. Let $\pi_x(\mathcal{D}') = \pi_{x_1}(\mathcal{D}') \cdot e \cdot \pi_{x_3}(\mathcal{D}')$ such that $\delta^*_{\mathcal{D}'}(\pi_{x_1}(\mathcal{D}') \cdot e) = d$ is the last visit of r to d. Let $x = x_1 \cdot x_2 \cdot x_3$, where x_1 and $x_1 \cdot x_2$ are suffix-less. That is, assuming $t = \delta^*_{\mathcal{D}'}(\pi_{x_1}(\mathcal{D}'))$ and $\nu'(t) = i$, then x_2 is the sub word of x that is induced by the component $\mathcal{B}_i$, thus $\delta^*_i(x_2) = e \in \mathcal{E}$. We distinguish between two cases. In the first case $x_2 = \epsilon$, thus $x_1 = \epsilon$ and $d'^0 = d$, and we define $y = x$ as in the base case. In the second case, $x_2 \neq \epsilon$. By the induction hypothesis, there is a suffix-less (w.r.t $\mathcal{D}$) word $y_1 \in (\Sigma_I \times \Sigma_O)^*$ such that $\delta^*_{\mathcal{D}}(\pi_{y_1}(\mathcal{D})) = \delta^*_{\mathcal{D}'}(\pi_{x_1}(\mathcal{D}')) = t \in D'$ and $\delta^*_{\mathcal{S}}(y_1) = \delta^*_{\mathcal{S}}(x_1)$. If $\delta_{\mathcal{D}}(t, e) = d$, then $\delta_{\mathcal{D}'}(t, e) = d$ and $y = y_1 \cdot x_2 \cdot x_3$ clearly meets the requirements of the claim. Assume $\delta_{\mathcal{D}}(t, e) = d'$. Note that $y_1 \cdot x_2$ is a suffix-less (w.r.t $\mathcal{D}$) word with $\delta^*_{\mathcal{D}}(\pi_{y_1 \cdot x_2}(\mathcal{D})) = d'$. Let $\delta^*_{\mathcal{S}}(y_1 \cdot x_2) = s$. Since $S_{d'} \subseteq S_d$, there is a suffix-less (w.r.t $\mathcal{D}$) word $z \in (\Sigma_I \times \Sigma_O)^*$ such that $\delta_{\mathcal{D}}(\pi_z(\mathcal{D})) = d$ and $\delta^*_{\mathcal{S}}(z) = s$. We define $y = z \cdot x_3$, which clearly meets the requirements of the claim, and we are done. □

Before we turn to the lower bound, we argue that the exponential blow-up proven in Theorem 4.5 cannot be avoided:

Theorem 4.8 *For every $n \geqslant 1$, there is an open library $\mathcal{L}$ and specification $\mathcal{S}_n$ such that the size of $\mathcal{L}$ is constant, the size of $\mathcal{S}_n$ is $O(n^2)$, and every cheapest correct design for $\mathcal{S}_n$ that uses components from $\mathcal{L}$ has at least 2^n states.*

Proof: Consider the specification $\mathcal{S}_n$ and library $\mathcal{L}$ that are described in Lemma 4.2. We claim that every correct design that costs 1 cannot count #'s and should thus remember vectors in $\{\tilde{0}, \tilde{1}\}^n$. Assume towards contradiction that there is a correct

design $\mathcal{D}$ with $cost(\mathcal{D}) = 1$ and $\mathcal{D}$ has less than 2^n states. Thus, $nused(\mathcal{D}, \mathcal{B}_1) = 1$. Since $\mathcal{D}$ must assign initial control to $\mathcal{B}_1$, its initial state d^0 is labeled with 1. We claim that if $\mathcal{B}_1$ relinquishes control after reading #, it is assigned control again. Indeed, if one of the other components gains control, for the input word $\#\#\tilde{0}^{n+1}$ the output sequence is a sequence in $\_^*$. Recall that we require that after reading a word in $\#^*(\tilde{0} + \tilde{1})^n$ there is an output of n letters in $\{0, 1\}$, thus we reach a contradiction to the correctness of $\mathcal{D}$. We conclude that the initial state has a $e^1_\#$-labeled self loop, thus $\delta_{\mathcal{D}}(d^0, e^1_\#) = d^0$.

Since $\mathcal{D}$ has less that 2^n states and the components $\mathcal{B}_2$, $\mathcal{B}_3$, and $\mathcal{B}_4$ relinquish control after reading a letter in $\{\tilde{0}, \tilde{1}\}$, there are two words $a, b \in \{\tilde{0}, \tilde{1}\}^n$ such that $a \neq b$ and $\delta^*_{\mathcal{L},\mathcal{D}}(a) = \delta^*_{\mathcal{L},\mathcal{D}}(b)$. Let $0 \leqslant i \leqslant n-1$ such that, WLog, $1 = a_i \neq b_i = 0$. By the above, for the initial state $q^0_{\mathcal{L},\mathcal{D}}$ of $\mathcal{T}_{\mathcal{L},\mathcal{D}}$ we have $\delta_{\mathcal{L},\mathcal{D}}(q^0_{\mathcal{L},\mathcal{D}}, \#) = q^0_{\mathcal{L},\mathcal{D}}$. Thus, $\delta^*_{\mathcal{L},\mathcal{D}}(\#^i a\tilde{0}^i) = \delta^*_{\mathcal{L},\mathcal{D}}(\#^i b\tilde{0}^i)$. Recall that $\mathcal{S}_n$ requires that either 0 or 1 is output after these words are read, thus either $\mathcal{B}_3$ or $\mathcal{B}_4$ gain control. In the first case, the output letter is 0 and the input word $\#^i a\tilde{0}^{i+1}$ produces an output that violates the specification, and in the second case 1 is output and the input word $\#^i b\tilde{0}^{i+1}$ produces an output that violates the specification, and we are done. □

Theorem 4.9 *The BCD problem for open libraries is NEXPTIME-complete.*

Proof: Membership in NEXPTIME follows from Theorem 4.5 and the fact we can check the correctness of a design and calculate its cost in polynomial time.

For the lower bound, we describe a reduction from *EXP-TILING*. Consider an input to EXP-TILING $\langle T, V, H, n\rangle$, where $T = \{t_1, \ldots, t_m\}$ is a set of tiles, $V, H \subseteq T \times T$ are *vertical* and *horizontal* relations, respectively, and $n \in \mathbb{N}$ is an index. We say that $\langle T, V, H, n\rangle \in$ EXP-TILING if it is possible to fill a $2^n \times 2^n$ square with the tiles in T that respects the two relations. Formally, $\langle T, V, H, n\rangle \in$ EXP-TILING if there is a function $f : 2^n \times 2^n \to T$ such that for $a, b \in 2^n$, if $a < 2^n$ then $V(f(a, b), f(a + 1, b))$, and if $b < 2^n$, then $H(f(a, b), f(a, b + 1))$.

Given an input $\langle T, V, H, n\rangle$, we construct an input $\langle \mathcal{L}, \mathcal{S}, k\rangle$ to the open-BCD problem such that there is an exponential tiling iff there is a correct design $\mathcal{D}$ with $cost(\mathcal{D}) \leqslant 2^{2n+1} + 1$. The idea behind the reduction is similar to that of Lemma 4.2. We define $\Sigma_I = \{\tilde{0}, \tilde{1}, \#, c, v, h, \_\}$ and $\Sigma_O = \{0, 1, \_\} \cup T$. For $x \in \{0, 1\}^n$, we use $\tilde{x}$ to refer to the $\{\tilde{0}, \tilde{1}\}$ copy of x. The library $\mathcal{L}$ has the same components as in Lemma 4.2 with an additional *tile component* $\mathcal{B}_t$ for every $t \in T$. The component $\mathcal{B}_t$ outputs t in its initial state, and when reading c, v, or h, it relinquishes control. When reading every other letter, it enters an accepting sink. The construction costs of the components in $\mathcal{L}$ is 0. We define $q\text{-}cost(\mathcal{B}_1) = 2^{2n} + 1$, and $q\text{-}cost(\mathcal{B}_t) = 1$ for all $t \in T$. The other components' quality cost is 0.

Consider a correct design $\mathcal{D}$ with $cost(\mathcal{D}) \leqslant 2^{2n+1} + 1$. We define $\mathcal{S}$ so that a correct design must use $\mathcal{B}_1$ at least once, thus $\mathcal{D}$ uses it exactly once. Intuitively, $a \cdot b$, for $a, b \in \{0,1\}^n$, can be thought of as two coordinates in a $2^n \times 2^n$ square. We define $\mathcal{S}$ so that after reading the word $\tilde{a} \cdot \tilde{b} \in \{\tilde{0}, \tilde{1}\}^{2n}$, a component is output, which can be thought of as the tile in the (a, b) coordinate in the square. The next letter that can be read is either c, v, or h. Then, $\mathcal{S}$ enforces that the output is $a \cdot b$, $(a+1) \cdot b$, and $a \cdot (b+1)$, respectively. Thus, we show that $\mathcal{D}$ uses exactly 2^{2n} tile components and the tiling that it induces is legal.

Recall that $\mathcal{D}$ uses $\mathcal{B}_1$ exactly once and uses at most 2^{2n} tile components. We describe the specification $\mathcal{S}$ as an intersection of the languages of three DFAs. The first DFA $\mathcal{S}_c$ is similar to the specification in Theorem 4.8. The differences are that there are $2n$ chains and after the sequence of $2n$ input letters, a letter $\langle t, c \rangle$, for $t \in T$, must be read. Thus, it guarantees that when $\mathcal{D}$ reads $\tilde{a}\tilde{b}c\_^{2n}$, for $\tilde{a}, \tilde{b} \in \{\tilde{0}, \tilde{1}\}^n$, it outputs a word $\_^{2n}tab$, where $t \in T$ and recall that $a, b \in \{0,1\}^n$ represent the same vectors as $\tilde{a}$ and $\tilde{b}$, respectively.

Consider $a, b \in \{0,1\}^n$, and let $d_{a,b}$ be the state of $\mathcal{D}$ that is reached after reading the word $\tilde{a}\tilde{b}$. Then, control must be assigned to a tile component, thus $t = \nu(\delta^*_{\mathcal{L},\mathcal{D}}(\tilde{a}\tilde{b}))$ is a tile component. Consider the input word $w = c\_^{2n}$. If w is read when $\mathcal{D}$ is at state $d_{a,b}$, the word $a \cdot b$ must be output. Recall that $\mathcal{D}$ uses at most 2^{2n} tile components. A key observation is the following. Consider a state d of $\mathcal{D}$ that is labeled by a tile component. If $\mathcal{D}$ is in state d and ab is output when reading w, then $d = d_{a,b}$.

The next DFA from which $\mathcal{S}$ is devised is $\mathcal{S}_v$. It guarantees that when $\mathcal{D}$ reads the word $\tilde{a}\tilde{b}vc\_^{2n}$ it outputs $\_^{2n}t_it_j(a+1)b$ for $t_i, t_j \in T$ with $V(t_i, t_j)$. Finally, in order to deal with the tiles in the top row, S_v accepts every word that starts with a prefix in $\#^*\tilde{1}^n(\tilde{0}+\tilde{1})^nv$. The DFA $\mathcal{S}_h$ is defined similarly to $\mathcal{S}_v$.

We define a tiling f by $f(a, b) = \nu(d_{a,b})$ for $a, b \in \{0,1\}^n$. We show that the above observation implies that f respects V, and the proof for H similar. Consider $a, b \in \{0,1\}^n$. If $a = 1^n$, then $f(a, b)$ is in the top row, and there is nothing to prove. Otherwise, we view a and b as numbers and claim that $V(f(a, b), f(a+1, b))$. Recall that after reading ab, $\mathcal{D}$ reaches the state $d_{a,b}$ and the tile $f(a, b)$ is output. By the above, $\mathcal{S}_v$ guarantees that if v is read, the next tile that is output respects V. We claim that $f(a+1, b)$ is output. Indeed, when reading $c\_^{2n}$, $\mathcal{S}_v$ guarantees that $(a+1)b$ must be output. By the observation above, $d_{(a+1),b}$ is the only state of $\mathcal{D}$ from which an input of $c\_^{2n}$ produces an output of $(a+1)b$.

The other direction, namely, if there is a tiling of $2^n \times 2^n$, then there is a design that costs $2^{2n} + 1$, is dual to the above. □

5 Computation-Based Cost

While the cost model we use in Section 4 is suited for some settings, e.g., in cases where the goal is to minimize the number of states in the system, in other settings a *computation-based* cost model is more appropriate. For example, in a system that issues grants upon requests, a goal of the designer can be to design a system that minimizes the waiting time for a grant once a request is received. A standard model for reasoning about such costs of computations is *lattice automata* [24]. Such an automaton assigns to each word a value which is an element of some lattice.

We study the synthesis problem from component libraries where the specification is given by a deterministic lattice automaton (LDFA, for short) and the components are box LDFAs. Thus, our goal is to compose the components in the library to construct an LDFA that is equivalent to the specification LDFA, where equivalence means that the two automata assign the same values to all words.

The compleity of problems on lattice automata typically coincide with the complexity of the corresponding problem in the Boolean setting. An exception is the problem of minimization of LDFAs, which is NP-complete [**?**]. It is shown in [**?**], that there is no canonical minimal LDFA for a latticed language. Recall that minimal DFAs play a key role in our upper bound for the design problem in the closed setting (Theorem 3.1) as we assumed the specification is given as such a DFA.

Even with no canonical minimal LDFA for the language of the specification, we show that the design problem can be solved in polynomial time. We assume the specification is given as a *separable* LDFA, which is a type of LDFAs we introduce here. In such an LDFA, every two states have a separating word (similar to minimal DFAs). That is, if there are words w_1 and w_2 whose runs reach two different states, then there is a word w such that $w_1 \cdot w$ gets a different value than $w_2 \cdot w$. We show that every latticed language has a separable LDFA of polynomial size. This result might be of independent interest.

5.1 Lattice automata

Let $\langle A, \leqslant \rangle$ be a partially ordered set, and let P be a subset of A. An element $a \in A$ is an *upper bound* on P if $a \geqslant b$ for all $b \in P$. Dually, a is a *lower bound* on P if $a \leqslant b$ for all $b \in P$. An element $a \in A$ is the *least element of* P if $a \in P$ and a is a lower bound on P. Dually, $a \in A$ is the *greatest element of* P if $a \in P$ and a is an upper bound on P. A partially ordered set $\langle A, \leqslant \rangle$ is a *lattice* if for every two elements $a, b \in A$ both the least upper bound and the greatest lower bound of $\{a, b\}$ exist, in which case they are denoted $a \vee b$ (a *join* b) and $a \wedge b$ (a *meet* b), respectively. A lattice is *complete* if for every subset $P \subseteq A$ both the least upper

bound and the greatest lower bound of P exist, in which case they are denoted $\bigvee P$ and $\bigwedge P$, respectively. In particular, $\bigvee A$ and $\bigwedge A$ are denoted $\top$ (*top*) and $\bot$ (*bottom*), respectively. A lattice $\langle A, \leqslant \rangle$ is finite if A is finite. Note that every finite lattice is complete. A lattice $\langle A, \leqslant \rangle$ is *distributive* if for every $a, b, c \in A$, we have $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ and $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$.

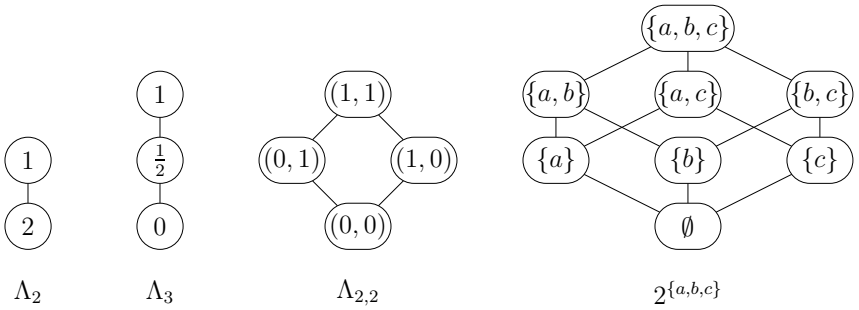


Figure 3: Some lattices

In Figure 3 we describe some finite lattices. The elements of the lattice Λ_2 are the usual truth values 1 (*true*) and 0 (*false*) with the order $0 \leqslant 1$. The lattice Λ_n contains the values $0, 1, \ldots, n-1$, with the order $0 \leqslant 1 \leqslant \ldots \leqslant n-1$. The lattice $\Lambda_{2,2}$ is the Cartesian product of two Λ_2 lattices, thus $(a, b) \leqslant (a', b')$ if both $a \leqslant a'$ and $b \leqslant b'$. Finally, the lattice $2^{\{a,b,c\}}$ is the power set of $\{a, b, c\}$ with the set-inclusion order. In this lattice, *join* and *meet* coincide with union and intersection, respectively, and we have, for example, $\{a\} \vee \{b\} = \{a, b\}$, $\{a\} \wedge \{b\} = \bot$, $\{a, c\} \vee \{b\} = \top$, and $\{a, c\} \wedge \{b\} = \bot$.

Consider a lattice Λ (we abuse notation and refer to Λ also as a set of elements, rather than a pair of a set with an order on it). For a set X of elements, an Λ-*set over* X is a function $S : X \to \Lambda$ assigning to each element of X a value in Λ. Thus, $S \in \Lambda^X$. It is convenient to think about $S(x)$ as the truth value of the statement "x is in S". We say that an Λ-set S is *Boolean* if $S(x) \in \{\top, \bot\}$ for all $x \in X$.

Consider a lattice Λ and an alphabet Σ. An Λ-*language* is an Λ-set over Σ^*. Thus, an Λ-language $L : \Sigma^* \to \Lambda$ assigns a value in Λ to each word over Σ.

A *deterministic lattice automaton* on finite words (LDFA, for short) is a tuple $\mathcal{A} = \langle \Lambda, \Sigma, Q, Q_0, \delta, F \rangle$, where Λ is a finite lattice, Σ is a finite alphabet, Q is a finite set of states, $Q_0 \in \Lambda^Q$ is an Λ-set of initial states, $\delta \in \Lambda^{Q \times \Sigma \times Q}$ is an Λ-transition-relation, and $F \in \Lambda^Q$ is an Λ-set of accepting states.

The fact that $\mathcal{A}$ is deterministic is reflected in two conditions on Q_0 and δ. First, there is at most one state $q \in Q$, called the *initial state* of $\mathcal{A}$, such that $Q_0(q) \neq \bot$. In addition, for every state $q \in Q$ and letter $\sigma \in \Sigma$, there is at most one state $q' \in Q$, called the σ-*destination* of q, such that $\delta(q, \sigma, q') \neq \bot$. The *run* of an LDFA on a word $w = \sigma_1 \cdot \sigma_2 \cdots \sigma_n$ is a sequence $r = q_0, \ldots, q_n$ of $n+1$ states, where q_0 is the initial state of $\mathcal{A}$, and for all $1 \leqslant i \leqslant n$, it holds that q_i is the σ_i-*destination* of

q_{i-1}. We extend the notion of destination to words. For a word $w \in \Sigma^*$ and a state $q \in Q$, we use $\delta^*(q, w)$ to refer to the w-destination of q. When $q \in Q_0$, we omit it and use $\delta^*(w)$. The *value* of w is $val(w) = Q_0(q_0) \wedge \bigwedge_{i=1}^{n} \delta(q_{i-1}, \sigma_i, q_i) \wedge F(q_n)$. Intuitively, $Q_0(q_0)$ is the value of q_0 being initial, $\delta(q_{i-1}, \sigma_i, q_i)$ is the value of q_i being a successor of q_{i-1} when σ_i is the input letter, $F(q_n)$ is the value of q_n being accepting, and the value of w is the meet of all these values. The *traversal value* of w is $trav\text{-}val(w) = Q_0(q_0) \wedge \bigwedge_{i=1}^{n} \delta(q_{i-1}, \sigma_i, q_i)$, and its *acceptance value* is $F(q_n)$. The Λ-language of $\mathcal{A}$, denoted $L(\mathcal{A})$, maps each word w to the value of its run in $\mathcal{A}$. In case such a run does not exist, the value of the word is $\bot$. An example of an LDFA can be found in Figure 5. We say that two LDFA $\mathcal{A}$ and $\mathcal{B}$ are *equivalent* iff they assign the same values to all words, thus for $w \in \Sigma^*$, we have $val(\mathcal{A}, w) = val(\mathcal{B}, w)$.

Note that traditional deterministic automata over finite words (DFA, for short) correspond to LDFA over the lattice Λ_2. Indeed, over Λ_2, a word is mapped to the value $\top$ iff the run on it uses only transitions with value $\top$ and its final state has value $\top$.

An LDFA is *simple* if Q_0 and δ are Boolean. Note that the traversal value of a run r of a simple LDFA is either $\bot$ or $\top$, thus the value of r is induced by F. For simplicity, in such LDFAs we assume $\delta : Q \times \Sigma \to Q$.

The following lemma was proven in [24] and we give it here for completeness.

Lemma 5.1 *[24] Every LDFA over a lattice Λ with n states has an equivalent simple LDFA with $n \cdot |\Lambda|$ states.*

Proof: Consider an LDFA $\mathcal{D} = \langle \Lambda, \Sigma, D, d_0, \delta, F \rangle$. Intuitively, in the simple LDFA $\mathcal{D}'$ we "remember" the lattice value of a run of $\mathcal{D}$. Formally, let $\mathcal{D}' = \langle \Lambda, \Sigma, D \times \Lambda, \langle d_0, \top \rangle, \delta', F' \rangle$, where $F'(\langle d, \ell \rangle) = \ell \wedge F(d)$ and we define δ' below. Recall that for every $d \in D$ and $\sigma \in \Sigma$, there is at most one state $d' \in D$ such that $\delta(d, \sigma, d') \neq \bot$. We define $\delta'(\langle d, \ell \rangle, \sigma) = \langle d', \ell \wedge \delta(d, \sigma, d') \rangle)$. Clearly, the size of $\mathcal{D}'$ is $|D| \cdot |\Lambda|$, and it is simple. Moreover, it is not hard to see that $\mathcal{D}$ and $\mathcal{D}'$ are equivalent. □

5.2 Separable LDFAs

Recall that in the Boolean setting, our solution to the closed design problem relied on the fact that the specification was given as a *minimal* DFA. Specifically, we used the fact that states in such a DFA have *separating words*. It is known that there is no canonical minimal LDFA for a lattice language [21]. However, we show below that it is possible to assume that the specification is given by means of an LDFA whose states have separating words.

Formally, consider an LDFA $\mathcal{D} = \langle \Lambda, \Sigma, D, d_0, \delta, F \rangle$. We say that two states $d_1, d_2 \in D$ have a separating word, if there is a word $w \in \Sigma^*$ such that for every two

words $w_1, w_2 \in \Sigma^*$ with $\delta^*_{\mathcal{D}}(w_1) = d_1$ and $\delta^*_{\mathcal{D}}(w_2) = d_2$, we have $val(\mathcal{D}, w_1 \cdot w) \neq val(\mathcal{D}, w_2 \cdot w)$. We say that $\mathcal{D}$ is *separable* if every two states have a separating word.

For example, consider the LDFA $\mathcal{D}$ that is depicted in Figure 5. The alphabet of $\mathcal{D}$ is $\Sigma = \{1, 2, 3\}$ and it is defined over the lattice $\langle 2^{a,b,c}, \subseteq \rangle$. The states' names appear above them and their acceptance value inside. We show that $\mathcal{D}$ is not separable. For this, we show that the states d_1 and d_2 are not separable, as for the words $w_1 = 1$ and $w_2 = 2$, we have $\delta^*_{\mathcal{D}}(w_1) = d_1$ and $\delta^*_{\mathcal{D}}(w_2) = d_2$, but there is no word w such that $val(\mathcal{D}, w_1 \cdot w) \neq val(\mathcal{D}, w_2 \cdot w)$. Indeed, words with a prefix w_1 or w_2 have either a value of $\{a\}$ or a value of $\bot$. If 1^* is read after reading w_1 or w_2, then the value is $\{a\}$, and otherwise it is $\bot$. Note that it is not possible to simply merge d_1 and d_2 as that would result in a change of the value of either the word 32 or the word 42.

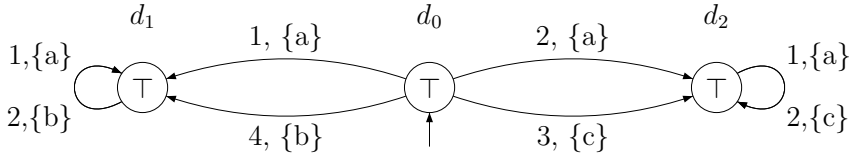


Figure 4: An example of an LDFA in which the states d_1 and d_2 are not separable.

We show that every simple LDFA $\mathcal{D}$ has an equivalent simple separable LDFA $\mathcal{D}'$. By Lemma 5.1, this would imply that every LDFA has an equivalent separable LDFA. Let $\mathcal{D} = \langle \Lambda, \Sigma, D, d_0, \delta, F \rangle$. We define an equivalence relation on D and use it to construct $\mathcal{D}'$. Consider a join-irreducible element $a \in JI(\Lambda)$. We construct a DFA $\mathcal{D}_a$ with $L(\mathcal{D}_a) = \{w \in \Sigma^* : val(\mathcal{D}, w) \geqslant a\}$ as follows. We define $\mathcal{D}_a = \langle \Sigma, D, d_0, \delta_a, F_a \rangle$, where for $q \in Q$ and $\sigma \in \Sigma$, we have $\delta_a(q, \sigma) = q'$ iff $\delta(q, \sigma, q') = \top$, and $d \in F_a$ iff $a \leqslant F(d)$. Since $\mathcal{D}_a$ is a DFA, its language has a canonical minimal DFA $\mathcal{D}'_a$. Moreover, the states of $\mathcal{D}_a$ refine these of $\mathcal{D}'_a$ such that there is a mapping f_a from the states of $\mathcal{D}_a$ to these of $\mathcal{D}'_a$ such that for two states $d_1, d_2 \in D$, we have $f_a(d_1) = f_a(d_2)$ iff for every word $w \in \Sigma^*$, we have $w \in L(\mathcal{D}^{d_1}_a)$ iff $w \in L(\mathcal{D}^{d_2}_a)$.

We are ready to define the equivalence relation on D. We define $d_1 \sim d_2$ iff $F(d_1) = F(d_2)$ and for every $a \in JI(\Lambda)$ we have $f_a(d_1) = f_a(d_2)$. Let $D_\sim$ be the partition of D according to $\sim$.

Consider two states $d_1, d_2 \in D$ such that $d_1 \sim d_2$ and a letter $\sigma \in \Sigma$. We claim that $d'_1 = \delta(d_1, \sigma) \sim \delta(d_2, \sigma) = d'_2$. Assume otherwise, thus either $F(d'_1) \neq F(d'_2)$ or there is $a \in JI(\Lambda)$ such that $f_a(d'_1) \neq f_a(d'_2)$. Assume that the second case holds. Then, there is a word $w \in \Sigma^*$, such that $a \leqslant val(\mathcal{D}^{d'_1}, w)$ and $a \not\leqslant val(\mathcal{D}^{d'_2}, w)$, or the other way around. But then, the word $\sigma \cdot w$ is separating for d_1 and d_2, contradicting the fact that $f_a(d_1) = f_a(d_2)$. The proof for the first case is similar.

Let $\mathcal{D}' = \langle \Lambda, \Sigma, D_\sim, d'_0, \delta', F' \rangle$, where $d'_0 \in D_\sim$ is such that $d_0 \in d'_0$, and δ' and F' are defined as follows. For $A \in D_\sim$, $d \in A$, and $\sigma \in \Sigma$, we define $\delta'(A, \sigma) = A'$ iff

$\delta(d, \sigma) \in A'$. Then, $F'(A) = F(d)$. By the above, δ' is well defined.

For example, consider the simple LDFA $\mathcal{D}$ that is equivalent to the one in Figure 5. Here, we do not state the values of the transitions as they are all $\top$. Note that the states $\langle d_1, \{a\}\rangle$ and $\langle d_2, \{a\}\rangle$ are not separable and in $\mathcal{D}'$ they are merged.

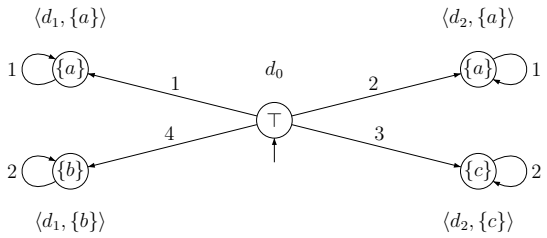


Figure 5: A simple LDFA that is equivalent to the one in Figure 5.

We claim that $\mathcal{D}'$ is equivalent to $\mathcal{D}$. Consider a word $w = w_1, \ldots, w_n \in \Sigma^*$ and let $r = r_0, r_1, \ldots, r_n$ and $r' = r'_0, r'_1, \ldots, r'_n$ be the runs of $\mathcal{D}$ and $\mathcal{D}'$ on w, respectively. It is not hard to prove by induction on i that for every $0 \leqslant i \leqslant n$, we have $r_i \in r'_i$. Since $\mathcal{D}$ and $\mathcal{D}'$ are simple, we have $val(\mathcal{D}, w) = F(r_n)$ and $val(\mathcal{D}', w) = F'(r'_n)$. By the claim, $r_n \in r'_n$, and by the definition of F', we have $F'(r'_n) = F(r_n)$, and we are done.

Finally, we claim that $\mathcal{D}'$ is separable. Consider two words $w_1, w_2 \in \Sigma^*$ such that $A_1 = \delta^*_{\mathcal{D}'}(w_1) \neq \delta^*_{\mathcal{D}'}(w_2) = A_2$. If $F(A_1) \neq F(A_2)$, then ϵ is a separating word. Assume $F(A_1) = F(A_2)$, and we show that they have a separating word. Let $d_1 = \delta^*_{\mathcal{D}}(w_1)$ and $d_2 = \delta^*_{\mathcal{D}}(w_2)$. By the above, we have $d_1 \in A_1$ and $d_2 \in A_2$. Since $A_1 \neq A_2$, there is $a \in JI(\Lambda)$ such that $f_a(d_1) \neq f_a(d_2)$. Thus, there is a word $w \in \Sigma^*$ such that $a \leqslant val(\mathcal{D}^{d_1}, w)$ and $a \nleqslant val(\mathcal{D}^{d_2}, w)$, or the other way around. Note that since $\mathcal{D}$ is simple, we have $val(\mathcal{D}, w_1 \cdot w) = val(\mathcal{D}^{d_1}, w)$ and $val(\mathcal{D}, w_2 \cdot w) = val(\mathcal{D}^{d_2}, w)$. Since $\mathcal{D}$ and $\mathcal{D}'$ are equivalent, they assign the same values to the words $w_1 \cdot w$ and $w_2 \cdot w$, thus we have $a \leqslant val(\mathcal{D}', w_1 \cdot w)$ and $a \nleqslant val(\mathcal{D}', w_1 \cdot w)$. Recall that two lattice values $x, y \in \Lambda$ are equal iff for every $a \in JI(\Lambda)$, we have $a \leqslant x$ iff $a \leqslant y$. Thus, we have that $val(\mathcal{D}', w_1 \cdot w) \neq val(\mathcal{D}', w_1 \cdot w)$, and we are done.

Note that the size of $\mathcal{D}'$ is at most the size of $\mathcal{D}$. We conclude with the following.

Theorem 5.2 *Every simple LDFA with n states has an equivalent simple LDFA with at most n states.*

Recall that in Lemma 5.1, the construction of a simple LDFA results in a blowup in the size of lattice. Thus, we have the following.

Corollary 5.3 *Every LDFA with n states over a lattice Λ has an equivalent simple separable LDFA of size at most $n \cdot |\Lambda|$.*

5.3 The closed-lattice synthesis problem

Consider a library $\mathcal{L}$ of box-LDFAs and a design $\mathcal{D}$. Recall that a design in the Boolean setting is a recipe to construct an LDFA by glueing the components in $\mathcal{L}$. Given a design $\mathcal{D}$, we refer to $\mathcal{A}_{\mathcal{L},\mathcal{D}}$ as the compositional LDFA that is constructed using $\mathcal{D}$ and the components from $\mathcal{L}$. We can now define the *lattice closed-design* problem. Given a library of box-LDFAs $\mathcal{L}$ and a specification LDFA $\mathcal{S}$, decide whether there is a design $\mathcal{D}$ such that $\mathcal{A}_{\mathcal{L},\mathcal{D}}$ is equivalent to $\mathcal{S}$. Recall that in the lattice setting this means that $\mathcal{S}$ and $\mathcal{A}_{\mathcal{L},\mathcal{D}}$ assign the same values to all words. We assume that the components in $\mathcal{L}$ as well as $\mathcal{S}$ are all defined with respect to the same lattice.

Theorem 5.4 *The lattice closed design problem can be solved in polynomial time.*

Proof: The solution here is similar to that of the closed design problem in Theorem 3.1. Given a library $\mathcal{L}$ of box-LDFAs and a specification LDFA $\mathcal{S}$, we construct a full information safety game $\mathcal{G}_{\mathcal{L},\mathcal{S}}$ such that Player 1 wins iff there is a design for $\mathcal{S}$ using the components of $\mathcal{L}$. Recall that a safety game is a turn-based two player game in which Player 1 wins iff the token that the players move stays within the "safe" vertices. Again, similar to the Boolean setting, Player 1's strategies correspond to designs. He select the first component to gain control, and once a component relinquishes control, he selects which component gains control next.

Player 2 challenges Player 1's choice of design. In the Boolean setting, he selects the word that is read while a component is in control, which amounts to selecting an exit state from which the component relinquishes control. In the lattice setting, different runs that exit through the same exit state might have different traversal values. Player 1 should not know what the traversal value is. This has the sense of partial information, which caused the exponential blowup in the open setting. However, we are able to bypass this problem. Assume Player 1 assigns control to a components $\mathcal{B}_i$, and Player 2 selects the exit state e. Then, we maintain both the join and the meet of all possible values of runs that exit through e.

Formally, assume $\mathcal{L}$ consists of components of the form $\mathcal{B}_i = \langle \Lambda, \Sigma, C_i, c_i^0, F_i, E_i \rangle$, for $i \in [n]$. As in the Boolean setting, we denote by $\mathcal{C}$, $\mathcal{C}_0$, and $\mathcal{E}$, the union of the sets of states, initial states, and exit states, respectively, of the components. Let $\mathcal{S} = \langle \Lambda, \Sigma, S, s_0, F_S \rangle$. By Corollary 5.3, we can assume that $\mathcal{S}$ is a simple separable LDFA.

We construct a game $\langle V, E, V_0, \alpha \rangle$, where $V_1 = \mathcal{E} \times \Lambda \times \Lambda \times S$ and $V_2 = \mathcal{C}_0 \times \Lambda \times \Lambda \times S$, $V_1 = \mathcal{C}_0 \times \{\top\} \times \{\top\} \times \{s_0\}$, and E and α are defined as follows. All the vertices in V_1 are safe, i.e., they are in α. We describe when a vertex $\langle c_0^i, \ell_\downarrow, \ell_\uparrow, s \rangle$ in V_2 is not

in α. We alter the definitions we had in the Boolean setting of "infix witness" and "suffix witness" to incorporate the lattice values. First, we have that $\langle c_0^i, \ell_\downarrow, \ell_\uparrow, s\rangle$ is an infix witness if there exists words $w_1, w_2 \in \Sigma^*$ such that $\delta_i^*(w_1) = \delta_i^*(w_2) \in \mathcal{E}$ and $\delta_{\mathcal{S}}^*(s, w_1) \neq \delta_{\mathcal{S}}^*(s, w_2)$. Second, we have that $\langle c_0^i, \ell_\downarrow, \ell_\uparrow, s\rangle$ is a suffix witness if there exists a word $w \in \Sigma^*$ such that either $val(\mathcal{B}_i, w) \wedge \ell_\downarrow \neq val(\mathcal{S}^s, w)$ or $val(\mathcal{B}_i, w) \wedge \ell_\uparrow \neq val(\mathcal{S}^s, w)$.

We describe the edges of the game. First, edges leaving Player 1 vertices are as in the Boolean setting and corresponding to choosing the next component to gain control; we have $\langle\langle e, \ell_\downarrow, \ell_\uparrow, s\rangle, \langle c_0^i, \ell_\downarrow, \ell_\uparrow, s\rangle\rangle \in E$, for every $i \in [n]$. Vertices in $V_2 \backslash \alpha$ have no outgoing transitions. Consider a vertex $v = \langle c_0^i, \ell_\downarrow, \ell_\uparrow, s\rangle \in V_2 \cap \alpha$. Edges leaving v correspond to a choice of exit state e of $\mathcal{B}_i$. Since $v \notin \alpha$, there is a state $s' \in S$ such that every word $w \in \Sigma^*$ that has $\delta^*(w) = e$ also has $\delta_{\mathcal{S}}^*(s, w) = s'$. Finally, the traversal values of these words might be different. We update the meet and join of all these traversal values. Formally, there is an edge $\langle\langle c_0^i, \ell_\downarrow, \ell_\uparrow, s\rangle, \langle e, \ell'_\downarrow, \ell'_\uparrow, s'\rangle\rangle$ iff there exists a word $w \in \Sigma^*$ such that $\delta_i^*(w) = e$ and $\delta_{\mathcal{S}}^*(s, w) = s'$, and $\ell'_\downarrow = \ell_\downarrow \wedge \bigwedge_{w \in \Sigma^* : \delta_i^*(w) = e} trav\text{-}val(\mathcal{B}_i, w)$ and $\ell'_\uparrow = \ell_\uparrow \wedge \bigvee_{w \in \Sigma^* : \delta_i^*(w) = e} trav\text{-}val(\mathcal{B}_i, w)$.

We claim that Player 1 wins $\mathcal{G}_{\mathcal{L},\mathcal{S}}$ iff there is a correct design for $\mathcal{S}$ using the component of $\mathcal{L}$. For the first direction, assume Player 1 has a winning strategy $f_{\mathcal{D}}$ and let $\mathcal{D}$ be the corresponding design as in Theorem 3.1. We claim that $\mathcal{D}$ is a correct design. Assume towards contradiction that there is a word $w \in \Sigma^*$ such that $val(\mathcal{A}_{\mathcal{L},\mathcal{D}}, w) \neq val(\mathcal{S}, w)$. Let $w = w_1 \cdot w_2 \cdot \ldots w_m$ be the partition of w according to the components that process it in $\mathcal{A}_{\mathcal{L},\mathcal{D}}$, and let $\ell_1, \ell_2, \ldots, \ell_m$ be the traversal values of each of the sub-words in the corresponding component. Let $\mathcal{B}_j$ be the component that processes w_m. Note that $val(\mathcal{A}_{\mathcal{L},\mathcal{D}}, w) = \bigwedge_{1 \leqslant i \leqslant m-1} \ell_i \wedge val(\mathcal{B}_j, w_m)$. Consider the Player 2 strategy f_w that, intuitively, selects the word $w_1 \cdot \ldots \cdot w_{m-1}$. The vertex in the game $\mathcal{G}_{\mathcal{L},\mathcal{S}}$ that the game reaches is of the form $v = \langle c_j^0, \ell_\downarrow, \ell_\uparrow, s\rangle \in V_2$. It is not hard to show that $\ell_\downarrow \leqslant \bigwedge_{1 \leqslant i \leqslant m-1} \ell_i \leqslant \ell_\uparrow$.

We claim that $v \notin \alpha$, which will contradict the fact that $f_{\mathcal{D}}$ is winning. Recall that two lattice elements are equal iff their order with respect to all join irreducible elements is the same. We distinguish between two cases. First, there is an element $a \in JI(\Lambda)$ such that $a \leqslant val(\mathcal{A}_{\mathcal{L},\mathcal{D}}, w)$ and $a \not\leqslant val(\mathcal{S}, w)$. We claim that $a \leqslant (\ell_\downarrow \wedge val(\mathcal{B}_j, w_m))$. Indeed, since $a \leqslant val(\mathcal{A}_{\mathcal{L},\mathcal{D}}, w)$, we have $a \leqslant val(\mathcal{B}_j, w_m)$ as well as $a \leqslant \ell_i$, for $1 \leqslant i \leqslant m-1$. By the above, the latter implies that $a \leqslant \ell_\uparrow$. On the other hand, we claim that $a \not\leqslant val(\mathcal{S}^s, w_m)$. It is not hard to see that $\delta_{\mathcal{S}}^*(w_1 \cdot \ldots \cdot w_{m-1}) = s$. Since $\mathcal{S}$ is simple, $val(\mathcal{S}, w) = val(\mathcal{S}^s, w_m)$, thus the claim follows. We conclude that $\ell_\uparrow \wedge val(\mathcal{B}_j, w_m) \neq val(\mathcal{S}^s, w_m)$, implying that $v \notin \alpha$, and we are done. Showing that $v \notin \alpha$ when $a \not\leqslant val(\mathcal{A}_{\mathcal{L},\mathcal{D}}, w)$ and $a \leqslant val(\mathcal{S}, w)$, is done similarly using $\ell_\downarrow$.

For the other direction, assume there is a correct design $\mathcal{D}$, and we show that the

corresponding Player 1 strategy $f_{\mathcal{D}}$ is a winning strategy in $\mathcal{G}_{\mathcal{L},\mathcal{S}}$. Assume otherwise, and let f_2 be a Player 2 strategy that is winning against $f_{\mathcal{D}}$. Let $v = \langle c_i^0, \ell_\downarrow, \ell_\uparrow, s\rangle \notin \alpha$ that the game reaches. Assume $v \notin \alpha$ because of an infix witness. The contradiction is attained similarly to Theorem 3.1. Recall that in this case there are words $w_1, w_2 \in \Sigma^*$ such that $\delta_i^*(w_1) = \delta_i^*(w_2) \in \mathcal{E}$ but $s_1 = \delta_{\mathcal{S}}^*(s, w_1) \neq \delta_{\mathcal{S}}^*(s, w_2) = s_2$. Since $\mathcal{S}$ is separable, there is a separating word w for s_1 and s_2, thus $val(\mathcal{S}^{s_1}, w) \neq val(\mathcal{S}^{s_2}, w)$. But, since $\mathcal{B}_i$ exists from the same exit state when reading w_1 and w_2, the LDFA $\mathcal{A}_{\mathcal{L},\mathcal{D}}$ assigns the same values to the words with suffix $w_1 \cdot w$ and $w_2 \cdot w$, while they get different values in $\mathcal{S}$.

Assume $v \notin \alpha$ because of a suffix witness. Thus, there is a word $w \in \Sigma^*$ such that either $\ell_\downarrow \wedge val(\mathcal{B}_j, w) \neq val(\mathcal{S}^s, w)$ or $\ell_\uparrow \wedge val(\mathcal{B}_j, w) \neq val(\mathcal{S}^s, w)$. We prove for the first case and the second is similar. Assume the play $out(f_{\mathcal{D}}, f_2)$ traverses the components $\mathcal{B}_{i_1}, \ldots, \mathcal{B}_{i_m}$ and exit states $e_1, \ldots, e_m$. We choose words $w_1, \ldots, w_m$ that the components traverse. Thus, we need, for every $1 \leqslant j \leqslant m$ that $\delta_{i_j}^*(w_j) = e_j$. Again, we distinguish between two cases. First, let $a \in JI(\Lambda)$ such that $a \leqslant (\ell_\downarrow \wedge val(\mathcal{B}_j, w))$ and $a \not\leqslant val(\mathcal{S}^s, w)$. Since $a \leqslant \ell_\downarrow$, for every $1 \leqslant j \leqslant m$, every word $u \in \Sigma^*$ such that $\delta_{i_j}^*(u) = e_j$, has $a \leqslant trav\text{-}val(\mathcal{B}_{i_j}, u)$. So we can choose any such word u as w_j. Note that since the intermediate vertices in the play $out(f_{\mathcal{D}}, f_2)$ are in α, it is not hard to show that $\delta_{\mathcal{S}}^*(w_1 \cdot \ldots \cdot w_m) = s$. So, we have $a \leqslant val(\mathcal{A}_{\mathcal{L},\mathcal{D}}, w_1 \cdot \ldots w_m \cdot w)$ while $a \not\leqslant val(\mathcal{S}, w_1 \cdot \ldots w_m \cdot w)$, and we are done. For the second case, $a \not\leqslant (\ell_\downarrow \wedge val(\mathcal{B}_j, w))$ and $a \leqslant val(\mathcal{S}^s, w)$. If $a \not\leqslant val(\mathcal{B}_j, w)$, then we choose for each component some word as in the above. If $a \not\leqslant \ell_\downarrow$, then there is $1 \leqslant j \leqslant m$ and a word $w_j \in \Sigma^*$ such that $\delta_{i_j}^*(w_j) = e_j$ and $a \not\leqslant trav\text{-}val(\mathcal{B}_{i_j}, w_j)$. We choose the other words as in the above, so that we have $a \not\leqslant val(\mathcal{A}_{\mathcal{L},\mathcal{D}}, w_1 \cdot \ldots \cdot w_m)$, and the proof is similar to the above. □

6 Libraries with Costs and Multiple Users

In this section we study the setting in which several designers, each with his own specification, use the library. The construction cost of a component is now shared by the designers that use it, with the share being proportional to the number of times the component is used. For example, if $c\text{-}cost(\mathcal{B}) = 8$ and there are two designers, one that uses $\mathcal{B}$ once and a second that uses $\mathcal{B}$ three times, then the construction costs of $\mathcal{B}$ of the two designers are 2 and 6, respectively. The quality cost of a component is not shared. Thus, the cost a designer pays for a design depends on the choices of the other users and he has an incentive to share the construction costs of components with other designers. We model this setting as a multi-player game, which we dub *component library games* (CLGs, for short). The game can be

thought of as a one-round game in which each player (user) selects a design that is correct according to his specification. In this section we focus on closed designs.

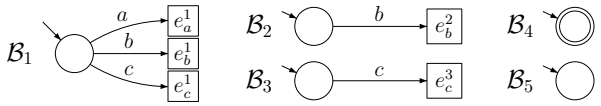
Formally, a CLG is a tuple $\langle \mathcal{L}, \mathcal{S}_1, \ldots, \mathcal{S}_k \rangle$, where $\mathcal{L}$ is a closed component library and, for $1 \leqslant i \leqslant k$, the DFA $\mathcal{S}_i$ is a specification for Player i. A strategy of Player i is a design that is correct with respect to $\mathcal{S}_i$. We refer to a choice of designs for all the players as a *strategy profile*. Consider a profile $P = \langle \mathcal{D}_1, \ldots, \mathcal{D}_k \rangle$ and a component $\mathcal{B} \in \mathcal{L}$. The construction cost of $\mathcal{B}$ is split proportionally between the players that use it. Formally, for $1 \leqslant i \leqslant k$, recall that we use $nused(\mathcal{B}, \mathcal{D}_i)$ to denote the number of times $\mathcal{D}_i$ uses $\mathcal{B}$. For a profile P, let $nused(\mathcal{B}, P)$ denote the number of times $\mathcal{B}$ is used by all the designs in P. Thus, $nused(\mathcal{B}, P) = \sum_{1 \leqslant i \leqslant k} nused(\mathcal{B}, \mathcal{D}_i)$. Then, the construction cost that Player i pays in P for $\mathcal{B}$ is $c\text{-}cost_i(P, \mathcal{B}) = c\text{-}cost(\mathcal{B}) \cdot \frac{nused(\mathcal{B}, \mathcal{D}_i)}{nused(\mathcal{B}, P)}$. Since the quality costs of the components is not shared, it is calculated as in Section 4. Thus, the cost Player i pays in profile P, denoted $cost_i(P)$ is $\sum_{\mathcal{B} \in \mathcal{L}} c\text{-}cost_i(P, \mathcal{B}) + nused(\mathcal{B}, \mathcal{D}_i) \cdot q\text{-}cost(\mathcal{D}_i)$. We define the cost of a profile P, denoted $cost(P)$, as $\sum_{i \in [k]} cost_i(P)$.

For a profile P and a correct design $\mathcal{D}$ for Player i, let $P[i \leftarrow \mathcal{D}]$ denote the profile obtained from P by replacing the choice of design of Player i by $\mathcal{D}$. A profile P is a *Nash equilibrium* (NE) if no Player i can benefit by unilaterally deviating from his choice in P to a different design; i.e., for every Player i and every correct design $\mathcal{D}$ with respect to $\mathcal{S}_i$, it holds that $cost_i(P[i \leftarrow \mathcal{D}]) \geqslant cost_i(P)$.

Theorem 6.1 *There is a CLG with no NE.*

Proof: We adapt the example for multiset cost-sharing games from [7] to CLGs. Consider the two-player CLG over the alphabet $\Sigma = \{a, b, c\}$ in which Player 1 and 2's specifications are (the single word) languages $\{ab\}$ and $\{c\}$, respectively. The library is depicted in Figure 6, where the quality costs of all components is 0, $c\text{-}cost(\mathcal{B}_1) = 12$, $c\text{-}cost(\mathcal{B}_2) = 5$, $c\text{-}cost(\mathcal{B}_3) = 1$, and $c\text{-}cost(\mathcal{B}_4) = c\text{-}cost(\mathcal{B}_5) = 0$. Both players have two correct designs. For Player 1, the first design uses $\mathcal{B}_1$ twice and the second design uses $\mathcal{B}_1$ once and $\mathcal{B}_2$ once. There are also uses of $\mathcal{B}_4$ and $\mathcal{B}_5$, but since they can be used for free, we do not include them in the calculations. For Player 2, the first design uses $\mathcal{B}_2$ once, and the second design uses $\mathcal{B}_1$ once. The table in Figure 6 shows the players' costs in the four possible CLG's profiles, and indeed none of the profiles is a NE. □

We study computational problems for CLGs. The most basic problem is the *best-response problem* (BR problem, for short). Given a profile P and $i \in [k]$, find the cheapest correct design for Player i with respect to the other players' choices in P. Apart from its practical importance, it is an important ingredient in the solutions to the other problems we study. The next problem we study is finding the



	$\{\mathcal{B}_1, \mathcal{B}_1\}$	$\{\mathcal{B}_1, \mathcal{B}_2\}$
$\{\mathcal{B}_3\}$	$5, 12$	$5, 13$
$\{\mathcal{B}_1\}$	$4, 8$	$6, 7$

Figure 6: The library of the CLG with no NE, and the costs of the players in its profiles.

social optimum (SO, for short), namely the profile that minimizes the total cost of all players; thus the one obtained when the players obey some centralized authority. For both the BR and SO problems, we study the decision (rather than search) variants, where the input includes a threshold μ. Finally, since CLGs are not guaranteed to have a NE, we study the problem of deciding whether a given CLG has a NE. We term this problem $\exists$NE.

Definition 6.1 *We define the decision problems formally as follows. Let $\mathcal{G}$ be a CLG.*

BR *An input $\langle \mathcal{G}, P, i, \mu \rangle$ is in BR, where P is a profile, $i \in [k]$, and $\mu \in \mathbb{R}$, iff there is a design $\mathcal{D}_i$ that is correct with respect to $\mathcal{S}_i$ such that $cost_i(P[i \leftarrow \mathcal{D}_i]) \leqslant \mu$.*

SO *An input $\langle \mathcal{G}, \mu \rangle$ is in SO, where $\mu \in \mathbb{R}$, iff there is a profile P such that $cost(P) \leqslant \mu$.*

$\exists$NE *An input $\langle \mathcal{G} \rangle$ is in $\exists$NE iff there is a NE profile in $\mathcal{G}$.*

Note that the BCD problem studied in Section 4 is a special case of BRP when there is only one player. Also, in a setting with a single player, the SO and BR problems coincide, thus the lower bound of Theorem 4.3 applies to them. In Lemma 4.1 we showed that if there is a correct design $\mathcal{D}$ with $cost(\mathcal{D}) \leqslant \mu$, then there is also a correct design $\mathcal{D}'$, based on a memoryless strategy and hence having polynomially many states, such that for every component $\mathcal{B}$, we have $nused(\mathcal{D}', \mathcal{B}) \leqslant nused(\mathcal{D}, \mathcal{B})$. The arguments there apply in the more general case of CLGs. Thus, we have the following.

Theorem 6.2 *The BR and SO problems are NP-complete.*

We continue to study the $\exists$NE problem. We show that $\exists$NE is complete for Σ_2^P – the second level of the polynomial hierarchy. Namely, decision problems solvable in polynomial time by a nondeterministic Turing machine augmented by an *oracle* for an NP-complete problem. An oracle for a computational problem is a black box that is able to produce a solution for any instance of the problem in a single operation. Thus, for every problem $P \in \Sigma_2^P$ there is a machine such that for every $x \in P$ there is

a polynomial accepting computation (with polynomial many queries to the oracle). As co-NP is the dual complexity class of NP, the dual complexity class of Σ_2^P is Π_2^P. Thus, a problem P is Σ_2^P-complete iff its complement $\overline{P}$ is Π_2^P-complete.

Theorem 6.3 *The ∃NE problem is Σ_2^P-complete.*

Proof: For the upper bound, we describe a nondeterministic Turing machine with an oracle to SBR problem – the strict version of the BR problem, where we seek a design whose cost is strictly smaller than μ. Given a CLG $\mathcal{G} = \langle \mathcal{L}, \mathcal{S}_1, \ldots, \mathcal{S}_k \rangle$, we guess a profile $P = \langle \mathcal{D}_1, \ldots, \mathcal{D}_k \rangle$, where for $1 \leqslant i \leqslant k$, the design $\mathcal{D}_i$ has at most $|\mathcal{C}_0 \times S_i|$ states, where S_i are the states of $\mathcal{S}_i$. First, we check whether P is a profile of correct designs. That is, for $i \in [k]$, we check whether $\mathcal{D}_i$ is a correct design with respect to $\mathcal{S}_i$, which can be done in polynomial time. If there is an incorrect design we terminate and reject. Next, we check whether P is a NE profile by checking if there is a player that has a beneficial move from P. That is, for $i \in [k]$, we feed the oracle the input $\langle \mathcal{G}, P, i, cost_i(P) \rangle$. If the oracle answers YES, then Player i can benefit from deviating and P is not a NE in which case we reject. On the other hand, if for every $i \in [k]$ the oracle answers NO, then P is a NE in which case we accept. Clearly the machine recognizes ∃NE. Note that if $\mathcal{G} \in$ ∃NE, one of the profiles P we guess is a NE, and the computation in which we guess P is a polynomial accepting computation.

For the lower bound, we show a reduction from the complement of the Π_2^P-complete problem *min-max vertex cover* [23] (MMVC, for short). The input to the MMVC problem is $\langle G, I, J, N, \mu \rangle$, where $G = \langle V, E \rangle$ is an undirected graph, I and J are sets of indices, $N : V \to \{V_{i,j} \subseteq V : i \in I \text{ and } j \in J\}$ partitions the vertices, and $\mu \in \mathbb{N}$ is a value. Note that since G is undirected, its edges are sets with two vertices. We refer to the sets in the partition $\{V_{i,j}\}_{i \in I, j \in J}$ as *neighborhoods* and for $v \in V$ we refer to $N(v)$ as the neighborhood of v. Note that there is a coarser partition of V, namely $\{V_i\}_{i \in I}$, where $V_i = \bigcup_{j \in J} V_{i,j}$. We refer to the elements in this partition as *districts* and, for $v \in V$, use $D(v)$ to denote the district v belongs to. For a function $t : I \to J$ we define $V_t = \bigcup_{i \in I} V_{i,t(i)}$. Intuitively, t is a choice of neighborhood in each district. Let $G_t = \langle V_t, E_t \rangle$ be the induced subgraph of G on the vertex set V_t. Formally, for $e \in E$, we have $e \in E_t$ iff $e \subseteq V_t$. For a graph G, we say that $V' \subseteq V$ is a vertex cover of G if for every $e = \{u, v\} \in E$ we have $V' \cap \{u, v\} \neq \varnothing$. An input $\langle G, I, J, N, \mu \rangle$ is in MMVC iff for any choice of neighborhoods in the districts given by a function t, the smallest vertex cover of the resulting graph is at most μ. Formally, $\max_{t \in J^I} \min\{|V'| : V' \subseteq V_t \text{ is a vertex cover of } G_t\} \leqslant \mu$. We assume without loss of generality that $\mu \leqslant |V|$.

Consider an input $\langle G, I, J, N, \mu \rangle$ to MMVC. We construct a CLG $\mathcal{G}$ with library $\mathcal{L}$ and specifications $\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_{|I|}$ such that $\mathcal{G}$ has a NE iff $\langle G, I, J, N, \mu \rangle \notin$

MMVC. The alphabet Σ consists of letters $i, \tilde{i} \in I$, $v, \tilde{v} \in V$, and $e \in E$. Let $E = \{e_1, \ldots, e_m\}$. The specification of Player 0 consists of words of length $3|E|$ of the form $v_1\tilde{v_1}e_1 \ldots v_m\tilde{v_m}e_m$, for some $v_1, \ldots, v_m \in V$ (allowing duplicates). For $i \in I$, the specifications of Player i consist of the single word $i \cdot (\tilde{i})^\ell$, where ℓ is a polynomial in $|V|$, which we define in the following.

We describe the components of $\mathcal{L}$ (see Figure 7). When describing the components' costs we only refer to the construction cost as their quality costs are 0. The simplest components are $\mathcal{B}_{acc}$ and $\mathcal{B}_{rej}$. They consist of a single initial state that is accepting in $\mathcal{B}_{acc}$ and rejecting in $\mathcal{B}_{rej}$. The cost of both these components is 0. The next component is $\mathcal{B}_0$, which is exactly $\mathcal{S}_0$, and its cost is $\mu + 1$. The rest of the components have no accepting states. We describe these components by the words that they can process. For each word there is a unique disjoint path from the initial to a separate exit state. For every neighborhood $V_{i,j}$ there is a component $\mathcal{B}_{i,j} \in \mathcal{L}$ that costs $(3|E| + 2)(\mu + 1)$. We refer to these components as *neighborhood components*. The single-lettered words it can process are i, $\tilde{i}$, and v for $v \in V_{i,j}$. Also, it can process the words $v\tilde{v}e$ for $v \in V \backslash V_{i,j}$ and $e \in E$, and $\tilde{v}e$ for $v \in V_{i,j}$ and $e \in E$ such that $e \cap (V_i \backslash V_{i,j}) \neq \varnothing$. For every $v \in V$ there is a component $\mathcal{B}_v$ that costs 1. We refer to these components as *vertex components*. The words it can process are $\tilde{u}e$ for $u \in N(v)$ and $e \in E$ such that $v \in e$. For $i \in I$ there is a component $\mathcal{B}_i$ that can process the word $\tilde{i}$ and costs a very small value $\xi > 0$. The construction is clearly polynomial in the input.

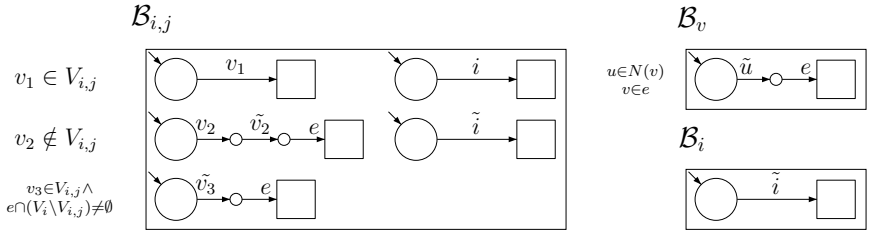


Figure 7: Some of the components in the library produced by the reduction from MMVC.

We claim that $\langle G, I, J, N, \mu \rangle \notin$ MMVC iff $\mathcal{G} = \langle \mathcal{L}, \mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_{|I|} \rangle \in \exists$NE. For the first direction, assume $\langle G, I, J, N, \mu \rangle \notin$ MMVC. Thus, there is a function $t : I \to J$ such that every vertex cover of V_t has more than μ vertices. We claim that the CLG $\mathcal{G}$ has a NE. Consider the profile P in which Player 0 uses only the component $\mathcal{B}_0$ and, for $i \in I$, Player i uses the design $\mathcal{D}_{i,t(i)}$, which we describe below, and uses only the neighborhood component $\mathcal{B}_{i,t(i)}$ and the components $\mathcal{B}_{acc}$ and $\mathcal{B}_{rej}$. For $i \in I$ and $j \in J$, we describe the design $\mathcal{D}_{i,j}$. In $\mathcal{D}_{i,j}$, the component $\mathcal{B}_{i,j}$ gains initial control. If the component relinquishes control after reading the single-lettered word i, it regains control. If it relinquishes control after reading any other word, the design assigns

control to $\mathcal{B}_{rej}$. Similarly, for ℓ times, control is given to $\mathcal{B}_{i,j}$ assuming it reads the word $\tilde{i}$, and otherwise control is given to $\mathcal{B}_{rej}$. After $\mathcal{B}_{i,j}$ gains control $\ell + 1$ times, control is assigned to $\mathcal{B}_{acc}$. Clearly, $L(\mathcal{D}_{i,j}) = L(\mathcal{S}_i)$, thus it is a correct design for Player i.

Assume towards contradiction that P is not a NE. Thus, there is a player that benefits from deviating. We start by showing that, for $i \in I$, Player i cannot benefit from deviating. Note that for $j \in J$, no other player uses a component $\mathcal{B}_{i,j}$ in P. Thus, deviating to a design $\mathcal{D}_{i,j}$, for $j \neq t(i)$, would result in the same cost for Player i, and deviating to any other correct design would increase his cost, and is clearly not beneficial.

We continue to show that Player 0 cannot benefit from a deviation. Assume towards contradiction that there is a correct design $\mathcal{D}$ such that $cost_0(P) > cost_0(P[0 \leftarrow \mathcal{D}])$. Consider the set $V' \subseteq V$ of vertices that correspond to vertex components that are used in $\mathcal{D}$, thus $v \in V'$ iff $\mathcal{B}_v \in used(\mathcal{D})$. Recall that $c\text{-}cost(\mathcal{B}_0) = \mu + 1$, thus $cost_0(P) = \mu + 1$. Since the construction costs of the vertex components is 1 and Player 1 does not share them, we have $|V'| \leqslant \mu$.

We claim that there is a vertex cover $V'' \subseteq V' \cap V_t$ for G_t. Since $|V'| \leqslant \mu$, this would contradict our assumption that $\langle G, I, J, N, \mu \rangle \notin$ MMVC. Recall that the cost of using a neighborhood component without sharing is more than $\mu + 1$. Since we assume Player 0 benefits from deviating to $\mathcal{D}$, he must share all his uses of these components. Since the neighborhood components that are used by the players $1, \ldots, |I|$ are exactly these that are dictated by t, every neighborhood component $\mathcal{B}_{i,j} \in used(\mathcal{D})$ has $j = t(i)$.

We claim that for every $e \in E_t$ there is a vertex $v \in V' \cap V_t$ such that $v \in e$. Let $E_t = \{e_{i_1}, \ldots, e_{i_l}\}$ and $j \in [l]$. Consider the word $w = v\tilde{v}e_1 v\tilde{v}e_2 \ldots v\tilde{v}e_{i_j - 1}$ for some $v \in V$. Since w is a prefix of a word in $L(\mathcal{S}_0)$, the run of $\mathcal{A}_{\mathcal{L},\mathcal{D}}$ on w does not get stuck. Since the last letter in w is in E the component that gains control after reading w is a neighborhood component, thus it is $\mathcal{B}_{i,t(i)}$ for some $i \in I$.

Consider the word $w' = w \cdot u\tilde{u}e_{i_j}$ for $u \in V_{i,t(i)}$. Again, since w' is a prefix of a word in $L(\mathcal{S}_0)$ the run of $\mathcal{A}_{\mathcal{L},\mathcal{D}}$ on w' does not get stuck. Since $u \in V_i$, $\mathcal{B}_{i,t(i)}$ relinquishes control after reading u. We claim that a vertex component $\mathcal{B}_v$ must gain control next. Indeed, since $u \in V_{i,t(i)}$, the only neighborhood component that is a candidate to process the word $\tilde{u}e$ is $\mathcal{B}_{i,t(i)}$. However, since $u \in G_t$ if it has an endpoint that belongs to the district V_i, then the endpoint is in $V_{i,t(i)}$. Thus, $\mathcal{B}_{i,t(i)}$ cannot process $\tilde{u}e$ and it is processed by a vertex component $\mathcal{B}_v$. Note that since it can process the word $\tilde{u}e$, we have $u \in N(v)$, thus $N(v) = V_{i,t(i)}$. Moreover, $v \in e$. Clearly $v \in V'$ as $\mathcal{D}$ uses $\mathcal{B}_v$, and we are done.

For the second direction, assume $\langle G, I, J, N, \mu \rangle \in$ MMVC. Assume towards contradiction that there is a NE profile P in $\mathcal{G}$. We distinguish between two cases.

In the first case, Player 0 does not use $\mathcal{B}_0$. Recall that Player 0 has a correct design that uses only $\mathcal{B}_0$ and costs $\mu + 1$. Moreover, every correct design that does use $\mathcal{B}_0$ must use a neighborhood component $\mathcal{B}_{i,j}$, which costs $(3|E| + 2)(\mu + 1)$. Since P is a NE, Player i, the only player that can use $\mathcal{B}_{i,j}$, uses $\mathcal{B}_{i,j}$ at least $3|E| + 1$ times. We claim that Player i has a beneficial deviation from P, contradicting the fact that P is a NE. First, we bound $cost_i(P)$. Clearly, Player 0 uses $\mathcal{B}_{i,j}$ at most $3|E|$ times, thus $cost_i(P) \geqslant \frac{3|E|+1}{3|E|+3|E|+1} \cdot cost(\mathcal{B}_{i,j})$. We describe a beneficial deviation for Player i. Consider the design $\mathcal{D}$ that assigns initial control to $\mathcal{B}_{i,j}$. If it relinquishes control after reading anything different from i, $\mathcal{D}$ assigns control to $\mathcal{B}_{rej}$. Otherwise, the component $\mathcal{B}_i$, which can process only the word $\tilde{i}$, gains control ℓ consecutive times after which $\mathcal{B}_{acc}$ gains control. It is not hard to see that $L(\mathcal{D}) = L(\mathcal{S}_i)$. Since $nused(\mathcal{D}, \mathcal{B}_{i,j}) = 1$ and $nused(\mathcal{D}, \mathcal{B}_i) = \ell$, we have $cost_i(P[i \leftarrow \mathcal{D}]) \leqslant \frac{1}{2} cost(\mathcal{B}_{i,j}) + \xi$. For sufficiently small ξ, we have $cost_i(P) > cost_i(P[i \leftarrow \mathcal{D}])$, and we are done.

In the second case, the design Player 0 chooses in P is the design that uses only the component $\mathcal{B}_0$. Thus, $cost_0(P) = \mu + 1$. Recall that for every $i \in I$ and $j \in J$, the design $\mathcal{D}_{i,j}$ is a correct design for Player i that uses only the neighborhood component $\mathcal{B}_{i,j}$. It is not hard to see that since P is a NE, every Player i chooses some design $\mathcal{D}_{i,j}$. Let $t : I \to J$ be the function that corresponds to these players choices. Since $\langle G, I, J, N, \mu \rangle \in$ MMVC, there is a vertex cover $V' \subseteq V_t$ of G_t such that $|V'| \leqslant \mu$.

We construct a design $\mathcal{D}$ for Player 0 that is a beneficial deviation from P. Recall that Player 0's specification is the set of words of length $3|E|$ of the form $v_1\tilde{v_1}e_1 \dots v_m\tilde{v_m}e_m$, where $E = \{e_1, \dots, e_m\}$ and $v_1, \dots, v_m \in V$ (allowing duplicates). The definition of $\mathcal{D}$ is inductive. Let $1 \leqslant l \leqslant |E|$. Consider a word $w \in \Sigma^*$ of length $3(l-1)$ that can be extended to a word in $L(\mathcal{S}_0)$. That is, there is a word $x \in \Sigma^*$ such that $w \cdot x \in L(\mathcal{S}_0)$. For $v \in V$, let $w_v = w \cdot v\tilde{v}e_{l+1}$. Note that w_v can also be extended to a word in $L(\mathcal{S}_0)$ (possibly with ϵ). Assuming the run of $\mathcal{A}_{\mathcal{L},\mathcal{D}}$ on w does not get stuck and control is relinquished from some component after reading w, we describe how $\mathcal{D}$ assigns control next such that the run of $A_{\mathcal{L},\mathcal{D}}$ on every w_v does not get stuck and control is relinquished after reading w_v.

We distinguish between two cases. In the first case, $e_l \notin E_t$. Thus, there is a vertex $v \in e_l \backslash V_t$. Let V_i be v's district, thus $V_i = D(v)$. Let $V_{i,j} \subseteq V_i$ be the neighborhood in V_i that is selected by t, thus $j = t(i)$. Note that this since $v \notin V_t$, it does not belong to $V_{i,j}$. The component to which $\mathcal{D}$ assigns control after reading w is the neighborhood component $\mathcal{B}_{i,j}$. Consider a vertex $u \in V$. If $u \notin V_{i,j}$, then when reading $u\tilde{u}e$ the run in $\mathcal{B}_{i,j}$ does not get stuck and control is relinquished at its end. If $u \in V_{i,j}$, then $\mathcal{B}_{i,j}$ relinquishes control after reading u. In such a case we define $\mathcal{D}$ to reassign control to $\mathcal{B}_{i,j}$. Note that the run of $\mathcal{B}_{i,j}$ on $\tilde{u}e_l$ does not get stuck. Indeed, the vertex $v \in e_l$ is a witness to the fact that $(V_i \backslash V_{i,j}) \cap e_l \neq \varnothing$. Clearly,

control is relinquished after $\mathcal{B}_{i,j}$ reads $\tilde{u}e_l$. If in one of the times $\mathcal{B}_{i,j}$ gains control it relinquishes it after reading any other word $x \in \Sigma^*$, then there is no $y \in \Sigma^*$ such that $w \cdot x \cdot y \in L(\mathcal{S}_0)$, and we assign control to $\mathcal{B}_{rej}$.

In the second case, $e_l \in E_t$. Thus, there is a vertex v in the vertex cover V' such that $v \in e_l$. Let $N(v) = V_{i,j}$. Note that since $v \in V_t$, we have $t(i) = j$. The component to which $\mathcal{D}$ assigns control after reading w is the neighborhood component $\mathcal{B}_{i,j}$. Consider a vertex $u \in V$. The case where $u \notin V_{i,j}$ is similar to the previous case. For $u \in V_{i,j}$, when $\mathcal{B}_{i,j}$ reads u, it relinquishes control. In such a case, $\mathcal{D}$ assigns control to $\mathcal{B}_v$. Since $v \in V_{i,j}$ and $v \in e_l$, the component $\mathcal{B}_v$ can process the word $\tilde{u}e_l$, and it relinquishes control after its end. Similarly to the above, if $\mathcal{B}_{i,j}$ or $\mathcal{B}_v$ relinquish control after reading any other word $x \in \Sigma^*$, then we assign control to $\mathcal{B}_{rej}$. Finally, if $l = |E|$, we assign control $\mathcal{B}_{acc}$.

Correctness of the design $\mathcal{D}$ is immediate from the construction. We claim that $cost_0(P[0 \leftarrow \mathcal{D}]) < \mu + 1$. Note that the $\mathcal{D}$ uses two types of components. The first type is neighborhood components. Consider $\mathcal{B}_{i,j} \in used(\mathcal{D})$. Note that we constructed $\mathcal{D}$ so that $j = t(i)$. Thus, Player 0 shares the cost of $\mathcal{B}_{i,j}$ with Player i. Recall that Player i chooses the design $\mathcal{D}_{i,j}$ that uses $\mathcal{B}_{i,j}$ ℓ times. We define ℓ to be sufficiently large so that the proportion of the cost that Player 0 pays is less than $\frac{1}{2|E|}$. Thus, the total cost Player 0 endures for neighborhood components is less than 1. The second type of components $\mathcal{D}$ uses is vertex components. Since $\mathcal{D}$ uses vertex components that correspond to vertices in the vertex cover V', the number of such component that $\mathcal{D}$ uses is at most μ, which is the total cost for these components. Thus, $cost_0(P[0 \leftarrow \mathcal{D}]) < \mu + 1$, and we are done. □

7 Discussion

Traditional synthesis algorithms assumed that the system is constructed from scratch. Previous work adjusted synthesis algorithms to a reality in which systems are constructed from component libraries. We adjust the algorithms further, formalize the notions of quality and cost and seek systems of high quality and low cost. We argue that one should distinguish between quality considerations, which are independent of uses of the library by other designs, and pricing considerations, which depend on uses of the library by other designs.

Once we add multiple library users to the story, synthesis is modeled by a resource-allocation game and involves ideas and techniques form algorithmic game theory. In particular, different models for sharing the price of components can be taken. Recall that in our model, users share the price of a component, with the share being proportional to the number of uses. In some settings, a *uniform sharing*

rule may fit better, which also makes the game more stable. In other settings, a more appropriate sharing rule would be the one used in *congestion games* – the more a component is used, the higher is its price, reflecting, for example, a higher load. Somewhat surprising, games with congestion effects turn out to be more stable than cost-sharing games [9]. Still, the complexity of the decision problems we study here for CLGs match the ones for CLGs with congestion effects. Moreover, synthesis of different specifications in different times gives rise to *dynamic allocation* of components, and synthesis of collections of specifications by different users gives rise to *coalitions* in the games. These notions are well studied in algorithmic game theory and enable an even better modeling of the rich settings in which traditional synthesis is applied.

References

[1] M. Abadi, L. Lamport, and P. Wolper. Realizable and unrealizable concurrent program specifications. In *Proc. 25th ICALP*, LNCS 372, pages 1–17. Springer, 1989.

[2] S. Almagor, U. Boker, and O. Kupferman. Formalizing and reasoning about quality. In *Proc. 40th ICALP*, LNCS 7966, pages 15 – 27. Springer, 2013.

[3] G. Alonso, F. Casati, H.A. Kuno, and V. Machiraju. Web Services - Concepts, Architectures and Applications. *Data-Centric Systems and Applications*. Springer, 2004.

[4] R. Alur, K. Etessami, and P. Madhusudan. A temporal logic of nested calls and returns. In *Proc. 10th TACAS*, LNCS 2725, pages 67–79. Springer, 2004.

[5] B. Aminof, F. Mogavero, and A. Murano Synthesis of hierarchical systems. In *Sci. Comput. Program.*, volume 83, pages 56–79, 2014.

[6] G. Avni and O. Kupferman. When does abstraction help? *IPL*, 113:901–905, 2013.

[7] T. Tamir. *private communication.*

[8] G. Avni, O. Kupferman, and T. Tamir. Network-formation games with regular objectives. In *Proc. 17th FoSSaCS*, LNCS 8412, pages 119–133. Springer, 2014.

[9] G. Avni, O. Kupferman, and T. Tamir. Congestion Games with Multisets of Resources and Applications in Synthesis. In *Proc. 35th FSTTCS*, LIPIcs 45, pages 365–379, Schloss Dagstuhl, 2015.

[10] D. Berwanger and L. Doyen. On the power of imperfect information. In *Proc. 28th TST& TCS*, pages 73–82, 2008.

[11] A. Bohy, V. Bruyère, E. Filiot, and J-F. Raskin. Synthesis from LTL specifications with mean-payoff objectives. In *Proc. 19th TACAS*, LNCS 7795, pages 169–184. Springer, 2013.

[12] U. Boker, K. Chatterjee, T.A. Henzinger, and O. Kupferman. Temporal specifications with accumulative values. In *Proc. 26th LICS*, pages 43–52, 2011.

[13] L. de Alfaro, M. Faella, T.A. Henzinger, R. Majumdar, and M. Stoelinga. Model checking discounted temporal properties. *TCS*, 345(1):139–170, 2005.

[14] L. de Alfaro and T.A. Henzinger. Interface theories for component-based design. In *Proc. 1st EMSOFT*, LNCS 2211, pages 148–165. Springer, 2001.

[15] L. Doyen, T. A. Henzinger, B. Jobstmann, and T. Petrov. Interface theories with component reuse. In *Proc. 8th EMSOFT*, pages 79–88, 2008.

[16] J. Elgaard, N. Klarlund, and A. Möller. Mona 1.x: new techniques for WS1S and WS2S. In *Proc. 10th CAV*, LNCS 1427, pages 516–520. Springer, 1998.

[17] A. Fabrikant, C. Papadimitriou, and K. Talwar. The complexity of pure nash equilibria. In *Proc. 36th STOC*, pages 604–612, 2004.

[18] M. Faella, A. Legay, and M. Stoelinga. Model checking quantitative linear time logic. *ENTCS*, 220(3):61–77, 2008.

[19] E. Filiot, N. Jin, and J.-F. Raskin. Antichains and compositional algorithms for LTL synthesis. *FMSD*, 39(3):261–296, 2011.

[20] G. Gößler and J. Sifakis. Composition for component-based modeling. *Sci. Comput. Program.*, 55(1-3):161–183, 2005.

[21] S. Halamish and O. Kupferman. Minimizing deterministic lattice automata. In *Proc. 14th FoSSaCS*, LNCS 6604, pages 199 – 213. Springer, 2011.

[22] M. Jurdzinski. Small progress measures for solving parity games. In *Proc. 17th STACS*, LNCS 1770, pages 290–301. Springer, 2000.

[23] K-I. Ko and C-L. Lin. On the complexity of min-max optimization problems and their approximation. In *Minimax and Applications*, volume 4 of *Nonconvex Optimization and Its Applications*, pages 219–239. Springer, 1995.

[24] O. Kupferman and Y. Lustig. Lattice automata. In *Proc. 8th VMCAI*, LNCS 4349, pages 199 – 213. Springer, 2007.

[25] O. Kupferman, N. Piterman, and M.Y. Vardi. Safraless compositional synthesis. In *Proc. 18th CAV*, LNCS 4144, pages 31–44. Springer, 2006.

[26] O. Kupferman and M.Y. Vardi. Safraless decision procedures. In *Proc. 46th FOCS*, pages 531–540, 2005.

[27] Y. Lustig and M.Y. Vardi. Synthesis from component libraries. *STTT* 15:603–618, 2013.

[28] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. 16th POPL*, pages 179–190, 1989.

[29] J.-F. Raskin, K. Chatterjee, L. Doyen, and T. Henzinger. Algorithms for ω-regular games with imperfect information. *LMCS*, 3(3), 2007.

[30] T. Roughgarden and E. Tardos. How bad is selfish routing? *JACM*, 49(2):236–259, 2002.

[31] S. Safra. On the complexity of ω-automata. In *Proc. 29th FOCS*, pages 319–327, 1988.

[32] A.P. Sistla and E.M. Clarke. The complexity of propositional linear temporal logic. *JACM*, 32:733–749, 1985.

Congestion Games with Multisets of Resources and Applications in Synthesis*

Guy Avni[†] *Orna Kupferman*[‡] *Tami Tamir*[§]

Abstract

In classical congestion games, players' strategies are subsets of resources. We introduce and study *multiset congestion games*, where players' strategies are multisets of resources. Thus, in each strategy a player may need to use each resource a different number of times, and his cost for using the resource depends on the load that he and the other players generate on the resource.

Beyond the theoretical interest in examining the effect of a repeated use of resources, our study enables better understanding of non-cooperative systems and environments whose behavior is not covered by previously studied models. Indeed, congestion games with multiset-strategies arise, for example, in production planing and network formation with tasks that are more involved than reachability. We study in detail the application of synthesis from component libraries: different users synthesize systems by gluing together components from a component library. A component may be used in several systems and may be used several times in a system. The performance of a component and hence the system's quality depends on the load on it.

Our results reveal how the richer setting of multisets congestion games affects the stability and equilibrium efficiency compared to standard congestion games. In particular, while we present very simple instances with no pure Nash equilibrium and prove tighter and simpler lower bounds for equilibrium inefficiency, we are also able to show that some of the positive results known for affine and weighted congestion games apply to the richer setting of multisets.

1 Introduction

Congestion games model non-cooperative resource sharing among selfish players. Resources may be shared by the players and the cost of using a resource increases

*Published in the proceedings of the 35th Conference on Foundation of Software Technology and Theoretical Computer Science, LIPIcs 45, pages 365–379, Schloss Dagstuhl, 2015.

[†]School of Computer Science and Engineering, The Hebrew University, Jerusalem, Israel

[‡]School of Computer Science and Engineering, The Hebrew University, Jerusalem, Israel

[§]School of Computer Science, The Interdisciplinary Center, Israel

with the load on it. Such a cost paradigm models settings where high congestion corresponds to lower quality of service or higher delay. Formally, each resource e is associated with an increasing latency function $f_e : \mathbb{N} \to \mathbb{R}$, where $f_e(\ell)$ is the cost of a single use of e when it has load ℓ.

Previous work on congestion games assumes that players' strategies are subsets of resources, as is the case in many applications, most notably routing and network design. For example, in the setting of networks, players have reachability objectives and strategies are subsets of edges, each inducing a simple path from the source to the target [29, 3, 19]. We introduce and study multiset games, where players' strategies are multisets of resources. Thus, a player may need a resource multiple times – depending on the specific resource and strategy, and his cost for using the resource depends on the load that he and the other players generate on it. Formally, in *multiset congestion games* (MCGs, for short), a player that uses j times a resource e that is used ℓ times by all players together, pays $j \cdot f_e(\ell)$ for these uses.

Beyond the theoretical interest in examining the effect of multisets on the extensively-studied classical games, multiset congestion games arise naturally in many applications and environments. The use of multisets enables the specification of rich settings that cannot be specified by means of subsets. We give here several examples.

As a first example, consider *network formation*. In addition to reachability tasks, which involve simple paths (and hence, subsets of resources), researchers have studied tasks whose satisfaction may involve paths that are not simple. For example, a user may want to specify that each traversal of a low-security channel is followed by a visit to a check-sum node. A well-studied class of tasks that involve paths that need not be simple are these associated with a specific length, such as patrols in a geographical region. Several communication protocols are based on the fact that a message must pass a pre-defined length before reaching its destination, either for security reasons (e.g., in *Onion routing*, where the message is encrypted in layers [27]) or for marketing purposes (e.g., advertisement spread in social networks). In addition, tasks of a pre-defined length are the components of *proof-of-work* protocols that are used to deter denial of service attacks and other service abuses such as spam (e.g., [15]), and of several protocols for sensor networks [7]. The introduction of multiset corresponds to strategies that are not necessarily simple paths [5].

In *production systems* or in *planning*, a system is modeled by a network whose nodes correspond to configurations and whose edges correspond to actions performed by resources. Users have tasks, that need to be fulfilled by taking sequences of actions. This setting corresponds to an MCG in which the strategies of the players are multisets of actions that fulfill their tasks, which indeed often involve repeated execution of actions [13]; for example "once the arm is up, do not put it down until the block is placed". Also, multiset games can model *preemptive scheduling*, where

the processing of a job may split in several feasible ways among a set of machines.

Our last example, which we are going to study in detail, is *synthesis form component libraries*. A central problem in formal methods is synthesis [26], namely the automated construction of a system from its specification. In real life, hardware and software systems are rarely constructed from scratch. Rather, a system is typically constructed from a library of components by *gluing* components from a library (allowing multiple uses) [23]. For example, when designing an internet browser, a designer does not implement the TCP protocol but uses existing implementations as black boxes. The library of components is used by multiple users simultaneously, and the usages are associated with costs. The usage cost can either decrease with load (e.g., when the cost of a component represents its construction price, the users of a component share this price) as was studied in [4], or increase with load (e.g., when the components are processors and a higher load means slower performance). The later scenario induces an instance of an MCG.

Let us demonstrate the intricacy of the multiset setting with the question of the existence of a *pure Nash equilibrium* (PNE). That is, whether each instance of the game has a profile of pure strategies that constitutes a PNE – a profile such that no player can decrease his cost by unilaterally deviating from his current strategy. By [28], classical congestion games are potential games and thus always have a PNE. Moreover, by [19], in a symmetric congestion game, a PNE can be found in polynomial time. As we show in Example 1 below, a PNE might not exist in an MCG even in a symmetric two-player game over identical resources.

Example 1: Consider the following symmetric MCG with two players and three resources: a, b, and c. The players' strategy space is $\{a, a, b\}$ or $\{b, b, c\}$ or $\{c, c, a\}$. That is, a player needs to access some resource twice and the (cyclically) consequent resource once. The latency function of all three resources is the same, specifically, $f_a(\ell) = f_b(\ell) = f_c(\ell) = \ell^2$. The players' costs in all possible profiles are given in Table 1. We show that no PNE exists in this game. Assume first that the two players select distinct strategies, w.l.o.g. $\{a, a, b\}$ and $\{b, b, c\}$. In this profile, a is accessed twice, b is accessed three times, and c is accessed once. Thus, every access of a, b and c costs $4, 9$ and 1 respectively. The cost of Player 1 is $8 + 9 = 17$, while the cost of Player 2 is $18 + 1 = 19$. By deviating to $\{c, c, a\}$, the cost of Player 2 will reduce to 17 (while the cost of Player 1 will increase to 19). Thus, no PNE in which the players select different strategies exists. If the player select the the same strategy, then one resource is accessed 4 times, and one resource is accessed twice, implying that the cost of both players is $2 \cdot 16 + 1 \cdot 4 = 36$, and any deviation is profitable. We conclude that no PNE exists in the game. □

We study and answer the following questions in general and for various classes of multiset congestion games (for formal definitions, see Section 2): (*i*) Existence

	$\{a,a,b\}$	$\{b,b,c\}$	$\{c,c,a\}$
$\{a,a,b\}$	$36,36$	$19,17$	$17,19$
$\{b,b,c\}$	$17,19$	$36,36$	$19,17$
$\{c,c,a\}$	$19,17$	$17,19$	$36,36$

Table 1: Players costs. Each entry describes the cost of Player 1 followed by the cost of Player 2.

of a PNE. (*ii*) An analysis of *equilibrium inefficiency*. A *social optimum* (SO) of the game is a profile that minimizes the total cost of the players; thus, the one obtained when the players obey some centralized authority. It is well known that decentralized decision-making may lead to solutions that are sub-optimal from the point of view of society as a whole. We quantify the inefficiency incurred due to selfish behavior according to the *price of anarchy* (PoA) [22] and *price of stability* (PoS) [3] measures. The PoA is the worst-case inefficiency of a PNE (that is, the ratio between the cost of a worst PNE and the SO). The PoS is the best-case inefficiency of a Nash equilibrium (that is, the ratio between the cost of a best PNE and the SO). (*iii*) *Computational complexity* of finding a PNE.

Before we turn to describe our results, let us review related work. *Weighted* congestion games (WCGs, for short), introduced in [25], are congestion games in which each player i has a *weight* $w_i \in \mathbb{N}$, and his contribution to the load of the resources he uses as well as his payments are multiplied by w_i. WCGs can be viewed as a special case of MCGs, where each resource in a strategy for Player i repeats w_i times. A different extension of WCGs in which players may use a resource more than once is *integer splittable WCGs* [24, 30]. These games model the setting in which a player has a number (integer) of tasks he needs to perform and can split them among the resources. For example, in the network setting, a player might need to send $\ell \in \mathbb{N}$ packets from vertex s to t. He can send the packets on different paths, but a packet cannot be split. MCGs are clearly more general than WCGs and integer splittable WCGs – the ability to repeat each resource a different number of times lead to a much more complex setting. Thus, it is interesting to compare our results with these known for these games.

It is shown in [17, 21] that the existence of a PNE in WCGs depends on the latency function: when the latency functions are either affine or exponential, WCGs are guaranteed to admit a PNE, whereas WCGs with a polynomial latency function need not have a PNE. In [24], the author shows that PNE always exists when the latency functions are linear using a *potential function* argument. This argument fails when the latency functions are convex, but [30] are still able to show that there is

always a PNE in these games. We are able to show that the exact potential function of [17] applies also to (the much richer) affine MCGs (that is, MCGs with a affine latency function), and thus they always admit a PNE. As demonstrated in Example 1, very simple MCGs with quadratic latency functions might have no PNE.

We turn on to results in the front of equilibrium inefficiency. In congestion games with affine latency functions, both the PoA and PoS measures are well understood. It was shown in [12] that PoS $\geqslant 1 + \frac{1}{\sqrt{3}} \approx 1.577$ and is at most 1.6. A tight upper bound was later shown in [10]. Also, PoA $= \frac{5}{2}$ [12]. Going one step towards our setting to the study of affine WCGs, [6] shows that PoA $= 1 + \phi$, where $\phi \approx 1.618$ is the golden ratio. The PoS question is far from being settled. Only recently, [9] shows a first upper bound of 2 for PoS in linear WCGs, which is a subclass of affine WCGs. As far as we know, the only lower bound that is known for affine WCGs is the lower bound from the unweighted setting. So there is a relatively large gap between the upper- and lower-bounds for the PoS in these games.

We bound the potential function in order to show that every affine MCG G has PoS$(G) < 2$. This improves and generalizes the result in [9]. Our most technically-challenging result is the PoS lower-bound proof, which involves the construction of a family $\mathcal{G}$ of linear MCGs. Essentially, the game $G_k \in \mathcal{G}$ is parameterized by the number of players and defined recursively. The use of multisets enables us to to define a game in which, although the sharing of resources dramatically changes between its profiles, the cost a player pays is equal in all of them. For $k = 17$ we obtain that PoS$(G_{17}) > 1.631$. This is the first lower bound in these models that exceeds the 1.577 lower bound in congestion games. Finally, the PNE in $\mathcal{G}$ is achieved with dominant strategies, so our bound holds for stronger equilibrium concepts.

As for the PoA, we show that MCGs with latency functions that are polynomials of degree at most d have PoA $= \Phi_d^{d+1}$, where Φ_d is the unique nonnegative real solution to $(x+1)^d = x^{d+1}$. Observe that Φ_d is a natural generalization of the golden ratio to higher degrees. Specifically, $\Phi_1 = \phi$. For the upper bound, we adjust the upper-bound proof of [2] to our setting. We show a simplified matching lower bound; a simple two-player MCG with only two resources and latency functions of the form $f(\ell) = \ell^d$. For general latency functions we show that the PoA can be arbitrarily high.

We turn to study the application of synthesis from component libraries by multiple players. Recall that in this application, different users synthesize systems from components. A component may be used in several systems and may be used several times in a system. The quality of a system depends on the load on its components. This gives rise to an MCG, which we term a *component library game* (CLG, for short). On the one hand, a CLG is a special case of MCG, so one could expect

positive results about MCGs to apply to CLGs. On the other hand, while in MCGs the strategies of the players are given explicitly by means of multisets of resources, in CLGs the strategies of the players are given symbolically by means of a specification deterministic finite automaton – the one whose language has to be composed from the library's components.

We prove that every MCG has a corresponding CLG, implying that negative results for MCGs apply to CLGs. Moreover, we show that the succinctness of the presentation of the strategies makes decision problems about MCGs more complex in the setting of CLGs. We demonstrate this by studying the complexity of the *best-response* problem – deciding whether a player can benefit from a unilateral deviation from his strategy, and the problem of deciding whether a PNE exists in a given game. For the best-response problem, which is in P for MCGs, we prove NP-completeness for CLGs. The problem of deciding the existence of a PNE is known to be strongly NP-complete for weighted symmetric congestion games. For network congestion games with player specific cost functions, this problem is NP-complete for arbitrary networks, while a PNE can be found efficiently for constant size networks [1]. We provide a simpler hardness proof for MCGs, which is valid also for a constant number of resources, and we show that for CLGs the problem is Σ_2^P-complete. As good news, we are able to prove a "small-design property" for CLGs, which bounds the number of strategies that one needs to consider and enables us to lift to CLGs the positive results for MCGs with linear latency functions. Thus, such CLGs always have a PNE and their PoS is at most 2.

Due to space constraints, some proofs and examples are given in the appendix.

2 Preliminaries

A *multiset* over a set E of elements is a generalization of a subset of E in which each element may appear more than once. For a multiset A over E and an element $e \in E$, we use $A(e)$ to denote the number of times e appears in A, and use $e \in A$ to indicate that $A(e) \geqslant 1$. When describing multisets, we use e^m, for $m \in \mathbb{N}$, to denote m occurrences of e.

A *multiset congestion game* (MCG) is a tuple $G = \langle K, E, \{\Sigma_i\}_{i \in K}, \{f_e\}_{e \in E} \rangle$, where $K = \{1, \ldots, k\}$ is a set of players, E is a set of *resources*, for every $1 \leqslant i \leqslant k$, the *strategy space* Σ_i of Player i is a collection of multisets over E, and for every resource $e \in E$, the *latency function* $f_e : \mathbb{N} \to \mathbb{R}$ is a non-decreasing function. The MCG G is an *affine* MCG if for every $e \in E$, the latency function f_e is affine, i.e., $f_e(x) = a_e x + b_e$, for non-negative constants a_e and b_e. Similarly, we say that G is a *linear* MCG if it is affine and for $e \in E$ we have $b_e = 0$. We assume w.l.o.g. that

for $e \in E$ we have $a_e \geqslant 1$. Classical congestion games are a special case of MCGs where the players' strategies are sets of resources. Weighted congestion games can be viewed as a special case of MCGs, where for every $1 \leqslant i \leqslant k$, multiset $s_i \in \Sigma$ and $e \in s_i$, we have $s_i(e) = w_i$.

A profile of a game G is a tuple $P = \langle s_1, s_2, \ldots, s_k \rangle \in (\Sigma_1 \times \Sigma_2 \times \ldots \times \Sigma_k)$ of strategies selected by the players. For a resource $e \in E$, we use $L_{e,i}(P)$ to denote the number of times e is used in P by Player i. Note that $L_{e,i}(P) = s_i(e)$. We define the *load* on e in P, denoted $L_e(P)$, as the number of times it is used by all players, thus $L_e(P) = \sum_{1 \leqslant i \leqslant k} L_{e,i}(P)$ [1].

In classical congestion games, all players that use a resource e pay $f_e(\ell)$, where ℓ is the number of players that use e. As we formalize below, in MCGs, the payment of a player for using a resource e depends on the number of times he uses it. Given a profile P, a resource $e \in E$, and $1 \leqslant i \leqslant k$, the *cost of e for Player i in P* is $cost_{e,i}(P) = L_{e,i}(P) \cdot f_e(L_e(P))$. That is, for each of the $L_{e,i}(P)$ uses of e, Player i pays $f_e(L_e(P))$. The cost of Player i in the profile P is then $cost_i(P) = \sum_{e \in E} cost_{e,i}(P)$ and the cost of the profile P is $cost(P) = \sum_{1 \leqslant i \leqslant k} cost_i(P)$. We also refer to the cost of a resource e in P, namely $cost_e(P) = \sum_{i \in K} cost_{e,i}(P)$.

Consider a game G. For a profile P, player $i \in K$, and a strategy $s'_i \in \Sigma$ for Player i, let $P[i \leftarrow s'_i]$ denote the profile obtained from P by replacing the strategy for Player i by s'_i. A profile P is a *pure Nash equilibrium* (PNE) if no Player i can benefit from unilaterally deviating from his strategy in P to another strategy; i.e., for every player i and every strategy $s'_i \in \Sigma$ it holds that $cost_i(P[i \leftarrow s'_i]) \geqslant cost_i(P)$.

We denote by OPT the cost of a social-optimal solution; i.e., $OPT = \min_P cost(P)$. It is well known that decentralized decision-making may lead to sub-optimal solutions from the point of view of society as a whole. We quantify the inefficiency incurred due to self-interested behavior according to the *price of anarchy* (PoA) [22] and *price of stability* (PoS) [3] measures. The PoA is the worst-case inefficiency of a Nash equilibrium, while the PoS measures the best-case inefficiency of a Nash equilibrium. Formally,

Definition 2.1 *Let $\mathcal{G}$ be a family of games, and let G be a game in $\mathcal{G}$. Let $\Upsilon(G)$ be the set of Nash equilibria of the game G. Assume that $\Upsilon(G) \neq \varnothing$.*

- *The* price of anarchy *of G is the ratio between the* maximal *cost of a PNE and the social optimum of G. That is, $PoA(G) = \max_{P \in \Upsilon(G)} cost(P)/OPT(G)$. The* price of anarchy *of the family of games $\mathcal{G}$ is $PoA(\mathcal{G}) = sup_{G \in \mathcal{G}} PoA(G)$.*

[1] Since our strategies are multisets, we have that $s_i(e)$, for all i and e, is an integer. Our considerations, however, are independent of this, thus all our results are valid also for games in which strategies might include fractional demands for resources. In non-splittable (atomic) games, the players must select a single strategy, even if fractional demands are allowed.

- *The* price of stability *of G is the ratio between the* minimal *cost of a PNE and the social optimum of G. That is, $PoS(G) = \min_{P \in \Upsilon(G)} cost(P)/OPT(G)$. The* price of stability *of the family of games $\mathcal{G}$ is $PoS(\mathcal{G}) = sup_{G \in \mathcal{G}} PoS(G)$.*

3 Existence of a Pure Nash Equilibrium

As demonstrated in Example 1, MCGs are less stable than weighted congestion games:

Theorem 3.1 *There exists a symmetric two-player MCG with identical resources and quadratic latency function that has no PNE.*

On the positive side, we show that a PNE exists in all MCGs with affine latency functions. We do so by showing that an exact potential function exists, which is a generalization of the one in [9, 18].

Theorem 3.2 *Affine MCGs are potential games.*

Proof: For a profile P and a resource $e \in E$, define

$$\Phi_e(P) = a_e \cdot \Big(\sum_{i=1}^{k} \sum_{j=i}^{k} L_{e,i}(P) \cdot L_{e,j}(P) \Big) + \Big(b_e \cdot \sum_{i=1}^{k} L_{e,i}(P) \Big).$$

Also, $\Phi(P) = \sum_{e \in E} \Phi_e(P)$. In the appendix, we prove that Φ is an exact potential function. □

The negative result in Theorem 3.1 gives rise to the decision problem ∃PNE; given an MCG, decide whether it has a PNE. Being a generalization of WCGs, the hardness results known for WCGs imply that ∃PNE is NP-hard [14]. Using the richer definition of MCGs, we show below a much simpler hardness proof. We also show hardness for games with a constant number of resources, unlike congestion games with user-specific cost functions [1].

Theorem 3.3 *Given an instance of an MCG, it is strongly NP-complete to decide whether the game has a PNE, as well as to find a PNE given that one exists. For games with a constant number of resources, the problems are NP-Complete.*

Remark 3.3.1: In *splittable (non-atomic) games*, each player can split his task among several strategies. This can be seen as if each player is replaced by $M \to \infty$ identical players all having the same strategy space scaled by $1/M$. This model suits several applications, in particular planning of preemptive production. Splittable games are well-understood in classical and weighted congestion games [29, 8]. In Appendix B we define the corresponding MCG and show that the positive PNE-existence result, known for weighted congestion games, carry over to games with multisets of resources. □

4 Equilibrium Inefficiency in MCGs

4.1 The Price of Stability

The PoS problem in affine congestion games is settled: [12, 10] show that PoS $= 1 + \frac{1}{\sqrt{3}} \approx 1.577$. For affine WCGs, the problem was open for a long time, and only recently progress was made by [9], who showed that PoS $\leqslant 2$ for linear WCGs. As far as we know, there is no known lower bound for linear WCGs that exceeds the 1.577 bound for unweighted games. We show that every affine MCG G has $\text{PoS}(G) < 2$. Thus, we both improve the result to include affine functions, tighten the bound, and generalize it. For the lower bound, we show a family of linear MCGs $\mathcal{G}$ that has $\text{PoS}(\mathcal{G}) > 1.631$. We start with the upper bound.

Theorem 4.1 *Every affine MCG G has $PoS(G) < 2$.*

Proof: Consider an affine MCG G and a profile P. It is not hard to see that for the potential function Φ that is presented in Theorem 3.2 we have $\Phi(P) \leqslant cost(P)$. Moreover, for $e \in E$ we have $2\Phi_e(P) = cost_e(P) - a_e \sum_{1 \leqslant i \leqslant k} L_{e,i}^2(P) - b_e \sum_{1 \leqslant i \leqslant k} L_{e,i}(P)$. Thus, $\Phi(P) > \frac{1}{2} cost(P)$. The theorem follows using standard techniques: $cost(O) \geqslant \Phi(O) \geqslant \Phi(N) > \frac{1}{2} cost(N)$, where O is the social optimum and N is a PNE that is reached from O by a sequence of best-respond moves of the players. Then, $PoS(G) \leqslant \frac{cost(N)}{cost(O)} < 2$. The details of the proof can be found in the appendix. □

Note that while the PoS can get arbitrarily close to 2, it is strictly smaller than 2 for every game instance. The proof in [9], on the other hand, only shows PoS $\leqslant 2$ for the family of affine MCGs, and our result does not improve this bound.

For the lower bound, we show a family of linear MCG $\mathcal{G} = \{G_k\}_{k \geqslant 2}$ that are parameterized by the number of players. Using a computerized simulation, we obtain that for the game with 17 players, we have $\text{PoS}(G_{17}) > 1.631$. We leave open the problem of calculating the exact value the PoS tends to as the number of players increases. The graph depicted in Figure 3 in the appendix, of $\text{PoS}(G_k)$ as a function of k, hints that the answer is only slightly higher than 1.631.

The PNE in the games in the family is achieved with dominant strategies, and thus it is resistant to stronger types of equilibria.

Theorem 4.2 *There is a linear MCG G with $PoS(G) > 1.631$.*

Proof: We define a family of games $\{G_k\}_{k \geqslant 2}$ as follows. The game G_k is played by k players, thus $K_k = \{1, \ldots, k\}$. For Player 1, all strategies $\Sigma_1^k = \{O_1^k\}$ consists of a single multiset. For ease of presentation we sometimes refer to O_1^k as N_1^k. For $i \geqslant 2$, the strategy space of Player i consists of two multisets, $\Sigma_i^k = \{O_i^k, N_i^k\}$. We

define G_k so that for all $k \geqslant 2$, the profile $\bar{O}_k = \langle O_1^k, \ldots, O_k^k \rangle$ is the social optimum and the profile $\bar{N}_k = \langle N_1^k, \ldots, N_k^k \rangle$ is the only PNE.

When describing the games in the family, we partition the resources into *types* and describe a multiset as a collection of triples. A triple $\langle t, y, l \rangle$ stands for y different resources of type t, each appearing l times. For example, $\{\langle a, 2, 1\rangle, \langle b, 1, 3\rangle, \langle c, 2, 2\rangle\}$ stands for the multiset $\{a_1, a_2, b_1, b_1, b_1, c_1, c_1, c_2, c_2\}$. In all games and resources, there are two types of latency functions; the identity function, or identity *plus* epsilon, where the second type of function are linear functions of the form $f(x) = (1 + \epsilon) \cdot x$, for some $\epsilon > 0$. The latency function of resources of the same type is the same, and we use the terms "a has identity latency" and "b has identity plus ϵ latency" to indicate that all the resources a' of type a have $f_{a'}(j) = j$ and all the resources b' of type b have $f_{b'}(j) = (1 + \epsilon) \cdot j$, for all numbers j of uses.

The definition of G_k is complicated and we start by describing the idea in the construction of G_2 and G_3. In the appendix we also describe G_4. We start by describing G_2. The game G_2 is defined with respect to two types of resources, a and b, with identity and identity plus ϵ latency, respectively. We define Player 1's strategy space $\Sigma_1^2 = \{O_1^2\}$ and Player 2's strategy space $\Sigma_2^2 = \{O_2^2, N_2^2\}$, with $O_1^2 = N_2^2 = \langle a, 2, 1\rangle$ and $O_2^2 = \langle b, 1, 2\rangle$. That is, $\Sigma_1^2 = \{\{a_1, a_2\}\}$ and $\Sigma_2^2 = \{\{a_1, a_2\}, \{b_1, b_1\}\}$. Clearly, the profile $\bar{N}_2 = \langle O_1^2, N_2^2 \rangle$ is the only PNE in G_2.

We continue to describe G_3. The game G_3 is defined with respect to four types of resources, a, b, c^1 and c^2, where b has identity plus ϵ latency, c^1 has identity plus ϵ' latency, and the other resources have identity latency. Let $x_3 = 3! = 6$. We define $\Sigma_1^3 = \{O_1^3\}$, $\Sigma_2^2 = \{O_2^3, N_2^3\}$, and $\Sigma_3^3 = \{O_3^3, N_3^3\}$, with $O_1^3 = N_2^3 = \langle a, x_3, 1\rangle$, $O_2^3 = \langle b, \frac{x_3}{2}, 2\rangle$, $O_3^3 = \{\langle c^1, \frac{x_3}{3}, 3\rangle, \langle c^2, \frac{x_3}{2}, 1\rangle\}$, and $N_3^3 = \{\langle b, \frac{x_3}{2}, 1\rangle, \langle a, x_3, 1\rangle\}$. We claim that $\bar{N}_3 = \langle O_1^3, N_2^3, N_3^3 \rangle$ is the only PNE. Our goal here is not to show a complete proof, but to demonstrate the idea of the construction. It is not hard to see that Player 2 deviates to N_2^3 from the profile $\bar{O}_3 = \langle O_1^3, O_2^3, O_3^3 \rangle$, Player 3 deviates from the resulting profile $\bar{N}_3 = \langle O_1^3, N_2^3, N_3^3 \rangle$. The crux of the construction is to keep Player 2 from deviating back from $\bar{N}_3$. Note that since Player 3 uses the b-type resources once in $\bar{N}_3$, when Player 2 deviates from N_2^3 to O_2^3, their load increases to 3. Thus, $cost_2(\bar{N}_3[2 \leftarrow O_2^3]) = 3(3 \cdot 2 \cdot (1 + \epsilon)) > 6(3 \cdot 1) = cost_2(\bar{N}_3)$ and the deviation is not beneficial.

We define the game G_k, for $k \geqslant 2$, as follows. Let $x_k = k!$. Player 1's strategy space consists of a single multiset $O_1^k = \langle e_{1,1}, x_k, 1\rangle$. For $2 \leqslant i \leqslant k$, assume we have defined the strategies and resources for players $1, \ldots, i-1$. We define Player i's strategies as follows. We start with the multiset N_i^k, which does not introduce new resources. We define $N_i^k = \cup_{1 \leqslant j \leqslant i-1} \{\langle t, x, 1\rangle : \langle t, x, l\rangle \subseteq O_i^k\}$. The definition of O_i^k is more involved, but the idea is simple. We define O_i^k so that it satisfies two properties. First, O_i^k uses new resources. That is, for every $1 \leqslant j \leqslant i-1$, both

$O_i^k \cap O_j^k = \varnothing$ and $O_i^k \cap N_j^k = \varnothing$. Consider the profile P_i in which, for every $1 \leqslant j < i$, Player j uses N_j^k and, for every $i \leqslant l \leqslant k$, Player l uses O_l^k. We define O_i^k so that when all resources have identity latency, $cost_i(P_i) = cost_i(P_i[i \leftarrow N_i^k])$. For every multiset $\langle e_{j,a}, x_{j,a}, 1\rangle$ in N_i^k, which we have just defined, we introduce a multiset $\langle e_{i,b}, x_{i,b}, l_{i,b}\rangle$ in O_i^k that uses new resources, where b is a unique index that is arbitrarily chosen, and x_b^i and l_b^i are defined as follows. Let $l = |\{j : e_{j,a} \in N_j^k\}|$. We define $l_{i,b} = l + 1$ and $x_{i,b} = x_{j,a}/l_{i,b}$. Since O_i^k uses new resources, showing the first property is easy. In the appendix we show it satisfies a much stronger property.

Claim 4.2.1: Consider $k \in \mathbb{N}$, a profile P in G_k, and $1 < i \leqslant k$. Assume Player i plays O_k^i in P. When the latency functions are identity, we have $cost_i(P) = cost_i(P[i \leftarrow N_k^i])$.

To complete the construction, we define the latency functions so that for every $2 \leqslant i \leqslant k$, we have that $e_{i,1}$-type resources have identity plus ϵ_i latency for $0 < \epsilon_2 < \ldots < \epsilon_k$. By Claim 4.2.1 there are such values that make N_i^k a dominant strategy for Player i. Thus, the only PNE in G_k, for $k \geqslant 2$, is the profile $\bar{N}_k = \langle O_1^k, N_2^k, \ldots, N_k^k\rangle$. Next, we identify the social optimum.

Claim 4.2.2: The profile $\bar{O}_k = \langle O_1^k, \ldots, O_k^k\rangle$ is the social optimum.

Once we identify $\bar{O}_k$ as the social optimum and $\bar{N}_k$ as the only PNE, the calculation of the PoS boils down to calculating their costs, which we do using a computer. Specifically, we have $\text{PoS}(G_{17}) = 1.6316$, and we depict the values of G_k, for $2 \leqslant k \leqslant 17$, in Figure 3 in the appendix. □

Remark 4.2.1: We conjecture that the correct value for the PoS is closer to our lower bound of 1.631 rather than to the upper bound of 2. In the appendix we show a more careful analysis of the potential function than the one in Theorem 4.1 that shows that for every linear MCG G we have $PoS(G) \leqslant 2 - \frac{\sum_{e \in E} \sqrt{cost_e(N_G)}}{cost(O_G)}$, where N_G and O_G denote the cheapest PNE and the social optimum of G, respectively. Also, we show that for every $n \geqslant 2$, for the MCG G_n that is described in Theorem 4.2, the inequality in the expression is essentially an equality. □

Remark 4.2.2: We can alter the family in Theorem 4.2 to have quadratic latency functions instead of identity functions. Although Claim 4.2.1 does not hold in the altered family, a computerized simulation shows that the N strategies are still dominant strategies. Also, using a computerized simulation, we show that the PoS for G_{15} is 2.399, higher than the upper bound of 2.362 for congestion games, which is shown in [9, 11]. □

4.2 The Price of Anarchy

In this section we study the PoA for MCGs. We start with MCGs with polynomial latency functions and show that the upper bound proven in [2] for WCGs can be adjusted to our setting. Being a special case of MCGs, the matching lower bound for WCGs applies too. Still, we present a different and much simpler lower-bound example, which uses a two-player singleton MCG. In a singleton game, each strategy consists of (multiple accesses to) a single resource. Finally, when the latency functions are not restricted to be polynomials, we show that the PoA is unbounded, and it is unbounded already in a singleton MCG with only two players.

We start by showing that the PoA in polynomial MCGs is not higher than in polynomial WCGs. The proof adjusts the one known for WCGs [2] to our setting. For $d \in \mathbb{N}$, we denote by $\mathcal{P}_d$ the set of polynomials of degree at most d.

Theorem 4.3 *The PoA in MCGs with latency functions in $\mathcal{P}_d$ is at most Φ_d^{d+1}, where Φ_d is the unique nonnegative real solution to $(x+1)^d = x^{d+1}$.*

Next, we show a matching lower bound that is stronger and simpler than the one in [2].

Theorem 4.4 *For $d \in \mathbb{N}$, the PoA in two-player singleton MCG with latency functions in $\mathcal{P}_d$ is at least Φ_d^{d+1}.*

Proof: Let $d \in \mathbb{N}$. Consider the two-player singleton MCG G with resources $E = \{e_1, e_2\}$, strategy spaces $\Sigma_1 = \{e_1^x, e_2^y\}$ and $\Sigma_2 = \{e_1^y, e_2^x\}$, and for $\ell \in \mathbb{R}$, we define the latency functions $f_{e_1}(\ell) = f_{e_2}(\ell) = \ell^d$. We define $x = \Phi_d$ and $y = 1$. Since $x > y$ the social optimum is attained in the profile $\langle e_1^y, e_2^y \rangle$ and its cost is $2y^d = 2$. Recall that in MCGs, the players' strategies are multisets. In particular, x should be a natural number. To fix this, we consider a family of MCGs in which the ratio between x and y tends to the ratio above.

We claim that the profile $N = \langle e_1^x, e_2^x \rangle$ is a PNE. This would imply that $\mathrm{PoA}(G) = \frac{2x^{d+1}}{2} = \Phi_d^{d+1}$, which would conclude the proof. We continue to prove the claim. The cost of a player in N is $x \cdot x^d = x^{d+1}$ and by deviating, the cost changes to $y \cdot (x+y)^d = (x+1)^d$. Our definition of x implies that $x^{d+1} = (x+1)^d$. Thus, the cost does not change after deviating. Since the players are symmetric, we conclude that the profile N is a PNE, and we are done. □

Finally, by taking variants with factorial latency functions to the game described in Theorem 4.4, we are able to increase the PoA in an unbounded manner.

Theorem 4.5 *The PoA in two-player MCGs is unbounded.*

5 Synthesis from Component Libraries

In this section we describe the application of MCGs in synthesis from component libraries. As briefly explained in Section 1, in this application, different users synthesize systems by gluing together components from a component library. A component may be used in several systems and may be used several times in a system. The performance of a component and hence the system's quality depends on the load on it. We describe the setting in more detail, formalize it by means of MCGs, and relate to the results studied in earlier sections.

Today's rapid development of complex and safety-critical systems requires reliable verification methods. In *formal methods*, we reason about systems and their specifications by solving mathematical questions about their models. A central problem in formal methods is synthesis, namely the automated construction of a system from its specification. In real life, systems are rarely constructed from scratch. Rather, a system is typically constructed from a library of components by *gluing* components from the library [23]. In this setting, the input to the synthesis problem is a specification and a library of components, and the goal is to construct from the components a system that exhibits exactly the behaviors specified in the specification.

Remark 5.0.1: The above setting corresponds to *closed systems*, whose behavior is independent of their environment. It is possible to generalize the definitions to *open systems*, which interact with their environment. In [4], we studied both the closed and open settings in the context of cost-sharing (rather than congestion) games. The technical challenges that have to do with the system being open are orthogonal to these that arise from the congestion effects, and on which we focus in this work. □

In our setting, we use deterministic finite automata (DFAs, for short) to model the specification and use *box-DFAs* to model the components in the library. Formally, a DFA is $\mathcal{A} = \langle \Sigma, Q, \delta, q_0, F \rangle$, where Σ is an alphabet, Q is a set of states, $\delta : Q \times \Sigma \to Q$ is a partial transition function, $q_0 \in Q$ is an initial states, and $F \subseteq Q$ is a set of accepting states. The *run* of $\mathcal{A}$ on a word $w = w_1, \ldots w_n \in \Sigma^*$ is the sequence of states $r = r_0, r_1, \ldots, r_n$ such that $r_0 = q_0$ and for every $0 \leqslant i \leqslant n-1$, we have $r_{i+1} = \delta(r_i, w_{i+1})$. Now, a box-DFA $\mathcal{B}$ is a DFA augmented with a set of *exit states*. When a run of $\mathcal{B}$ reaches an exit state, it moves to another box-DFA, as we formalize below.

The input to the synthesis from component libraries problem is a specification DFA $\mathcal{S}$ over an alphabet Σ and a library of box-DFAs components $\mathcal{L} = \{\mathcal{B}_1, \ldots, \mathcal{B}_n\}$. The goal is to produce a *design*, which is a recipe to compose the components from $\mathcal{L}$ to a DFA. A design is correct if the language of the system it induces coincides

with that of the specification.

Intuitively, the design can be thought of as a scheduler; it passes control between the different components in $\mathcal{L}$. When a component $\mathcal{B}_i$ is in *control*, it reads letters in Σ, visits the states of $\mathcal{B}_i$, follows its transition function, and if the run terminates, it is accepting iff it terminates in one of $\mathcal{B}_i$'s accepting states. A component relinquishes control when the run reaches one of its exit states. It is then the design's duty to choose the next component, which gains control through its initial state.

Formally (see an example in Figure 1), a *transducer* is a DFA that has, in addition to the input alphabet that labels the transitions, also an output alphabet that labels the states. Also, a transducer has no rejecting states. Let $[n] = \{1, \ldots, n\}$. A design is a transducer $\mathcal{D}$ whose input alphabet is the set $\mathcal{E}$ of all exit states of all the components in $\mathcal{L}$ and whose output alphabet is $[n]$. We can think of $\mathcal{D}$ as running beside the components. When a component reaches an exit state e, then $\mathcal{D}$ reads the input letter e, proceeds to its next state, and outputs the index of the component to gain control next. Note that the components in the library are black boxes: the design $\mathcal{D}$ does not read the alphabet Σ of the components and has no information about the states that the component visits. It only sees which exit state have been reached. Given a library $\mathcal{L}$ and a design $\mathcal{D}$, their *composition* is a DFA $\mathcal{A}_{\mathcal{L},\mathcal{D}}$ obtained by composing the components in $\mathcal{L}$ according to $\mathcal{D}$. We say that a design $\mathcal{D}$ is *correct* with respect to a specification DFA $\mathcal{S}$ iff $L(\mathcal{A}_{\mathcal{L},\mathcal{D}}) = L(\mathcal{S})$. In the appendix we construct $\mathcal{A}_{\mathcal{L},\mathcal{D}}$ formally.

For example, consider the library $\mathcal{L} = \{\mathcal{B}_1, \mathcal{B}_2\}$ over the alphabet $\Sigma = \{a, b, c\}$, and the design $\mathcal{D}$ that are depicted in Figure 1. We describe the run on the word bc. The component that gains initial control is $\mathcal{B}_1$ as the initial state of $\mathcal{D}$ outputs 1. The run in $\mathcal{B}_1$ proceeds with the letter b to the exit state e_1 and relinquishes control. Intuitively, control is passed to the design that advances with the letter e_1 to the state that outputs 2. Thus, the component $\mathcal{B}_2$ gains control, and it gains it through its initial state. Then, the letter c is read, $\mathcal{B}_2$ proceeds to the exit state e_3 and relinquishes control. The design advances with the letter e_3 to a state that outputs 1, and control is assigned to $\mathcal{B}_1$. Since the initial state of $\mathcal{B}_1$ is rejecting, the word ab is rejected. As a second example, consider the word ab. Again, $\mathcal{B}_1$ gains initial control. After visiting the exit state e_2, control is reassigned to $\mathcal{B}_1$. Finally, after visiting the state e_1, control is assigned to $\mathcal{B}_2$, where the run ends. Since the initial state of $\mathcal{B}_2$ is accepting, the run is accepting.

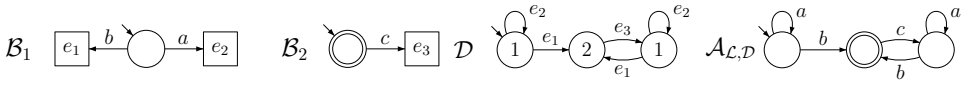


Figure 1: An example of a library $\mathcal{L} = \{\mathcal{B}_1, \mathcal{B}_2\}$, a design $\mathcal{D}$, and the resulting composition $\mathcal{A}_{\mathcal{L},\mathcal{D}}$.

The synthesis problem defined above is aimed at synthesizing correct designs. We now add costs to the setting. A *component library game* (CLG, for short) is a tuple $\langle K, \mathcal{L}, \{\mathcal{S}_i\}_{i\in K}, \{f_{\mathcal{B}}\}_{\mathcal{B}\in\mathcal{L}}\rangle$, where $K = \{1, \ldots, k\}$ is a set of players, $\mathcal{L}$ is a collection of box-DFAs, the objective of Player $i \in K$ is given by means of a specification DFA $\mathcal{S}_i$, and, as in MCGs, the latency function $f_{\mathcal{B}}$ for a component $\mathcal{B} \in \mathcal{L}$ maps the load on $\mathcal{B}$ to its cost with this load. For $i \in K$, the set of strategies for Player i is the set of designs that are correct with respect to $\mathcal{S}_i$. A CLG corresponds to an MCG with a slight difference that there might be infinitely many correct designs. Consider a profile $P = \langle \mathcal{D}_1, \ldots, \mathcal{D}_k\rangle$. For a component $\mathcal{B} \in \mathcal{L}$, we use $L_{\mathcal{B},i}(P)$ to denote the number of times Player i uses $\mathcal{B}$ in P. Recall that each state in the transducer $\mathcal{D}_i$ is labeled by a component in $\mathcal{L}$. We define $L_{\mathcal{B},i}(P)$ to be the number of states in $\mathcal{D}_i$ that are labeled with $\mathcal{B}$. The rest of the definitions are as in MCGs.

We first show that every MCG can be translated to a CLG:

Theorem 5.1 *Consider a k-player MCG G. There is a CLG G' between k players with corresponding profiles. Formally, there is a one-to-one and onto function f from profiles of G to profiles of G' such that for every profiles P in G and Player $i \in [k]$, we have that $cost_i(P) = cost_i(f(P))$.*

Proof: Consider an MCG $\langle K, E, \{\Sigma_i\}_{i\in K}, \{f_e\}_{e\in E}\rangle$. Recall that Σ_i is the set of strategies for Player i that consists of multisets over E. We construct a CLG with alphabet $E \cup \bigcup_{i\in K} \Sigma_i$. For $i \in K$, the specification $\mathcal{S}_i$ for Player i consists of $|\Sigma_i|$ words. Every strategy $s = \{e_1, \ldots, e_n\}$ (allowing duplicates) in Σ_i contributes to $L(S)$ the word $s \cdot e_1 \cdot e_2 \cdot \ldots \cdot e_n$. We construct a library $\mathcal{L}$ with $|E| + \sum_{i\in K} |\Sigma_i|$ components of two types: a *strategy component* $\mathcal{B}_s$ for each $s \in \Sigma_i$ and a *resource component* $\mathcal{B}_e$ for each $e \in E$. In addition, $\mathcal{L}$ contains the component $\mathcal{B}_{acc}$ that is depicted in Figure 2. Intuitively, a correct design must choose one strategy component $\mathcal{B}_s$ and then use the component $\mathcal{B}_e$ the same number of times e appears in s. We continue to describe the components. For $s \in \Sigma_i$, the component $\mathcal{B}_s$ relinquishes control only if the letter s is read. It accepts every word in $L(\mathcal{S}_i)$ that does not start with s. For $e \in E$, the resource component $\mathcal{B}_e$ has an initial state with an e-labeled transition to an exit state. Finally, the latency function for the resource components coincides with latency functions of the resources in the MCG, thus for $e \in E$, we have $f_{\mathcal{B}_e} = f_e$. The other latency functions are $f \equiv 0$. In the appendix we prove that there is a cost-preserving one-to-one and onto correspondence between correct designs with respect to $\mathcal{S}_i$ and strategies in Σ_i, implying the existence of the required function between the profiles. □

Theorem 5.1 implies that the negative results we show for MCGs apply to CLGs:

Corollary 5.2 *There is a CLG with quadratic latency functions with no PNE; for*

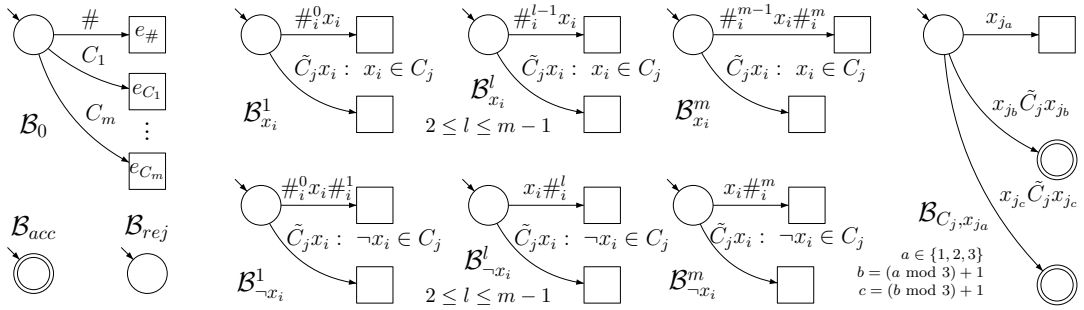


Figure 2: The components in the library $\mathcal{L}$.

CLGs with affine latency functions, we have $PoS(CLG) > 1.631$*; for* $d \in \mathbb{N}$*, the PoA in a two-player singleton MCG with latency functions in* $\mathcal{P}_d$ *is at least* Φ_d^{d+1}.

Remark 5.2.1: We note that the positive results for CLGs with linear latency functions, namely existence of PNE and $PoS(CLG) \leqslant 2$, do not follow immediately from Theorem 3.2, as its proof relies on the fact that an MCG has only finitely many profiles. Since the strategy space of a player might have infinitely many strategies, a CLG might have infinitely many profiles. In order to show that CLGs with linear latency functions have a PNE we need Lemma 5.3 below, which implies that even in games with infinitely many profiles, there is a best response dynamics that only traverses profiles with "small" designs. Such a traversal is guaranteed to reach a PNE as there are only finitely many such profiles. □

Computational complexity We turn to study two computational problems for CLGs: finding a *best-response* and deciding the existence of a PNE. We show that the succinctness of the representation of the objectives of the players in CLGs makes these problems much harder than for MCGs. Our upper bounds rely on the following lemma. The lemma is proven in [4] for cost-sharing games, and the considerations in the proof there applies also for congestion games.

Lemma 5.3 *Consider a library* $\mathcal{L}$*, a specification* $\mathcal{S}$*, and a correct design* $\mathcal{D}$*. There is a correct design* $\mathcal{D}'$ *with at most* $|\mathcal{S}| \cdot |\mathcal{L}|$ *states, where* $|\mathcal{L}|$ *is the number of states in the components of* $\mathcal{L}$*, such that for every component* $\mathcal{B} \in \mathcal{L}$*, the number of times* $\mathcal{D}'$ *uses* $\mathcal{B}$ *is at most the number of times* $\mathcal{D}$ *uses* $\mathcal{B}$*.*

We start with the best-response problem (BR problem, for short): Given an MCG $\mathcal{G}$ between k players, a profile P, an index $i \in K$, and $\mu \in \mathbb{R}$, decide whether Player i has a strategy S_i' such that $cost_i(P[i \leftarrow S_i']) \leqslant \mu$.

Theorem 5.4 *The BR problem for MCGs is in P. For CLGs it is NP-complete, and NP-hardness holds already for games with one player and linear latency functions.*

Proof: Showing that the BR problem is in P for MCGs follows easily from the fact the set of strategies for Player i is given implicitly and calculating the cost for a player in a profile can be done in polynomial time.

The upper bound for CLGs follows from Lemma 5.3, which implies an upper bound on the size of the cheapest correct designs. Since checking whether a design is correct and calculating its cost can both be done in polynomial time, membership in NP follows.

We continue to the lower bound. We describe the intuition of the reduction and the formal definition along with the correctness proof can be found in the appendix. Given a 3SAT formula φ with clauses $C_1, \ldots, C_m$ and variables $x_1, \ldots, x_n$, we construct a library $\mathcal{L}$ and a specification $\mathcal{S}$ such that there is a design $\mathcal{D}$ that costs at most $\mu = nm + m$ iff φ is satisfiable. The library $\mathcal{L}$ consists of an *initial component* $\mathcal{B}_0$, *variable components* $\mathcal{B}^j_{x_i}$ and $\mathcal{B}^j_{\neg x_i}$ for $j \in [m]$ and $i \in [n]$, *clause components* $\mathcal{B}_{C_j, x_{j_k}}$ for $j \in [m]$ and $k \in \{1, 2, 3\}$, and component $\mathcal{B}_{acc}$ and $\mathcal{B}_{rej}$. The components of the library are depicted in Figure 2. The latency function of the variable components is the identity function $f(x) = x$, thus using such a component once costs 1. The latency functions of the other components is the constant function $f \equiv 0$, thus using such components any number of times is free.

Intuitively, a correct design corresponds to an assignment to the variable and must use nm variable components as follows. For $i \in [n]$, either use all the components $\mathcal{B}^1_{x_i}, \ldots, \mathcal{B}^m_{x_i}$ or all the components $\mathcal{B}^1_{\neg x_i}, \ldots, \mathcal{B}^m_{\neg x_i}$ with a single use each. Thus, a correct design implies an assignment $\eta : \{x_1, \ldots, x_n\} \to \{T, F\}$. Choosing $\mathcal{B}^1_{x_i}, \ldots, \mathcal{B}^m_{x_i}$ corresponds to $\eta(x_i) = F$ and choosing $\mathcal{B}^1_{\neg x_i}, \ldots, \mathcal{B}^m_{\neg x_i}$ corresponds to $\eta(x_i) = F$.

Additionally, in order to verify that a correct design corresponds to a satisfying assignment, it must use m clause components and m more variable components as follows. Consider a correct design $\mathcal{D}$, and let $\eta : \{x_1, \ldots, x_n\} \to \{T, F\}$ be the corresponding assignment as described above. For every $j \in [m]$, $\mathcal{D}$ must use a clause component $\mathcal{B}_{C_j, x_i}$, where recall that the clause C_j includes a literal $\ell \in \{x_i, \neg x_i\}$. Using the component $\mathcal{B}_{C_j, x_i}$ requires $\mathcal{D}$ to use a variable component $\mathcal{B}^t_\ell$, for some $t \in [m]$. So, a correct design uses a total of $nm + m$ components with identity latency. If $\eta(\ell) = F$, then $\mathcal{B}^t_\ell$ is already in use and a second use will cost more than 1, implying that the design costs more than $nm + m$. □

The next problem we study is deciding the existence of a PNE. As we show in Theorem 3.3, the problem is NP-complete for MCGs. As we show below, the succinctness of the representation makes this problem harder for CLGs.

Theorem 5.5 *The ∃PNE problem for CLGs is Σ^P_2-complete.*

Proof: The upper bound is easy and follows from Lemma 5.3. For the lower

bound we show a reduction from the complement of *not all equal* $\forall\exists$ *3SAT* (NAE, for short), which is known to be Σ_2^P-complete [16]. An input to NAE is a 3CNF formula φ over variables $x_1, \ldots, x_n, y_1, \ldots, y_n$. It is in NAE if for every assignment $\eta : \{x_1, \ldots, x_n\} \to \{T, F\}$ there is an assignment $\rho : \{y_1, \ldots, y_n\} \to \{T, F\}$ such that every clause in φ has a literal that gets value truth and a literal that gets value false (in η or ρ, according to whether the variable is an x or a y variable). We say that such a pair of assignments $\langle \eta, \rho \rangle$ is *legal* for φ.

Given a 3CNF formula φ, we construct a CLG G with three players such that $\varphi \in$ NAE iff G does not have a PNE. We describe the intuition of the reduction. The details can be found in the appendix. There is a one-to-one correspondence between Player 3 correct designs and assignments to the variables $\{x_1, \ldots, x_n\}$. For an assignment $\eta : \{x_1, \ldots, x_n\} \to \{T, F\}$ we refer to the corresponding correct design by $\mathcal{D}_\eta$. Consider a legal pair of assignments $\langle \eta, \rho \rangle$, and assume Player 3 chooses the design $\mathcal{D}_\eta$. Similarly to the proof of Theorem 5.4, the library contains variable components with identity latency function. We construct the library and the players' objectives so that there is a correct design $\mathcal{D}_\rho$ for Player 1 that uses $mn + 2m$ variable components each with load 1 iff $\langle \eta, \rho \rangle$ is a legal pair for φ. More technically, both $\mathcal{D}_\eta$ and $\mathcal{D}_\rho$ use mn variable components that correspond to the variables $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$, respectively. For every $j \in [m]$, assuming the j-th clause is $\ell_j^1 \vee \ell_j^2 \vee \ell_j^3$, the design $\mathcal{D}_\rho$ must use two additional variable components $\mathcal{B}_{\ell_j^a}^{t_1}$ and $\mathcal{B}_{\ell_j^b}^{t_2}$, for $a \neq b \in \{1, 2, 3\}$ and $t_1, t_2 \in [m]$, which corresponds to η or ρ assigning value true to ℓ_j^a and value false to ℓ_j^b.

Player 1 has an additional correct design $\mathcal{D}_{ALL}$ in which he does not share any components regardless of the other players' choices. Player 2 has two possible designs $\mathcal{D}_A$ and $\mathcal{D}_B$. Assume Player 3 chooses a design $\mathcal{D}_\eta$. We describe the interaction between Player 1 and Player 2. We define the library and the players' objectives so that when Player 1 chooses some design $\mathcal{D}_\rho$, Player 2 prefers $\mathcal{D}_B$ over $\mathcal{D}_A$, thus $cost_2(\langle \mathcal{D}_\rho, \mathcal{D}_A, \mathcal{D}_\eta \rangle) > cost_2(\langle \mathcal{D}_\rho, \mathcal{D}_B, \mathcal{D}_\eta \rangle)$. When Player 2 plays $\mathcal{D}_B$, Player 1 prefers $\mathcal{D}_{ALL}$ over every design $\mathcal{D}_\rho$, thus $cost_1(\langle \mathcal{D}_\rho, \mathcal{D}_B, \mathcal{D}_\eta \rangle) > cost_1(\langle \mathcal{D}_{ALL}, \mathcal{D}_B, \mathcal{D}_\eta \rangle)$. When Player 1 chooses $\mathcal{D}_{ALL}$, Player 2 prefers $\mathcal{D}_A$ over $\mathcal{D}_B$, thus $cost_2(\langle \mathcal{D}_{ALL}, \mathcal{D}_B, \mathcal{D}_\eta \rangle) > cost_2(\langle \mathcal{D}_\eta, \mathcal{D}_A, \mathcal{D}_\eta \rangle)$. Finally, when Player 2 chooses $\mathcal{D}_A$, Player 1 prefers the design $\mathcal{D}_\rho$ iff the pair $\langle \eta, \rho \rangle$ is legal for φ, thus $cost_1(\langle \mathcal{D}_{ALL}, \mathcal{D}_A, \mathcal{D}_\eta \rangle) > cost_1(\langle \mathcal{D}_\rho, \mathcal{D}_A, \mathcal{D}_\eta \rangle)$, for a legal pair $\langle \eta, \rho \rangle$.

Thus, if $\varphi \in$ NAE, then for every assignment η, there is an assignment ρ such that $\langle \eta, \rho \rangle$ is a legal pair. Then, assuming Player 3 chooses a design $\mathcal{D}_\eta$, Player 1 prefers either choosing $\mathcal{D}_{ALL}$ or $\mathcal{D}_\rho$ over every other design, where $\langle \eta, \rho \rangle$ is a legal pair. By the above, there is no PNE in the game. If $\varphi \notin$ NAE, then there is an assignment η such that for every assignment ρ, the pair $\langle \eta, \rho \rangle$ is illegal. Then, the profile $\langle \mathcal{D}_{ALL}, \mathcal{D}_A, \mathcal{D}_\eta \rangle$ is a PNE, and we are done. □

References

[1] H. Ackermann and A. Skopalik. Complexity of pure Nash equilibria in player-specific network congestion games. *Internet Mathematics*, 5(4):321–515, 2008.

[2] S. Aland, D. Dumrauf, M. Gairing, B. Monien, and F. Schoppmann. Exact price of anarchy for polynomial congestion games. *SIAM J. Comput.*, 40(5):1211–1233, 2011.

[3] E. Anshelevich, A. Dasgupta, J. Kleinberg, E. Tardos, T. Wexler, and T. Roughgarden. The price of stability for network design with fair cost allocation. *SIAM J. Comput.*, 38(4):1602–1623, 2008.

[4] G. Avni and O. Kupferman. Synthesis from component libraries with costs. In *Proc. 25th CONCUR*, LNCS 8704, pages 156–172. Springer, 2014.

[5] G. Avni, O. Kupferman, and T. Tamir. Network-formation games with regular objectives. In *Proc. 17th FoSSaCS*, LNCS 8412, pages 119–133. Springer, 2014.

[6] B. Awerbuch, Y. Azar, and A. Epstein. The price of routing unsplittable flow. *SIAM J. Comput.*, 42(1):160–177, 2013.

[7] N. Basilico, N. Gatti, and F. Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *Proc. 8th AAMAS*, 2009.

[8] K. Bhawalkar, M. Gairing, and T. Roughgarden. Weighted congestion games: Price of anarchy, universal worst-case examples, and tightness. In *ESA (2)*, pages 17–28, 2010.

[9] V. Bilò. A unifying tool for bounding the quality of non-cooperative solutions in weighted congestion games. In *WAOA*, pages 215–228, 2012.

[10] I. Caragiannis, M. Flammini, C. Kaklamanis, P. Kanellopoulos, and L. Moscardelli. Tight bounds for selfish and greedy load balancing. *Algorithmica*, 61(3):606–637, 2011.

[11] G. Christodoulou and M. Gairing. Price of stability in polynomial congestion games. In *Proc. 40th ICALP*, pages 496–507, 2013.

[12] G. Christodoulou and E. Koutsoupias. On the price of anarchy and stability of correlated equilibria of linear congestion games. In *ESA*, pages 59–70, 2005.

[13] N. Daniele, F. Guinchiglia, and M.Y. Vardi. Improved automata generation for linear temporal logic. In *Proc. 11th CAV*, LNCS 1633, pages 249–260. Springer, 1999.

[14] J. Dunkel and A.S. Schulz. On the complexity of pure-strategy nash equilibria in congestion and local-effect games. *Mathematics of Operations Research*, 33(4):851–868, 2008.

[15] C. Dwork and M. Naor. Pricing via processing, or, combatting junk mail. In *Proc. CRYPTO*, pages 139–177, 2009.

[16] T. Eiter and G. Gottlob Note on the complexity of some eigenvector problems. Technical Report CD-TR 95/89, Christian Doppler Laboratory for Expert Systems, TU Vienna, 1995.

[17] D. Fotakis, S. Kontogiannis, and P. Spirakis. Selfish unsplittable flows. *Theoretical Computer Science*, 348(2-3):226–239, 2005.

[18] D. Fotakis, S. Kontogiannis, and P. Spirakis. Symmetry in Network Congestion Games: Pure Equilibria and Anarchy Cost. In *Proc. WAOA*, pages 161-175, 2005.

[19] A. Fabrikant, C. Papadimitriou, and K. Talwar. The complexity of pure Nash equilibria. In *Proc. 36th STOC*, pages 604–612, 2004.

[20] M. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. Freeman and Co., 1979.

[21] T. Harks and M. Klimm. On the existence of pure Nash equilibria in weighted congestion games. *Math. Oper. Res.*, 37(3):419–436, 2012.

[22] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. *Computer Science Review*, 3(2):65–69, 2009.

[23] Y. Lustig and M.Y. Vardi. Synthesis from component libraries. *STTT*, 15(5-6):603–618, 2013.

[24] C. Meyers. *Network flow problems and congestion games: complexity and approximation results*. PhD thesis, MIT, 2006.

[25] I. Milchtaich. Congestion games with player-specific payoff functions. *Games and Economic Behavior*, 13(1):111 – 124, 1996.

[26] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *POPL*, pages 179–190, 1989.

[27] M.G. Reed, P.F. Syverson, and D.M. Goldschlag. Anonymous connections and onion routing. *IEEE J. on Selected Areas in Communication*, 1998. Issue on Copyright and Privacy Protection.

[28] R.W. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2:65–67, 1973.

[29] T. Roughgarden and E. Tardos. How bad is selfish routing? *JACM*, 49(2):236–259, 2002.

[30] L. Tran-Thanh, M. Polukarov, A. C. Chapman, A. Rogers, and N. R. Jennings. On the existence of pure strategy nash equilibria in integer-splittable weighted congestion games. In *SAGT*, pages 236–253, 2011.

A Proofs

The function Φ in Theorem 3.2 is an exact potential function: Consider a profile P, and assume Player i deviates from his strategy s_i in P to a strategy s'_i, and let P' be the resulting profile. We claim that $\Phi(P) > \Phi(P')$. Moreover, $\Phi(P) - \Phi(P')$ is exactly the gain of Player i from the deviation.

Consider a transition $e \in E$. Note that for every $j \neq i$, we have $L_{e,j}(P) = L_{e,j}(P')$. Thus, for $j, j' \neq i$, the values $a_e \cdot L_{e,j}(P) \cdot L_{e,j'}(P)$ and $b_e \cdot L_{e,j}(P)$ appear both in $\Phi_e(P)$ and $\Phi_e(P')$. Thus,

$$\Phi_e(P) - \Phi_e(P') =$$

$$= a_e \cdot \sum_{j=1}^{k} L_{e,i}(P) \cdot L_{e,j}(P) + b_e \cdot L_{e,i}(P) - \left(a_e \cdot \big(\sum_{j=1}^{k} L_{e,i}(P') \cdot L_{e,j}(P') \big) + b_e \cdot L_{e,i}(P') \right) =$$

$$= L_{e,i}(P) \cdot \big(a_e \cdot L_e(P) + b_e \big) - L_{e,i}(P') \cdot \big(a_e \cdot L_e(P') + b_e \big) = cost_{e,i}(P) - cost_{e,i}(P').$$

Note that if $L_{e,i}(P) = L_{e,i}(P') = 0$, then $\Phi_e(P) - \Phi_e(P') = 0$. Thus,

$$\Phi(P) - \Phi(P') = \sum_{e \in E} \Phi_e(P) - \Phi_e(P') = \sum_{e \in E} cost_{e,i}(P) - cost_{e,i}(P') = cost_i(P) - cost_i(P').$$

Since we assume Player i improves his cost, we have $cost_i(P) > cost_i(P')$, and we are done. □

Proof of Theorem 3.3: The problem is clearly in NP. For the hardness proof we present a reduction from the 3-*Partition Problem*, which is known to be strongly NP-hard [20]. Given a set of $3m$ integers $\mathcal{A} = \{s_1, s_2, ..., s_{3m}\}$ whose total sum is mS, and for every $1 \leqslant i \leqslant 3m$ it holds that $S/4 < s_i < S/2$, the goal is to decide whether it is possible to partition $\mathcal{A}$ into sets of size 3, each with total sum S.

Given $\mathcal{A}$, define the following MCG G over $3m+2$ players and $m+4$ resources $\{e_1, e_2, \ldots, e_m, z, a, b, c\}$. The latency functions of the resources are $f_{e_1}(x) = \ldots = f_{e_m}(x) = x; f_z(x) = \lceil 2x/S \rceil \cdot S; f_a(x) = \max(0, (x-1)^2); f_b(x) = f_c(x) = x^2$. For every $1 \leqslant i \leqslant 3m$, the strategy space of Player i is $\{\{e_1^{s_i}\}, \ldots, \{e_m^{s_i}\}, \{z^{s_i}, a\}\}$. The additional two players play the no-PNE game introduced in Example 1, thus we have the same strategy space $\{\{a,a,b\}, \{b,b,c\}, \{c,c,a\}\}$. We show that a PNE exists in G if and only if a 3-partition of $\mathcal{A}$ exists.

If a 3-partition of $\mathcal{A}$ exists, then a PNE exists: the first $3m$ players can split between the resources $e_1, \ldots, e_m$, causing load S on each resource, thus, cost $s_i \cdot S$ for each player $1 \leqslant i \leqslant 3m$. A deviation to $\{z, a\}$ will cause cost $s_i \cdot S + 1$ and is therefore not beneficial. Since the latency function of a is lower than the latency functions of b and c, and the first $3m$ players do not cause any load on a, it is easy to verify that the profile $\{\{a,a,b\}$ and $\{c,c,a\}\}$ is stable for the two last players (achieving costs 9 and 12).

If a 3-partition does not exist, then we claim that in every profile, exactly one out of the first $3m$ players selects $\{z^{s_i}, a\}$. If no players selects this strategy, then at least one of the resources $e_1, \ldots, e_m$ has load larger than S and a deviation to $\{z^{s_i}, a\}$ is beneficial for any of the players using this resource. If more than one player selects this strategy, then the load on z is $L_z \geqslant 2s_{min} > S/2$. Combining the facts that $f_z(x) = \lceil 2x/S \rceil \cdot S$ and $L_z > S/2$, we get that $f_z(L_z) \geqslant 2S$, thus, deviating to a resource $e_1, \ldots, e_m$ with load less than $1.5S$ (by averaging argument, at least one such resource exists) is beneficial.

Given that exactly one player selects $\{z, a\}$, we have that the load on a is 1. Thus, the two last players face exactly the game introduced in Example 1 - that has no PNE. We conclude that a PNE exists if and only if a 3-partition exists. Moreover, even if a 3-partition exists, it is strongly NP-hard to find it, and therefore, finding a PNE is strongly NP-hard as well.

In a similar way, it is possible to define a reduction from *Equal-Partition*, in which $2n$ integers should split into two sets of the same cardinality and the same total sum. The induced game will be over six resources: the first two will be assigned the partition items, and the four additional items will have the same role as z, a, b and c in the above reduction. Thus, the hardness proof is valid also for games with a constant number of resources, unlike congestion games with user-specific cost functions [1]. □

Proof of Theorem 4.1: Our proof is based on the potential function defined in [9, 18] (see Theorem 3.2). Consider a affine MCG G. We claim that for every profile P in G we have:

$$\frac{1}{2} \cdot cost(P) < \Phi(P) \leqslant cost(P).$$

In order to prove the claim, we prove that for every $e \in E$:

$$\frac{1}{2} \cdot cost_e(P) < \Phi_e(P) \leqslant cost_e(P).$$

For the second inequality, recall that $cost_e(P) = \sum_{1 \leqslant i \leqslant k} L_{e,i}(P) \cdot \big(a_e \cdot L_e(P) + b_e\big)$ and $L_e(P) = \sum_{1 \leqslant i \leqslant k} L_{e,i}(P)$. Thus,

$$cost_e(P) = \sum_{i=1}^{k} L_{e,i}(P) \cdot \big(a_e \sum_{j=1}^{k} L_{e,j}(P) + b_e\big) = a_e \cdot \sum_{i=1}^{k} \sum_{j=1}^{k} L_{e,j}(P) \cdot L_{e,i}(P) + b_e \cdot \sum_{i=1}^{k} L_{e,i}(P) \geqslant$$

$$\geqslant a_e \cdot \sum_{i=1}^{k} \sum_{j=i}^{k} L_{e,j}(P) \cdot L_{e,i}(P) + b_e \cdot \sum_{i=1}^{k} L_{e,i}(P) = \Phi_e(P).$$

We continue to prove the first inequality. Using the same calculations as above, we have:

$$cost_e(P) - \Phi_e(P) = a_e \cdot \sum_{i=2}^{k} \sum_{j=1}^{i-1} L_{e,j}(P) \cdot L_{e,i}(P) =$$

$$= a_e \cdot \sum_{i=1}^{k} \sum_{j=i+1}^{k} L_{e,j}(P) \cdot L_{e,i}(P) = \Phi_e(P) - a_e \sum_{i=1}^{k} L_{e,i}(P)^2 - b_e \sum_{i=1}^{k} L_{e,i}(P).$$

Thus,

$$cost_e(P) = 2 \cdot \Phi_e(P) - a_e \sum_{i=1}^{k} L_{e,i}(P)^2 - b_e \sum_{i=1}^{k} L_{e,i}(P).$$

Since a_e, b_e, and $L_{e,i}(P)$, for $1 \leqslant i \leqslant k$, are all positive, we get that $cost_e(P) < 2 \cdot \Phi_e(P)$, and we are done. □

A description of G_4 from Theorem 4.2:

The game G_4 is defined with respect to resources $a, b, c^1, c^2, d^1, d^2, d^3$, and d^4, where the latency of a, b, c^1, and c^2 are as in G_3, d^1 has latency identity plus ϵ'' and d^2, d^3, and d^4 have identity latency. Let $x_4 = 4!$. The strategies of players $1, 2$ and 3, are the same as in G_3 only that we use x_4 instead of x_3. We define $O_4^4 = \{\langle d^1, \frac{x_4}{4}, 4\rangle, \langle d^2, \frac{x_4}{4}, 2\rangle, \langle d^3, \frac{x_4}{3}, 1\rangle, \langle d^4, \frac{x_4}{2}, 1\rangle\}$ and $N_4^4 = \{\langle a, x_4, 1\rangle, \langle b, \frac{x_4}{2}, 1\rangle, \langle c^1, \frac{x_4}{3}, 1\rangle, \langle c^2, \frac{x_4}{2}, 1\rangle\}$. Again, we claim that $\bar{N}_4$ is the only PNE in the game. Note that with a slight adjustment due to the transition from x_3 to x_4, our calculations in G_3 are valid here. So, to illustrate the idea why $\bar{N}_4$ is the only PNE, we show that Player 4 deviates from $\bar{N}_4[4 \leftarrow O_4^4]$ to N_4^4, and that players 2 and 3 do not benefit by deviating from $\bar{N}_4$ to O_2^4 and O_3^4, respectively. Thus, $cost_4(\bar{N}_4[4 \leftarrow O_4^4]) = \frac{x_4}{4}(4 \cdot 4 \cdot (1 + \epsilon'')) + \frac{x_4}{2 \cdot 2}(2 \cdot 2) + \frac{x_4}{3} + \frac{x_4}{2}$. In the profile $\bar{N}_4[4 \leftarrow O_4^4]$, the load on the a-type resources is 3, the load on the b-type resources is 1, and the load on the c^1- and c^2-type resources is 0. Thus, in $\bar{N}_4$, the loads increase to $4, 2, 1$, and 1, respectively, and $cost_4(\bar{N}_4) = x_4(4 \cdot 1) + \frac{x_4}{2}(2 \cdot 1 \cdot (1 + \epsilon)) + \frac{x_4}{3}(1 \cdot 1 \cdot (1 + \epsilon')) + \frac{x_4}{2}(1 \cdot 1)$. We choose $\epsilon > \epsilon'$ so that $cost_4(\bar{N}_4[4 \leftarrow O_4^4]) > cost_4(\bar{N}_4)$. We continue to show that players 2 and 3 do

not benefit by deviating from $\bar{N}_4$ to O_2^4. In $\bar{N}_4$, the load on the b-type resources is 2, and in $\bar{N}_4[2 \leftarrow O_2^4]$ the load increases to 4 as Player 2 has two uses of every b-type resource. Thus, $cost_2(\bar{N}_4) = 24(4{\cdot}1) < 12(4{\cdot}2{\cdot}(1{+}\epsilon)) = cost_2(\bar{N}_4[2 \leftarrow O_2^4])$. Similarly, $cost_3(\bar{N}_4) = 24(4{\cdot}1){+}12(2{\cdot}1{\cdot}(1{+}\epsilon)) > 8(4{\cdot}3{\cdot}(1{+}\epsilon')){+}12(2{\cdot}1) = cost_3(\bar{N}_4[3 \leftarrow O_3^4])$. Recall that $x_4 = 4! = 24$. Thus, $cost(\bar{N}_4) = 24(4 \cdot 4) + 12(2 \cdot 2 \cdot (1 + \epsilon)) + 8 + 12$ and $cost(\bar{O}_4) = 24 + 12(2 \cdot 2(1 + \epsilon)) + 8(3 \cdot 3(1 + \epsilon')) + 12 + 6(4 \cdot 4(1 + \epsilon'')) + 6(2 \cdot 2) + 8 + 12$, and thus, $\text{PoS}(G_4) = 1.527$.

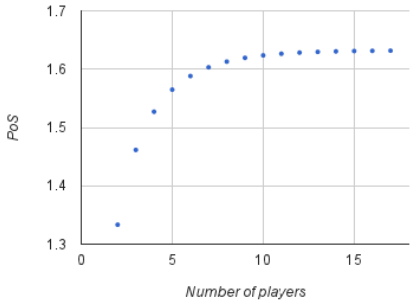


Figure 3: A graph of $\text{PoS}(G_k)$ as a function of k. The value of $\text{PoS}(G_{17})$ is 1.6316.

Proof of Claim 4.2.1: We prove the claim by induction on k. For the base case, $k = 2$ and the correctness of the claim is shown in the description of G_2 above. We assume correctness for $k - 1$ and prove for k. Consider $1 \leqslant i \leqslant k$ and a profile $P = \langle P_1^k, \ldots, P_k^k \rangle$ having $P_i^k = O_i^k$. Let $P' = P[i \leftarrow N_i^k]$. We claim that $cost_i(P) > cost_i(P')$.

We distinguish between two cases. In the first case, $i \neq k$. Let $Q = \langle P_1^{k-1}, \ldots, P_{k-1}^{k-1} \rangle$, where for $2 \leqslant j \leqslant k$, if $P_j^k = N_j^k$, then $P_j^{k-1} = N_j^{k-1}$ and otherwise $P_j^k = O_j^k$ and $P_j^{k-1} = O_j^{k-1}$. Let $Q' = Q[i \leftarrow N_i^{k-1}]$. By the induction hypothesis, $cost_i(Q) = cost_i(Q')$. We distinguish between two cases. In the first case $P_k^k = O_k^k$, and by the construction, Player k does not share any of its resources. It is not hard to prove that $\frac{1}{x_k} \cdot cost_i(P) = \frac{1}{x_{k-1}} \cdot cost_i(Q)$ and $\frac{1}{x_k} \cdot cost_i(P') = \frac{1}{x_{k-1}} cost_i(Q')$, and the claim follows.

For the second case, $P_k^k = N_k^k$. Recall that N_k^k has a single use for every resource in N_j^k or O_j^k, for $1 \leqslant j \leqslant k - 1$. Specifically, for every $e \in O_i^k$ we have $L_e(P) = L_e(Q) + 1$, and for every $e \in N_i^k$ we have $L_e(P') = L_e(Q') + 1$. Thus,

$$cost_i(P) = \sum_{e \in O_i^k} L_{e,i}(P) \cdot f_e(L_e(P)) = \sum_{e \in O_i^k} L_{e,i}(P) \cdot f_e(L_e(Q){+}1) = \sum_{e \in O_i^k} L_{e,i}(P) \cdot f_e(L_e(Q)) + \sum_{e \in O_i^k} L_{e,i}(P).$$

Since $L_{e,i}(P) = L_{e,i}(Q)$, we have

$$\sum_{e \in O_i^k} L_{e,i}(P) \cdot f_e(L_e(Q)) = \sum_{e \in O_i^k} L_{e,i}(Q) \cdot f_e(L_e(Q)) = cost_i(Q).$$

By the induction hypothesis, $cost_i(Q) = cost_i(Q')$. Going the opposite direction, we have $cost_i(Q') = \sum_{e \in N_i^k} L_{e,i}(P') \cdot f_e(L_e(Q'))$. Note that $\sum_{e \in O_i^k} L_{e,i}(P) = |O_i^k|$. We show in the construction that $|O_i^k| = |N_i^k| = \sum_{e \in N_i^k} L_{e,i}(P')$. To conclude,

$$cost_i(P) = \sum_{e \in N_i^k} L_{e,i}(P') \cdot f_e(L_e(Q')) + \sum_{e \in N_i^k} L_{e,i}(P') = \sum_{e \in N_i^k} L_{e,i}(P') \cdot f_e(L_e(P')) = cost_i(P').$$

We continue to prove the second case in which $i = k$. Recall that we construct O_k^k and N_k^k to that $cost_k(\bar{O}_k) = cost_k(\bar{N}_k)$. We claim that $cost_k(\bar{O}_k) = cost_k(P)$ and $cost_k(\bar{N}_k) = cost_k(P')$, which would conclude the proof. Recall that in the construction of O_k^k we use new resources. Thus, Player k does not share any resources when he plays O_k^k, and we have $cost_k(\bar{O}) = cost_k(P)$. We continue to show that $cost_k(\bar{N}_k) = cost_k(P')$. It is not hard to prove by induction on k, that for every two profiles S and S' in G_k, we have $\sum_{e \in E_k} L_e(S) = \sum_{e \in E_k} L_e(S')$, where E_k are the resources of the game G_k. When Player k plays N_k^k, he uses every resource in $E_k \backslash O_k^k$ exactly once. Since O_k^k uses new resources, when Player k plays N_k^k, the load on $e \in O_k^k$ is 0, and $\sum_{e \in E_k} L_e(\bar{N}_k) = \sum_{e \in (E_k \backslash O_k^k)} L_e(\bar{N}_k)$. Let $y = \sum_{2 \leqslant i \leqslant k-1} \frac{x_k}{i} \epsilon_i$. To conclude the proof,

$$\begin{aligned} cost_k(\bar{N}_k) &= \sum_{e \in N_i^k} L_{e,i}(\bar{N}_k) \cdot f_e(L_e(\bar{N}_k)) = y + \sum_{e \in N_i^k} L_e(\bar{N}_k) \cdot 1 = \\ &= y + \sum_{e \in E} L_e(\bar{N}_k) = y + \sum_{e \in E} L_e(P') \cdot 1 = cost_k(P'). \end{aligned}$$

□

Proof of Claim 4.2.2: Assume towards contradiction that the social optimum profile P^* is not $\bar{O}$. Let $1 \leqslant i \leqslant k$ be the highest index of a player that uses his N_i^k strategy in P^*. We claim that the profile P' in which Player i plays O_i^k has $cost(P^*) > cost(P')$. For $i < j \leqslant k$, in P^*, Player j uses his O_j^k strategy, and thus does not share any resources with Player i in P'. It follows that $cost_j(P^*) = cost_j(P')$. For $1 \leqslant j < i$, since Player i shares resources with Player j in P^* and does not share resources in P', we have that $cost_j(P^*) > cost_j(P')$. Moreover, since the e_1-type resources have identity latency, we have $cost_j(P^*) \geqslant 1 + cost_j(P')$. By the construction, we have $cost_i(P^*) = cost_i(P') + \epsilon_i$, and it is possible to select ϵ_i so that $cost(P') < cost(P^*)$, which is a contradiction to the minimality of P^*, and we are done. □

Proof of Remark 4.2.1: For the upper bound, consider a linear MCG G and let N be a PNE profile that is reached by a sequence of best-response moves from the social optimum O. We tighten our analysis from Theorem 4.1. For $e \in E$, we have $cost_e(N) = 2cost(N) - a_e \cdot \sum_{1 \leqslant i \leqslant k} L_{e,i}^2(N) \geqslant \sqrt{a_e} \cdot \sum_{1 \leqslant i \leqslant k} L_{e,i}(N) = \sqrt{a_e} \cdot L_e(N) = \sqrt{cost_e(N)}$. Thus, $cost_e(N) \geqslant 2 - \sqrt{cost_e(N)}$ and $cost(N) \geqslant 2 - \sum_{e \in E} \sqrt{cost_e(N)}$.

Let N' be the cheapest PNE profile. Then, $\sum_{e\in E}\sqrt{cost_e(N)} \geqslant \sum_{e\in E}\sqrt{cost_e(N')}$ and $\text{PoS}(G) \leqslant 2 - cost^{-1}(O)\cdot\sum_{e\in E}\sqrt{cost_e(N')}$.

For the lower bound, we show that the inequality is essentially an equality for the MCGs in the family described in Theorem 4.2. Consider $k \in \mathbb{N}$. Since in the social optimum in G_k the resources are used by exactly one player, we have $\Phi(\bar{O_k}) = cost(\bar{O_k})$. Claim 4.2.1 implies that, essentially, $\Phi(\bar{O_k}) = \Phi(\bar{N_k})$. Finally, since in $\bar{N_k}$ a resource is used by a player at most one time, we have $\Phi(\bar{N_k}) = \frac{1}{2}cost(\bar{N_k}) + \frac{1}{2}\sum_{e\in E}\sqrt{cost_e(\bar{N_k})}$. Thus, $\text{PoS}(G_k) = 2 - cost^{-1}(O_k)\cdot\sum_{e\in E}\sqrt{cost_e(\bar{N_k})}$. Since $\bar{N_k}$ is the only PNE in G_k, we are done.

Proof of Theorem 4.3: Recall that $\mathcal{P}_d$ is the set of polynomials of degree at most d. The proof relies on the following lemma [2]: Let $d \in \mathbb{N}$. Then,

$$\min_{(\lambda,\mu)\in\mathbb{R}^2}\{\frac{\lambda}{1-\mu} : \forall x,y \in \mathbb{R}, f \in \mathcal{P}_d \text{ we have } y\cdot f(x+y) \leqslant \lambda\cdot y\cdot f(y) + \mu\cdot x\cdot f(x)\} = \Phi_d^{d+1}. \tag{1}$$

Consider an MCG G, and let N be a PNE profile and O be the social optimum profile. Consider $1 \leqslant i \leqslant k$. Recall that for a profile P and resource $e \in E$, we use P_e to denote the load on e in P, the latency function on $e \in E$ is $f_e \in \mathcal{P}_d$, and for $1 \leqslant i \leqslant k$, we use P^i to denote the multiset that Player i chooses in P and $P^i(e)$ the number of times Player i uses e in P^i. By the definition of a PNE, we have $cost_i(N) \leqslant cost_i(N[i \leftarrow O^i])$. Next,

$$cost_i(N[i \leftarrow O^i]) = \sum_{e\in E} O^i(e)\cdot f_e(N_e - N_e^i + O_e^i) \leqslant \sum_{e\in E} O^i(e)\cdot f_e(N_e + O_e),$$

where the last inequality follows from the fact that the latency functions we consider are monotonically increasing.

$$cost(N) = \sum_{1\leqslant i\leqslant k} cost_i(N) \leqslant \sum_{1\leqslant i\leqslant k}\sum_{e\in E} O^i(e)\cdot f_e(N_e + O_e) = \sum_{e\in E} O_e\cdot f_e(N_e + O_e).$$

Let $\lambda, \mu \in \mathbb{R}$ that minimize the expression in (1). Then,

$$\sum_{e\in E} O_e\cdot f_e(N_e + O_e) \leqslant \sum_{e\in E} \lambda\cdot O_e\cdot f(O_e) + \mu\cdot N_e\cdot f(N_e) = \lambda\cdot cost(O) + \mu\cdot cost(N).$$

Rearranging yields the theorem.

Proof of Theorem 4.5: We show a family of two-player congestion games in which the PoA is arbitrarily high. For $n \in \mathbb{N}$, we define the game G_n. The resources are $E = \{e_1, e_2\}$. The strategy spaces of the players are mirrored. Player 1's strategy space is $\Sigma_1 = \{e_1, e_2^n\}$ and Player 2 strategy space is $\Sigma_2 = \{e_1^n, e_2\}$. The latency functions are both the factorial function, i.e., $f_{e_1}(l) = f_{e_2}(l) = l!$.

We continue to calculate $\mathrm{PoA}(G_n)$. Clearly, the social optimum is attained in the profile $\langle e_1, e_2\rangle$ and its cost is 2. We claim that the profile $N = \langle e_2^n, e_1^n\rangle$ is a PNE. Indeed, $cost_1(N) = n \cdot n!$, and by deviating to e_1, his cost increases to $1 \cdot (n+1)!$. The proof for Player 2 is dual. Since $cost(N) = 2n \cdot n!$, we have $\mathrm{PoA}(G_n) = n \cdot n!$, and we are done. □

A formal construction of the DFA $\mathcal{A}_{\mathcal{L},\mathcal{D}}$: Recall that the library of components is $\mathcal{L} = \{\mathcal{B}_1, \ldots, \mathcal{B}_n\}$. For $i \in [n]$, we have $\mathcal{B}_i = \langle \Sigma, B_i, \delta_i, b_i^0, F_i, E_i\rangle$, where $E_i \subseteq B_i$ is a set of exit states having $b_i^0 \notin E_i$ and $F_i \cap E_i = \varnothing$. The states of the components are disjoint, thus $B_i \cap B_j = \varnothing$, for $j \neq i$. We denote by $\mathcal{B}$, $\mathcal{F}$, and $\mathcal{E}$ the union of all states, accepting states, and exit states, respectively, thus $\mathcal{B} = \bigcup_{i\in[n]} B_i$, $\mathcal{F} = \bigcup_{i\in[n]} F_i$, and $\mathcal{E} = \bigcup_{i\in[n]} E_i$. Recall that a design $\mathcal{D}$ is a transducer over input alphabet $\mathcal{E}$ and output alphabet $[n]$. Consider a design $\mathcal{D} = \langle \mathcal{E}, [n], D, \delta_{\mathcal{D}}, d^0, \nu\rangle$, where $\nu : D \to [n]$.

We construct the composition system $\mathcal{A}_{\mathcal{L},\mathcal{D}} = \langle \Sigma, Q_{\mathcal{L},\mathcal{D}}, \delta_{\mathcal{L},\mathcal{D}}, q^0_{\mathcal{L},\mathcal{D}}, F_{\mathcal{L},\mathcal{D}}\rangle$ as follows. The set of states $Q_{\mathcal{L},\mathcal{D}} \subseteq (\mathcal{B}\backslash\mathcal{E}) \times D$ consists of pairs of a *component state* from $\mathcal{B}$ and an *design state* from D. The component states are consistent with ν, thus $Q_{\mathcal{L},\mathcal{D}} = \bigcup_{i\in[n]}(B_i\backslash E_i) \times \{q : \nu(q) = i\}$. In exit states, the composition immediately moves to the initial state of the next component, which is why the component states of $\mathcal{A}_{\mathcal{L},\mathcal{D}}$ do not include $\mathcal{E}$. Consider a state $\langle b, q\rangle \in Q_{\mathcal{L},\mathcal{D}}$ and a letter $\sigma \in \Sigma$. Let $i \in [n]$ be such that $b \in B_i$. When a run of $\mathcal{A}_{\mathcal{L},\mathcal{D}}$ reaches the state $\langle b, q\rangle$, the component $\mathcal{B}_i$ is in control. Recall that b is not an exit state. Let $b' = \delta_i(b, \sigma)$. If $b' \notin E_i$, then $\mathcal{B}_i$ does not relinquish control after reading σ and $\delta_{\mathcal{L},\mathcal{D}}(\langle b, q\rangle, \sigma) = \langle b', q\rangle$. If $b' \in E_i$, then $\mathcal{B}_i$ relinquishes control through b', and it is the design's task to choose the next component to gain control. Let $q' = \delta(q, b')$ and let $j = \nu(q')$. Then, $\mathcal{B}_j$ is the next component to gain control (possibly $j = i$). Accordingly, we advance $\mathcal{D}$ to q' and continue to the initial state of $\mathcal{B}_j$. Formally, $\delta_{\mathcal{L},\mathcal{D}}(\langle b, q\rangle, \sigma) = \langle b_j^0, q'\rangle$. (Recall that $b_j^0 \notin E_j$, so the new state is in $Q_{\mathcal{L},\mathcal{D}}$.) Note also that a visit in b' is skipped. The component that gains initial control is chosen according to $\nu(d^0)$. Thus, $q^0_{\mathcal{L},\mathcal{D}} = \langle b_j^0, d^0\rangle$, where $j = \nu(d^0)$. Finally, the accepting states of $\mathcal{A}_{\mathcal{L},\mathcal{D}}$ are these in which the component state is accepting, thus $F_{\mathcal{L},\mathcal{D}} = \mathcal{F} \times D$. (Recall that $\mathcal{F} \cap \mathcal{E} = \varnothing$, so $F_{\mathcal{L},\mathcal{D}} \subseteq Q_{\mathcal{L},\mathcal{D}}$.)

Proof of Theorem 5.1: Consider an MCG $\langle K, E, \{\Sigma_i\}_{i\in K}, \{f_e\}_{e\in E}\rangle$. Recall that Σ_i is the set of strategies for Player i that consists of multisets over E. We construct a CLG with alphabet $E \cup \bigcup_{i\in K} \Sigma_i$. For $i \in K$, the specification $\mathcal{S}_i$ for Player i consists of $|\Sigma_i|$ words. Every strategy $s = \{e_1, \ldots, e_n\}$ (allowing duplicates) in Σ_i contributes to $L(S)$ the word $s \cdot e_1 \cdot e_2 \cdot \ldots \cdot e_n$. We construct a library $\mathcal{L}$ with $|E| + \sum_{i\in K} |\Sigma_i|$ components of two types: a *strategy component* $\mathcal{B}_s$ for each $s \in \Sigma_i$ and a *resource component* $\mathcal{B}_e$ for each $e \in E$. In addition, $\mathcal{L}$ contains the component

$\mathcal{B}_{acc}$ that is depicted in Figure 2. Intuitively, a correct design must choose one strategy component $\mathcal{B}_s$ and then use the component $\mathcal{B}_e$ the same number of times e appears in s. We continue to describe the components. For $s \in \Sigma_i$, the component $\mathcal{B}_s$ relinquishes control only if the letter s is read. It accepts every word in $L(\mathcal{S}_i)$ that does not start with s. For $e \in E$, the resource component $\mathcal{B}_e$ has an initial state with an e-labeled transition to an exit state. Finally, the latency function for the resource components coincides with latency functions of the resources in the MCG, thus for $e \in E$, we have $f_{\mathcal{B}_e} = f_e$. The other latency functions are $f \equiv 0$. We prove that there is a cost-preserving one-to-one and onto correspondence between correct designs with respect to $\mathcal{S}_i$ and strategies in Σ_i, implying the existence of the required function between the profiles.

Consider $i \in K$. We claim that there is a one-to-one and onto correspondence between correct designs with respect to $\mathcal{S}_i$ and strategies in Σ_i. Thus, there is a one-to-one and onto correspondence between the sets of profiles in the MCG and the CLG. We describe the design $\mathcal{D}_s$ that corresponds to the strategy $s = \{e_1, \ldots, e_n\} \in \Sigma_i$. The design $\mathcal{D}_s$ assigns initial control to the strategy component $\mathcal{B}_s$. Recall that $\mathcal{B}_s$ relinquishes control after reading the letter s. Then, $\mathcal{D}_s$ assigns control to the components $\mathcal{B}_{e_1}, \ldots, \mathcal{B}_{e_n}$ and finally control is assigned to $\mathcal{B}_{acc}$. Clearly the word $s \cdot e_1 \cdot e_2 \cdots e_n \in L(\mathcal{S})$ is accepted, no other word that starts with s is accepted, and since $\mathcal{B}_s$ accepts every word in $L(\mathcal{S}_i)$ that does not start with s, we have that $\mathcal{D}_s$ is correct. It is not hard to see that there are no other correct designs. To see that the correspondence is onto, note that since control has to be assigned to some component $\mathcal{B}_s$, each correct design corresponds to a strategy $s \in \Sigma_i$. Finally, consider a profile $P = \langle \mathcal{D}_{s_1}, \ldots, \mathcal{D}_{s_k} \rangle$ that corresponds to the profile $P' = \langle s_1, \ldots, s_k \rangle$. Clearly, for $i \in K$, we have $cost_i(P) = cost_i(P')$, and we are done.

Proof of Theorem 5.4: We describe the specification $\mathcal{S}$. The alphabet is $\Sigma = \{\#\} \cup \bigcup_{i \in [n]} \{\#_i^0, \ldots, \#_i^m, x_i\} \cup \bigcup_{j \in [m]} \{C_j, \tilde{C}_j\}$. The language $L(\mathcal{S})$ includes $3m + 1$ words. A long word $\# \cdot w_1 \cdot \ldots \cdot w_n$, where for $i \in [n]$, we have $w_i = \#_i^0 x_i \#_i^1 \ldots x_i \#_i^m$, and $3m$ short words, which we describe next. For $j \in [m]$, let $C_j = \ell_j^1 \vee \ell_j^2 \vee \ell_j^3$. For $k \in \{1, 2, 3\}$, let j_k be the index of the variable in the literal ℓ_j^k. That is, $\ell_j^1 \in \{x_{j_1}, \neg x_{j_1}\}$, $\ell_j^2 \in \{x_{j_2}, \neg x_{j_2}\}$, and $\ell_j^3 \in \{x_{j_3}, \neg x_{j_3}\}$. For $k \in \{1, 2, 3\}$, we have $C_j x_{j_k} \tilde{C}_j x_{j_k} \in L(S)$. Clearly it is possible to construct $\mathcal{S}$ so that its size is polynomial in n and m.

The components of the library $\mathcal{L}$ are $\mathcal{B}_0$, $\mathcal{B}_{x_i}^j$ and $\mathcal{B}_{\neg x_i}^j$ for $j \in [m]$ and $i \in [n]$, and $\mathcal{B}_{C_j, x_{j_k}}$ for $j \in [m]$ and $k \in \{1, 2, 3\}$. The components of the library are depicted in Figure 2. The latency functions of the components $\mathcal{B}_0$ and $\mathcal{B}_{C_j, x_{j_k}}$, for $j \in [m]$ and $k \in \{1, 2, 3\}$ is the constant function $f \equiv 0$. The latency function of the components $\mathcal{B}_{x_i}^j$, for $j \in [m]$ and $i \in [n]$ is the identity function $f(x) = x$. Thus, using a

component $\mathcal{B}^j_{x_i}$ once costs 1.

Assume that φ is satisfiable, and let $\eta : \{x_1, \ldots, x_n\} \to \{T, F\}$ be a satisfying assignment to the variables. We describe a correct design $\mathcal{D}$ with $cost(\mathcal{D}) = \mu = nm + m$. The component that gains control first is $\mathcal{B}_0$. We distinguish between two types of words. First, assume $\mathcal{B}_0$ exists through $e_\#$ after reading #. Thus, the only word that should be accepted from this point is the suffix $w_1 \cdot \ldots w_m$ of the long word in $L(S)$, where recall that for $i \in [n]$ we have $w_i = \#^0_i x_i \#^1_i \ldots x_i \#^m_i$. For $i \in [n]$, assuming the prefix $\#w_1 \cdot \ldots \cdot w_{i-1}$ has been read. If $\eta(x_i) = T$, then control is assigned to $\mathcal{B}^1_{\neg x_i}$, and otherwise control is assigned to $\mathcal{B}^1_{x_i}$. Then, the next $m - 1$ assignments of control, assuming the prefix that is read is a prefix of w_i, are to the components $\mathcal{B}^2_{\neg x_i}, \ldots, \mathcal{B}^m_{\neg x_i}$ if $f(x_i) = T$, and otherwise to $\mathcal{B}^2_{x_i}, \ldots, \mathcal{B}^m_{x_i}$. If some other word is read, control is assigned to $\mathcal{B}_{rej}$. Finally, control is assigned to $\mathcal{B}_{acc}$. Note that so far we have used nm components with identity latency, each with one use, for a total cost of nm.

Recall that $L(\mathcal{S})$ has $3m$ short words of the form $C_j x_{j_l} \tilde{C}_j x_{j_k}$, where $j \in [m]$ and $k \in \{1, 2, 3\}$. When reading the prefix C_j of such a word, the component $\mathcal{B}_0$ relinquishes control through e_{C_j}. Assume, WLog, that $f(\ell^1_j) = T$ (there must be such a literal as f is a satisfying assignment). We refer to x_{j_1} as the *witness* for C_j. The component that gains control after $\mathcal{B}_0$ is $\mathcal{B}_{C_j, x_{j_1}}$, which relinquishes control only if x_{j_1} is read. Let $0 \leqslant j' < j$ be the number of clauses with index less than j that have x_{j_1} as their witness. The component that gains control after $\mathcal{B}_{C_j, x_{j_1}}$ is $\mathcal{B}^{j'+1}_{x_{j_1}}$. This component has many exit states. If it exists after reading $\tilde{C}_j x_{j_1}$, then control is assigned to $\mathcal{B}_{acc}$, and otherwise control is assigned to $\mathcal{B}_{rej}$. Note that we have used another m components with identity latency, where each component is used once. Thus, the total cost of $\mathcal{D}$ is $nm + m$. It is not hard to see that $\mathcal{D}$ is correct, and we are done.

For the other direction, assume there is a correct design $\mathcal{D}$ that costs at most $nm + m$. We define an assignment $\eta : \{x_1, \ldots, x_n\} \to \{T, F\}$. Let $i \in [n]$. Since the long word $\#w_1 \ldots w_n$ is in $L(\mathcal{S})$, it must be in $L(\mathcal{A}_{\mathcal{L},\mathcal{D}})$. Note that the only components that can process the letter $\#^0_i$ are $\mathcal{B}^1_{x_i}$ and $\mathcal{B}^1_{\neg x_i}$. Thus, after reading $\#w_1 \ldots w_{i-1}$, one of these components must gain control. We define $\eta(x_i) = T$ if $\mathcal{B}^1_{\neg x_i}$ gains control and $\eta(x_i) = F$ if $\mathcal{B}^1_{x_i}$ gains control.

We claim that η is a satisfying assignment for φ. In order to do so, we make two observations. First, recall that, for $i \in [n]$, if $\eta(x_i) = T$, then $\mathcal{B}^1_{\neg x_i}$ gains control after $\#w_1 \ldots w_{i-1}$ is read. If the prefix $\#^1_i x_i$ of w_i is read, then $\mathcal{B}^2_{\neg x_i}$ must gain control as this is the only component that can process the next prefix $\#^2_i x_i$ of w_i. Generalizing this observation, after reading $\#w_1 \ldots w_{i-1}$, if $\eta(x_i) = T$, when reading the word w_i, the components that are in control are $\mathcal{B}^1_{\neg x_i}, \ldots, \mathcal{B}^m_{\neg x_i}$, and if $\eta(x_i) = F$, then $\mathcal{B}^1_{x_i}, \ldots, \mathcal{B}^m_{x_i}$ are in control. Thus, $\mathcal{D}$ uses at least nm components with identity

latency.

Next, let $j \in [m]$ with $C_j = \ell_j^1 \vee \ell_j^2 \vee \ell_j^3$ such that $\ell_j^k \in \{x_{j_k}, \neg x_{j_k}\}$, for $k \in \{1,2,3\}$. Note that after reading the letter C_j control must be assigned to a component $\mathcal{B}_{C_j,x_{j_k}}$ as $C_j x_{j_k} \tilde{C}_j x_{j_k} \in L(\mathcal{S})$ and these are the only components that can process words that start with one of the three letters x_{j_1}, x_{j_2}, or x_{j_3}. Denote by ℓ_j and x_j the literal and variable with $\ell_j \in \{x_j, \neg x_j\}$ such that $\mathcal{B}_{C_j,x_j}$ gains control from $\mathcal{D}$ after C_j is read. When x_j is read in $\mathcal{B}_{C_j,x_j}$, control is relinquished. Note that the component that gains control next must be a component $\mathcal{B}^t_{\ell_j}$, for $t \in [m]$, as these are the only components that can process $\tilde{C}_j x_j$. Let d be the state in $\mathcal{D}$ that is reached after reading $C_j x_j$[2]. We claim that there is no other word $w \in \Sigma^*$ such that the run of $\mathcal{D}$ on w reaches d. Indeed, the word $\tilde{C}_j x_j$ is accepted from d and there is only one word in $L(\mathcal{S})$ that ends with this suffix.

Since we have already counted nm uses of components with identity latency and $\mathcal{D}$ costs at most $nm+m$, we conclude that $\mathcal{D}$ uses exactly $nm+m$ such components each with a single use. Specifically, for $j \in [m]$, the component $\mathcal{B}^t_{\ell_j}$ that gains control after $C_j x_j$ is read is used once and it is used when reading the word $C_j x_j \tilde{C}_j x_j$. Thus, when reading the long word $\#w_1 \dots w_m$ it does not gain control, and by our definition of η, we have $\eta(\ell_j) = T$. We conclude that η is a satisfying assignment, and we are done.

Proof of Theorem 5.5: We now describe the reduction in detail. Let $\varphi = C_1 \wedge \ldots \wedge C_m$, where for $j \in [t]$ we have $C_j = \ell_j^1 \vee \ell_j^2 \vee \ell_j^3$, we construct a CLG G with three players such that $\varphi \in$ NAE iff G does not have a PNE. We describe the specifications of the players. The alphabet is $\Sigma = \{\#, \&, T, F, a, b\} \cup \bigcup_{i\in[n]}\{\#^0_{x_i}, \dots, \#^m_{x_i}, x_i,\ \#^0_{y_i}, \dots, \#^m_{y_i}, y_i\} \cup \bigcup_{j\in[m]}\{C_j, \tilde{C}_j\} \cup \bigcup_{j\in[m]}\{\ell_j^1, \ell_j^2, \ell_j^3\}$. The specification $\mathcal{S}_3$ for Player 3 consists of a single word $w_1 \cdot \ldots \cdot w_n$, where for $i \in [n]$, we have $w_i = \#^0_{x_i} x_i \#^1_{x_i} \dots x_i \#^m_{x_i}$. The specification $\mathcal{S}_2$ for Player 2 consists of two words a^3 and b^5. The specification $\mathcal{S}_1$ for Player 1 consists of $6m+1$ words. Similarly to Player 3, it has a long word $\# \cdot v_1 \cdot \ldots \cdot v_n$, where for $i \in [n]$, we have $v_i = \#^0_{y_i} y_i \#^1_{y_i} \dots y_i \#^m_{y_i}$, and $6m$ short words, which we describe next. For $j \in [m]$, recall that $C_j = \ell_j^1 \vee \ell_j^2 \vee \ell_j^3$. For $k \in \{1,2,3\}$, we have $C_j \ell_j^k T \tilde{C}_j T \ell_j^k, C_j \ell_j^k F \tilde{C}_j F \ell_j^k \in L(\mathcal{S}_1)$. Clearly it is possible to construct $\mathcal{S}_1, \mathcal{S}_2$, and $\mathcal{S}_3$ so that their size is polynomial in n and m.

We continue to describe the library $\mathcal{L}$. For every $i \in [n]$ and $t \in [m]$, there are *variable components* $\mathcal{B}^t_{x_i}$, $\mathcal{B}^t_{\neg x_i}$, $\mathcal{B}^t_{y_i}$, and $\mathcal{B}^t_{\neg y_i}$, which have identity latency functions and are similar to these in the lower bound proof of Theorem 5.4. The *upper* part of the components is almost identical to these depicted in Figure 2. For example,

[2]Recall that $\mathcal{D}$ reads exit states and not words over Σ, so this notation is not formal and we mean the exit states that are read during the run on the word.

$\mathcal{B}^1_{x_i}$ relinquishes control after reading $\#^0_{x_i} x_i$. The *lower* part is slightly different. Consider $j \in [m]$ and a literal ℓ that appears in C_j. Intuitively, as in Theorem 5.4, designs of Player 1 and Player 3 correspond to assignments and must use either all the components $\mathcal{B}^1_\ell, \ldots, \mathcal{B}^m_\ell$ or all the components $\mathcal{B}^1_{\neg\ell}, \ldots, \mathcal{B}^m_{\neg\ell}$. Choosing the first corresponds to assigning value false to ℓ and choosing the second corresponds to assigning value true to ℓ. The components $\mathcal{B}^1_\ell, \ldots, \mathcal{B}^m_\ell$ have a path labeled $\tilde{C}_j T \ell$ that leads to an exit state. The components $\mathcal{B}^1_{\neg\ell}, \ldots, \mathcal{B}^m_{\neg\ell}$ have a path labeled $\tilde{C}_j F \ell$ that leads to an exit state.

For every $j \in [m]$, there are six components $\mathcal{B}_{C_j,\ell^a_j,\ell^b_j}$ for $a \neq b \in \{1,2,3\}$ with latency function $f \equiv 0$. The component $\mathcal{B}_{C_j,\ell^a_j,\ell^b_j}$ corresponds to assigning value true to ℓ^a_j and value false to ℓ^b_j. Let $c \in \{1,2,3\}$ such that $c \neq a$ and $c \neq b$. The component $\mathcal{B}_{C_j,\ell^a_j,\ell^b_j}$ relinquishes control after reading $\ell^a_j T$ and $\ell^b_j T$, after which a correct design must assign control to components $\mathcal{B}^{t_1}_{\ell^a_j}$ and $\mathcal{B}^{t_2}_{\neg\ell^b_j}$, for some $t_1, t_2 \in [m]$, in order to accept the suffices $\tilde{C}_j T \ell^a_j$ and $\tilde{C}_j F \ell^b_j$, respectively. The other words that start with C_j are accepted *for free*. The component $\mathcal{B}_{C_j,\ell^a_j,\ell^b_j}$ accepts the words $\ell^a_j F \tilde{C}_j F \ell^a_j, \ell^b_j T \tilde{C}_j T \ell^b_j, \ell^c_j T \tilde{C}_j T \ell^c_j$, and $\ell^c_j F \tilde{C}_j F \ell^c_j$.

Similarly to the proof of Theorem 5.4, a correct design $\mathcal{D}_\eta$ for Player 3 corresponds to an assignment $\eta : \{x_1, \ldots, x_n\} \to \{T, F\}$ and a correct design $\mathcal{D}_\rho$ corresponds to an assignment $\rho : \{y_1, \ldots, y_n\} \to \{T, F\}$. Moreover, every correct design $\mathcal{D}_\rho$ must use $mn + 2m$ variable components. Assuming Player 3 chooses the design $\mathcal{D}_\eta$, then there is a Player 1 correct design $\mathcal{D}_\rho$ that uses only variable components with load 1 iff the pair $\langle \eta, \rho \rangle$ is legal for φ.

We describe the rest of the components in the library. There are components $\mathcal{B}_{acc}$ and $\mathcal{B}_{rej}$ that have constant latency function 0 and appear at the *end* of runs. We do not specify when they gain control below as it is immediate. There are components $\mathcal{B}^A_0$, $\mathcal{B}^B_0$, A, and B, with latency functions 0, 0, $f_A(x) = x^2 + 125$ and $f_B(x) = 4x^2$, respectively. The component $\mathcal{B}^A_0$ relinquishes control after reading a and accepts the word b^5. Dually, the component $\mathcal{B}^B_0$ relinquishes control after reading b and accepts the word a^3. The components A and B relinquish control after reading a and b, respectively. Thus, a correct Player 2 design must assign initial control to either $\mathcal{B}^A_0$ or $\mathcal{B}^B_0$, and then assign control two times to A and four times to B, respectively. The component $\mathcal{B}_{ALL}$ is identical to $\mathcal{S}_1$. Thus, Player 1 has a correct design that uses only it, and we refer to this design as $\mathcal{D}_{ALL}$. Regardless of what design the other players choose, Player 1 pays $1100 + mn + 2m$ for $\mathcal{D}_{ALL}$. The component $\mathcal{B}_0$ has a constant latency function 33. It relinquishes control through a unique exit state if $\#, \&$, or C_j, for $j \in [m]$ is read. Thus, a correct Player 1 design that does not use $\mathcal{B}_{ALL}$ must assign initial control to $\mathcal{B}_0$. When $\mathcal{B}_0$ relinquishes control after reading $\#$, control must be assigned to mn variable components, after reading C_j,

for some $j \in [m]$, control must be assigned to some component $\mathcal{B}_{C_j,\ell_j^a,\ell_j^b}$ as described in the above, and after reading &, control must be assigned to A for six consecutive times and then to B for one time.

We prove that the reduction is correct. Assume that $\varphi \notin$ NAE, thus there is an assignment $\eta : \{x_1, \ldots, x_n\} \to \{T, F\}$ such that for every assignment $\rho : \{y_1, \ldots, y_n\} \to \{T, F\}$, we have that $\langle \eta, \rho \rangle$ is illegal for φ. We claim that the profile $P = \langle \mathcal{D}_{ALL}, \mathcal{D}_B, \mathcal{D}_\eta \rangle$ is a PNE. Clearly Player 2 does not deviate. Since Player 1 uses $\mathcal{D}_{ALL}$ it does not use any of the variable components and all the designs of Player 3 cost the same, thus Player 3 does not deviate as well. Assume Player 1 benefits from deviating to a design $\mathcal{D}_\rho$. Since Player 2 uses $\mathcal{D}_B$, Player 1 pays for his uses of $\mathcal{B}_0$, A, and B a cost of $33 + 966 + 100 = 1099$. Recall that $\mathcal{D}_\rho$ uses $nm + 2m$ variable components. Since $cost_1(P) = 1100 + nm + 2m$ and we assume that deviating to $\mathcal{D}_\rho$ is beneficial, the load on every variable component that is used in $\mathcal{D}_\rho$ must be 1. Thus, similarly to the proof of Theorem 5.4, we can show that $\langle \eta, \rho \rangle$ is legal for φ, which is a contradiction. Next, assume that $\varphi \in$ NAE. We show that there is no PNE in G. Consider a profile in which Player 3 chooses the design $\mathcal{D}_\eta$, and let ρ be an assignment such that $\langle \rho, \eta \rangle$ is legal for φ. It is not hard to see that choosing $\mathcal{D}_\eta$ or $\mathcal{D}_{ALL}$ dominates every other choice of design for Player 1. In Table 2 we show that no matter which design Player 2 chooses, there is no PNE, and we are done.

	$\mathcal{D}_\rho$	$\mathcal{D}_{ALL}$
$\mathcal{D}_A$	$1099 + mn + 2m, 400$	$1100 + mn + 2m, 256$
$\mathcal{D}_B$	$1138 + mn + 2m, 378$	$1100 + mn + 2m, 288$

Table 2: Players costs. Each entry describes the cost of Player 1 followed by the cost of Player 2.

B Splittable (Non-Atomic) Games

In a splittable game each player can split his task among several strategies. This model suits several applications, in particular planning of preemptive production. Splittable games are well-understood in classical and weighted congestion games [29, 8]. We define the corresponding MCG and show that the positive PNE-existence result, known for weighted congestion games, carry over to games with multisets of resources.

Recall that Σ_i is the strategy space of Player i, where $\Sigma_i = \{m_{i,1}, \ldots, m_{i,c_i}\}$ and

for each $1 \leqslant j \leqslant c_i$, $m_{i,j}$ is a multiset over E. A splitted strategy for a player i is given by $\hat{s}_i = \langle \alpha_{i,1}, \ldots, \alpha_{i,c_i} \rangle$ such that $\sum_{j=1}^{c_i} \alpha_j = 1$.

A profile of a game G is a set $P = \langle \hat{s}_1, \hat{s}_2, \ldots, \hat{s}_k \rangle$ of strategies selected by the players. For a resource $e \in E$, the load that player i generates on e is $L_{e,i}(P) = \sum_{j=1}^{c_i} \alpha_j \cdot m_{i,j}(e)$. The *load* on e in P, is $L_e(P) = \sum_{1 \leqslant i \leqslant k} L_{e,i}(P)$.

The players' costs are defined as in the unsplittable model, that is, given a profile P, a resource $e \in E$, and $1 \leqslant i \leqslant k$, the *cost of e for Player i in P* is $cost_{e,i}(P) = L_{e,i}(P) \cdot f_e(L_e(P))$. The cost of Player i in the profile P is then $cost_i(P) = \sum_{e \in E} cost_{e,i}(P)$.

A profile P is a PNE if for every player i the cost of i when playing $\langle \alpha_{i,1}, \ldots, \alpha_{i,c_i} \rangle$ is not higher than playing $\langle \ldots, \alpha_{i,v_1} + \delta, \ldots, \alpha_{i,v_2} - \delta, \ldots \rangle$ for all v_1, v_2 and δ. In particular, it means that the marginal cost for all strategies with $\alpha_i > 0$ is the same. The proof of the following theorem is identical to the corresponding proof for classical CGs [29].

Theorem B.1 *Every splittable MCG has a PNE.*

For example, consider the instance in Example 1, where $\Sigma_i = \langle \{a^2, b\}, \{b^2, c\}, \{c^2, a\} \rangle$ for $i = 1, 2$. The profile $P = \langle \{\frac{1}{2}, \frac{1}{3}, \frac{1}{6}\}, \{\frac{1}{6}, \frac{1}{3}, \frac{1}{2}\} \rangle$ is a (non-unique) PNE. For each $e \in \{a, b, c\}$, we have $L_a(P) = 2 \cdot \frac{1}{2} + 1 \cdot \frac{1}{6} + 2 \cdot \frac{1}{6} + 1 \cdot \frac{1}{2} = 2$, $L_b(P) = 2$ and $L_c(P) = 2$, thus $cost_1(P) = cost_2(P) = 4 \cdot 3 = 12$. This profile is a PNE, since all strategies have the same marginal cost.

Dynamic Resource Allocation Games*

Guy Avni[†] *Thomas A. Henzinger*[‡] *Orna Kupferman*[§]

Abstract

In *resource allocation games*, selfish players share resources that are needed in order to fulfill their objectives. The cost of using a resource depends on the load on it. In the traditional setting, the players make their choices concurrently and in one-shot. That is, a strategy for a player is a subset of the resources. We introduce and study *dynamic* resource allocation games. In this setting, the game proceeds in phases. In each phase each player chooses one resource. A scheduler dictates the order in which the players proceed in a phase, possibly scheduling several players to proceed concurrently. The game ends when each player has collected a set of resources that fulfills his objective. The cost for each player then depends on this set as well as on the load on the resources in it – we consider both congestion and cost-sharing games. A prime application of dynamic resource allocation games is the setting of networks in which players choose their routes edge by edge, with choices depending on earlier choices of other players. We study the stability of dynamic resource allocation games, where the appropriate notion of stability is that of subgame perfect equilibrium, study the inefficiency incurred due to selfish behavior, and also study problems that are particular to the dynamic setting, like constraints on the order in which resources can be chosen or the problem of finding a scheduler that achieves stability.

1 Introduction

Resource allocation games (RAGs, for short) [23] model settings in which selfish agents share resources that are needed in order to fulfill their objectives. The cost of using a resource depends on the load on it. Formally, a k-player RAG G is given by a set E of resources and a set of possible strategies for each player. Each strategy is a subset of resources, fulfilling some objective of the player. Each resource $e \in E$ is associated with a latency function $\ell_e : \mathbb{N} \to \mathbb{R}$, where $\ell_e(\gamma)$ is the cost of a single

*Submitted for publication.

[†]School of Computer Science and Engineering, The Hebrew University, Jerusalem, Israel

[‡]IST Austria

[§]School of Computer Science and Engineering, The Hebrew University, Jerusalem, Israel

use of e when it has load γ. For example, in *network formation games* (NFGs, for short) [2], a network is modeled by a directed graph, and each player has a source and a target vertex. In the corresponding RAG, the resources are the edges of the graph and the objective of each player is to connect his source and target. Thus, a strategy for a player is a set of edges that form a simple path from the source to the target. Each edge may participate in paths of several players. When an edge e is used by m players, each of them pays $\ell_e(m)$ for his use.

A key feature of RAGs is that the players choose how to fulfill their objectives *in one shot* and *concurrently*. Indeed, a strategy for a player is a subset of the resources – chosen as a whole, and the players choose their strategies simultaneously. In many settings, however, resource sharing proceeds in a different way. First, in many settings, the choices of the players are made resource by resource as the game evolves. For example, when the network in an NFG models a map of roads and players are drivers choosing routes, it makes sense to allow each driver not to commit to a full route in the beginning of the game but rather to choose one road (edge) at each junction (vertex), gradually composing the full route according to the congestion observed. Second, players may not reach the junctions together. Rather, in each "turn" of the game, only a subset of the players (say, these that have a green light) proceed and chose their next road.

As another example to a rich composition and scheduling of strategies, consider the setting of *synthesis from component libraries* [17], where a designer synthesizes a system from existing components.It is shown in [4, 6] that when multiple designers use the same library, a RAG arises. Here too, the choice of components may be made during the design process and may evolve according to choices of other designers.

In this work we introduce and study *dynamic resource allocation games*, which allow the players to choose resources in an iterative and non-concurrent manner. A dynamic RAG is given by a pair $\mathcal{G} = \langle G, \nu \rangle$, where G is a k-player RAG and $\nu : \{1, \ldots, k\} \to \{1, \ldots, k\}$ is a *scheduler*. A dynamic RAG proceeds in *phases*. In each phase, each player chooses one resource. A phase is partitioned into at most k*turns*, and the scheduler dictates which players proceed in each turn: Player i moves at turn $\nu(i)$. Note that the scheduler may assign the same turn to several players, in which case they choose a resource simultaneously in a phase. Once all turns have been taken, a phase is concluded and a new phase begins. There are two "extreme" schedulers: (1) A *sequential* scheduler assigns different turns to all players, i.e., ν is a permutation, reflecting the fact that the players make their choices sequentially, one player in each turn. (2) A *concurrent* scheduler assigns the same turn to all the player; i.e., $\nu(i) = 1$ for all $i \in \{1, \ldots, k\}$, reflecting the fact that all players proceed concurrently in the first (and only) turn in each phase. A *strategy* for a player in a dynamic RAG maps the history of choices made by the players so far

(that is, the choices of all players in earlier phases as well as the choices of players that proceed in earlier turns in the current phase) to the player's next choice. A player finishes playing once the resources she has chosen satisfy the objective. The game terminates once all players finish playing. A strategy profile in the game is a vector of strategies – one for each player. The outcome of a profile is an assignment of a set of resources to each player. The cost of each player in a profile is induced by the costs of the resources in his set, which depends on their load and latency functions as in usual RAGs.

We illustrate the intricacy of the selecting the resources in phases in the following example.

Example 1.1 *Consider the* 4*-player network formation game that is depicted in Figure 1. The interesting edges have names, e.g.,* $a, b, c \ldots$*, and their latency function is depicted below the edge. For example, we have* $\ell_a(x) = x$ *and* $\ell_{c_1}(x) = 10x$*. The other edges have latency function* 0*. The source and target of a node of Player* i *are depicted with a node called* s *and* t*, respectively, and with a subscript* i*. For example, Player* 2*'s source is* $s_{1,2}$ *and he has two targets* t_2^L *and* t_2^R*. The players' strategies are paths from one of their sources to one of their targets.*

Consider a dynamic version of the game in which Player i *chooses an edge at turn* i*. At first look, it seems that edge* g *will never be chosen. However, we show that Player* 1*'s optimal strategy uses it. Player* 1 *has three options in the first turn, either choose* g*,* a*, or* b*. Assume he chooses* a *(and dually* b*). Then, we claim that Player* 2 *will choose* b*. Note that Players* 3 *and* 4 *move oposite of Player* 2 *no matter how Player* 1 *moves, as they prefer avoiding a load of* 2 *on* c_1 *and* c_2*, which costs* 20 *each, even at the cost of a load of* 3 *on* f*, which costs only* 3*. Knowing this, Player* 2 *prefers using* b *alone over sharing* a *with Player* 1*. Since the loads on* a *and* e *are* 1 *and* 3*, respectively, Player* 1*'s cost is* $1 + 3 = 4$*.*

On the other hand, if Player 1 *chooses* g *in the first phase, he postpones revealing his choice between left and right. If Player* 2 *proceeds left, then Players* 3 *and* 4 *proceed right, and Player* 1 *proceeds left in the second phase. Now, the load on* a *and* e *is* 2 *and* 1*, respectively, thus Player* 1*'s cost is* $\frac{1}{2} + 2 + 1 = 3\frac{1}{2}$*.* □

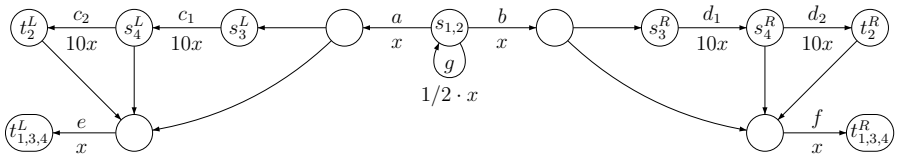


Figure 1: A network formation game in which it is beneficial to select a path that is not simple.

The concept of what we refer to as a dynamic game is old and dates back to Von Neumann's work on *extensive form games* [20]. Most work on RAGs consider the simultaneous setting. However, there have been different takes on adding dynamicity to RAGs. In [19], the authors refine the notion of NE by considering *lookahead* equilibria; a player predicts the reactions of the other players to his deviations, and he deviates only if the outcome is beneficial. The depth of lookahead is bounded and is a parameter to the equilibria. A similar setting was applied to RAGs in [7], where the players are restricted to choose a *best-response* move rather than a deviation that might not be immediately beneficial. Concurrent ongoing games are commonly used in formal methods to model the interaction between different components of a system (c.f., [1]). In such a game, multiple players move a token on a graph. At each node, each player selects a move, and the transition function determines the next position of token, given the vector of moves the players selected. The objectives of the players refer to the generated path and no costs are involved. Closest to our model is the model of [16], and its subsequent works [8, 10]. They study RAGs in which players arrive and select strategies one by one, yet in one shot.

Our dynamic games differ from all of these games in two aspects. We allow the players to reveal their choice of resources in parts, thus we allow "breaking" the strategies into parts. Moreover, the choices the players make in all these games are either concurrent of sequential, and we allow a mix between the two. These two concepts are natural and general, and can be applied to other games and settings.

The first question that arises in the context of games, and on which we focus in this work, is the existence of a *stable outcome* of the game. In the context of RAGs, the most prominent stability concept is that of a *Nash equilibrium* (NE, for short) – a profile such that no player can decrease his cost by unilaterally deviating from his current strategy. It is well known that every RAG has an NE [23]. The definition of an NE applies to all games, and can also be applied to our dynamic RAGs. As we demonstrate in Example 2.1, the dynamic setting calls for a different stability concept, as some NEs need not be achievable by rational players in the dynamic setting. Essentially, it follows from the fact that rational players take the history of the game into account when they make their choices in intermediate phases, ruling out some choices that are rational only in a concurrent and one-shot setting. To overcome this limitation of NE, the notion of *subgame perfect equilibrium* (SPE, for short) was introduced in [27], which we define formally in Section 2.

Classifying RAGs, we refer to the type of their latency functions as well as the type of the objectives of the players. *Congestion games* [24] are RAGs in which the latency functions are increasing, whereas in *cost-sharing games* [2], each resource has a cost that is split between the players that use it (in particular, the latency functions are decreasing). In terms of objectives, we consider *singleton* RAGs, in

which the objectives of the players are singletons of resources, and *symmetric* RAGs, in which all players have the same objective.

Our most interesting results are in terms of equilibrium existence. It is easy to show, and similar results are well known, that every dynamic RAG with a sequential scheduler has an SPE. The proof uses backwards induction on the tree of all possible outcomes of the game (see Theorem 3.1 for details). One could hope to achieve a similar proof also for schedulers that are not sequential, especially given the fact that every RAG has an NE. Quite surprisingly, however, we show that this is not the case. For congestion games, we show examples of a singleton congestion game and a symmetric congestion game with no SPE. Moreover, the latency function in both cases is linear. On the positive side, we show that singleton and symmetric congestion games are guaranteed to have an SPE for every scheduler. For cost-sharing games, we also show an example with no SPE. In the cost-sharing setting, however, we show that singleton objectives are sufficient to guarantee the existence of an SPE in all schedules. It follows that singleton dynamic congestion games are less stable than singleton dynamic cost-sharing games. This is interesting, as in the on-shot concurrent setting, congestion games are known to be more stable than cost-sharing games in various parameters. One would expect that this "order of stability" would carry over to the dynamic setting, as is the case in other extensions of the traditional setting. For example, an NE is not guaranteed for *weighted* cost-sharing games [9] as well as very restrictive classes of *multiset* cost-sharing games [5], whereas every linear weighted congestion game [12] and even linear multiset congestion game is guaranteed to have an NE [6]. Also, as we detail in Section 4, there are classes of congestion games that are guaranteed to have a *strong equilibrium* whereas the cost-sharing counterpart does not [13].

It is well known that decentralized decision-making may lead to solutions that are sub-optimal from the point of view of society as a whole. In simultaneous games, the standard measures to quantify the inefficiency incurred due to selfish behavior is the *price of anarchy* (PoA) [15] and *price of stability* (PoS) [2]. In both measures we compare against the *social optimum* (SO, for short), namely the cheapest profile. The PoA is the worst-case inefficiency of an NE (that is, the ratio between the cost of a worst NE and the SO). The PoS is the best-case inefficiency of a Nash equilibrium (that is, the ratio between the cost of a best NE and the social optimum). For the dynamic setting we adjust these two measures to consider SPEs rather than NEs, and we refer to them as DPoA and DPoS. We study the equilibrium inefficiency in the classes of games that have SPEs. We show that the DPoA and DPoS in dynamic singleton cost-sharing games as well as dynamic singleton congestion games coincide with the PoA and PoS in the corresponding simultaneous class. As mentioned above, [16, 8, 10] study games in which players arrive one after. Since their games are

sequential, they always have an SPE. They study the *sequential PoA*, and show that it can either be equal, below, or above the PoA of the corresponding class of RAGs.

We then turn to study computational problems for dynamic RAGs. First, we study the problem of deciding whether a given dynamic RAG has an SPE. We show that the problem is PSPACE-complete for both congestion and cost-sharing games. Our lower bound for cost-sharing games implies that finding an SPE in sequential games is PSPACE-hard. To the best of our knowledge, while this problem was solved in [16] for congestion games, we are the first to solve it for cost-sharing games. We also study the problem of finding a schedule that admits an SPE under given constraints on the order the players move, and show that this problem is also PSPACE-complete. Finally, we consider dynamic games in which there is an order on the resources that the players choose. So, if for two resources e_1 and e_2, we have $e_1 \prec e_2$, then a player cannot choose e_1 in a later phase than e_2. The motivation for an order on resources is natural. For example, returning to network formation games, a driver can only extend the path he chooses as the choices are made during driving. We show that all our results carry over to the ordered case.

Due to lack of space, some proofs and examples are given in the appendix.

2 Preliminaries

Resource allocation games For $k \geqslant 1$, let $[k] = \{1, \ldots, k\}$. A *resource-allocation game* (RAG, for short) is a tuple $G = \{[k], E, \{\Sigma_i\}_{i\in[k]}, \{\ell_e\}_{e\in E}\}$, where $[k]$ is a set of k players; E is a set of resources; for $i \in [k]$, the set $\Sigma_i \subseteq 2^E$ is a set of objectives[1] for Player i; and, for $e \in E$, we have that $\ell_e : \mathbb{N} \to \mathbb{R}$ is a latency function. The game proceeds in one-round in which the players select simultaneously one of their objectives. A *profile* $P = \langle \sigma_1, \ldots, \sigma_k \rangle \in \Sigma_1 \times \ldots \times \Sigma_k$ is a choice of an objective for each player. For $e \in E$, we denote by $nused(P, e)$ the number of times e is used in P, thus $nused(P, e) = |\{i \in [k] : e \in \sigma_i\}|$. For $i \in [k]$, the *cost* of Player i in P, denoted $cost_i(P)$, is $\sum_{e\in\sigma_i} \ell_e(nused(P, e))$.

Classes of RAGs are characterized by the type of latency functions and types of objectives. In *congestion* games (CGs, for short), the latency functions are increasing. An exceptionally stable class of CGs are ones in which the latency functions are linear (c.f., [12, 6]); every resource $e \in E$ has two constants a_e and b_e, and the latency function is $\ell_e(x) = a_e \cdot x + b_e$. In *cost-sharing* games (SG, for short), each resource $e \in E$ has a *cost* c_e and the players that use the resource share its cost,

[1] We use "objectives" here rather than "strategies" as the second will later be used for dynamic games.

thus the latency function for e is $\ell_e(x) = \frac{c_e}{x}$. In particular, the latency functions are decreasing. We use DCGs and DSGs to refer to dynamic CGs and dynamic SGs, respectively. In terms of objectives, we study *symmetric* games, where the players' sets of objectives are equal, thus $\Sigma_i = \Sigma_j$ for all $i, j \in [k]$, and *singleton* games, where each $\sigma \in \Sigma_i$ is a singleton, for every $i \in [k]$.

Dynamic resource allocation games A *dynamic* RAG is pair $\mathcal{G} = \langle G, \nu \rangle$, where G is a RAG and $\nu : [k] \to [k]$ is a *scheduler*. Intuitively, in a dynamic game, rather than revealing their objectives at once, the game proceeds in *phases*: in each phase, each player reveals one resource in his objective. Each phase is partitioned into at most k *turns*. The scheduler dictates the order in which the players proceed in a phase by assigning to each player his turn in the phases. If the scheduler assigns the same turn to several players, they select a resource concurrently. Once all players take their turn, a phase is concluded and a new phase begins. There are two "extreme" schedulers: (1) players get different turns, i.e., ν is a permutation, (2) all players move in one turn, i.e., $\nu \equiv 1$. We refer to games with these schedulers as *sequential* and *concurrent*, respectively. Note that ν might not be an onto function. For simplicity, we assume that, for $j > 1$, if turn j is assigned a player, then so is turn $j - 1$. We use t_ν to denote the last turn according to ν, thus $t_\nu = \max_i \nu(i)$.

Let $E_\perp = E \cup \{\perp\}$, where $\perp$ is a special symbol that represents the fact that a player finished playing. Consider a turn $j \in [k]$. We denote by before(j) the set of players that play before turn j; thus before$(j) = \{i \in [k] : \nu(i) < j\}$. A player has full knowledge of the resources that have been chosen in previous phases and the resources chosen in previous turns in the current phase. A strategy for Player i in $\mathcal{G}$ is a function $f_i : (E_\perp^{[k]})^* \cdot (E_\perp^{\text{before}(\nu(i))}) \to E_\perp$. A *profile* $P = \langle f_1, \ldots, f_k \rangle$ is a choice of a strategy for each player. The *outcome* of the game given a profile P, denoted $out(P)$, is an infinite sequence of functions $\pi^1, \pi^2, \ldots$, where for $i \geqslant 1$, we have $\pi^i : [k] \to E_\perp$. We define the sequence inductively as follows. Let $m \geqslant 1$ and $j \in [k]$. Assume $m - 1$ phases have been played as well as $j - 1$ turns in the m-th phase, thus $\pi^1, \pi^2, \ldots, \pi^{m-1}$ are defined as well as $\pi^m_{j-1} : \text{before}(j) \to E_\perp$. We define π^m_j as follows. Consider a player i with $\nu(i) = j$. The resource Player i chooses in the m-th phase is $f_i(\pi^1, \ldots, \pi^{m-1}, \pi^m_{j-1})$. Finally, we define $\pi^m = \pi^m_{t_\nu}$.

We restrict attention to *legal* strategies for the players, namely ones in which the collection of resources chosen by Player i in all phases is an objective in Σ_i. Also, once Player i chooses $\perp$, then he has finished playing and all his choices in future phases must also be $\perp$. Formally, for a profile $P = \langle f_1, \ldots, f_k \rangle$ with $out(P) = \pi^1, \pi^2, \ldots$ and $i \in [k]$, let $out_i(P)$ be $\pi^1(i), \pi^2(i), \ldots$. For $j \geqslant 1$, let $e_j = \pi^j(i)$ be the resource Player i selects in the j-th phase. Thus, $out_i(P)$ is an infinite sequence over $E_\perp$. We say that f_i is legal if (1) there is an index m such that $e_j \in E$ for all $j < m$

and $e_j = \bot$ for all $j \geqslant m$, and (2) the set $\{e_1, \ldots, e_{m-1}\}$ is an objective in Σ_i. (In particular, a player cannot select a resource multiple times nor a resource that is not a member in his chosen objective). We refer to an outcome in which the players use legal strategies as a *legal outcome* and a prefix of a legal outcome as a *legal history*.

In $out(P)$, every player selects a set of resources. The cost of a player is calculated similarly to RAGs. That is, his cost for a resource e, assuming the load on it is γ, is $\ell_e(\gamma)$, and his total cost is the sum of costs of the resources he uses. When the outcome of a profile P in a dynamic RAG coincides with the outcome of a profile Q in a RAG G, we say that P and Q are *matching* profiles.

Equilibria concepts A *Nash equilibrium*[2] (NE, for short) in a game is a profile in which no player has an incentive to unilaterally deviate from his strategy. Formally, for a profile P, let $P[i \leftarrow f_i']$ be the profile in which Player i switches to the strategy f_i' and all other players use their strategies in P. Then, a profile P is a NE if for every $i \in [k]$ and every legal strategy f_i' for Player i, we have $cost_i(P) \leqslant cost_i(P[i \leftarrow f_i'])$. It is well known that every RAG is guaranteed to have an NE [23].

The definition of NE applies to all games, in particular to dynamic ones. Every NE Q in a RAG G matches an NE in a dynamic game $\langle G, \nu \rangle$, for some scheduler ν, in which the players ignore the history of the play and follow their objectives in Q. However, such a strategy is not rational. Thus, one could argue that an NE is not necessarily achievable in a dynamic setting. We illustrate this in the following example.

Example 2.1 *Consider a two-player DCG with resources* $\{a, b\}$*, latency functions* $\ell_a(x) = x$ *and* $\ell_b(x) = 1.5x$*, and objectives* $\Sigma_1 = \Sigma_2 = \{\{a\}, \{b\}\}$*. Consider the sequential scheduling in which Player* 1 *moves first followed by Player* 2*. Since the players' objectives are singletons, the dynamic game consists of one phase. Consider the Player* 2 *strategy* f_2 *that "promises" to select the resource* a *no matter what Player* 1 *selects, thus* $f_2(a) = f_2(b) = a$*. Let* f_1^a *and* f_1^b *be the Player* 1 *strategies in which he selects* a *and* b*, respectively, thus* $f_1^a(\epsilon) = a$ *and* $f_1^b(\epsilon) = b$*. Note that these are all of Player* 1*'s possible strategies. The profile* $P = \langle f_1^b, f_2 \rangle$ *is an NE. Indeed, Player* 2 *pays* 1*, which is the least possible payment, so he has no incentive to deviate. Also, by deviating to* f_1^a*, Player* 1*'s payoff increases from* 1.5 *to* 2*, so he has no incentive to deviate either. Note, however, that this strategy of Player* 2 *is not rational. Indeed, when it is Player* 2*'s turn, he is aware of Player* 1*'s choice. If Player* 1 *plays* f_1^a*, then a rational Player* 2 *is not going to choose* a*, as this results in a cost of* 2*, whereas by* b*, his cost will be* 1.5*. Thus, an NE profile with* f_2 *may*

[2]Throughout this paper, we consider *pure* strategies and deviations, as is the case in the vast literature on RAGs.

not be achievable. □

To overcome this issue, the notion of *subgame perfect equilibrium* (SPE, for short) was introduced. In order to define SPE, we need to define a subgame of a dynamic game. Let $\mathcal{G} = \langle G, \nu \rangle$, where $G = \langle [k], E, \{\Sigma_i\}_{i\in[k]}, \{c_e\}_{e\in E} \rangle$. It is helpful to consider the *outcome tree* $\mathcal{T}_\mathcal{G}$ of $\mathcal{G}$, which is a finite rooted tree that contains all the legal histories of $\mathcal{G}$. Each internal node in $\mathcal{T}_\mathcal{G}$ corresponds to a legal history, its successors correspond to possible extensions of the history, and each leaf corresponds to a legal outcome. Consider a legal history h. We define a dynamic RAG $\mathcal{G}_h$, which, intuitively, is the game as $\mathcal{G}$ after the history h has been played. More formally, the outcome tree of $\mathcal{G}_h$ is the subtree $\mathcal{T}^h_\mathcal{G}$ whose root is the node h. We define the costs in $\mathcal{G}_h$ so that the costs of the players in the leaves of $\mathcal{T}^h_\mathcal{G}$ are the same as the corresponding leaves in $\mathcal{T}_\mathcal{G}$. Assume that h ends at the m-th turn. A profile P in $\mathcal{G}$ corresponds to a trimming of $\mathcal{T}_\mathcal{G}$ in which the internal node h has exactly one child $h \cdot \overline{\sigma}$, where $\overline{\sigma}$ is the set of choices of the players in $\nu^{-1}(m)$ when they play according to their strategies in P. The profile P induces a profile P^h in $\mathcal{G}_h$, where the trimming of $\mathcal{T}^h_\mathcal{G}$ according to P^h coincides with the trimming of $\mathcal{G}$ according to P. We formally define the outcome tree and a subgame in Appendix A.

Definition 2.1 *A profile P is an SPE if for every legal history h, the profile P^h is a NE in $\mathcal{G}_h$.*

Note that the profile $P = \langle f^b_1, f_2 \rangle$ in the example above is an NE but not an SPE. Indeed, for the history $h = a$, the profile P^h is not a NE in $\mathcal{G}_h$ as Player 2 can benefit from unilaterally deviating as described above.

3 Existence of SPE in Dynamic Congestion Games

It is easy to show that every sequential dynamic game has an SPE by unwinding the outcome tree, and similar results have been shown before (c.f., [16]). The proof can be found in Appendix B.

Theorem 3.1 *Every sequential dynamic game has an SPE.*

One could hope to prove that a general dynamic game $\mathcal{G}$ also has an SPE using a similar unwinding of $\mathcal{T}_\mathcal{G}$. Possibly using the well-known fact that every CG is guaranteed to have an NE [23]. Unfortunately, and somewhat surprisingly, we show that this is not possible. We show that (very restrictive) DCGs might not have an SPE. For the good news, we identify a maximal fragment that is guaranteed to have an SPE.

Recall that a CG is singleton when the players' objectives consist of singletons of resources, and a CG is symmetric if all the players agree on their objectives. We start with the bad news and show that symmetric DCGs and singleton DCGs need not have an SPE, even with linear latency functions. We then show that the combination of these two restrictions is sufficient for existence of an SPE in a DCG.

Theorem 3.2 *There is a symmetric linear DCG with no SPE.*

Proof: We first describe a linear DCG with no SPE, and then alter it to make it symmetric. Consider the following three-player linear CG G with resources $E = \{a, a', b, b', c\}$ and linear latency functions $\ell_a(x) = \ell_b(x) = x$, $\ell_{a'}(x) = \frac{3}{4}x$, $\ell_{b'}(x) = 1\frac{1}{4}$, and $\ell_c(x) = x + \frac{2}{3}$. Let $\Sigma_1 = \Sigma_2 = \{\{a, a'\}, \{b, b'\}, \{c\}\}$ and $\Sigma_3 = \{\{c\}, \{a', b\}\}$. Consider the dynamic game $\mathcal{G}$ in which Players 1 and 2 move concurrently followed by Player 3. Formally, $\mathcal{G} = \langle G, \nu \rangle$, where $\nu(1) = \nu(2) = 1$ and $\nu(3) = 2$.

We claim that there is no SPE in $\mathcal{G}$. Note that since the players' objectives are disjoint, then once a player reveals the first choice of resource, he reveals the whole objective he chooses, thus we analyze the game as if it takes place in one phase in which the players' reveal their whole objective. The profiles in which Players 1 and 2 choose the same objective are clearly not a SPE as they are not an NE in the game $\mathcal{G}_\epsilon$. As for the other profiles, in Figure 2, we go over half of them, and show that none of them is an SPE. The other half is dual. The root of each tree is labeled by the objectives of Players 1 and 2, and its branches according to Player 3's objectives. In the leaves we state Player 3's payoff. In an SPE, Player 3 performs a best-response according to the objectives he observes as otherwise the subgame is not in an NE. We depict his choice with a bold edge. Beneath each tree we note the payoffs of all the players in the profile, and the directed edges represent the player that can benefit from unilaterally deviating. In Appendix C, we construct a symmetric DCG $\mathcal{G}'$ by altering the game $\mathcal{G}$ above. We do this by adding a fourth player and three new resources so that $\mathcal{G}'$ simulates $\mathcal{G}$. □

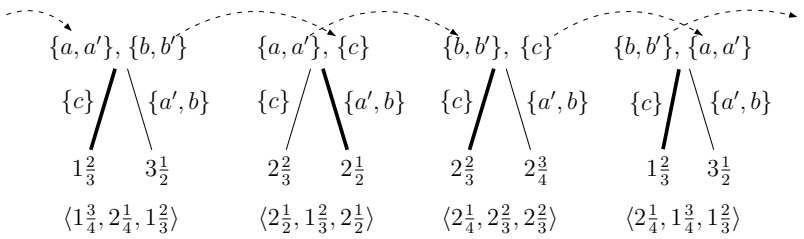


Figure 2: Profiles in the game $\mathcal{G}$ with no SPE.

Theorem 3.3 *There is a singleton linear DCG with no SPE.*

Proof: Consider the four-player linear singleton CG G with resources $E = \{a, b, c, d\}$ and linear latency functions $\ell_a(x) = 4\frac{1}{2} \cdot x$, $\ell_b(x) = 2\frac{1}{2} \cdot x$, $\ell_c(x) = 3x$, and $\ell_d(x) = 4x$. Let $\Sigma_1 = \{\{a\}, \{c\}\}$, $\Sigma_2 = \{\{b\}, \{d\}\}$, $\Sigma_3 = \{\{b\}, \{c\}\}$, and $\Sigma_4 = \{\{c\}, \{d\}\}$. Consider the dynamic game $\mathcal{G}$ in which Players 1 and 2 move concurrently, then Player 3, and finally Player 4. Formally, $\mathcal{G} = \langle G, \nu \rangle$, where $\nu(1) = \nu(2) = 1$, $\nu(3) = 2$, and $\nu(4) = 3$.

We go over all the profiles in $\mathcal{G}$ and show that none of them is an SPE. The profiles are depicted in Figure 3. Similar to Theorem 3.2, the root of each tree is labeled by the objective of Players 1 and 2, its branches according to Players 3 and 4's objectives, and in the leaves we state the payoffs of Players 3 and 4 assuming they choose their best choice given the other players' choices. □

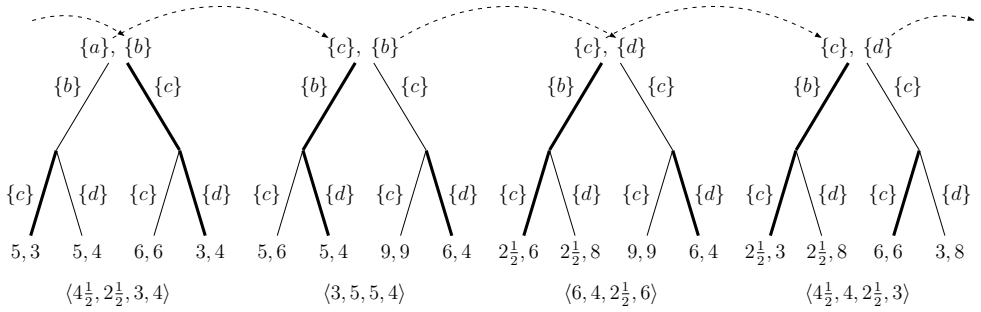


Figure 3: The profiles of the singleton DCG with no SPE. Bold edges depict Players 3 and 4's best choices given the other players choices. Directed edges represent the player that can benefit from unilaterally deviating.

We now prove that combining the two restrictions does guarantee the existence of SPE. We note that while our negative results hold for linear DCGs, which tend to be stabler than other DCGs, our positive result holds for every increasing latency functions.

Theorem 3.4 *Every symmetric singleton DCG has an SPE.*

Proof: Consider a symmetric singleton DCG $\mathcal{G} = \langle G, \nu \rangle$. Recall that since G is a singleton game, every outcome of $\mathcal{G}$ consists of one phase. Let P be an NE in G (recall that according to [23] an NE exists in every CG). Since G is symmetric, we can assume that, for $1 \leqslant j < k$, the players that move in the j-th turn do not pay more than the players that move after them. Formally, for $i, i' \in [k]$, if $\nu(i) < \nu(i')$, then $cost_i(P) \leqslant cost_{i'}(P)$. In particular, the players who move in the first turn pay the least, and the players that move in the last turn pay the most. We construct a profile Q in $\mathcal{G}$ and show that it is an SPE. Intuitively, in Q, the players follow their objectives in P assuming the previous players also follow it. Since the costs are increasing with turns, if Player i deviates, a following Player j will prefer switching

resources with Player i and also switching the costs. Thus, the deviation is not beneficial for Player i. In Appendix D, We construct Q formally and prove that it is an SPE. □

4 Existence of SPE in Dynamic Cost-sharing Games

Cost sharing games tend to be less stable than congestion games in the concurrent setting; for example, very simple fragments of multiset cost-sharing games do not have an NE [5] while linear multiset congestion games are guaranteed to have an NE [6]. In this section we are going to show that, surprisingly, there are classes of games in which an SPE exists only in the cost-sharing setting. Still, SPE is not guaranteed to exist in general DSGs. We start with the bad news.

Theorem 4.1 *There is a DSG with no SPE.*

Proof: Consider the following four-player SG G with resources $E = \{a, a', a'', b, b', b'', c, c', c''\}$ and costs $c_a = c_b = c_c = 6$, $c_{a'} = c_{b'} = c_{c'} = 4$, and $c_{a''} = c_{b''} = c_{c''} = 3$. Let $\Sigma_1 = \{\{a, a'\}, \{b, b''\}\}$, $\Sigma_2 = \{\{b, b'\}, \{c, c''\}\}$, $\Sigma_3 = \{\{c, c'\}, \{a, a''\}\}$, and $\Sigma_4 = \{\{a, a'\}, \{b, b'\}, \{c, c'\}\}$. Consider the dynamic game $\mathcal{G}$ in which players 1,2, and 3 move concurrently followed by Player 4. Formally, $\mathcal{G} = \langle G, \nu \rangle$, where $\nu(1) = \nu(2) = \nu(3) = 1$ and $\nu(4) = 2$.

We claim that there is no SPE in $\mathcal{G}$. Similar to Theorem 3.2, since the players' objectives are disjoint, we analyze the game as if it takes place in one phase. In Figure 4, we depict some of the profiles and show that none of them are an SPE. As in Theorem 3.2, the root of each tree is labeled by the objectives of Players $1, 2$, and 3, its branches according to Player 4's choices, and in the leaves we state the cost of Player 4 assuming he chooses his best choice given the other players' choices. Finally, it is not hard to show that every profile not on the cycle of profiles cannot be an SPE. □

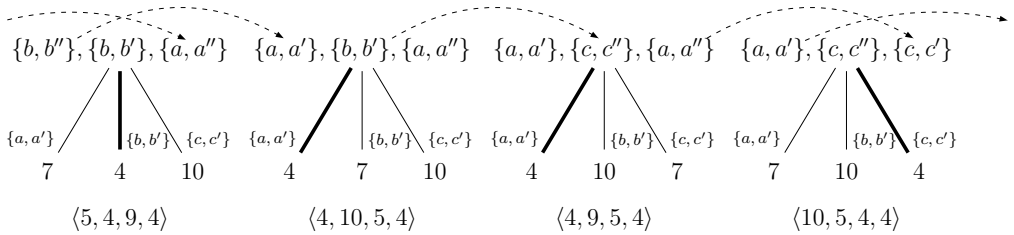


Figure 4: Profiles in the game with no SPE. Bold edges depict Player 4's best choice given the other players choices. Directed edges represent the player that can benefit from unilaterally deviating.

Recall that singleton DCGs are not guaranteed to have an SPE (Theorem 3.3). On the other hand, we show below that singleton DSGs are guaranteed to have an SPE. In order to find an SPE in such a game, we use a firmer notion of an equilibria in SGs.

A *strong equilibria* (SE, for short) [3] which is stable against deviations of *coalitions* of players rather than deviations of a single player as in NEs. Formally, consider a singleton SG $G = \langle [k], E, \{\Sigma_i\}_{i\in[k]}, \{c_e\}_{e\in E}\rangle$, a profile $P = \langle \sigma_1, \ldots, \sigma_k\rangle$, a coalition of players $C \subseteq [k]$, and a joint move $S \in \bigcup_{i\in C} \Sigma_i^{\{i\}}$ for the members of the coalition. We denote by $P[C \leftarrow S] = \langle \sigma'_1, \ldots, \sigma'_k\rangle$ the profile in which the players in C switch to their objective in S, thus $\sigma'_i = S(i)$ for every $i \in C$, and $\sigma'_i = \sigma_i$ for $i \notin C$. We say that S is beneficial for C if it is beneficial for all the members of the coalition, thus for every $i \in C$, we have $cost_i(P[C \leftarrow S]) < cost_i(P)$. We say that P is an SE if there is no coalition that has a beneficial move.

We show a connection between strong equilibria and SPEs in singleton SGs. It is shown in [13] that every singleton SG has an SE.

Theorem 4.2 *Consider a singleton DSG $\mathcal{G} = \langle G, \nu\rangle$. Then, every strong equilibrium in G matches an SPE of $\mathcal{G}$. In particular, every singleton DSG has an SPE.*

Proof: We describe the intuition of the proof and the details can be found in Appendix E. Consider a singleton DSG $\mathcal{G} = \langle G, \nu\rangle$, and let Q be an SE in G. We describe a profile P in $\mathcal{G}$ that matches Q, and we claim that it is an SPE. Consider a history h that ends in the i-th turn. Assume the players that play in h follow their objective in Q. Then, the players who play next, namely these in $\nu^{-1}(i+1)$, also follow Q. Thus, P matches Q. The definition of the strategies in P for histories that do not follow Q is inductive: assume only the players in $\nu^{-1}(i)$ choose differently than in Q, then the subgame $\mathcal{G}_h$ is a singleton DSG. We find a strong equilibrium in $\mathcal{G}_h$ and let the players in $\nu^{-1}(i+1)$ choose according to it.

We claim that P is an SPE. Assume towards contradiction that there is a Player i who can benefit from a unilateral deviation to a resource e. Such a deviation initiates subsequent deviations from players who choose in later turns than Player i. We consider the outcome of the game, and the players that play differently than in Q. Let $I, D \subseteq [k]$ be the set of players whose cost increases and decreases, respectively, with respect to their cost in Q. We make several observations. Since Q is an SE, we have $I \neq \varnothing$. Moreover, there must be a player $j_I \in I$ who deviates to e. Assume Player j_I chooses e' in Q. Since $\mathcal{G}$ is a SG, there must be a player $j_D \in D$ who chooses e' in Q, but deviates following Player i's deviation. We continue recursively and identify a sequence of resources $e = e_1, e_2, \ldots$ such that for every $j \geqslant 1$, there are $j_I \in I$ and $j_D \in D$ such that both players deviate into e_j. Moreover, Players j_I and $(j+1)_D$ use the same objective in Q. A contradiction follows from the fact that

since there are finitely many resources, the sequence has a loop. □

Remark 4.3 One could suspect that existence of strong equilibria in the underlying RAG implies existence of an SPE in the dynamic game. However, [13] shows that singleton CGs are guaranteed to have an SE, while we show in Theorem 3.3 that singleton DCGs are not guaranteed to have an SPE. In fact, [13] shows that every NE in a singleton CG is also an SE, while it is shown in [25] that this is not the case for singleton SGs.

One could also suspect that Theorem 4.2 generalizes to richer types of objectives. That is, we can ask whether, for an DSG $\mathcal{G} = \langle G, \nu \rangle$, an SE in G matches an SPE in $\mathcal{G}$. Theorem 4.1 shows that this is not the case as in the SG there, the profile $\langle \{b, b''\}, \{b, b'\}, \{a, a''\}, \{b, b'\} \rangle$ is an SE.

Remark 4.4 Consider a symmetric DSG $\mathcal{G} = \langle G, \nu \rangle$. The social optimum profile O in G is attained when all the players choose the same cheapest objective, namely the objective with the minimal sum of resource costs. It is not hard to see that O is an NE as a deviation results in a more expensive objective with less sharing. Recall that we study SPE in dynamic games as NE might contain strategies that will not be used by rational players. Consider a profile P in $\mathcal{G}$ that matches O (note that there can be many such profiles, and some can consist of strategies that are chosen by rational players). The same arguments stated above imply that P is an NE. Nevertheless, P may not be an SPE, as $\mathcal{G}$ might contain a subgame with no SPE.

5 Equilibrium Inefficiency

It is well known that decentralized decision-making may lead to sub-optimal solutions from the point of view of society as a whole. We define the cost of a profile P, denoted $cost(P)$, to be $\sum_{i \in [k]} cost_i(P)$. We denote by OPT the cost of a social-optimal solution; i.e., $OPT = \min_P cost(P)$. Two standard measures that quantify the inefficiency incurred due to self-interested behavior are the *price of anarchy* (PoA) [15, 21] and *price of stability* (PoS) [2, 26]. The PoA is the worst-case inefficiency of an NE; The PoA of a game G is the ratio between the cost of the most expensive NE and the cost of the social optimum. The PoS measures the best-case inefficiency of an NE, and is defined similarly with the cheapest NE. The PoA of a family of games $\mathcal{F}$ is $\sup_{G \in \mathcal{F}} \text{PoA}(G)$, and the definition is similar for PoS.

In dynamic games we consider SPE rather than NE. We adapt the definitions above accordingly, and we refer to the new measures as *dynamic PoA* and *dynamic PoS* (DPoA and DPoS, for short). We study the equilibrium inefficiency in the classes of games that are guaranteed to have an SPE, namely singleton DSGs and symmetric singleton DCGs.

The lower bounds for the PoA and PoS for singleton SG and singleton symmetric CGs follow to the dynamic setting as we can consider the scheduler in which all players choose simultaneously in the first turn. The upper bounds on the DPoS for singleton symmetric DCGs follow from the fact that every NE in singleton symmetric CGs matches an SPE in the corresponding dynamic game. For singleton DSG, recall that an SE in the traditional game matches an SPE in the dynamic game. It is shown in [29] that singleton SGs have an SE whose cost is at most $\log(k) \cdot OPT$, which coincides with the $\log(k)$ lower bound. Finally, the upper bound on the DPoA for singleton DSGs follows from the same argument as traditional games. For congestion games, it follows by applying a recent result by [10] to our setting. The details can be found in Appendix F.

Theorem 5.1 *The DPoA and DPoS in singleton DSGs and singleton symmetric DCGs coincide with the PoA and PoS in singleton SGs and singleton symmetric CGs, respectively.*

6 Deciding the Existence of SPE

In the previous sections we showed that dynamic RAGs are not guaranteed to have an SPE. A natural decision problem arises, which we refer to as ∃SPE: given a dynamic RAG, decide whether it has an SPE. We show that the problem is PSPACE-complete in DSGs as well as DCGs. We start with the lower bound. The crux of the proof is given in the following lemma.

Lemma 6.1 *Given a QBF instance ψ, there is a fully sequential game $\mathcal{G}_\psi$ that is either a DCG or a DSG, and two constants $\gamma, \delta > 0$, such that in every SPE P in $\mathcal{G}_\psi$, (1) if ψ is true, then $cost_1(P) < \gamma$, and (2) if ψ is false, then $cost_1(P) > \delta$.*

Proof: For DCGs, such a construction is described in [16], which uses a construction by [28] in order to simulate the logic of a NAND gate by means of a CG. For SGs, we describe the construction below, which is inspired by the construction in [16].

The input to the TQBF problem is a Boolean circuit ψ with inputs $x_1, \ldots, x_n$, where the variables are partitioned into sets of *existential* variables $E_1, \ldots, E_m$ and *universal* variables $A_1, \ldots, A_m$. We say that ψ is *true* (in which case it is in TQBF) iff $\exists E_1 \forall A_1 \ldots \exists E_m \forall A_m \psi$. Wlog, we assume that ψ is composed only of NAND gates. This is indeed wlog from every Boolean circuit φ we can construct an equivalent circuit φ', with only NAND gates in polynomial time.

We describe a DSG that simulates the logic of a NAND gate. We refer to the game as a *gate game* (see an example in Figure 5). The gate game simulates a gate

with two inputs and r outputs. It has $r+2$ players, where the first two players correspond to the two inputs of the gate, and we refer to them as I_1 and I_2, and the other players, which we refer to as $O_1, \ldots, O_r$, correspond to the outputs to the gate. The resources are $\{i_0^j, i_1^j : j \in \{1,2\}\} \cup \{g^j, o_0^j, o_1^j : j \in \{1,\ldots,r\}\}$ standing for *input*, *gate*, and *output* resources. The costs of the input resources is 1, the costs of the gate resources is 3ϵ, and every 0-output resource costs 1 while 1-output resources cost $1 + 1.1\epsilon$. Each player has two objectives: a 1 objective and a 0 objective. For $j = 1, 2$, let $\Sigma_{I_j} = \{\{i_0^j\}, \{i_1^j, g^1, \ldots, g^r\}\}$. For $j = 1, \ldots, r$, let $\Sigma_{O_j} = \{\{o_1^j\}, \{g^j, o_0^j\}\}$. Finally, the game is sequential. The exact order is not important as long as the input players play before the output players.

In Appendix G we prove that a gate game simulates the logic of a NAND gate. Namely, in the unique SPE of the game, the output players select their 1-objectives iff the input players select their 0-objectives. We also show how to connect gate games to construct a game that simulates the logic of a given circuit. □

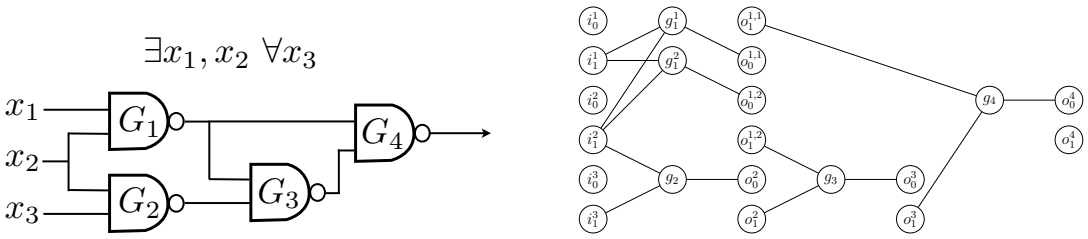


Figure 5: An input ψ to TQBF and the resulting dynamic game. An edge between two resources represents the fact that they belong to the same objective of one of the players.

To conclude the lower-bound proof, we combine between the game that is constructed in Lemma 6.1 and a game that has no SPE as in the examples we show in the previous sections. For the upper bound, consider a dynamic RAG $\mathcal{G}$, and let $\mathcal{T}_\mathcal{G}$ be the outcome tree of $\mathcal{G}$. Recall that there is a one-to-one correspondence between leaves in $\mathcal{T}_\mathcal{G}$ and legal outcomes of $\mathcal{G}$. In order to decide in PSPACE whether $\mathcal{G}$ has an SPE, we guess a leaf l in $\mathcal{T}_\mathcal{G}$ and verify that it is an outcome of an SPE. Thus, we ask if there is an SPE P in $\mathcal{G}$ whose outcome corresponds to l.

Theorem 6.2 *The ∃SPE problem is PSPACE-complete for dynamic RAGs.*

7 Efficient Stable Scheduling

The underlying assumption in game theory is that the players are selfish yet an authority may construct components of the game, and its challenge is to do so in a way that leads to stability. In some settings, the RAG is fixed and the authority

only has the power schedule the players. We assume that we are given a set S of constraints on schedulers that the authority can impose. A constraint $s \in S$ is either a *sequential constraint*, of the form $i_1 < i_2$, for $i_1, i_2 \in [k]$, stating that Player i_1 moves before Player i_2, or a *concurrent constraint*, of the form $i_1 = i_2$, stating that the Players i_1 and i_2 move concurrently. Scheduling has a price. The input also contains a value function $c : S \to \mathbb{Q}$ that assigns to each constraint a cost or a reward. Intuitively, when $c(s) = \gamma \geqslant 0$, it means that the authority pays γ in order to force s in ν. Then, when $c(s) = \gamma \leqslant 0$, it means that the authority is rewarded γ if ν respects s. Consider a scheduler ν. Let $R \subseteq S$ be the set of constraints satisfied by ν. Then, $cost_{S,c}(\nu) = \sum_{s \in R} c(s)$.

The *budgeted scheduling problem* (BS problem, for short) gets as input a RAG G, a set of constraints S, a value function $c : S \to \mathbb{Q}$, and a budget β. The goal is to decide whether there is a scheduler ν with $cost_{S,c}(\nu) \leqslant \beta$ such that the dynamic game $\langle G, \nu \rangle$ has an SPE. The proof of the following theorem can be found in Appendix I.

Theorem 7.1 *The BS problem is PSPACE-complete.*

8 Games with Ordered Resources

In many settings it makes sense to restrict the order in which the players select their objectives. For example, recall that a network formation games is played on a network, and each player chooses a path that connect his source and target vertices. When decisions are taken while the path is being generated, it is often the case that a player cannot select the edges on his path in any order. Rather, he must extend his path one edge at a time. The corresponding constraints state that if a player uses an edge $\langle u, v \rangle$, then, unless u is the source vertex, in a previous phase an edge that ends in u must have been chosen.

Generally speaking, we assume a dynamic game is also given by a partial order $\prec$ on the resources. A legal strategy is one that does not violate the order. Thus, if the sequence of choices for a player in some outcome is $e_1, e_2, \ldots, e_m$, then there are no $1 \leqslant i < j \leqslant m$ such that $e_j \prec e_i$. We restrict the players to choose only legal strategies and we assume there is at least one legal strategy for each player.

Theorem 8.1 *Our results in terms of SPE existence, equilibrium inefficiency, and computational complexity coincide for ordered dynamic games and dynamic games.*

Proof: Note that ordered dynamic games generalize dynamic games as we allow using the empty partial order. So, all our bad news follow to this setting. In terms of equilibrium existence and inefficiency, our good news are for singleton games. Such

games cannot be ordered. As for computational complexity, our upper bounds can easily be adapted to ordered games. □

9 Discussion and Future Research

We studied an addition of dynamics to resource allocation games. The dynamics we studied extend traditional RAGs in two natural ways: the players select one resource at a time rather than choosing their whole objective in one shot, and the players perform choices in an ordering that is neither sequential nor fully concurrent. A future research direction is to apply these two natural extensions to other games.

We studied stable solutions of dynamic RAGs while focusing on SPE. We showed classes of dynamic RAGs that are guaranteed to have an SPE, and showed that most classes need not have one. The existence of a stable outcome is crucial, and a natural future research direction is a quest for stable classes. In addition to restrictions on the setting, like symmetry or singleton objectives, one can think about different mechanisms for allocating costs to resources – possibly ones that take the dynamics into account, and, similar to issues we have studied in the paper, the ability to stability via controlled scheduling.

Finally, the question of the benefits of choosing redundant resources becomes of special interest in the dynamic setting. Recall that we require the set of resources that a player chooses in an outcome of a game to be one of his objectives. Allowing players to use redundant resources amounts to weakening this requirement and replacing it by one in which the set (in fact, multiset) of resources that a player chooses contains one of his objectives. In the traditional setting, it is not hard to see that choosing a redundant resource cannot be beneficial. In the dynamic setting this is not the case, as a player may choose a redundant resource in order to mislead the other players or in order to "pass" in his turn (a variant of Example 1.1, in which we remove Player 1's objectives that include the resource g, demonstrates this). An interesting direction for future research is the study of a game in which players are allowed to choose redundant resources.

References

[1] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, 2002.

[2] E. Anshelevich, A. Dasgupta, J. Kleinberg, E. Tardos, T. Wexler, and T. Roughgarden. The price of stability for network design with fair cost allocation. *SIAM J. Comput.*, 38(4):1602–1623, 2008.

[3] R. Aumann. Acceptable points in games of perfect information. *Contributions to the Theory of Games*, 4:287–324, 1959.

[4] G. Avni and O. Kupferman. Synthesis from component libraries with costs. In *Proc. 25th Int. Conf. on Concurrency Theory*, pages 156–172, 2014.

[5] G. Avni, O. Kupferman, and T. Tamir. Network-formation games with regular objectives. In *Proc. 17th Int. Conf. on Foundations of Software Science and Computation Structures*, volume 8412 of *Lecture Notes in Computer Science*, pages 119–133. Springer, 2014.

[6] G. Avni, O. Kupferman, and T. Tamir. Congestion games with multisets of resources and applications in synthesis. In *Proc. 35th Conf. on Foundations of Software Technology and Theoretical Computer Science*, pages 365–379, 2015.

[7] V. Bilò, A. Fanelli, and L. Moscardelli. On lookahead equilibria in congestion games. In *Web and Internet Economics - 9th International Conference, WINE 2013, Cambridge, MA, USA, December 11-14, 2013, Proceedings*, pages 54–67, 2013.

[8] J. R. Correa, J. de Jong, B. de Keijzer, and M. Uetz. The curse of sequentiality in routing games. In *Web and Internet Economics - 11th International Conference, WINE 2015, Amsterdam, The Netherlands, December 9-12, 2015, Proceedings*, pages 258–271, 2015.

[9] H. Chen and T. Roughgarden. Network design with weighted players. *Theory Comput. Syst.*, 45(2):302–324, 2009.

[10] J. de Jong and M. Uetz. The sequential price of anarchy for atomic congestion games. In *Web and Internet Economics - 10th International Conference, WINE 2014, Beijing, China, December 14-17, 2014. Proceedings*, pages 429–434, 2014.

[11] D. Fotakis. Stackelberg strategies for atomic congestion games. In *Algorithms - ESA 2007, 15th Annual European Symposium, Eilat, Israel, October 8-10, 2007, Proceedings*, pages 299–310, 2007.

[12] T. Harks and M. Klimm. On the existence of pure Nash equilibria in weighted congestion games. *Math. Oper. Res.*, 37(3):419–436, 2012.

[13] R. Holzman and N. Law-Yone. Strong equilibrium in congestion games. *Games and Economic Behavior*, 21(1-2):85–101, 1997.

[14] R. Koch and M. Skutella. Nash equilibria and the price of anarchy for flows over time. *Theory Comput. Syst.*, 49(1):71–97, 2011.

[15] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. *Computer Science Review*, 3(2):65–69, 2009.

[16] R. P. Leme, V. Syrgkanis, and E. Tardos. The curse of simultaneity. In *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 60–67, 2012.

[17] Y. Lustig and M.Y. Vardi. Synthesis from component libraries. *STTT*, 15(5-6):603–618, 2013.

[18] I. Milchtaich. Congestion games with player-specific payoff functions. *Games and Economic Behavior*, 13(1):111 – 124, 1996.

[19] V. S. Mirrokni, N. Thain, and A. Vetta. A theoretical examination of practical game playing: Lookahead search. In *Algorithmic Game Theory - 5th International Symposium, SAGT 2012, Barcelona, Spain, October 22-23, 2012. Proceedings*, pages 251–262, 2012.

[20] J. Neumann. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320.

[21] C. H. Papadimitriou. Algorithms, games, and the internet. In *Proc. 33rd ACM Symp. on Theory of Computing*, pages 749–753, 2001.

[22] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. 16th ACM Symp. on Principles of Programming Languages*, pages 179–190, 1989.

[23] R.W. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2:65–67, 1973.

[24] T. Roughgarden and E. Tardos. How bad is selfish routing? *Journal of the ACM*, 49(2):236–259, 2002.

[25] O. Rozenfeld and M. Tennenholtz. Strong and correlated strong equilibria in monotone congestion games. In *Internet and Network Economics, Second International Workshop, WINE 2006, Patras, Greece, December 15-17, 2006, Proceedings*, pages 74–86, 2006.

[26] A. S. Schulz and N. E. Stier Moses. On the performance of user equilibria in traffic networks. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 12-14, 2003, Baltimore, Maryland, USA.*, pages 86–87, 2003.

[27] R. Selten. Spieltheoretische behandlung eines oligopolmodells mit nachfrageträgheit. *Zeitschrift für die gesamte Staatswissenschaft*, 121, 1965.

[28] A. Skopalik and B. Vöcking. Inapproximability of pure nash equilibria. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 355–364, 2008.

[29] V. Syrgkanis. The complexity of equilibria in cost sharing games. In *Proc. of the 6th International Conference on Internet and Network Economics*, WINE'10, pages 366–377, 2010.

A Full Definitions: Outcome Trees and Subgames

We denote by $\mathcal{T}_\mathcal{G}$, the outcome tree of a dynamic RAG $\mathcal{G} = \langle G, \nu \rangle$. The root of $\mathcal{T}_\mathcal{G}$ corresponds to the empty prefix ϵ. Recall that t_ν is the maximal turn in ν. Let $m \geqslant 1$, and $j \in [t_\nu]$, and consider a node that corresponds to a prefix h of a legal outcome, after $m-1$ phases have been played as well as $j-1$ turns in the m-th phase. Thus, $h = \pi^1, \ldots, \pi^{m-1}, \pi^m_{j-1}$, where $\pi^l_r : [k] \to E_\perp$ and $\pi^m_{j-1} : \text{before}(j) \to E_\perp$. We say that h is *controlled* by the players in $\nu^{-1}(j)$. Note that if ν is sequential, then each node is controlled by exactly one player. The children of the node h contain all the possible extensions of h with a legal *joint move* by the players in $\nu^{-1}(j)$. Thus, a child h' of h is a prefix of a legal outcome and it is of the form $h' = \pi^1, \ldots, \pi^{m-1}, \pi^m_j$, where $\pi^m_j : \text{before}(j+1) \to E_\perp$, and π^m_j and π^m_{j-1} agree on before(j). The edge $\langle h, h' \rangle$ in $\mathcal{T}_\mathcal{G}$ corresponds to the joint move $\overline{\sigma} : \nu^{-1}(j) \to E_\perp$, where for every $i \in \nu^{-1}(j)$ we have $\overline{\sigma}(i) = \pi^m_j(i)$. We sometimes use $h \cdot \overline{\sigma}$ to refer to h'. Note that if $j = t_\nu$, then h' is of the form $\pi^1, \ldots, \pi^m, \pi^{m+1}_1$. Finally, h is a leaf if $j = t_\nu$ and all players have finished playing in the m-th phase, thus the choices of all the players in the next phases must be $\perp$. Clearly, $\mathcal{T}_\mathcal{G}$ is a finite tree.

Consider a profile $P = \langle f_1, \ldots, f_k \rangle$. It is possible to trim $\mathcal{T}_\mathcal{G}$ according to P so that each internal node h has exactly one child $h \cdot \overline{\sigma}$, where $\overline{\sigma}$ is the joint objective in which the players who control h follow P. That is, for each Player i who controls h we have $\overline{\sigma}(i) = f_i(h)$. Note that by trimming $\mathcal{T}_\mathcal{G}$ according to P, we leave exactly one leaf l that is reachable from the root. Note that both $out(P)$ as well as the leaf l correspond to the same profile in the underlying RAG G. For every $i \in [k]$, the cost of Player i in l is $cost_i(P)$.

We proceed to define a subgame. Let $h = \pi^1, \ldots, \pi^{m-1}, \pi^m_{j-1}$ be a legal history. Note that $m-1$ phases have been played as well as $j-1$ turns in the m-th phase, and the players that should play next are $\nu^{-1}(j)$. The subgame $\mathcal{G}_h = \langle G_h, \nu_h \rangle$ of $\mathcal{G}$ is a dynamic RAG with $G_h = \langle [k], E, \{\Sigma^h_i\}_{i \in [k]}, \{f_e\}_{e \in E} \rangle$, where the sets Σ^h_i of objectives are defined as follows. Consider $i \in [k]$. Let $e_1, \ldots, e_m$ be the choices of Player i in h over $E_\perp$. Let $m' \leqslant m$ be the last index such that $e_{m'} \in E$. Let $\sigma^h_i = \{e_1, \ldots, e_{m'}\}$ be the edges in E that Player i collects during h, and let $nused(h, e)$ be the load

generated on e in h by all the players, thus $nused(h,e) = |\{i \in [k] : e \in \sigma_i^h\}|$. The set of objectives of Player i in G_h is Σ_i^h. Each objective in Σ_i^h corresponds to an objective in Σ_i (that is, Player i's objectives in G), minus the resources that have been collected in h, thus $\Sigma_i^h = \{\sigma \backslash \sigma_i^h : \sigma \in \Sigma_i\}$. The cost Player i pays in a profile $P = \langle \sigma_1, \ldots, \sigma_k \rangle$ takes into account also the use of resources in the history h, thus $cost_i(P) = \sum_{e \in (\sigma_i(e) \cup \sigma_i^h(e))} \ell_e(nused(h,e) + nused(P,e))$. Recall that $j-1$ turns have been played in the last phase in h. Thus, we "shift" the scheduler ν_h by j. For $l \in [k]$, the players $\nu^{-1}(l)$ who are scheduled to play in the l-th turn by ν, are scheduled to play in the $\big((l-j) \mod k\big) + 1$ turn by ν_h. In particular, for every $i \in \nu^{-1}(j)$, we have $\nu_h(i) = 1$. Finally, consider a strategy f in $\mathcal{G}$. We define a strategy f^h in the game $\mathcal{G}_h$ by $f^h(x) = f(h \cdot x)$. Given a profile $P = \langle f_1, \ldots, f_k \rangle$, the corresponding profile in $\mathcal{G}_h$ is $P^h = \langle f_1^h, \ldots, f_k^h \rangle$.

B Proof of Theorem 3.1

Given a sequential game $\mathcal{G}$, we construct an SPE by "unwinding" the outcome tree $\mathcal{T}_\mathcal{G}$ in a backwards inductive manner. Consider an internal node h in $\mathcal{T}_\mathcal{G}$. Since $\mathcal{G}$ is sequential, there is a unique Player i who controls h. If h is a leaf, then $\perp$ is the only choice Player i can make, and we set his strategy to select $\perp$. Assume h is an internal node, and let $e_1, \ldots, e_m$ be the possible resources Player i can select in h. Let $h_1, \ldots, h_m$ be the children of the node h in $\mathcal{T}_\mathcal{G}$. Assume by induction that there is an SPE in the games $\mathcal{G}_{h_1}, \ldots, \mathcal{G}_{h_m}$, and let $\gamma_1, \ldots, \gamma_m$ be Player i's costs in these SPEs. We set Player i's choice in h to be a resource e_l that minimizes his payment, thus $\gamma_l = \min\{\gamma_1, \ldots, \gamma_m\}$. Clearly, this profile is an NE in the subgame $\mathcal{G}_h$. We refer to this choice of Player i as a *best response* to the history h that he observes. Note that every choice of Player i that achieves a higher cost is not an NE. In particular, if at every node h there is a unique best response for the player that controls h, then there is a unique SPE in $\mathcal{G}$.

C Proof of Theorem 3.2

To conclude the proof, we construct a symmetric CG G' by altering the game G above. We add a fourth player and three new resources d, e, and f, with latency functions $\ell_d(x) = 10x$, $\ell_e(x) = 25x$, and $\ell_f(x) = 30$. The other resources are as in G. The objectives of the players' are symmetric. They consist of $\Sigma_1 \cup \Sigma_2$, where we add d to every strategy in Σ_1 and e to every strategy in Σ_2. We also add a singleton objective $\{f\}$. Formally, the objectives are $\{\{f\}\} \cup \{\sigma \cup \{d\} : \sigma \in \Sigma_1\} \cup \{\sigma \cup \{e\} : \sigma \in \Sigma_2\}$. Finally, we define a scheduler ν' that is similar to ν only that Player 4

moves last, thus $\nu'(1) = \nu'(2) = 1$, $\nu'(3) = 2$, and $\nu'(4) = 3$.

We claim that $\mathcal{G}' = \langle G', \nu' \rangle$ has no SPE. We claim that in every profile that is a candidate to be an SPE, the choice of Players 1 and 2 in the first phase is d, the choice of Player 3 is e, and the choice of Player 4 is f. This follows from the following three properties: (1) the costs of these three resources is much higher than that of the other resources, so the players' best response in the first phase is to minimize the cost they pay for them, (2) using d at most twice is more beneficial than using e, and using d three times is more costly than using e once, and (3) using e once is more beneficial than using f once, and using f once is more beneficial than using e twice. Once the first phase is played, the analysis is the same as in the game $\mathcal{G}$, which we proved not to have an SPE.

D Proof of Theorem 3.4

We construct a profile $Q = \langle f_1, \ldots, f_k \rangle$ in $\mathcal{G}$ and show that it is an SPE. Consider a history h. For a resource $e \in E$, we define $nused(h, e)$ and $nused(P, e)$ to be the loads on e in h and P, respectively. We say that h is *consistent with* P if for every $e \in E$, we have $nused(h, e) \leqslant nused(P, e)$. When P is clear from the context we do not state it implicitly.

We first define the strategies in Q w.r.t. consistent histories. Consider such a history h that ends before the j-th turn, thus the players who control the node h in $\mathcal{T}_\mathcal{G}$ are $\nu^{-1}(j)$. We define the strategies in Q as if the players in $\nu^{-1}(j)$ move in a sequential order. Let $i_1, i_2, \ldots, i_n$ be an arbitrary order on the players in $\nu^{-1}(j)$. Then, for $1 \leqslant l \leqslant n$, we define the strategy f_{i_l} of Player i_l to perform a best response to the objectives of players in before(j), whose objectives he observes, and as if he also observes the objectives of the players $i_1, \ldots, i_{l-1}$ who also move in the j-th turn.

We formally define f_{i_l}. Let h' be the history h concatenated with the objectives of the players $i_1, \ldots, i_{l-1}$. We say that a resource $e \in E$ is *full* in h' if $nused(h', e) = nused(P, e)$. Recall that f_{i_l} is defined w.r.t. h. We define $f_{i_l}(h)$ to be a resource e that is not full after h' and, assuming all resources will eventually be filled up, choosing e will cost the least for Player i_l, thus e minimizes $\{\ell_e(nused(P, e)) : nused(h', e) < nused(P, e)\}$. Note that since players never choose a resource that is full, the history h concatenated with the choices in the j-th turn, is a consistent history.

We have not yet defined the strategies in Q w.r.t. histories that are inconsistent with P. Still, we can show that for every history h that is consistent with P, the profile Q^h is an NE in $\mathcal{G}_h$. Assume that h ends before the j-th turn. We first show that for every Player $i \in \nu^{-1}(j)$, choosing a resource that is not full in h *dominates*

choosing a resource that is already full. That is, no matter how the other players move, it is always more beneficial to choose a resource that is not yet full over one that is full. Then, all that is left in order to prove that Q^h is an NE, is to show that no player can benefit from deviating to a resource that is not full. Such a deviation results in a history that is consistent with P for which we have defined the strategies in Q. Intuitively, such a deviation is not beneficial as we defined Q so that players that move first pay less. By deviating, a player will "switch" costs with a player that moves after him, thus his cost cannot decrease.

We make an observation before proving the claim. Assume the players $\nu^{-1}(j)$ select a joint objective $\overline{\sigma}$ such that $h \cdot \overline{\sigma}$ is a history consistent with P. It is not hard to see that $out(Q^{h \cdot \overline{\sigma}})$ is also consistent with P. Thus, for each Player $i \in \nu^{-1}(j)$, we have $cost_i(Q^h) = cost_{i'}(P)$, for some $i' \in [k]$ (possibly $i' = i$). Note that Q^h is a profile in the game $\mathcal{G}_h$ whereas P is a profile in the game G.

We proceed to prove that for every $i \in \nu^{-1}(j)$, Player i cannot benefit from unilaterally deviating from the profile Q^h in the game $\mathcal{G}_h$. Assume towards contradiction that Player i can benefit from unilaterally deviating and choosing a resource $e \in E$. Recall that since $\mathcal{G}$ is a singleton game, Player i does not move again. Let $\overline{\sigma}$ be the joint move at h according to Q^h, and let $h' = h \cdot \overline{\sigma}$. We distinguish between two cases. First, assume the resource e is full in the history h'. Thus, $nused(h', e) = nused(P, e)$. Note that Player i's deviation forms a history that is not consistent with P and we have not yet defined the strategies in Q w.r.t such histories. Still, we can show that such a deviation is not beneficial. Note that the load on e, no matter how the other players move is at least $nused(P, e) + 1$. Recall that $cost_i(Q^h) = cost_{i'}(P)$, for some $i' \in [k]$. Since G is a symmetric game, Player i' can choose the objective e, thus $e \in \Sigma_{i'}$. Since P is an NE, we have $cost_{i'}(P) \leqslant cost_{i'}(P[i' \leftarrow e])$. Note that the load on e in the profile $P[i' \leftarrow e]$ is exactly $nused(P, e) + 1$. Since G is a congestion game, the cost of e increases with the load on it. Thus, the cost Player i pays after deviating is at least $cost_{i'}(P[i' \leftarrow e])$, which is not beneficial, and we reach a contradiction.

In the second case, the resource e to which Player i deviates is not full in the history h'. Let h'' be the history after Player i's deviation. Then, h'' is a history consistent with P, and Q is defined w.r.t h''. Let P' be the profile in G that corresponds to the outcome of Q in the subgame $\mathcal{G}_{h''}$. Let $l \in [k]$ be the player that selects e in P'. Thus, $cost_i(P') = cost_l(Q)$. Recall that according to Q, at h, Player i should select the cheapest resource that is not full. Since Player i deviates, the resource e is not that cheapest resource. So, Player l moves after Player i, where by "after" we either mean that $\nu(i) < \nu(l)$ or $\nu(i) = \nu(l)$ but i comes before l in the arbitrary order we fix for the players in $\nu(i)$-th turn. Thus, $cost_l(Q) \geqslant cost_i(Q)$, and the deviation is not beneficial for Player i.

To conclude the proof, we complete the definition of the strategies in Q. The definition is inductive. Let h' be a history consistent with P, and let $\overline{\sigma}$ be a joint move such that the history $h = h' \cdot \overline{\sigma}$ is inconsistent. Since G is a singleton symmetric game, so is the game G_h. Thus, we find an NE profile P' in G_h and continue as in the above.

E Proof of Theorem 4.2

Let $P = \langle f_1, \ldots, f_k \rangle$ be the profile that is described in the body of the paper whose outcome coincides with the SE Q. We claim that P is an SPE. Consider a history h that ends before the j-th turn, where assume that the players in h follow their choices in Q. For the other histories the proof is similar. Assume towards contradiction that there is $i \in \nu^{-1}(j)$ such that Player i can benefit from unilaterally deviating to a strategy f'_i. We think of the outcome of $P[i \leftarrow f'_i]$ as a profile $Q' = \langle \sigma'_1, \ldots, \sigma'_k \rangle$ in G. Let $C \subseteq [k]$ be the players whose objectives in Q' differ from their objectives in Q. Let $I, D \subseteq C$ be the partition of C to players whose payoff in Q' increase and decrease, respectively, with respect to their outcome in Q. Formally, if $i \in I$, then $cost_i(Q) \geqslant cost_i(Q')$, and if $i \in D$, then $cost_i(Q) < cost_i(Q')$. Since Q is an SE, the coalition C cannot all benefit, thus $I \neq \varnothing$.

Consider a resource σ such that there is a player $j \in D$ with $\sigma'_j = \sigma$. We make three observations. (1) there is $j_1 \in I$ such that $\sigma'_{j_1} = \sigma$. Otherwise, in the game G, the coalition of players in C that play σ in Q' can benefit from deviating from Q, contradicting the fact that it is an SE.

(2) There is $j_2 \in D$ such that Players j_1 and j_2 choose the same objective in Q, thus $\sigma_{j_1} = \sigma_{j_2}$. Assume otherwise. Let Player j be the first player that "leaves" σ_{j_2}, thus Player j chooses σ_{j_2} in Q and not in Q' and this is the first such player to choose. Note that if $j \in I$, then by staying in σ_{j_2}, his cost cannot increase, thus the deviation is not beneficial, and we reach a contradiction.

(3) Note that $\sigma'_{j_1} \neq \sigma'_{j_2}$. Indeed, players j_1 and j_2 pay the same in Q while Player j_1's payoff in Q' is higher than it is in Q and Player j_2's payoff is lower.

Recall that we assume that Player i's objective in Q differs from his objective in Q' and that $i \in D$. We find a sequence of resources $e_1, e_2, \ldots$, and for every $j \geqslant 1$, we associate with the resource e_j two players $j_i \in I$ and $j_d \in D$ such that $\sigma'_{j_i} = \sigma'_{j_d} = e_j$. First, we define $e_1 = \sigma'_i$ and we associate i with e_1, thus $1_d = i$. Consider $j \geqslant 1$. Assume there is a player $j_d \in D$ with $\sigma'_{j_d} = e_j$. By (1), there is a $j_i \in I$ with $\sigma'_{j_i} = e_j$, and we associate j_i with e_j. By (2), there is $(j+1)_d \in D$ such that Players $(j+1)_d$ and j_i choose the same resource in P_h, thus $\sigma_{j_i} = \sigma_{(j+1)_d}$. We define the next resource in the sequence e_{j+1} to be $\sigma'_{(j+1)_d}$, which by (3) is different

from e_j. We associate Player $(j+1)_d \in D$ with e_{j+1}, and continue inductively.

Since there are finitely many resources, there is a loop $e_r, e_{r+1}, \ldots, e_{s-1}, e_r$ in the sequence above. For $r \leqslant j \leqslant s-1$, note that $cost_{j_d}(Q') = cost_{j_i}(Q')$ and $cost_{j_i}(Q) = cost_{(j+1)_d}(Q)$. Moreover, $cost_{j_i}(Q') \geqslant cost_{j_i}(Q)$ and $cost_{j_d}(Q) > cost_{j_d}(Q')$. Combining the above we have $cost_{j_d}(Q) > cost_{j_d}(Q') = cost_{j_i}(Q') \geqslant cost_{j_i}(Q) = cost_{(j+1)_d}(Q)$, and we reach a contradiction.

F Proof of Theorem 5.1

The lower bounds for both measures are easy. Since we consider singleton games, an NE in a singleton RAG is an SPE in the corresponding dynamic RAG with the concurrent scheduler, i.e., all the players move simultaneously in the first round.

We continue to the upper bounds, and start with SGs. In SGs, we have PoS = $\log(k)$ and PoA = k [2]. The proof for the upper bound for the DPoA is the same as in RAGs: if a player's cost is more than k times his cost in the social optimum in some SPE, then he can deviate to his objective in the social optimum and reduce his cost. For the upper bound on the DPoS, we note that the proof in Theorem **??** shows that a specific profile of an SG is an outcome of an SPE. It is shown in [29] that the cost of this profile is at most $\log(k) \cdot OPT$, thus we have DPoS = $\log(k)$.

We conclude by studying CGs. In singleton symmetric CGs, we have PoA = 4/3 [11] and we are not aware of a tight bound on the PoS. For the DPoS, Theorem 3.4 shows that every NE in a symmetric singleton CG corresponds to an SPE. For the DPoA, we use a claim from [10]. They show that with a sequential scheduler, every outcome of an SPE in a symmetric singleton CG is an NE in the underlying simultaneous game. Their proof works also for schedulers that are not sequential.

G Proof of Lemma 6.1

Note that the players' objectives are disjoint, so we analyze the game as if it takes place in one phase. We claim that a gate game simulates the semantics of a NAND gate. Assume both input players select their 1 objective, which corresponds to the case in which the input of the gate is two 1's, thus the outputs should be 0. For $j \in [r]$, we show that choosing the 0 objective is dominant for Player O_j. Indeed, if Player O_j plays his 0 objective, the cost of the resource g^j is split between three players, so the total cost for Player O_j is $1+\epsilon$. On the other hand, the cost of the 1 objective is $1+1.1\epsilon$. Assume now that one of the input players chooses his 0 objective, thus the outputs should be 1. Then, choosing the 1 objective is dominant for Player O_j as the cost of the 0 objective is at least $1+1.5\epsilon$ while the cost of the

1 objective remains $1 + 1.1\epsilon$.

Next, we describe how to connect two gate games. Let $\mathcal{G}$ and $\mathcal{G}'$ be gate games as in the above. We connect the corresponding gates such that the j-th output of the first gate is fed as the first input to the second gate. In the combined game, the players of $\mathcal{G}$ move before the players in $\mathcal{G}'$. Also, we merge between the output resources of Player O_j in $\mathcal{G}$ with the input resources of Player I'_1 in $\mathcal{G}'$. More formally, let $\Sigma_{O_j} = \{\{o_1^j\}, \{g^j, o_0^j\}\}$ in $\mathcal{G}$ and $\Sigma_{I'_1} = \{\{i_0'^j\}, \{i_1'^j, c'^1, \ldots, c'^r\}\}$ in $\mathcal{G}'$. Then, in the combined game, we have $o_1^j = i_1'^1$ and $o_0^j = i_0'^1$. The cost of the first resource is 1 and the second is $1 + 1.1\epsilon$.

We claim that it is dominant for Player I'_1 to match Player O_j's choice of objective. Intuitively, this follows from the fact that the input and output resources cost much more than the gate resources, so sharing the cost of the first is more beneficial than sharing the second. More formally, assume Player O_j plays his 1 objective. If Player I'_1 chooses his 1 objective, the cost of the resource $i_1'^1$ is split between the two players. So, Player I'_1's cost, no matter what the other players play is a bit over $\frac{1}{2}$. On the other hand, if he chooses his 0 objective, his cost is 1. Choosing the 1 objective in this case is clearly dominant. The case where Player O_j chooses his 0 objective is dual.

We proceed to describe the game $\mathcal{G}_\psi$ (see for example Figure 5). In $\mathcal{G}_\psi$, there is a gate game that corresponds to every NAND gate in ψ, and the games are connected as in the above. For example, in Figure 5, for $i \in [4]$, let $\mathcal{G}_i$ be the gate game that corresponds to the gate G_i. Consider the gate G_1. One of its outputs is fed as input to G_4 and the other to G_3. In $\mathcal{G}_1$, the first pair of output resources are $o_1^{1,1}$ and $o_0^{1,1}$, which also serve as input resources in the gate game $\mathcal{G}_4$. Thus, the set of objectives of the first input player in $\mathcal{G}_4$ is $\{\{o_0^{1,1}\}, \{o_1^{1,1}, g_4\}\}$. Similarly, the second pair of output resources are $o_1^{1,2}$ and $o_0^{1,2}$, which also serve as input resources in the game $\mathcal{G}_3$.

We are left to describe the inputs and output of the circuit and how they interact. Assume the inputs to ψ are the variables $x_1, \ldots, x_n$. Then, in addition to the players who simulate the NAND gates, the game $\mathcal{G}_\psi$ includes also n *variable players*. As in the above, each variable player $j \in [n]$ has a 0 and 1 objective, which corresponds to an assignment to the variable x_j. Player x_j serves as the input player in every gate game that x_j appears in. So, Player x_j has a 0 objective which is $\{i_0^j\}$ and a 1 objective, which includes an input resource i_1^j as well as gate resources as in the above. For example, in Figure 5, the variable x_2 is fed as input to the gates C_1 and C_2, so the set of objectives of Player x_2 is $\{\{i_0^2\}, \{i_1^2, g_1^1, g_1^2, c_2\}\}$.

Recall that there is a partition $E_1, \ldots, E_m, A_1, \ldots, A_m$ of $x_1, \ldots, x_n$. The scheduler in $\mathcal{G}_\psi$ schedules the variable players that correspond to the set of variables E_1 to move first, followed by the players who correspond to the set of variables A_1,

followed by E_2, etc. The other players in $\mathcal{G}_\psi$ who simulate the NAND gates follow according to the rules above.

Finally, there is a special NAND gate in ψ with only one output, where the output of this gate is the output of the whole circuit. We refer to this gate as the *final gate* and to the corresponding gate game as the *final gate game*. In Figure 5, the final gate is C_4. Let Player O be the output player in the final gate game. Recall that each variable player x_j has a 1 objective with a resource i_1^j and a 0 objective with an input resource i_0^j. Further recall that in a gate game, the 0 objective of the output players includes a gate resource. We define Player O's 0 objective to include the gate resource g of the final gate game as well as all the input resources of the universal variable players, thus it is $\{g\} \cup \{i_0^j, i_1^j : x_j \in A_1, \ldots, A_m\}$. The 1 objective of Player O includes the input resources of all the existential variables players as well as another gate resource g' with cost 1.1ϵ, which we use to maintain the gate semantics, thus the 0 objective is $\{c'\} \cup \{i_0^j, i_1^j : x_j \in E_1, \ldots, E_m\}$. We assume Wlog that the number of existential and universal variables in ψ is the same. So, Player O's cost for the input resources is the same in both of his objectives no matter what the other players choose. Thus, the NAND semantics of the output gate is maintained.

Let Player x_1 be the first variable player to move in $\mathcal{G}_\psi$. We claim that if ψ is true, then in every SPE P, we have $cost_{x_1}(P) < \frac{1}{2} + \epsilon'$, and if ψ is false, then $cost_{x_1}(P) \geqslant 1$. Assume that x_1 is an existential variable, and the proof is dual for universal variables. Note that a cost of slightly over $\frac{1}{2}$ for Player x_j is achieved when Player O shares the cost of the input resource Player x_j uses. Thus, Player x_j, as well as all the existential players, have an incentive that the output of the circuit ψ is 1. Indeed, if the output is 0, the Player O shares the input resources with the universal players. Thus, if ψ is true, then the existential players can follow their assignments and guarantee a cost of slightly over $\frac{1}{2}$. On the other hand, if ψ is false, the universal players can guarantee a cost of slightly over $\frac{1}{2}$, making the cost of the existential players slightly over 1.

H Proof of Theorem 6.2

We start with the lower bound and the case of dynamic CGs. Given a QBF instance ψ, we apply the construction in Lemma 6.1 to get a sequential CG $\mathcal{G}_\psi$ and $\gamma, \delta > 0$ such that if ψ is true, then $cost_1(\mathcal{G}_\psi) < \gamma$, and if ψ is false, then $cost_1(\mathcal{G}_\psi) > \delta$. Let $\mathcal{G}_1$ be the first 3-player game with no SPE that is described in Theorem 3.2. We construct a dynamic CG $\mathcal{G}'$, by merging $\mathcal{G}_\psi$ and $\mathcal{G}_1$, such that $\mathcal{G}'$ has an SPE iff ψ is true.

We proceed to construct $\mathcal{G}'$. We add two players $k+1$ and $k+2$ to $\mathcal{G}_\psi$ that take the roles of the first two players in $\mathcal{G}_1$. The role of Player 3 in $\mathcal{G}_1$ is played by Player 1 in $\mathcal{G}_\psi$, so we add the objectives $\{a', b\}$ and $\{c\}$ to Σ_1. Players $k+1$ and $k+2$ move first concurrently followed by Player 1 and the rest of the players in $\mathcal{G}_\psi$, which move sequentially. Recall that in every profile that is a candidate to be an SPE in $\mathcal{G}_1$, Player 3's cost is at least $1\frac{2}{3}$ and at most $2\frac{2}{3}$. We alter the latency functions in $\mathcal{G}_1$ so that in these profiles Player 3 pays at least γ and at most δ. Let P be the SPE in $\mathcal{G}_\psi$. We claim that if $cost_1(P) < \gamma$, then $\mathcal{G}'$ has an SPE. Indeed, it is not hard to see that the following profile is an SPE: Players $k+1$ and $k+2$ choose $\langle\{a, a'\}, \{c\}\rangle$, and the other players play according to P in every subgame. On the other hand, if $cost_1(P) > \delta$, then, there is no SPE. Indeed, Player 1 prefers the objectives originating from $\mathcal{G}_1$ over these in $\mathcal{G}$, and there is no SPE in the topmost subgame as shown in Theorem 3.2.

The case of dynamic SGs is similar. Let $\mathcal{G}_\psi$ be the output of the construction in Lemma 6.1 and $\mathcal{G}_2$ the dynamic SG with no SPE that is described in Theorem 4.1. We merge $\mathcal{G}_\psi$ with $\mathcal{G}_2$ by letting the first player in $\mathcal{G}_\psi$ take the role of the fourth player in $\mathcal{G}_2$ similar to the above. Note that in every candidate profile for an SPE in $\mathcal{G}_2$, Player 4's cost is 4. We alter the costs in $\mathcal{G}_\psi$ so that $\gamma = 4 - \epsilon$ and $\delta = 4 + \epsilon$. Let P be an SPE in $\mathcal{G}_\psi$. We claim that if $cost_1(P) < \gamma$, then $\mathcal{G}'$ has an SPE. Indeed, it is not hard to see that the following profile is an SPE: the first three players in $\mathcal{G}_2$ choose $\langle\{b, b''\}, \{b, b'\}, \{a, a''\}\rangle$, and the other players play according to P in every subgame. On the other hand, if $cost_1(P) > \delta$, then Player 4 always prefers his objective in $\mathcal{G}_2$ over his objective in P, and by the reasoning in Theorem 4.1, there is no SPE in the topmost subgame, and we are done.

We continue to study the upper bound. Consider a dynamic RAG $\mathcal{G}$, and let $\mathcal{T}_\mathcal{G}$ be the outcome tree of $\mathcal{G}$. Recall that there is a one-to-one correspondence between leaves in $\mathcal{T}_\mathcal{G}$ and legal outcomes of $\mathcal{G}$. We guess a leaf l in $\mathcal{T}_\mathcal{G}$ and verify that it is an outcome of an SPE. Thus, we ask if there is an SPE P in $\mathcal{G}$ whose outcome corresponds to l.

In order to decide whether such a profile P exists, we traverse the path from l to the root of $\mathcal{T}_\mathcal{G}$. Consider an internal node h and its child on this path $h \cdot \overline{\sigma}$. Intuitively, our guess of l fixes the joint objective in h to be $\overline{\sigma}$. Consider a Player i who controls h and a resource e'_i different from $\overline{\sigma}(i)$. Thus, $\overline{\sigma}[i \leftarrow e'_i]$ is a unilateral deviation of Player i. We ask if it is possible to define the strategies in the other internal nodes such that the deviation is not beneficial for Player i. Formally, we ask if there is an SPE P' in the subgame $\mathcal{G}_{\overline{\sigma}[i \leftarrow e'_i]}$ in which the cost of Player i is at least his cost in l. This is done by calling the algorithm recursively with a slight change to the base case. Recall that in order for P to be an SPE, it must be an NE in *every* internal node in $\mathcal{T}_\mathcal{G}$. Thus, we verify that an SPE exists in every child $h \cdot \overline{\sigma}'$

of h that we have not yet considered. Thus, $\overline{\sigma}$ and $\overline{\sigma}'$ differ by at least two entries. This is again done by a recursive call to the algorithm. Clearly, the algorithm uses polynomial space.

I Proof of Theorem 7.1

For the upper bound, given an input $\langle G, \langle S, c\rangle, \beta\rangle$, for G with k players, we go over all schedulers $\nu : [k] \to [k]$. For each scheduler ν, we check whether $cost_{S,c}(\nu) \leqslant \beta$ and if so, we use the algorithm in Theorem 6.2 in order to decide whether the game $\langle G, \nu\rangle$ has an SPE. In case it does, we accept. Clearly, the algorithm runs in polynomial space and accepts iff the input is legal.

For the lower bound, we show a reduction from ∃SPE. Given an input $\langle G, \nu\rangle$ to ∃SPE we construct an input $\langle G, \langle S, c\rangle, -(k-1)^2\rangle$ to the BS problem as follows. For every $i_1, i_2 \in [k]$, we add to S a constraint according to the order of i_1 and i_2 in ν. For example, if $\nu(i_1) < \nu(i_2)$, then we add to S the constraint $i_1 < i_2$. All constraints have cost -1. Clearly, the only scheduler that has cost at most $-(k-1)^2$ is ν, thus $\langle G, \nu\rangle \in$ ∃SPE iff $\langle G, \langle S, c\rangle, -(k-1)^2\rangle \in$ BS, and we are done.[3]

[3]Note that often it is possible to force ν with less than $(k-1)^2$ constraints. We have no reason however to minimize the number of constraints in the reduction.

6 Discussion

In the first part of this thesis we studied topics in the border between formal methods and algorithmic game theory. We showed a flow of ideas between the two areas while focusing mainly on different aspects of resource-allocation games. For the direction from formal methods to AGT, we generalized network formation games by allowing the players to have richer objectives than reachability. The paths the players select in the new game are no longer simple, so each strategy uses a multiset of edges rather than a set of edges as in network formation games. We applied a similar generalization to resource allocation games, and introduced and studied multiset resource allocation games. We show that these generalizations reduce stability in most cases while maintaining it in others.

Another extension of classic resource allocation games is our addition of dynamics in the process of choosing the resources. While games in AGT are typically "one round" games, ongoing games are natural in formal methods. So, the definitions in this game are close in nature to games in formal methods whereas the questions we ask are typical AGT questions.

For the direction from AGT to formal methods, we consider the problem of synthesis in a multi-player setting and introduce new approaches for the definition of cost of a synthesized system. We consider systems generated from components. Each synthesized system has a cost, which is affected by its underlying components. For example, the cost can be the price the players pay for the manufacturing of the component, thus an increase of load has a positive effect as more players "split the bill". Dually, the components can be processors, where higher load means a decrease in performance. Thus, on top of the game theoretic approach to synthesis, in which each player tries to find a correct system we point to an "outer" game that concerns costs. Here, the strategies of a player are the correct systems and we are looking for a stable solution; namely one in which the players have no incentive to change their system.

The work presented in the thesis sets the stage for further research in the field. We describe some directions for future research. The immediate directions are the problems we left open; e.g., the exact value of the PoS in affine multiset congestion games. Next, recall that our extensions of resource allocation games are mostly less stable than traditional games. Specifically, most multiset resource allocation games are not guaranteed to have an NE, and most dynamic RAGs are not guaranteed to have an SPE. An interesting direction would be to "restore stability" to these games, where the rough goal is to ensure that the game under consideration has a stable outcome. Such a goal was studied for resource allocation games with no NE [2]. One way to achieve this would be to weaken the equilibrium we chose as a

measure for stability, while keeping it strong enough to be interesting. Another way would be to consider subclasses of the general game in either its structure or the objectives of the players. Finally, we can make assumptions that increase stability like the ability to control some of the players [3]. That is, an authority can choose some subset of the players and assign strategies to them, which they cannot change. The other players act as normal players.

We proceed to describe less concrete directions. Resource allocation games can be thought of as games that we find in the "wild". These are mathematical objects that attempt to model real-life settings. Our goal is to analyze these objects using several measures: existence of equilibrium, the inefficiency of equilibria, etc. RAGs are only one class of such "games in the wild" that are studied in AGT. Another class is *network creation games* [1], which takes a different approach to networks from network *formation* games, which we studied. There, the players model routers, and they are the nodes of the network. Each router decides to which other routers it is connected directly, so the strategies of each player consist of subsets of her adjacent edges. Choosing a subset of edges corresponds to creating them. Once all routers choose a strategy a graph is formed. The routers' goal is to be "highly connected" to the other routers in the network (for some definition of "highly connected"). Clearly, constructing more edges increases connectivity. However, buying an edge has a cost. If the price is high, a player might prefer to sacrifice her connectivity and rely on edges that other players bought. It is interesting to apply concepts from formal methods to this game or one of its many variants. For example, other than the simple connectivity criteria, we can study a setting in which each router has a specification that allows only a subset of paths, or a quantitative specification that assigns values to paths and the goal would be to connectivity of high quality.

Finally, studying "games in the wild" is only one area in AGT. It is interesting to study other areas that have meeting points with formal methods. For example, in *algorithmic mechanism design* (AMD, for short), an authority tries to maximize the social welfare or some other optimization goal, in the context of selfish players. It is common to try and find a mechanism that is *incentive compatible*, namely a mechanism in which the players achieve their best outcome by acting according to their true preferences. Our work on repairing multi-player concurrent games draws ideas from AMD. Indeed, we assumed an authority can alter the graph on which the game takes place. The objective of the authority is to increase stability. But the game is played by selfish players who only care about their personal gain.

The second part of the thesis focuses on lifting traditional formal methods to the quantitative setting. We studied abstraction and simulation of weighted automata and reasoning on partial-specified systems. Our work on adding costs to synthesis from component libraries can be viewed as such a work. Here too, there are several

concrete directions for future research in each of the works we studied. For example, we have not considered synthesis from component libraries with costs (even with a single designer) with infinite-word specifications. We note that our definition of cost for a design can be applied with no change to this setting.

We also considered the problem of stochastization of WFAs. We studied a fragment of the general case, where the given WFA's ambiguity is bounded. We left the exact complexity of the problem open. Also, it is interesting to find other fragments for which stochastization is decidable. Stochastization can be thought of a method to avoid determinization, which is not always possible for WFAs, by constructing a probabilistic WFA. We had strong constraints on the structure of the probabilistic WFA, namely, it should be the same as the original WFA. It is interesting to weaken these constraints while retaining decidability. Finally, stochastization is a general concept that can be applied to other models such as Boolean automata that accepts either finite or infinite words.

References

[1] A. Fabrikant, A. Luthra, E. Maneva, C. Papadimitriou, and S. Shenker. On a network creation game. In *ACM Symposium on Principles of Distributed Computing*, 2003.

[2] K. Kollias and T. Roughgarden. Restoring pure equilibria to weighted congestion games. *ACM Trans. Economics and Comput.*, 3(4):21, 2015.

[3] T. Roughgarden. Stackelberg scheduling strategies. *SIAM J. Comput.*, 33(2):332–350, 2004.

כאשר ייתכן שמספר שחקנים יבחרו באותו תור. רוח ההגדרות של המשחק דומה למשחקים בשיטות פורמליות ואנו חוקרים את ייציבות בדומה לשאלות שנשאלות בתורת המשחקים.

תורת המשחקים היא כמותית מטבעה: בכל תוצאה של משחק, לכל שחקן יש מחיר שאותו הוא מנסה למזער. שיטות פורמליות לאומת זאת, הן בוליאניות: מערכת מספקת את המפרט או לא. בחלק האחרון של התזה, אנו חוקרים הרחבה של שיטות פורמליות על ידי הוספת כמותיות. כאן למשל נשאל באיזו מידה מערכת מספקת את המפרט שלה. ישנו מאמץ גדול בשנים האחרונות להוספת כמותיות לשיטות פורמליות ואנו לוקחים חלק במאמץ זה. "אוטומטים ממושקלים" הם מודל נפוץ בשביל להביע מפרט ממושקל או למדל מערכת ממושקלת. אנו חוקרים כמה בעיות על מודל זה. אנו מציעים יוריסטיקה כדי להתגבר על חוסר הכריעות של בעית הכלת השפות של אוטומטים אלו, אנו חוקרים השלמת משקלות של אוטומטים, שממדלת מערכות חסרות, ואנו חוקרים החלפת חוסר-דטרמינזם בהסתברות באוטומטים ממושקלים.

תקציר

עבודה זו חוקרת את התחום שבין שיטות פורמליות ותורת המשחקים החישובית. המטרה בשיטות פורמליות היא חקירה פורמלית של מערכות. לכן, מושגים כמו "מפרט" ו"התנהגות מתמשכת" נפוצים ונחקרים רבות בתחום זה. תורת המשחקים החישובית הוא תחום פורח בשנים האחרונות, ונמצא במפגש שבין תורת המשחקים ומדעי המחשב. שאלות נפוצות בתחום זה הן היציבות של משחקים שונים.

תחילה, אנו חוקרים מעבר רעיונות משיטות פורמליות לתורת המשחקים. אנו חוקרים משחק "בניית רשת", שהוא ידוע בתורת המשחקים. במשחק הקלאסי, ישנה רשת ולכל שחקן יש קודקוד מקור ויעד. אסטרטגיה של שחקן היא בחירת מסלול בין שני קודקודים אלו. לכל קשת במסלול יש מחיר, והשחקנים שמשתמשים בה, חולקים בינהם את המחיר. אנו מציעים הרחבה של משחק זה, בה לכל שחקן יש מפרט, כל קשת ברשת מסומנת באות, ואסטרטגיה של שחקן היא בחירת מסלול שעומד במפרט שלו. ההבדל המשמעותי בין המשחק שלנו והמשחק הקלאסי הוא שמסלול ששחקן בוחר כבר לא בהרכח חסר-מעגלים. כלומר, במשחק החדש, ייתכן ששחקן יבחר את אותה קשת מספר רב של פעמים. כעת, חלוקת המחיר פרופורציונלית למספר השימושים בקשת. אנו חוקרים את היציבות של המשחק החדש ביחס לשאלות הסטנדרטיות בתורת המשחקים: קיום של שווי משקל, יעילות שווי המשקל ושאלות חישוביות על משחקים אלו. כמו כן, אנו מכלילים את משחק הרשת למשחק "בחירת משאבים" בו אסטרטגיה של שחקן היא בחירת אוסף משאבים, כאשר ייתכן ושחקן ישתמש במשאב מספר רב של פעמים, זאת בניגוד למשחק הקלאסי בו ייתכן לכל היותר שימוש יחיד במשאב.

בכיוון השני, מתורת המשחקים לשיטות פורמליות, אנו מרחיבים את בעיית הסינטזה מרכיבים. בבעיה זו, ישנה ספריה של רכיבים והמטרה היא ליצור מערכת על ידי "הדבקה" של הרכיבים מהספרייה. אנו מרחיבים את הבעיה בשני אופנים. אנו מוסיפים מחיר לכל רכיב, ומאפשרים למספר רב של משתמשים להשתמש בספריה בעת ובעונה אחת. המחיר של רכיב מושפע מכמה פעמים הוא מופיע במערכות השונות. למשל, רכיב יכול למדל מעבד, ושימושים רבים גורמים לעומס, מה שמוריד את איכות המערכת. כל משתמש מעוניין למצוא מערכת שתהיה זולה ככל הניתן. כעת, נוצר משחק בחירת משאבים, כאשר המשאבים הם הרכיבים והמשתמשים הם השחקנים.

לבסוף, אנו חוקרים הוספת התנהגות מתמשכת למשחקי בחירת משאבים. אנו מקלים שתי הנחות מהמשחק הקלאסי: כל שחקן בוחר אסטרטגיה בבת אחת וכל השחקנים בוחרים במקביל. אנו חוקרים משחק שמתנהל בשלבים. בכל שלב כל שחקן בוחר משאב יחיד. כל שלב מחולק לתורות, וישנו "מתזמן" שקובע באיזה תור כל שחקן בוחר בכל שלב,

עבודה זו נעשתה בהדרכתה של אורנה קופרמן.

שיטות פורמליות כמותיות פוגשות את תורת המשחקים החישובית

חיבור לשם קבלת תואר דוקטור לפילוסופיה

גיא אבני

הוגש לסנט האוניברסיטה העברית בירושלים

פברואר 2016