Optimality and Resilience in Parity Games

Dissertation zur Erlangung des Grades
des Doktors der Naturwissenschaften
der Fakultät für Mathematik und Informatik
der Universität des Saarlandes

vorgelegt von
Alexander Dominik Weinert

Saarbrücken, 2018

Dean:	Prof. Dr. Sebastian Hack
Advisor:	PD Dr. Martin Zimmermann
Examination Board:	Prof. Dr. Verena Wolf (Chair) PD Dr. Martin Zimmermann Prof. Dr. Sven Schewe Prof. Dr. Veronique Bruyére Prof. Bernd Finkbeiner, Ph. D. Dr. Swen Jacobs (Academic Staff)
Thesis Defence:	December 14, 2018

Abstract

Modeling reactive systems as infinite games has yielded a multitude of results in the fields of program verification and program synthesis. The canonical parity condition, however, neither suffices to express non-functional requirements on the modeled system, nor to capture malfunctions of the deployed system. We address these issues by investigating quantitative games in which the above characteristics can be expressed.

Parity games with costs are a variant of parity games in which traversing an edge incurs some nonnegative cost. The cost of a play is the limit superior of the cost incurred between answering odd colors by larger even ones. We extend that model by using integer costs, obtaining parity games with weights, and show that the problem of solving such games is in NP $\cap$ coNP and that it is PTime-equivalent to the problem of solving energy parity games.

We moreover show that Player 0 requires exponential memory to implement a winning strategy in parity games with weights. Further, we show that the problem of determining whether Player 0 can keep the cost of a play below a given bound is ExpTime-complete for parity games with weights and PSpace-complete for the special cases of parity games with costs and finitary parity games, i.e., it is harder than solving the game. Thus, optimality comes at a price even in finitary parity games.

We further determine the complexity of computing strategies in parity games that are resilient against malfunctions. We show that such strategies can be effectively computed and that this is as hard as solving the game without disturbances.

Finally, we combine all these aspects and show that Player 0 can trade memory, cost, and resilience for one another. Furthermore, we show how to compute the possible tradeoffs for a given game.

Zusammenfassung

Die Modellierung von reaktiven Systemen durch unendliche Spiele ermöglichte zahlreiche Fortschritte in der Programmverifikation und der Programmsynthese. Die häufig genutzte Paritätsbedingung kann jedoch weder nichtfunktionale Anforderungen ausdrücken, noch Fehlfunktionen des Systems modellieren. Wir betrachten quantitative Spiele in denen diese Merkmale ausgedrückt werden können.

Paritätsspiele mit Kosten (PSK) sind eine Variante der Paritätsspiele in denen die Benutzung einer Kante nichtnegative Kosten verursacht. Die Kosten einer Partie sind der Limes Superior der Kosten zwischen ungeraden und den jeweils nächsten größeren geraden Farben. Wir erweitern dieses Modell durch ganzzahlige Kosten zu Paritätsspielen mit Gewichten (PSG). Wir zeigen, dass das Lösen dieser Spiele in $\mathrm{NP} \cap \mathrm{coNP}$ liegt, dass es PTime-äquivalent dazu ist, Energieparitätsspiele zu lösen und dass Spieler 0 exponentiellen Speicher benötigt, um zu gewinnen.

Ferner zeigen wir, dass das Problem, zu entscheiden, ob Spieler 0 die Kosten eines Spiels unter einer gegebenen Schranke halten kann, ExpTime-vollständig für PSG ist, sowie dass es PSpace-vollständig für die Spezialfälle PSK und finitäre Paritätsspiele (FPS) ist. Optimalität ist also selbst in FPS nicht kostenlos.

Außerdem bestimmen wir die Komplexität davon, Strategien in Paritätsspielen zu berechnen, die robust gegenüber Fehlfunktionen sind, zeigen, dass solche Strategien effektiv berechnet werden können und beweisen, dass dies nur linearen Mehraufwand bedeutet.

Darüberhinaus kombinieren wir die oben genannten Aspekte, zeigen, dass Spieler 0 Speicher, Kosten und Robustheit gegeneinander eintauschen kann und berechnen die möglichen Kompromisse.

Acknowledgements

According to a proverb, it takes a village to raise a child. The same is true for doing research and, in particular, for writing a dissertation. I would like to thank everyone without who this last years would not have been possible and not nearly as great.

First and foremost, I would like to thank Martin Zimmermann not only for allowing me to write this thesis under his supervision, but also for teaching me how to do research, for giving me the opportunity to present that research at conferences all over the world, for innumerable discussions and rounds of proof-reading, and of course for saving me from the clutches of an invading dragonfly.

I would also like to thank Bernd Finkbeiner for allowing me to work at the Reactive Systems Group, for providing me with the infrastructure to perform my research, and of course for refereeing this thesis. Thanks also go the members of the Reactive Systems Group, namely Chris, Felix, Hazem, Jana, Jesko, Leander, Malte, Max, Michael, Mouhammad, Noemi, Norine, Peter, and Swen for always being available for and open to all kinds of discussions. Moreover, I am grateful for all the help that Christa Schäfer provided, whether it had to do with navigating academic bureaucracy, with increasingly obtuse travel itineraries, or with organizing GandALF 2018 and my thesis defense.

Furthermore, I am indebted to a number of other researchers I collaborated with, namely Rajeev Alur, Loris D'Antoni, Daniel Neider, Mickael Randour, Sven Schewe, Paulo Tabuada, and Matthew Weaver. This research was generously funded by the German Research Foundation (DFG) and by the Saarbrücken Graduate School. Moreover, I would like to thank my reviewers Martin Zimmermann, Sven Schewe, Veronique Bruyére, and Bernd Finkbeiner for refereeing this thesis, as well as the other members of my examination board, namely Verena Wolf and Swen Jacobs.

These acknowledgements would be nowhere near complete without thanking my friends from my undergraduate studies in Aachen, namely Bertina, Christian, Jasper, Jan, Julia, Julian, Kevin, Lutz, Moritz, and Philipp for always being up for all kinds of shenanigans over the last ten years.

I also want to thank my parents, Elke and Edgar, for enabling me to develop and follow my interests and for supporting me since the summer of 1990. Finally, I want to thank my girlfriend and future wife Sabrina, for sticking with me through five years of studying in Aachen and three more in Saarbrücken and lovingly supporting me all the way. Without you, none of this would have been possible.

Contents

CHAPTER 1

Introduction

In a classical view, a program takes some input, performs a transformation on that input, and outputs the result before terminating. While such systems still exist, embedded systems have become more and more prevalent in recent years; A modern car typically comprises several hundreds if not thousands of embedded systems that work in unison to guarantee safety and functionality of the car. Such embedded systems are characterized by running indefinitely and by maintaining constant communication with their environment throughout their runtime. Since such systems are ubiquitous and are in control of a number of safety-critical systems, it is important to have expressive and well-understood formal methods for modeling such systems and their environments in order to allow for formal reasoning about the systems.

One such model that has yielded a host of advances in the fields of verification and synthesis of correct-by-construction reactive systems is the formalism of infinite games in which the environment and the system are viewed as two players that compete to satisfy some winning condition. Although this model is widely used for the design and analysis of embedded systems, we argue that it has shortcomings that prevent it from supporting in-depth analyses of the systems and from realistically modeling the systems. In this work, we address these shortcomings, mitigate them via extensions of the model of infinite games, and investigate and determine the overhead these extensions incur in terms of complexity.

We first describe this model and how it is used in the modeling of reactive systems in Section 1.1. Subsequently, in Section 1.2 and Section 1.3, we address the shortcomings of this model and detail our approach to mitigating them. Finally, in Section 1.4 we outline the structure of the remainder of this thesis.

1.1 Modeling Reactive Systems as Infinite Games

In their most basic forms, embedded systems can be viewed as combinatorial circuits comprised of logical gates and latches that are synchronized to some clock. In each time step, the circuit obtains some input bits and outputs the result of the evaluation of the gates and latches. As the construction of such circuits is, in general, quite involved and since they are often used in safety-critical roles, it is paramount to obtain formal methods to verify the correctness of a circuit with respect to some formal specification or, ideally, to automatically construct such a circuit given only the specification.

Church [Chu57] first raised the question of how to construct such a circuit that conforms to a given specification automatically and formulated the "synthesis problem" as follows:

> "Given a requirement which a circuit is to satisfy we may suppose the requirement expressed in some suitable logical system[...]. The synthesis problem is then to find [...] a circuit that satisfies the given requirement (or alternatively, to determine that there is no such circuit)."

For many such "suitable logical system[s]" the synthesis problem can be boiled down to solving a game between two players, which we call Player 0 and Player 1, and who, respectively, represent the circuit to be constructed and the environment that controls the input to the circuit. Player 0 then wins the game by satisfying the specification, while Player 1 wins by violating it. Throughout this work, for the sake of readability, we consider Player 0 to be female, while we consider Player 1 to be male.

Consider, e.g., a system that is in charge of controlling access to some resource that is shared between several clients: In each turn of the game modeling this scenario, Player 1, i.e., the environment, may cause one or more of the clients to request access to the resource. Conversely, Player 0, i.e., the system under construction, may grant one or multiple such requests. In order to satisfy the typical requirements of fairness and mutual exclusion, we declare Player 0 as the winner if she grants each request eventually and if she at no point grants access to more than one client in the same turn.

In this work, we consider games of infinite duration played on finite graphs, which suffice to capture a wide range of formal specifications. Such games are played between the two players by moving a token around a finite graph, the so-called arena of the game. The vertices of the arena are partitioned into those belonging to Player 0 and those belonging to Player 1.

In each turn, the player owning the vertex that currently carries the token picks a successor of that vertex and moves the token to that successor. After infinitely many such moves, the two players have constructed an infinite path through the graph, a so-called play. The winner of the resulting play is then determined via the winning condition, a set of plays that are proclaimed to be winning for Player 0.

Consider again the example of an embedded system tasked with controlling access to some shared resource. We show an arena modeling this situation for two clients in Figure 1.1. Here and in the remainder of this thesis, we draw vertices of Player 0

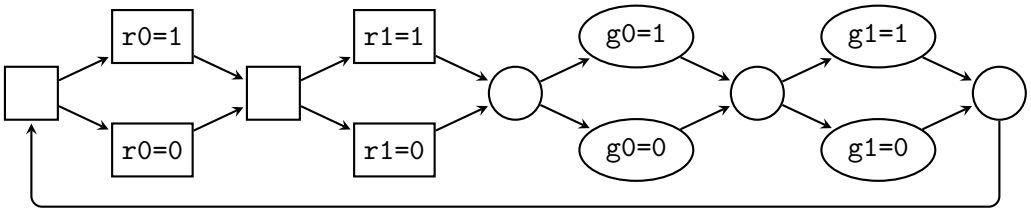


Figure 1.1: An arena modeling an arbiter in control of one resource with two clients.

and Player 1 as ellipses and rectangles, respectively. In that arena, we model one time step by moving the token from the leftmost vertex to the rightmost one. Returning the token to the leftmost vertex via the lower edge starts the next time step.

In the arena shown in Figure 1.1, the vertices labeled with `r0=1` and `r1=1` denote that the first and second client, respectively, request access to the resource in the current time step, while the vertices labeled `r0=0` and `r1=0` denote that they do not. Dually, the vertices labeled with `g0=1` and `g1=1` denote that the system grants access to the first and second client, while the vertices labeled with `g0=0` and `g1=0` denote that it does not grant access to the respective client in the current time step. By declaring exactly those plays winning in which each visit to the vertices `r0=1` and `r1=1` is eventually followed by a visit to the vertices `g0=1` and `g1=1`, respectively, we force Player 0 to implement an arbiter that eventually grants all requests for the resource in order to win the game.

The main object of interest in infinite games are strategies for the two players, i.e., objects describing the behavior of the players. A strategy for Player i prescribes for each play prefix ending in a vertex v of that player a successor of v to move to. If the strategy can be implemented using a finite-state machine that processes the history of the play and yields the next vertex to move to, we say that the strategy is finite-state. If, for a given game $\mathcal{G}$ and a vertex v of $\mathcal{G}$, there exists a strategy σ for Player i such that all plays that start in v and that follow the behavior described by σ satisfy (for Player 0) or violate (for Player 1) the winning condition, we say that Player i wins $\mathcal{G}$ from v. In the example shown in Figure 1.1, Player 0 wins from every vertex. This is witnessed, e.g., by the strategy which prescribes granting access to the first and to the second client in alternation.

Büchi and Landweber [BL69] showed that in all infinite games whose winning conditions are given by an ω-regular automaton that processes the resulting play, finite-state winning strategies for both players exist and can be computed effectively. Since a large number of "suitable logical system[s]" can be compiled into infinite games with ω-regular winning conditions and since each finite-state strategy for Player 0 can be transformed into a controller satisfying the specification given by the winning condition, the result of Büchi and Landweber provides a solution to Church's Problem for a wide range of specifications.

1.2 Measuring Degrees of Satisfaction

The canonical winning condition used in infinite games is the parity condition. This condition is induced by a labeling of the vertices of the underlying arena with natural numbers, so-called colors. The winner of the resulting play is then determined by the largest color seen infinitely often during the play. Player 0 wins the resulting play if and only if that color is even. Games that are equipped with a parity condition are called parity games. In such games, odd colors can be interpreted as requests, which are answered by larger even colors. In this formulation, the parity condition amounts to demanding that after a finite prefix all requests are answered.

The prevalence of this winning condition stems from the fact that a host of specification languages for reactive systems can be compiled into parity games. Prime among these specification languages is Linear Temporal Logic [Pnu77], one of the major logics for specifying properties of reactive systems. Furthermore, parity games have applications in other fields of theoretical computer science: The emptiness problem for parity automata over infinite trees can, e.g., be reduced to solving parity games [Wil01]. Moreover, the model-checking problem for modal μ-calculus can be reduced to that of solving parity games as well [Wil01].

Apart from their utility in a number of fields of computer science, parity games also feature a number of properties that make their use in solving the above problems attractive. In particular, Mostowski [Mos91] as well as Emerson and Jutla [EJ91] showed that if Player i wins a parity game from some vertex v, then she has a memoryless winning strategy from v, i.e., a strategy that prescribes the next vertex based only on the vertex currently holding the token.

The versatility of parity games together with the above result showing that they admit very simple winning strategies has led to them being the subject of a wide range of work in theoretical computer science. For a in-depth introduction to the uses of parity games and their connection to logics and automata, we refer to work by Grädel, Thomas, and Wilke [GTW02]. Here, we focus on the use of parity games for modeling embedded systems as described in the previous section.

While parity games suffice to model a wide range of scenarios in which such embedded systems may be deployed, one crucial shortcoming of such games is their inability to model quantitative conditions. Consider again our example of an embedded system controlling access to a resource, where in this case we consider the scenario of only a single client. It is straightforward to model this scenario as a parity game, where we use the odd color one to denote a request for the resource, while we denote a grant of that request with the even color two. We furthermore use the color zero to denote that access to the resource is neither requested nor granted. In using the parity condition, we relax our previous requirement of fairness and only require that all but finitely many requests of the client are answered.

Now consider the sequence of colors

$$1\,02\,1\,002\,1\,0002\,1\,00002\,1\,000002\,1\,0000002\,1\,00000002\cdots$$

This sequence satisfies the parity condition, as the largest color occurring infinitely

often is two. Intuitively, however, the behavior of the system is undesirable, as the environment has to wait longer and longer for the satisfaction of its requests. The parity condition is unable to express the requirement that this waiting time is bounded along the play.

If we imposed some fixed upper bound of, e.g., five, on the waiting time for requests to be answered, it would be straightforward to encode this bound in the arena of the parity game such that Player 0 loses a play if infinitely often requests are open for more than five steps. In general, however, one is interested in the question whether such an upper bound exists without explicitly encoding a fixed one in the constructed game.

Recently, there has been an increased focus of research on quantitative winning conditions which allow modeling such situations in which an unknown, but finite bound on some measure of cost on the resulting plays is required. In general, such quantitative winning conditions induce a finer structure on the set of plays than classical winning conditions: In classical parity games, the set of plays is partitioned into plays that are winning for Player 0 and those that are not. Quantitative conditions, in contrast, assign to each play some measure of cost and require Player 0 to ensure that the cost of a play remains finite. Hence, such quantitative conditions refine the picture of modeling embedded systems, as they enable more fine-grained analyses.

Among the notable examples of such quantitative conditions are the mean payoff condition [EM79, ZP95, BCD$^+$11, BFRR17] and its combination with the parity condition [CHJ05], the energy condition [CdAHS03, BFL$^+$08] and its combination with the parity condition [CD12], the average-energy condition [BMR$^+$18], as well as the finitary parity condition [CH06,CHH09,CF13] and its extension, the parity condition with costs [FZ14].

Among these extensions, finitary parity games [CH06, CHH09] and their extension to parity games with costs [FZ14] take a special role: In most combinations of the parity condition with quantitative conditions, the two conditions are independent, i.e., Player 0 has to satisfy both conditions individually. In the finitary parity condition, in contrast, these two conditions are conjoined, i.e., the quantitative condition measures a "degree of satisfaction" of the parity condition.

Similarly to our above example, in finitary parity games, the odd and even colors of parity games are interpreted as requests and responses, respectively. The cost of a play is determined as the limit of the number of turns it takes for an odd color to be answered by a larger even one. Player 0 wins a play if its cost is finite, i.e., if she is able to eventually bound the number of steps between requests and their respective responses. Chatterjee, Henzinger, and Horn [CHH09] showed that the problem of solving finitary parity games is simpler than that of solving parity games and that memoryless strategies still suffice for Player 0 to win, i.e., the finitary parity condition retains and even improves upon the desirable properties of classical parity games.

Moreover, recall the above scenario of an arbiter with a single client, in which we want Player 0 to win only if she is able to bound the waiting times for requests to be granted. The finitary parity condition allows us to easily model this requirement, since, e.g., the color sequence shown in the example above does not satisfy the finitary

parity condition.

While finitary parity games have attractive properties both in terms of the complexity of the problem of solving them and in terms of memory requirements, they feature a very simplistic cost model: Each step in the game induces unit cost of the modeled resource, e.g., time. This model is sufficient to express the above condition of Player 0 bounding the number of steps between requests and their responses if one chooses an appropriate underlying arena. It does not, however, allow to disentangle the measure of cost from the structure of the arena. Thus, the finitary parity condition is ill-suited to model scenarios in which, e.g., Player 0 and Player 1 may make multiple decisions in each time step: Each passing of control from one player to the other implies at least one unit of time passing. This is even illustrated by the simple scenario of an arbiter controlling access to a shared resource for two clients shown in Figure 1.1. Here, we modeled the scenario via an arena, in which only a single edge denotes an advance of time, while traversing any other edge did not denote time passing. Furthermore, the cost model of finitary parity games complicates the modeling of a resource other than time, which may not be drained uniformly in each transition.

In order to alleviate this shortcoming, Fijalkow and Zimmermann [FZ14] generalized the cost model of finitary parity games, obtaining parity games with costs. This model not only generalizes both the finitary parity condition and the parity condition, but it also disentangles the measure of cost from the arena of the game. The winning condition of parity games with costs is given by a coloring of the vertices and a labeling of the edges with nonnegative integers denoting the cost incurred when traversing the edge. Analogously to finitary parity games, the cost of a play is then defined as the limit of the cost incurred during the play infixes leading from a request for some odd color to its answer by a larger even color. Fijalkow and Zimmermann showed that the problem of solving parity games with costs is as hard as solving parity games and that memoryless strategies still suffice for Player 0 to win. Hence, according to the state of the art, solving such games is harder than solving finitary parity games, but only as hard as its subsumption of parity games requires it to be.

While parity games with costs allow for more precise modeling of quantitative conditions than finitary parity games, there still exist simple scenarios which cannot be captured even by such extended games.

Consider, e.g., a system with an attached battery that allows clients to request the current energy level of the battery from the system. The systems answers these requests only after a certain number of steps, e.g., after computing the current level, or after disconnecting consumers from the battery. In this scenario, one may want to require the controller to respond to requests for the current energy level of the battery with “up-to-date” information, i.e., the energy level should not be able to change arbitrarily between the request of a client and the response of the system. This situation cannot be modeled realistically using parity games with costs, as they are restricted to using nonnegative weights. Hence, such a game can only model either charging or discharging the battery, depending on the interpretation of the weights, but no parity game with costs can model a battery that is both charged and discharged over the course of a single play.

In this work, we alleviate the above problem by removing the restriction of parity games with costs, thus also allowing edges to be labeled with negative weights. We call the obtained games parity games with weights. Thus, we further extend the expressiveness of quantitative parity games and allow the modeling of resources that can be charged or drained. We show that solving such games is as hard as solving energy parity games, another widely used quantitative extension of parity games. We furthermore show that, in contrast to parity games with costs, Player 0 requires memory in order to implement winning strategies in parity games with weights. Hence, the added modeling capability afforded by parity games with weights comes at a price both in terms of the complexity of solving such games, as well as in terms of memory requirements.

Up to this point, we have only considered the boundedness problem for quantitative extensions of parity games, i.e., the problem of deciding whether or not Player 0 can ensure finite cost of a play. When the quantitative properties of the discussed extensions of parity games are used to model some resource, this amounts to determining whether there exists some finite amount of that resource that allows Player 0 to satisfy the quantitative winning condition. A larger amount of the modeled resource, however, in general comes at a price. Hence, it is desirable to determine the minimal amount of the resource that still allows the system to satisfy the specification. In terms of infinite game, this amounts to determining the minimal threshold b such that Player 0 is able to satisfy the quantitative condition with respect to b. As we obtain an upper bound on the maximal cost that Player 0 can enforce if she wins a parity game with weights from our algorithm solving such games, we find that determining the minimal such b amounts to solving the so-called threshold problem: Given some parity game with weights $\mathcal{G}$, some threshold b, and a vertex v of $\mathcal{G}$, determine whether Player 0 is able to enforce a cost of at most b when starting from v.

In this work, we show that the threshold problem is ExpTime-complete for parity games with weights and that it is PSpace-complete for the special cases of parity games with costs and finitary parity games. From the proof of membership in the respective complexity classes we also obtain that exponential memory suffices for both players to satisfy a given upper (for Player 0) or lower (for Player 1) bound on the cost. We moreover show that exponential memory is necessary for both players to do so even in the special case of finitary parity games.

Thus, playing optimally comes at a price, even in the special case of finitary parity games, both in terms of the complexity of computing a strategy that realizes an optimal bound (which rises from being solvable in polynomial time to requiring polynomial space), as well as in terms of the complexity of that strategy (which rises from memoryless strategies to strategies of exponential size).

In contrast to the above results, which yield necessity of exponential memory for Player 0 in order to play optimally, if her only aim is to finitely bound the cost of the play, then a positional strategy suffices to do so. While the exponential size of an optimal strategy may be prohibitive in the context of embedded systems, it may also be undesirable to only obtain some arbitrary finite upper bound on the cost of plays in the same scenarios. We furthermore show that this dichotomy is, in general, not

that strict, i.e., that there exists a gradual tradeoff between the memory required to implement a strategy and the upper bound on the cost of plays that it guarantees.

1.3 Modeling Real-World Challenges

In the previous section we have detailed shortcomings of the classical game-theoretic approach to modeling reactive systems. Namely, we have argued that the classical and canonical parity conditon is ill-suited to capturing quantitative requirements towards the system. We now furthermore argue that although the above approach of modeling reactive systems via infinite games has proven very successful, it is based on an overly optimistic world view: When modeling a reactive system as an infinite game, intuitively, each outgoing edge of a vertex of Player 0 denotes an action that the controller of the modeled system may take. The target of the edge denotes the state of the system and its environment after that action has been executed. Due to this model, we inherently presume that the engineer modeling the system takes into account the complete set of interactions between the system and its environment as well as all possible outcomes of all actions available to the system at any moment.

→Sec. 1.1, Page 3

Consider again, for example, the arbiter controlling access to a resource for two clients discussed above, which we modeled by the arena shown in →Figure 1.1. In constructing this arena, we presumed Player 0, i.e., the player representing the controller, to have full control over which client gains access to the resource. This model, however, is not entirely realistic: When controlling access to some physical resource, the actuator physically granting access to that resource may become stuck during operation, thus not permitting or preventing access to the clients as required by the controller. Similarly, when controlling access to some digital resource, such as shared memory, bit flips may cause access to be granted to unintended clients as well.

The above possibilities are not modeled in the arena shown in Figure 1.1. In fact, we argue that such malfunctions of the system cannot be effectively modeled using the standard notion of infinite games: Giving Player 0 control over the occurrence of such events would cause the system to disregard them, i.e., cause them to never occur. Dually, giving control over such occurrences to Player 1 would enable the (antagonistic) environment to cause such disturbances at will, thus preventing Player 0 from satisfying the specification in too many cases. In general, however it is necessary to have formal methods allowing for precise modeling of such effects: Excluding them from the design would yield a model that provides an overly optimistic and unrealistic view of real-life systems.

It is a topic of active research to compute controllers that are resilient against such malfunctions [TOLM12, ET14, HPSW16]. Related to the above problem of computing resilient controllers are various notions of fault tolerance [AAE04, BGH$^+$15, EKA08, GR09] and of robustness [BCG$^+$14, BCHJ09, BEJK14, BJP$^+$12, MRT13, TCRM14, TN16, BRS17, BBF$^+$18, NWZ18a]. In particular, a survey of a large body of work dealing with robustness in reactive synthesis has been presented by Bloem et al. [BEJK14]. The work mentioned above, however, usually considers specifications given in some logic,

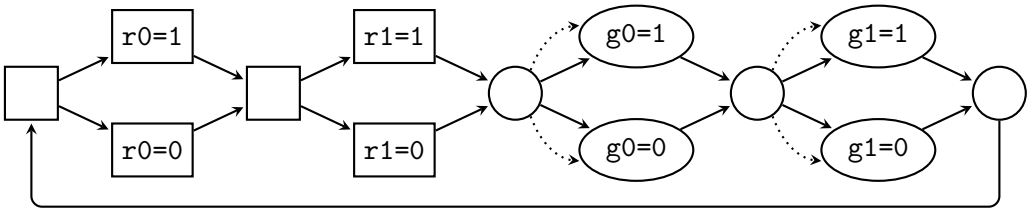


Figure 1.2: An arena modeling an arbiter in control of one resource with disturbances.

they treat disturbances as occurring antagonistically, i.e., as under the control of the environment, or they use a model of infinite games that differs significantly from the standard model of two-player, turn-based games introduced above.

In this work we follow the approach of Dallal, Neider, and Tabuada [DNT16], who have formalized the uncertainty about results of actions taken by Player 0 in the model of infinite games by introducing games with disturbances. Such games extend the standard model of infinite games as defined above by so-called disturbance edges. Whenever it is the turn of Player 0, after she has picked an outgoing edge of the current vertex, the play may instead move along one of the outgoing disturbance edges of that vertex, thus modeling the occurrence of a disturbance.

This models the intuitive behavior of malfunctions and other disturbances: Consider again the above example of an arbiter controlling access to a shared resource with two clients and recall that we argued previously that the choice of Player 0 of granting access to either client may be overridden by a malfunction of the system. We show an infinite game with disturbances taking such malfunctions into account in Figure 1.2. In that figure and in the remainder of this work, we denote disturbance edges by dotted lines. In the arena shown in that figure, whenever Player 0 has chosen to either grant or deny access to the resource to a client, a disturbance may occur and override her decision.

Crucially, in this model, disturbance edges are not under the control of either player, nor do they require an underlying model of their occurrence. Instead, they are treated as rare events. In this model, the question of determining the winning player from a given vertex is generalized: Instead of asking whether Player 0 wins a given game with disturbances from some given vertex, we instead ask how many disturbances may occur in order for her to still win the game. Furthermore, we ask to construct a strategy that is resilient against as many disturbances as possible, i.e., that still produces a winning play under the maximal number of disturbances that allow Player 0 to win at all.

Dallal, Neider, and Tabuada [DNT16] have solved the problem of computing such optimally resilient strategies only for safety games, i.e., for a very restricted class of infinite games in which it is the goal of Player 0 to avoid a given set of undesirable states. In that setting, they have shown that constructing optimally resilient strategies does not incur an overhead over solving the underlying game without disturbances. We first generalize their notion of resilience against disturbances to the setting of parity games and discuss the new phenomena that occur in this setting of more compli-

cated winning conditions. Subsequently, we show that, even for this greatly extended class of games, the problem of computing optimally resilient strategies is still only as hard as solving the underlying game without disturbances, as well as that optimally resilient strategies are not larger than winning strategies in the underlying game without disturbances. Hence, playing with respect to disturbances comes for free even in parity games.

Via these results, we raise the model of disturbances to the setting of parity games, i.e., the canonical games used for modeling embedded systems, and show how to solve the associated problems. Thus, we greatly increase the capability of the designer of a formal model of embedded systems to accurately capture real-life phenomena occurring when deploying such systems.

Finally, we investigate the notion of resilience in the quantitative parity games discussed in the previous section. In such games, there are two notions of quality for a given strategy, namely the upper bound on the cost of resulting plays, and the number of disturbances that allow Player 0 to still guarantee that upper bound. We show that computing the pairs of r and b such that Player 0 can still guarantee cost at most b if less than r disturbances occur is only as hard as solving the threshold problem for the underlying game.

1.4 Outline

We first formally define the prerequisite notions of infinite games and complexity theory in Chapter 2. In that chapter, we also give formal definitions of the parity condition, the finitary parity condition, and the parity condition with costs as discussed informally above.

Afterwards, we define the extension of parity games with costs to parity games with weights in Chapter 3 and prove the results on the complexity of solving such games and on the memory requirements. We moreover show that, if Player 0 has a winning strategy in such a game, then she has one that ensures an exponential upper bound on the cost of the resulting play. Furthermore, in Chapter 4 we investigate the problem of solving parity games with weights with respect to some given bound and prove the results claimed above. In that section, we furthermore investigate the problem for the special cases of parity games with costs and finitary parity games.

We then define the model of games with disturbances and show how to construct optimally resilient strategies, and prove that such strategies are not larger than winning strategies in Chapter 5.

After having thus defined and investigated the metric of cost and resilience in isolation, we investigate the tradeoffs between the different measures of quality discussed throughout this work in Chapter 6. Here, we exhibit the tradeoffs between cost and size of a strategy, as well as the tradeoff between cost and resilience of a strategy.

Finally, we conclude in Chapter 7 with an overview over open problems and with a discussion of future work.

CHAPTER 2

Preliminaries

In this chapter we define the mathematical notions required as prerequisites for the remainder of this work. We first introduce basic notions of infinite games in Section 2.1. Since a major contribution of this thesis is the characterization of the complexity of a number of decision problems, we subsequently discuss the complexity classes occurring in this work in Section 2.2. Finally, in Section 2.3 and Section 2.4, we present previous work regarding qualitative and quantitative games, respectively, upon which we build the contributions of this thesis.

We denote the nonnegative integers by $\mathbb{N}$ and the integers by $\mathbb{Z}$. Moreover, we define $\infty > n$ and $-\infty < n$ for all $n \in \mathbb{Z}$. As usual, we define $\infty + n = n + \infty = \infty$ and $-\infty - n = n - \infty = -\infty$ for all $n \in \mathbb{Z}$.

For any set L, we write L^*, L^+, and L^ω to denote the set of all finite sequences, all nonempty finite sequences, and all infinite sequences of elements from L, respectively. Moreover, for any two sets L, K, we write LK to denote the set of all sequences lk, where $l \in L$, $k \in K$. Finally, we write ϵ to denote the empty sequence.

2.1 Arenas and Strategies

In this work we exclusively consider two-player arena-based games with perfect information. Such games are played between the two players Player 0 and Player 1 who move a token around a graph, called the arena. Each vertex of the arena belongs to one of the two players. At the beginning of each turn, the token is situated at some vertex of the arena. The player owning that vertex picks an outgoing edge of the vertex, moves the token along that edge and the next turn begins. By iterating this process ad infinitum, the two players construct an infinite path through the arena, a so-called play.

Formally, an ARENA $\mathcal{A} = (V, V_0, V_1, E)$ consists of a finite, directed graph (V, E) and **Def.** arena

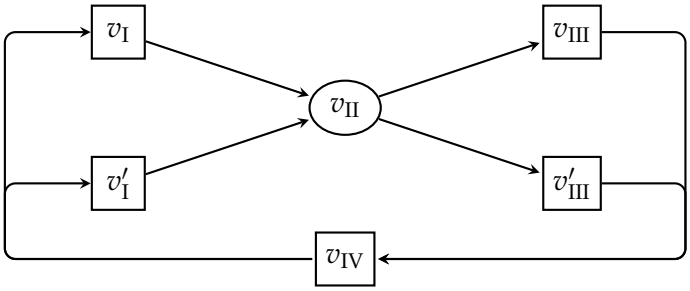


Figure 2.1: An arena with six vertices. Adapted from Chatterjee and Fijalkow [CF13].

a partition (V_0, V_1) of V into the vertices belonging to Player 0 and to Player 1. Since we later only consider infinite paths through arenas, we aim to avoid dead-ends in $\mathcal{A}$, hence we assume each vertex in $\mathcal{A}$ to be non-terminal. When drawing arenas, we draw the vertices of Player 0 as ellipses, while we draw the vertices of Player 1 as rectangles. If the owner of a vertex is unknown or irrelevant, we draw that vertex as a diamond. We define the size of $\mathcal{A}$ as $|\mathcal{A}| = |V|$.

Def. play
Def. play prefix

A PLAY in $\mathcal{A}$ is an infinite path ρ through (V, E). We define $|\rho| = \infty$. In contrast, a PLAY PREFIX is a finite path through (V, E). Each play is a member of V^ω, but not every infinite sequence of vertices from V is a play.

Example 2.1. Consider the arena $\mathcal{A}$ shown in Figure 2.1. The vertex v_{II} belongs to Player 0, while all other vertices, i.e., the vertices v_{I}, v'_{I}, v_{III}, v'_{III}, and v_{IV} belong to Player 1. Hence, whenever the token is at vertex v_{II}, Player 0 chooses to move the token to either v_{III} or to v'_{III}.

Dually, whenever the token is at any vertex other than v_{II}, Player 1 chooses a succeeding vertex to move the token to. Since, however, all vertices of Player 1 except for v_{IV} only have a single succeeding vertex, Player 1 only has to "choose" such a vertex when the token is at v_{IV}. At all other vertices, the play proceeds "automatically," i.e., Player 1 cannot influence the evolution of the play △

We formalize the notion of the two players choosing successor vertices via strategies. In general, players are unconstrained in their choice of successor vertex: They may resort to stochastic choices or pick the succeeding vertex completely arbitrarily. In this work, however, we only consider deterministic strategies for both players. Such strategies prescribe the next vertex to move to based only on the play prefix constructed so far.

Def. strategy
Def. positional strategy
Def. consistent play

Fix an arena $\mathcal{A} = (V, V_0, V_1, E)$. A STRATEGY for Player $i \in \{0, 1\}$ is a mapping $\sigma\colon V^*V_i \to V$ that satisfies $(v, \sigma(\pi v)) \in E$ for all play prefixes $\pi v \in V^*V_i$. We say that σ is POSITIONAL if we have $\sigma(\pi v) = \sigma(v)$ for all $\pi v \in V^*V_i$, i.e., if the strategy σ only uses the current vertex to determine the next move. In this case, we write $\sigma(v)$ to denote $\sigma(\pi v)$ for all $\pi \in V^*$. A play $v_0v_1v_2\cdots$ is CONSISTENT with a strategy σ for Player i, if we have $v_{j+1} = \sigma(v_0 \cdots v_j)$ for every $j \in \mathbb{N}$ where $v_j \in V_i$.

Example 2.2. Consider again the arena shown in Figure 2.1. A strategy for Player 0 is a function that, given a play prefix ending in v_{II}, yields either v_{III} or v'_{III}. Dually, a strategy for Player 1 is a function that maps play prefixes ending in v_{I} or v'_{I} to v_{II}, play prefixes ending in v_{III} or v'_{III} to v_{IV}, and play prefixes ending in v_{IV} to either v_{I} or to v'_{I}.

While both players have infinitely many strategies, they both only have two positional strategies: Player 0 has the two positional strategies σ and σ' given by $\sigma(\pi v_{\mathrm{II}}) = v_{\mathrm{III}}$ and $\sigma'(\pi v_{\mathrm{II}}) = v'_{\mathrm{III}}$ for all $\pi \in V^*$, respectively, while Player 1 has the two positional strategies τ and τ' defined via $\tau(\pi v_{\mathrm{IV}}) = v_{\mathrm{I}}$ and $\tau'(\pi v_{\mathrm{IV}}) = v'_{\mathrm{I}}$ for all $\pi \in V^*$, respectively.

The plays $(v_{\mathrm{I}} v_{\mathrm{II}} v_{\mathrm{III}} v_{\mathrm{IV}})^\omega$ and $(v_{\mathrm{III}} v_{\mathrm{IV}} v_{\mathrm{I}} v_{\mathrm{II}})^\omega$ are, e.g., consistent with both σ and τ. In contrast, the plays $(v_{\mathrm{I}} v_{\mathrm{II}} v'_{\mathrm{III}} v_{\mathrm{IV}})^\omega$ and $(v'_{\mathrm{III}} v_{\mathrm{IV}} v'_{\mathrm{I}} v_{\mathrm{II}})^\omega$ are both consistent with σ'. Furthermore, the former play is consistent with τ, while the latter one is consistent with τ'. △

Picking an initial vertex as well as strategies for the two players uniquely determines the resulting play.

Remark 2.3. *Let $\mathcal{A}$ be an arena, let σ and τ be strategies for Player 0 and Player 1, respectively, and let v be some vertex of $\mathcal{A}$. There exists a single, unique play in $\mathcal{A}$ that starts in v and is consistent with both σ and τ.*

The above, very general definition of strategies affords the player playing consistently with the strategy potentially infinite computational resources. In order to obtain a more realistic model of strategies, we define finite-state strategies. In such strategies, the respective player keeps track of the current "state" of the play using a finite-state machine that processes the play prefix constructed thus far.

A MEMORY STRUCTURE $\mathcal{M} = (M, init, upd)$ for an arena $\mathcal{A}$ with vertex set V and set E of edges consists of a finite set of memory states M, an initialization function $init\colon V \to M$, as well as an update function $upd\colon M \times E \to M$. We extend the update function to finite play prefixes via $upd^+(m, v) = m$ and $upd^+(m, \pi v v') = upd(upd^+(m, \pi v), (v, v'))$ for all $\pi \in V^*$ and $(v, v') \in E$. We define $|\mathcal{M}| = |M|$.

Def. memory structure

A NEXT-MOVE FUNCTION for Player i is a function $nxt\colon V_i \times M \to V$ that satisfies $(v, nxt(v, m)) \in E$ for all $v \in V_i$ and all $m \in M$. A memory structure $\mathcal{M}$ together with a next-move function nxt induce a strategy σ for Player i with memory $\mathcal{M}$ via

Def. next-move function

$$\sigma(v_0 \cdots v_j) = nxt(v_j, upd^+(init(v_0), v_0 \cdots v_j)) \ .$$

A strategy is called FINITE-STATE if it can be implemented by a memory structure. In a slight abuse of notation, we define the SIZE OF A FINITE-STATE STRATEGY $|M|$ as the size of a memory structure M implementing it.

Def. finite-state strategy

Def. size of strategy

An arena $\mathcal{A} = (V, V_0, V_1, E)$ and a memory structure $\mathcal{M} = (M, init, upd)$ for $\mathcal{A}$ induce the EXTENDED ARENA $\mathcal{A} \times \mathcal{M} = (V \times M, V_0 \times M, V_1 \times M, E')$ where we define E' as follows: We have $((v, m), (v', m')) \in E'$ if and only if $(v, v') \in E$ and $upd(m, (v, v')) = m'$. Every play $\rho = v_0 v_1 v_2 \cdots$ in $\mathcal{A}$ has a unique EXTENDED PLAY $\mathrm{ext}_{\mathcal{M}}(\rho) = (v_0, m_0)(v_1, m_1)(v_2, m_2) \cdots$ in $\mathcal{A} \times \mathcal{M}$ defined by $m_0 = init(v_0)$ and

Def. extended arena

Def. extended play

$$m_{j+1} = upd(m_j, (v_j, v_{j+1})), \text{ i.e., } m_j = upd^+(init(v_0), v_0 \cdots v_j) \ .$$

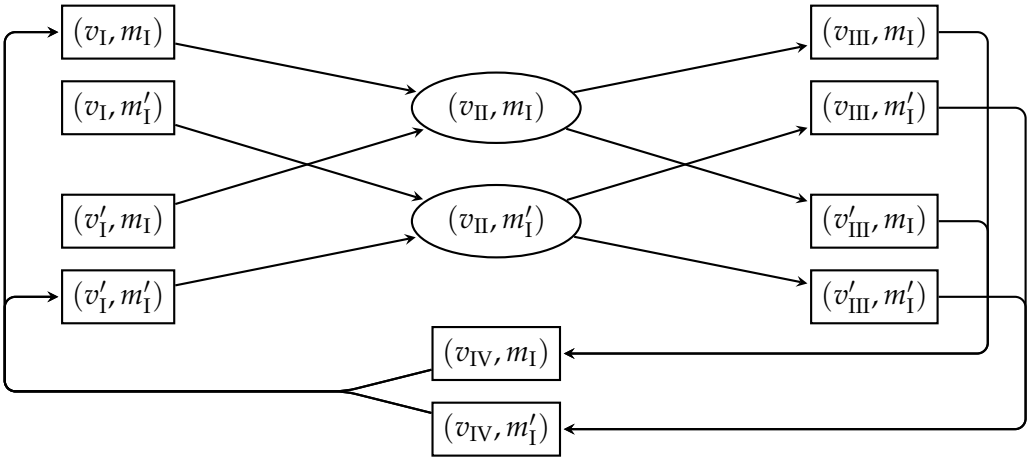


Figure 2.2: The arena from Figure 2.1 extended with the memory structure from Example 2.4.

If the memory structure $\mathcal{M}$ is clear from the context, we omit it from the notation for the extended play and just write $\mathrm{ext}(\rho)$. The extended play of a finite play prefix in $\mathcal{A}$ is defined analogously.

Example 2.4. Consider again the arena from Example 2.1 shown in Figure 2.1. Let σ be the strategy for Player 0 defined by

$$\sigma(v_0 \cdots v_j) = \begin{cases} v_\mathrm{III} & \text{if } j > 0 \text{ and if } v_{j-1} = v_\mathrm{I}, \text{ and} \\ v'_\mathrm{III} & \text{otherwise} \end{cases}$$

for all play prefixes $v_0 \cdots v_j$ with $v_j = v_\mathrm{II}$, i.e., σ prescribes for Player 0 to "mimick" the choices of upper and lower vertices made by Player 1, initially moving to v'_III if the play starts in v_II.

The strategy σ can be implemented using the memory structure $\mathcal{M} = (M, \mathit{init}, \mathit{upd})$ with $M = \{m_\mathrm{I}, m'_\mathrm{I}\}$, $\mathit{init}(v) = m_\mathrm{I}$ if $v = v_\mathrm{I}$ and $\mathit{init}(v) = m'_\mathrm{I}$ otherwise, and

$$\mathit{upd}(m, (v, v')) = \begin{cases} m_\mathrm{I} & \text{if } v' = v_\mathrm{I}, \text{ and} \\ m'_\mathrm{I} & \text{otherwise.} \end{cases}$$

We illustrate the extended arena $\mathcal{A} \times \mathcal{M}$ in Figure 2.2. Moreover, since the memory structure implementing σ has two elements, we directly obtain $|\sigma| = 2$. △

2.2 Complexity Classes

We assume the reader to be familiar with the notion of decision problems. Furthermore, we assume that the reader has already encountered the models of alternating, nondeterministic, unambiguous, and deterministic Turing machines and is familiar with the notion of a Turing machine deciding a problem [Tur37, Pap94, HMU01].

Class	Model	Bound	
		Time	Space
PTime	deterministic	polynomial	–
UP	unambiguous	polynomial	–
NP	nondeterministic	polynomial	–
APTime	alternating	polynomial	–
PSpace	deterministic	–	polynomial
ExpTime	deterministic	exponential	–

Table 2.3: Complexity classes used in this thesis.

In order to achieve uniform notation, however, we briefly recall their definitions informally: The states of an ALTERNATING TURING MACHINE $\mathfrak{T}$ are partitioned into existential and universal states. A run of $\mathfrak{T}$ is a tree that is rooted at the initial configuration of $\mathfrak{T}$. Each configuration that is in an existential state has a single outgoing edge to one of its successor configurations. In contrast, each configuration that is in a universal state has outgoing edges to each of its successor configurations. A run of $\mathfrak{T}$ is accepting if all branches of the run terminate in a configuration with an accepting state.

Def. alternating Turing machine

The other machines mentioned above result from syntactical and semantic restrictions of this model:

- A NONDETERMINISTIC TURING MACHINE is an alternating Turing machine without universal states,
- an UNAMBIGUOUS TURING MACHINE is a nondeterministic Turing machine that has at most one accepting run for each input, and
- a DETERMINISTIC TURING MACHINE is a nondeterministic Turing machine where each configuration has at most one successor configuration.

Def. nondeterministic Turing machine

Def. unambiguous Turing machine

Def. deterministic Turing machine

The complexity classes considered in this thesis arise from restricting the time or space available to Turing machines to decide an instance of a given problem. We give an overview over these complexity classes in Table 2.3. Furthermore, we denote the complement of UP and of NP by coUP and coNP, respectively.

By definition we have PTime $\subseteq$ UP $\subseteq$ NP $\subseteq$ APTime. We furthermore have PTime $\subseteq$ coUP $\subseteq$ coNP and coNP $\subseteq$ PSpace $\subseteq$ ExpTime. Results by Chandra, Kozen, and Stockmeyer [CKS81] furthermore yield APTime $=$ PSpace.

For a more thorough discussion of these models and complexity theory in general, we refer to introductory works, e.g., by Papadimitriou [Pap94].

2.3 Qualitative Games

Having defined arenas, we now turn our attention to games. As we have already described how the two players interact in order to construct a play, it remains to decide which player wins the resulting infinite play. To this end, we equip an arena with a

so-called winning condition, obtaining a game. For the remainder of this section, fix some arena $\mathcal{A}$ with vertex set V and set of edges E.

Def. winning condition

A WINNING CONDITION $\text{Win} \subseteq V^\omega$ is a set of infinite sequences of vertices from $\mathcal{A}$. We say that a play ρ of $\mathcal{A}$ is WINNING for Player 0 (according to Win) if $\rho \in \text{Win}$. In this case, we also say that ρ satisfies the winning condition Win. Dually, we say that a play ρ is winning for Player 1 if $\rho \in V^\omega \setminus \text{Win}$. A GAME $\mathcal{G} = (\mathcal{A}, \text{Win})$ consists of an arena $\mathcal{A}$ and a winning condition Win.

Def. winning play

Def. game

Def. 0-extendable

Def. 1-extendable

A winning condition Win is 0-EXTENDABLE if for all $\rho \in V^\omega$ and all $\pi \in V^*$, $\rho \in \text{Win}$ implies $\pi\rho \in \text{Win}$. Dually, Win is 1-EXTENDABLE if for all $\rho \in V^\omega$ and all $\pi \in V^*$, $\rho \notin \text{Win}$ implies $\pi\rho \notin \text{Win}$. If Win is both 0-extendable and 1-extendable, we say that Win is PREFIX-INDEPENDENT. Intuitively, if a winning condition is i-extendable, Player i may allow some finite prefix in which the winning condition is not yet satisfied and she is allowed to only "play to win" in the infinite.

Def. prefix-independent

Example 2.5. Let $\mathcal{A}$ be the arena with vertex set V shown in Figure 2.1. We define the winning condition Win via

$$\text{Win} = \big\{ v_0 v_1 v_2 \cdots \in V^\omega \mid (\exists^\omega j.\ (v_j = v_\text{I})) \Rightarrow (\exists^\omega j.\ (v_j = v_\text{III})) \wedge (\exists^\omega j.\ (v_j = v'_\text{I})) \Rightarrow (\exists^\omega j.\ (v_j = v'_\text{III})) \big\} \ ,$$

where we write $\exists^\omega$ to denote "there exist infinitely many." Thus, we demand that if v_I or v'_I are visited infinitely often, then v_III or v'_III, respectively, are visited infinitely often.

The plays $(v_\text{I} v_\text{II} v_\text{III} v_\text{IV})^\omega$ and $(v_\text{II} v'_\text{III} v_\text{IV} v'_\text{I})^\omega$ are winning for Player 0. Dually, the play $(v'_\text{I} v_\text{II} v_\text{III} v_\text{IV})^\omega$ is winning for Player 1.

Prepending a finite prefix to a play ρ does not change the set of vertices that appear infinitely often in ρ. Thus, the winning condition Win is both 0-extendable as well as 1-extendable, i.e., it is prefix-independent. △

Def. winning strategy

Let $\mathcal{G} = (\mathcal{A}, \text{Win})$ be a game. We say that a strategy σ for Player 0 is a WINNING STRATEGY for her from $v \in V$ if all plays that start in v and are consistent with σ are winning for her. Dually, a strategy τ for Player 1 is a winning strategy for him from $v \in V$ if every play that starts in v and is consistent with τ is not a member of Win.

Def. winning a game

Def. winning region

Def. solving

If Player i has a winning strategy from v, then we say Player i WINS $\mathcal{G}$ from v and lift this notion to sets of vertices by saying that Player i wins $\mathcal{G}$ from $V' \subseteq V$, if she wins $\mathcal{G}$ from each vertex in V'. The WINNING REGION of Player i is the set of vertices from which Player i wins $\mathcal{G}$. We denote the winning region of Player i in $\mathcal{G}$ by $W_i(\mathcal{G})$. SOLVING a game amounts to determining its winning regions and computing winning strategies for either player.

Remark 2.6. *For each game $\mathcal{G}$ we have $W_0(\mathcal{G}) \cap W_1(\mathcal{G}) = \emptyset$.*

Example 2.7. Let again $\mathcal{A}$ be the arena shown in Figure 2.1, let Win be the winning condition defined in Example 2.5 and let $\mathcal{G} = (\mathcal{A}, \text{Win})$. The strategy σ defined in Example 2.4 is winning for Player 0 from all vertices of $\mathcal{A}$. Hence, we obtain $W_0(\mathcal{G}) = V$ and, subsequently, $W_1(\mathcal{G}) = \emptyset$ due to Remark 2.6. △

Due to Remark 2.6 there exists no vertex v such that both players win $\mathcal{G}$ from v. This remark does not, however, imply that from each vertex either player wins. If this is the case, i.e., if $W_0(\mathcal{G}) \cup W_1(\mathcal{G}) = V$, then we say that $\mathcal{G}$ is DETERMINED.

Def. determined

All winning conditions considered in this work belong to the so-called BOREL HIERARCHY, which consists of sets $\mathbf{\Sigma}_j$ and $\mathbf{\Pi}_j$ for each $j \in \mathbb{N}$. In fact, this hierarchy furthermore contains ordinal levels that subsume all sets $\mathbf{\Sigma}_j$ and $\mathbf{\Pi}_j$. Since we, however, only require the levels $\mathbf{\Sigma}_j$ and $\mathbf{\Pi}_j$ for $j \in \mathbb{N}$ in this work, we omit these ordinal levels.

Def. Borel hierarchy

The Borel hierarchy is defined inductively and at the lowest level contains open sets and their complements. A set $L \subseteq V^\omega$ is OPEN if $L = \{\pi\rho \mid \pi \in K, \rho \in V^\omega\}$ for some set $K \subseteq V^*$. The Borel hierarchy is then defined via

Def. open set

- $\mathbf{\Sigma}_1 = \{L \subseteq V^\omega \mid L \text{ is open}\}$,
- $\mathbf{\Pi}_j = \left\{L \subseteq V^\omega \mid V^\omega \setminus L \in \mathbf{\Sigma}_j\right\}$ for $j \geq 1$, and
- $\mathbf{\Sigma}_{j+1} = \left\{L \subseteq V^\omega \mid L = \bigcup_{k\in\mathbb{N}} L_k, \text{ where } L_k \in \mathbf{\Pi}_j \text{ for all } k \in \mathbb{N}\ \right\}$ for $j \geq 1$.

We say that a winning condition $\text{Win} \subseteq V^\omega$ is BOREL if it is part of the Borel hierarchy, i.e., if $\text{Win} \in \mathbf{\Sigma}_j \cup \mathbf{\Pi}_j$ for some $j \in \mathbb{N}$.

Def. Borel winning condition

Proposition 2.8 ([Mar75]). *Let $\mathcal{G} = (\mathcal{A}, Win)$ be a game. If Win is Borel, then $\mathcal{G}$ is determined.*

The construction of the Borel hierarchy above only allows for union and complementation of sets. Due to De Morgan's rule, however, Proposition 2.8 yields that any game whose winning condition can be expressed as a Boolean combination of open sets, is determined.

Our definition so far allows for arbitrary and very complex winning conditions. In the following sections, we introduce common winning conditions and show how to solve games with such winning conditions.

2.3.1 Safety Games and Attractors

We start with a very simple class of winning conditions that require Player 0 to ensure that the play never reaches a given set of "unsafe" vertices.

Throughout this section, fix some arena $\mathcal{A}$ with vertex set V and set of edges E. Given a set of vertices $U \subseteq V$, the SAFETY CONDITION $\textsc{Safety}(U)$ is defined as

Def. safety condition

$$\textsc{Safety}(U) = \left\{v_0v_1v_2 \cdots \in V^\omega \mid \forall j \in \mathbb{N}.\, v_j \notin U\right\} \ ,$$

i.e., a play satisfies the safety condition if it does not contain vertices from U. We call any game $\mathcal{G} = (\mathcal{A}, \text{Win})$ where $\text{Win} = \textsc{Safety}(U)$ for some subset U of the vertices of $\mathcal{A}$ a SAFETY GAME.

Def. safety game

For the remainder of this section, fix some set $U \subseteq V$. We now show how to solve the game $\mathcal{G} = (\mathcal{A}, \textsc{Safety}(U))$.

To this end, we first note that $\textsc{Safety}(U)$ is in $\mathbf{\Pi}_1$ and thus Borel: We have

$$\textsc{Safety}(U) = V^\omega \setminus \left\{v_0v_1v_2 \cdots \in V^\omega \mid \exists j \in \mathbb{N}.\, v_j \in U\right\} \ ,$$

where the latter set clearly is open, i.e., a member of $\mathbf{\Sigma}_1$. Hence, we obtain determinacy of $\mathcal{G}$ via Proposition 2.8.

Remark 2.9. *Safety games are determined.*

In order to solve $\mathcal{G}$, we compute the set of vertices from which Player 1 can force the play to move to a vertex from U. As this notion is used in multiple places throughout this work, we introduce it here in a general formulation.

Let $X \subseteq V$ and let $i \in \{0,1\}$. We inductively determine the vertices from which Player i can force the play to visit a vertex from X by defining $\mathrm{Attr}_i^0(X) = X$ and

$$\mathrm{Attr}_i^j(X) = \mathrm{Attr}_i^{j-1}(X) \cup \left\{ v \in V_i \mid \exists v' \in \mathrm{Attr}_i^{j-1}(X).\, (v,v') \in E \right\} \cup \left\{ v \in V_{1-i} \mid \forall (v,v') \in E.\, v' \in \mathrm{Attr}_i^{j-1}(X) \right\} .$$

Def. *i*-attractor

The i-ATTRACTOR OF X is then defined as $\mathrm{Attr}_i(X) = \bigcup_{j \in \mathbb{N}} \mathrm{Attr}_i^j(X)$. If we want to emphasize the arena $\mathcal{A}$ the attractor is computed in, we write $\mathrm{Attr}_i^{\mathcal{A}}(X)$.

A straightforward induction yields that the computation of the attractor terminates after at most $|V|$ many steps.

Remark 2.10. *For all $i \in \{0,1\}$ and all $X \subseteq V$ we have $Attr_i(X) = Attr_i^{|V|}(X)$.*

Furthermore, Nerode, Remmel, and Yakhnis [NRY96] showed that the i-attractor of X can be computed in linear time in $|E|$.

Proposition 2.11 ([NRY96])**.** *Let $X \subseteq V$ and let $i \in \{0,1\}$.*

1. *Player i has a positional strategy σ_X such that each play starting in $Attr_i(X)$ that is consistent with σ_X contains a vertex from X.*
2. *The set $Attr_i(X)$ and the strategy σ_X can be computed in linear time in $|E|$.*

Def. attractor strategy

We call a strategy σ_X as defined in Proposition 2.11.1 a ATTRACTOR STRATEGY of Player i towards X.

→Sec. 2.1, Page 12

Example 2.12. Let $\mathcal{A}$ again be the arena from → Figure 2.1. The 0-attractor of $\{v_\mathrm{I}\}$ is the singleton set containing only v_I, as Player 1 may always choose to move to v'_I from v_IV, i.e., Player 0 cannot force the play to move to v_I from any vertex other than v_I.

The 0-attractor of $\{v_\mathrm{I}, v'_\mathrm{I}\}$, however, is the complete set of vertices of $\mathcal{A}$. While Player 0 cannot guarantee which vertex from that set will be visited, the structure of $\mathcal{A}$ guarantees that either of the vertices will eventually be visited. △

The i-attractor of X indeed contains all vertices from which Player i can force the play to enter X at some point: Let $v \in V \setminus \mathrm{Attr}_i(X)$. If $v \in V_i$, then v has only successors in $V \setminus \mathrm{Attr}_i(X)$. Dually, if $v \in V_{1-i}$, then v has at least one successor in $V \setminus \mathrm{Attr}_i(X)$. We illustrate this situation in Figure 2.4 for Player 0.

Def. trap

Moreover, we call any set that satisfies the same property as $V \setminus \mathrm{Attr}_i(X)$ a TRAP for Player i, as that player is unable to enforce leaving this set of vertices.

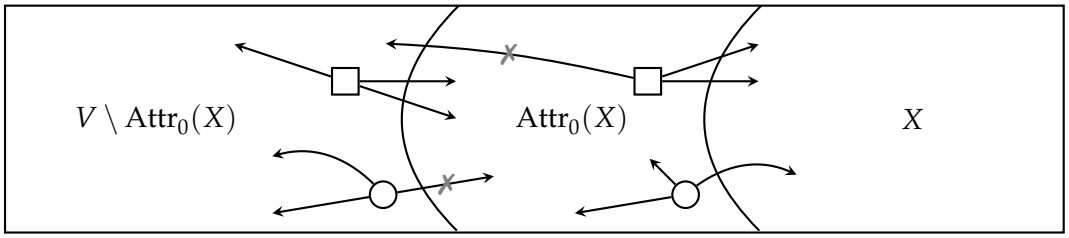


Figure 2.4: The 0-attractor of some set X and the corresponding trap for Player 0. Edges that cannot occur in the depicted situation are crossed out.

Remark 2.13. *The complement of an i-attractor is a trap for Player i.*

Player $1-i$ is always able to prevent a play that has entered a trap for Player i from leaving the trap again by simply taking edges leading back into the trap at every of his vertices.

Remark 2.14. *Let $i \in \{0,1\}$ and let $X \subseteq V$ be a trap for Player i. Player $1-i$ has a positional strategy τ_X such that no play starting in X that is consistent with τ_X contains a vertex from $V \setminus X$.*

We call a strategy τ from Remark 2.14 a TRAP STRATEGY for Player 1. Moreover, traps for Player i are closed under the computation of the $1-i$-attractor. **Def.** trap strategy

Remark 2.15. *If X is a trap for Player i, then $Attr_{1-i}(X)$ is also a trap for Player i.*

Using the notion of attractors it is straightforward to solve the safety game $\mathcal{G} = (\mathcal{A}, \textsc{Safety}(U))$. Solving $\mathcal{G}$ amounts to nothing more than computing the set of vertices from which Player 1 can enforce visiting an "unsafe" vertex, i.e., to computing the 1-attractor of U. Thus, we obtain $W_1(\mathcal{G}) = \mathrm{Attr}_1(U)$ and, consequently, $W_0(\mathcal{G}) = V \setminus W_1(\mathcal{G})$ due to Remark 2.9. As the attractor can be computed in linear time due to Proposition 2.11, the game $\mathcal{G}$ can be solved in linear time.

Remark 2.16. *The following problem is in* PTIME*:*

> *"Given a safety game $\mathcal{G} = (\mathcal{A}, \textsc{Safety}(U))$ and a vertex v of $\mathcal{A}$, does Player 0 win $\mathcal{G}$ from v?"*

Moreover, we observe that the attractor strategy of Player 1 towards $\mathrm{Attr}_1(U)$ and the trap strategy of Player 0 in $V \setminus \mathrm{Attr}_1(U)$ are winning strategies for either player from their respective winning region. Hence, both players have positional winning strategies.

Remark 2.17. *Let $\mathcal{G}$ be a safety game and let $i \in \{0,1\}$. Player i has a positional strategy that is winning for her from $W_i(\mathcal{G})$.*

2.3.2 Parity Games

Having discussed the very simple safety condition, we now consider the more involved parity condition. Again fix some arena $\mathcal{A}$ with vertex set V and set of edges E. A parity condition is given by coloring each vertex from V with some nonnegative integer. We interpret a visit to an odd color as a REQUEST that is ANSWERED by visiting a larger even color. Using this interpretation, we demand that Player 0 ensures that almost all, i.e., all but finitely many requests are eventually answered.

Def. request
Def. answer

Formally, a COLORING of a vertex set V is a function $\Omega\colon V \to \mathbb{N}$. Since we interpret odd colors as requests and larger even colors as answers, for each color $c \in \mathbb{N}$ we define

$$\mathrm{Ans}(c) = \{c' \in \mathbb{N} \mid c' \geq c \text{ and } c' \text{ is even}\}$$

as the set of colors that answer a request for color c. We say that a request for color c is OPEN in a play prefix $\pi = v_0 \cdots v_j$ if there exists a $j' \leq j$ such that $\Omega(v_{j'}) = c$, but there exists no $j'' \geq j'$ such that $\Omega(v_{j''}) \in \mathrm{Ans}(c)$. Furthermore, we say that a request is CLOSED if it is not open. Finally, we say that in a play $v_0 v_1 v_2 \cdots$ a visit to vertex $v_{j'}$ ANSWERS a request opened by visiting v_j if $\Omega(v_{j'}) \in \mathrm{Ans}(\Omega(v_j))$. Since for each even color c we have $c \in \mathrm{Ans}(c)$, only requests for odd colors can be open.

Def. open request
Def. closed request
Def. answer

Def. coloring

The PARITY CONDITION over Ω is then defined via

$$\textsc{Parity}(\Omega) = \{v_0 v_1 v_2 \cdots \in V^\omega \mid \exists j \in \mathbb{N}.\, \forall j' \geq j.\, \exists j'' \geq j'.\, \Omega(v_{j''}) \in \mathrm{Ans}(\Omega(v_{j'}))\} \quad ,$$

i.e., we require that after some finite prefix each request is answered. Equivalently, and more standard, a play satisfies the parity condition if the maximal color occurring infinitely often in a play is even. Analogously to safety games, we say that a game $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ is a PARITY GAME if $\mathrm{Win} = \textsc{Parity}(\Omega)$ for some coloring Ω of V. When drawing parity games, we label a vertex v with color c by v/c.

Def. parity condition

Def. parity game

Mostowski [Mos91] as well as Emerson and Jutla [EJ91] showed that parity games are determined, and that both players have positional winning strategies.

Proposition 2.18 ([Mos91, EJ91])**.** *If a game $\mathcal{G}$ is a parity game, then $\mathcal{G}$ is determined and both players have positional winning strategies from their winning regions.*

Example 2.19. We show an example of a parity game $\mathcal{G}$ in Figure 2.5. Player 0 wins $\mathcal{G}$ from every vertex using the positional strategy σ induced by $\sigma(v_{\mathrm{II}}) = v'_{\mathrm{III}}$, since the even color $\Omega(v'_{\mathrm{III}}) = 4$ is the largest color occurring in $\mathcal{G}$. △

Jurdzinski [Jur98] showed that the problem of solving parity games is in UP ∩ coUP, i.e., both the associated decision problem and its complement can be decided in polynomial time using an unambiguous Turing machine. Furthermore, Calude et al. [CJK+17] very recently showed that the problem can be solved in quasi-polynomial time.

Proposition 2.20 ([Jur98, CJK+17])**.** *The following problem is in* UP ∩ coUP*:*

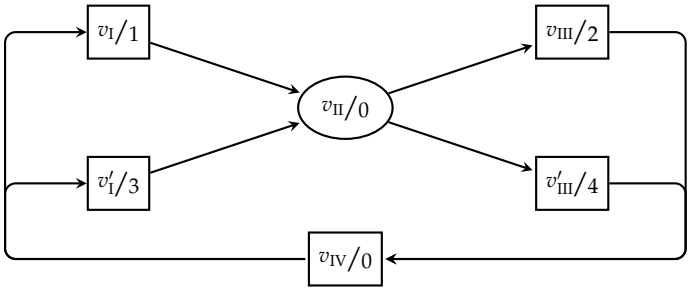


Figure 2.5: A parity game. Adapted from Chatterjee and Fijalkow [CF13].

> *"Given a parity game $\mathcal{G}$ with n vertices and d odd colors and a vertex v of $\mathcal{G}$, does Player 0 win $\mathcal{G}$ from v?"*

Moreover, the problem can be solved in time $\mathcal{O}(n^{\log d+6})$.

The problem of whether there exists a polynomial-time algorithm solving parity games is one of the major problems of theoretical computer science and has been open for decades [EJS93].

Recently, a number of algorithms have been developed that solve parity games in quasi-polynomial time [CJK+17,FJS+17,JL17,Leh18]. There exists, however, other work that shows that the approach taken by state-of-the-art algorithms will not yield sub-exponential runtime [Fri09, CDF+18].

2.4 Quantitative Games

Both winning conditions discussed thus far feature a very simple criterion for deciding whether or not a play is winning for Player 0: In safety games, either a play only visits safe vertices, or it does not. In parity games, the maximal color visited infinitely often is either even or odd. While this stark distinction is conceptually simple, it requires an external ordering of plays to reason about their "quality" apart from binary judgments.

We now discuss more involved winning conditions, in which each play is assigned a cost. The players then compete to minimize (for Player 0) or maximize (for Player 1) the cost of the constructed play, thus inducing a more gradual distinction between winning and losing plays.

Recall that in a parity game, Player 0 is only required to ensure that the maximal color seen infinitely often is even. Equivalently, when interpreting odd colors as requests and larger even colors as responses to these requests, this allows Player 0 to let the "waiting times" between requests and responses diverge, which is, in general, undesirable.

Example 2.21. Consider the parity game $\mathcal{G}$ shown in Figure 2.6. Player 0 wins $\mathcal{G}$ from all vertices. As she has no influence on the construction of the play in $\mathcal{G}$, this implies that all plays in $\mathcal{G}$ are winning for her.

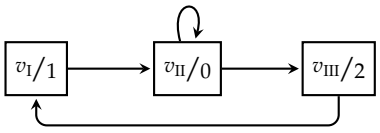


Figure 2.6: A parity game witnessing undesirable plays that Player 0 cannot avoid, but that are winning for her.

In particular the play

$$\rho = v_{\mathrm{I}}v_{\mathrm{II}}v_{\mathrm{III}} \cdot v_{\mathrm{I}}v_{\mathrm{II}}v_{\mathrm{II}}v_{\mathrm{III}} \cdot v_{\mathrm{I}}v_{\mathrm{II}}v_{\mathrm{II}}v_{\mathrm{II}}v_{\mathrm{III}} \cdot v_{\mathrm{I}}v_{\mathrm{II}}v_{\mathrm{II}}v_{\mathrm{II}}v_{\mathrm{II}}v_{\mathrm{III}} \cdots \ ,$$

is winning for Player 0, as all requests are eventually answered. The waiting times for the answers to these requests, however, diverge: The first request is answered after two steps, while the answer for the second request only occurs after three steps, and so on. △

In order to prevent Player 0 from winning plays with such divergent waiting times, Chatterjee and Henzinger [CH06] introduced the finitary parity condition, in which Player 0 not only has to satisfy the parity condition, but she also has to ensure an upper bound on the number of steps taken between requests and responses. Subsequently, Fijalkow and Zimmermann [FZ14] extended this very simple cost model by allowing arbitrary nonnegative weights along the edges of the arena, obtaining the parity condition with costs.

In this section, we introduce these two winning conditions formally and briefly discuss their major characteristics. Throughout this section, fix some arena $\mathcal{A}$ with vertex set V and set of edges E as well as a coloring Ω of V.

2.4.1 Finitary Parity Games

The motivation behind the introduction of the finitary parity condition is to only let Player 0 win those plays in which she is eventually able to ensure an upper bound on the waiting time between requests and responses. To this end, we first define the DISTANCE between a request and its earliest response via

Def. distance between request and response

$$\mathrm{Dist}(v_0v_1v_2\cdots, j) = \min\left\{j' - j \mid j' \geq j,\, \Omega(v_{j'}) \in \mathrm{Ans}(\Omega(v_j))\right\} \ ,$$

where $\min \emptyset = \infty$. This definition implies that, if the request for some odd color at position j is unanswered, we obtain $\mathrm{Dist}(\rho, j) = \infty$. Moreover, if $\Omega(v_j)$ is even, we have $\mathrm{Dist}(\rho, j) = 0$ due to $\Omega(v_j) \in \mathrm{Ans}(\Omega(v_j))$. We extend the notion of COST to plays by defining

Def. cost of play

$$\mathrm{Cost}(\rho) = \limsup_{j\to\infty} \mathrm{Dist}(\rho, j) \ .$$

Def. finitary parity condition

The FINITARY PARITY CONDITION, introduced by Chatterjee and Hezinger [CH06]

then requires Player 0 to eventually ensure a finite upper bound on the distance between requests and their responses. Formally, we define

$$\textsc{FinParity}(\Omega) = \{\rho \in V^\omega \mid \mathrm{Cost}(\rho) < \infty\} \quad .$$

Analogously to the safety condition and the parity condition, we call a game $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ a FINITARY PARITY GAME if $\mathrm{Win} = \textsc{FinParity}(\Omega)$ for some coloring Ω of $\mathcal{A}$. Since $\mathrm{Dist}(\rho, j) < \infty$ implies that the request at position j of ρ is eventually answered, $\rho \in \textsc{FinParity}(\Omega)$ implies that all but finitely many requests in ρ are answered. Hence, the finitary parity condition strengthens the parity condition.

Def. finitary parity game

Remark 2.22. *For each coloring Ω we have $\textsc{FinParity}(\Omega) \subseteq \textsc{Parity}(\Omega)$.*

Moreover, the finitary parity condition indeed implements the above intuition of preventing Player 0 from letting the waiting times between requests and responses diverge.

Example 2.23. Consider again the arena $\mathcal{A}$ and the coloring Ω shown in Figure 2.6 as well as the play ρ defined in Example 2.21 and recall $\rho \in \textsc{Parity}(\Omega)$. Since we have $\limsup_{j\to\infty} \mathrm{Dist}(\rho, j) = \infty$ we, in contrast, obtain $\rho \notin \textsc{FinParity}(\Omega)$, i.e., ρ is indeed losing for Player 0 in the finitary parity game $(\mathcal{A}, \textsc{FinParity}(\Omega))$. △

Chatterjee and Henzinger [CH06] showed that the problem of solving finitary parity games is in NP ∩ coNP and that such games can be solved in exponential time, which was later improved to PTIME by Chatterjee, Henzinger, and Horn [CHH09]. From the algorithm solving finitary parity games, the authors furthermore obtain an argument that finitary parity games are determined.

Proposition 2.24 ([CH06])**.** *Finitary parity games are determined.*

Moreover, the authors showed Player 0 still only requires positional strategies in order to win in finitary parity games. Dually, Player 1 has, in general, no finite-state winning strategy, as witnessed by the game shown in Figure 2.6.

Proposition 2.25 ([CH06])**.**

1. *Let $\mathcal{G}$ be a finitary parity game. Player 0 has a positional strategy that is winning for her from $W_0(\mathcal{G})$.*
2. *There exists a finitary parity game $\mathcal{G}$ with vertex set V such that $W_1(\mathcal{G}) = V$, but Player 1 has no finite-state winning strategy from any vertex of $\mathcal{G}$.*

Since positional strategies suffice for Player 0 to win from her winning region in finitary parity games, a straightforward argument yields that, if she is able to bound the distance between requests and responses at all, then she is able to bound this distance by the number of vertices in $\mathcal{A}$. In order to state this proposition succinctly, we lift the notion of costs to strategies.

To this end, we observe that the simple definition $\mathrm{Cost}(\sigma) = \sup_\rho \mathrm{Cost}(\rho)$, where ρ ranges over all plays consistent with σ would entail $\mathrm{Cost}(\sigma) = \infty$ if $W_0(\mathcal{G}) \neq V$.

Thus, in order to obtain a more useful notion, we include the starting vertex into the definition of the COST OF A STRATEGY and define

Def. cost of strategy

$$\mathrm{Cost}_v(\sigma) = \sup_{\rho} \mathrm{Cost}(\rho) \ ,$$

where ρ ranges over all plays starting in v and consistent with ρ.

Proposition 2.26 ([CH06]). *Let $\mathcal{G}$ be a finitary parity game with n vertices. Player 0 has a strategy σ such that $\max_{v \in W_0(\mathcal{G})} Cost_v(\sigma) \leq n$, with $\max \emptyset = 0$.*

While Chatterjee and Henzinger initially only showed that the problem of solving finitary parity games is in NP $\cap$ coNP, Chatterjee, Henzinger, and Horn [CHH09] later lowered that upper bound on the complexity by showing that finitary parity games can be solved in polynomial time.

Proposition 2.27 ([CHH09]). *The following problem is in* PTIME*:*

> *"Given a finitary parity game $\mathcal{G}$ with n vertices and a vertex v of $\mathcal{G}$, does Player 0 win $\mathcal{G}$ from v?"*

Moreover, it can be solved in time $\mathcal{O}(n^4)$.

Thus, there exist algorithms for solving finitary parity games that are asymptotically faster than the fastest currently known algorithms solving parity games.
2

2.4.2 Parity Games with Costs

The finitary parity condition introduced by Chatterjee, Henzinger and Horn has a very simple underlying cost model: Each traversal of an edge incurs unit cost, i.e., the cost of answering a request is the number of turns taken until it is answered.

Fijalkow and Zimmermann [FZ14] extended this simple cost model by allowing for arbitrary nonnegative weights of edges, obtaining parity games with costs. Such games allow more freedom in the modeling of resources such as time, as traversing an edge may consume an arbitrary amount of the resource, or none at all.

In a parity game with costs, each edge e is assigned a weight w and traversing e increases the cost associated with all open requests by w. In order to win, Player 0 is again required to eventually bound the costs incurred between requests and responses.

Formally, fix some arena $\mathcal{A}$ with vertex set V and set of edges E as well as some coloring Ω of V. A WEIGHT FUNCTION is a mapping $\mathrm{Weight}\colon E \to \mathbb{N}$ that assigns to each edge a nonnegative weight. We extend the weight function Weight to play prefixes and plays as usual: $\mathrm{Weight}(\epsilon) = \mathrm{Weight}(v) = 0$ for empty play prefixes and play prefixes of length one and $\mathrm{Weight}(\pi \cdot vv') = \mathrm{Weight}(\pi v) + \mathrm{Weight}(v, v')$ for all longer play prefixes.

Def. weight function

Using this notion of weight of a play infix, we are now able to define the cost of a request in a play or play prefix. As we consider a request for color c answered

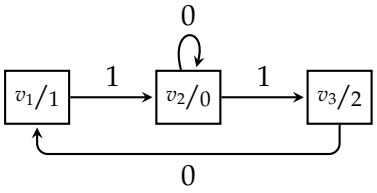


Figure 2.7: A parity game with costs.

as soon as the first color $c' \in \mathrm{Ans}(c)$ is visited, and since the weight of a play infix is monotonically increasing, we define the cost of the request for c as the minimal weight incurred along the infix leading to any first such color c'. Following this intuition, we define

$$\mathrm{Cor}(v_0 v_1 v_2 \cdots, j) = \min\left\{\mathrm{Weight}(v_j \cdots v_{j'}) \mid j' \geq j, \Omega(v_{j'}) \in \mathrm{Ans}(\Omega(v_j))\right\} \quad ,$$

where $\min \emptyset = \infty$.

We call $\mathrm{Cor}(\rho, j)$ the COST OF RESPONSE of the request posed at position j of ρ. Similarly to finitary parity games, we lift the notion of COST to plays via

Def. cost of response

Def. cost of play

$$\mathrm{Cost}(\rho) = \limsup_{j \to \infty} \mathrm{Cor}(\rho, j) \quad .$$

The weight function for edges is used implicitly in the definition of $\mathrm{Cost}(\rho)$. We ensure that the weight function used is always clear from the context.

Analogously to the finitary parity condition, the PARITY CONDITION WITH COSTS then requires Player 0 to bound the cost of plays. Formally, for a fixed coloring Ω and weight function Weight we define

Def. parity condition with costs

$$\textsc{CostParity}(\Omega, \mathrm{Weight}) = \{\rho \in V^\omega \mid \mathrm{Cost}(\rho) < \infty\} \quad .$$

Analogously to the previously discussed winning conditions, we call $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ a PARITY GAME WITH COSTS if $\mathrm{Win} = \textsc{CostParity}(\Omega, \mathrm{Weight})$ for some coloring Ω and some weight function Weight. We assume the weight function to be given in binary encoding. Thus, we have $|\mathcal{G}| = |\mathcal{A}| \lceil \log W \rceil$, where W is the largest weight occurring in $\mathcal{G}$.

Def. parity game with costs

Example 2.28. Consider the arena $\mathcal{A}$, the coloring Ω, and the weight function Weight shown in Figure 2.7. Moreover, consider once again the play

$$\rho = \underbrace{v_1 v_2 v_3}_{\pi_0} \cdot \underbrace{v_1 v_2 v_2 v_3}_{\pi_1} \cdot \underbrace{v_1 v_2 v_2 v_2 v_3}_{\pi_2} \cdot \underbrace{v_1 v_2 v_2 v_2 v_2 v_3}_{\pi_3} \cdots$$

defined in Example 2.21 with the infixes π_j of ρ defined as indicated. The play ρ satisfies the parity condition with costs, as we have $\mathrm{Weight}(\pi_j) = 2$ for each $j \in \mathbb{N}$. $\triangle$

When solving a parity game with costs, the actual weights assigned to edges are irrelevant. Instead it suffices to only distinguish for each edge e between the two cases $\mathrm{Weight}(e) = 0$ and $\mathrm{Weight}(e) > 0$.

Remark 2.29. *Let $\mathcal{G} = (\mathcal{A}, \textsc{CostParity}(\Omega, \mathit{Weight}))$ be a parity game with costs and define the weight function $\mathit{Weight}'(e) = \min\{\mathit{Weight}(e), 1\}$ for all edges e of $\mathcal{A}$. We then have $\textsc{CostParity}(\Omega, \mathit{Weight}) = \textsc{CostParity}(\Omega, \mathit{Weight}')$*

Fijalkow and Zimmermann showed that parity games with costs are determined, since their winning condition is Borel.

Proposition 2.30 ([FZ14])**.** *Parity games with costs are determined.*

Due to the flexibility afforded by including the weight function Weight as a parameter of the winning condition, the parity condition with costs subsumes both the parity condition as well as the finitary parity condition.

Remark 2.31. *For each coloring Ω and each weight function Weight we have that*

1. *if, for all $e \in E$, we have $\mathit{Weight}(e) = 0$, then we obtain $\textsc{CostParity}(\Omega, \mathit{Weight}) = \textsc{Parity}(\Omega)$, and that*
2. *if, for all $e \in E$, we have $\mathit{Weight}(e) > 0$, then we obtain $\textsc{CostParity}(\Omega, \mathit{Weight}) = \textsc{FinParity}(\Omega)$.*

The parity condition with costs strengthens the parity condition, but it generalizes the finitary parity condition.

Remark 2.32. *For each coloring Ω and each weight function Weight we have*

$$\textsc{FinParity}(\Omega) \subseteq \textsc{CostParity}(\Omega, \mathit{Weight}) \subseteq \textsc{Parity}(\Omega) \ .$$

Mogavero, Murano, and Sorrentino [MMS15] showed that the problem of solving a parity game with costs $\mathcal{G} = (\mathcal{A}, \textsc{CostParity}(\Omega, \mathrm{Weight}))$ with n vertices, d odd colors and largest weight W can be reduced to the problem of solving a parity game $\mathcal{G}' = (\mathcal{A} \times \mathcal{M}, \textsc{Parity}(\Omega'))$ with d colors, where $|\mathcal{M}| = dn$. Due to Proposition 2.20, we obtain that parity games with costs can be solved in quasipolynomial time and that both the corresponding decision problem and its complement can be solved in polynomial time on an unambiguous Turing machine.

Proposition 2.33 ([Jur98, MMS15, CJK+17])**.** *The following problem is in $\mathrm{UP} \cap \mathrm{coUP}$:*

> *"Given a parity game with costs $\mathcal{G}$ with n vertices, d odd colors, and largest weight W, and a vertex v of $\mathcal{G}$, does Player 0 win $\mathcal{G}$ from v?"*

Moreover, it can be solved in time $\mathcal{O}((n^4 d)^{\log d + 6})$.

Furthermore, Fijalkow and Zimmermann showed that Player 0 still does not require memory in order to win a parity game with costs. Since parity games with costs, however, subsume finitary parity games, as stated in Remark 2.31.2, the necessity of infinite memory for Player 1 follows directly from Proposition 2.25.2.

Proposition 2.34 ([FZ14])**.**

1. *Let $\mathcal{G}$ be a parity game with costs. Player 0 has a positional strategy that is winning for her from $W_0(\mathcal{G})$.*

	Complexity	Memory Player 0	Memory Player 1	Bounds
Safety	PTime	pos.	pos.	–
Parity	UP ∩ coUP quasi-polynomial	pos.	pos.	–
Finitary Parity	PTime	pos.	inf.	n
Parity with Costs	UP ∩ coUP quasi-polynomial	pos.	inf.	nW

Table 2.8: Characteristic properties of variants of parity games.

2. *There exists a parity game with costs $\mathcal{G}$ with vertex set V such that $W_1(\mathcal{G}) = V$, but Player 1 has no finite-state winning strategy from any vertex of $\mathcal{G}$.*

Finally, due to similar reasoning as in the case of finitary parity games, the existence of positional winning strategies for Player 0 implies that if Player 0 can eventually bound the cost incurred between requests and responses, then she can ensure that each answered request is followed by a response after visiting each vertex of the game at most once in the meantime.

Analogously to the case of finitary parity games, we lift the notion of cost to strategies for parity games with costs via $\mathrm{Cost}_v(\sigma) = \sup_\rho \mathrm{Cost}(\rho)$, where ρ ranges over all plays that start in v and that are consistent with σ.

Proposition 2.35 ([FZ14])**.** *Let $\mathcal{G}$ be a parity game with costs with n vertices and largest weight W. Player 0 has a strategy σ such that $\max_{v \in W_0(\mathcal{G})} Cost_v(\sigma) \leq nW$, with $\max \emptyset = 0$.*

2.5 Summary

We have defined arenas, games and strategies, as well as the safety condition and the parity condition. Moreover, we have introduced two quantitative extensions of the parity condition, namely the finitary parity condition and the parity condition with costs. Furthermore, we have discussed the characteristics of these winning conditions. We summarize these characteristics in Table 2.8.

Recall that the parity condition with costs only allows for nonnegative costs. In the following chapter, we lift this restriction and allow the use of arbitrary integer weights. We show how to solve the resulting games and we show that Player 0 now not only requires memory in order to implement a winning strategy, but that she in fact requires exponential memory. We furthermore show that exponential memory indeed suffices for her to win and that she, if she wins, can bound the maximal cost by an exponential value.

CHAPTER 3

Parity Games with Weights

In the previous section, we recalled the definition of two quantitative extensions of parity games. First, we recalled finitary parity games, introduced by Chatterjee and Henzinger [CH06]. In such games, Player 0 is required to eventually guarantee an upper bound on the number of edges traversed between a visit to a request for an odd color and the visit to a larger even color answering the request. This winning condition effectively models situations in which each step taken in the game incurs a unit cost of some resource, e.g., time. The requirement of an upper bound on the number of steps between requests and responses then amounts to demanding that Player 0 bounds the waiting time for responses.

Finitary parity games can be solved in polynomial time, as shown by Chatterjee, Henzinger, and Horn [CHH09]. Thus, according to the state of the art, solving such games is in fact easier than solving classical parity games. Moreover, there exist positional winning strategies for Player 0, while Player 1, in general, requires infinite memory to win.

Subsequently, we recalled parity games with costs, a generalization of finitary parity games due to Fijalkow and Zimmermann [FZ14]. In order to disentangle the cost incurred by a play infix from the structure of the underlying arena, and thus to allow for more flexibility in the modeling of resources, the authors extended the simple cost model of finitary parity games by allowing for arbitrary nonnegative weights: In parity games with costs, each edge is labeled with some nonnegative weight and it is the task of Player 0 to bound the weights accumulated between almost all visits to requests and their corresponding responses. Solving such games comes with an added difficulty over solving finitary parity games: While the weight incurred by a request is still monotonically increasing in this setting, it is not strictly monotonically increasing anymore, as a play may traverse edges of weight zero.

Due to this added complication, the problem of solving parity games with costs is

in UP ∩ coUP [MMS15]. While the problem of solving parity games with costs is thus believed to be harder than solving the special case of finitary parity games, it is only as hard as its subsumption of the problem of solving classical parity games requires it to be. The extension to nonnegative weights does not, however, entail an increase in the complexity of winning strategies: Positional strategies still suffice for Player 0 to win, i.e., winning strategies for her are not larger than in the special cases of parity games and finitary parity games.

➔ Page 4

We have argued in ➔ Section 1.2 that even this extended model of parity games with costs is insufficient to describe, e.g., scenarios in which the modeled systems contains a rechargeable resource. In order to mitigate this shortcoming, in this chapter, we further extend the model of parity games with costs by removing the restriction to nonnegative weights along the edges, obtaining parity games with weights.

This extension further complicates the problem of solving the resulting games: In contrast to parity games with costs, in parity games with weights the cost incurred of a request is not bounded by zero from below, nor does it develop monotonically. Hence, in a first step we adapt the definition of the cost of answering a request to take this non-monotonicity into account and demand that Player 0 not only bounds the costs incurred by request from above, as required by the parity condition with costs, but also from below.

➔ Sec. 2.4, Page 26

Moreover, recall that, when solving parity games with costs, we were able to only consider "abstract" costs due to ➔ Remark 2.29. This is no longer the case in parity games with weights, as the weight along a play infix may first increase, but later decrease and reach, e.g., zero again. In order to track such developments of the weight along a play infix correctly, we are required to retain precise information about the development of the weight, i.e., we are no longer able to replace the weights in a game by abstract ones.

We show that in spite of the more involved cost model and the complications described above, the problem of solving parity games with weights is only in NP ∩ coNP. While the generalization of the cost model thus only incurs a minor increase in the complexity of solving the resulting games, we furthermore show that Player 0 now requires exponential memory in order to implement a winning strategy. Exponential memory, however, also suffices for her to win. Hence, the additional modeling capability also comes at a price in terms of the memory required by Player 0 in order to implement a winning strategy.

The remainder of this chapter is structured as follows: First, we formalize the parity condition with weights in Section 3.1 and show that despite the additional complexity of the condition in comparison to the parity condition with costs, games with this novel condition are still determined.

Subsequently, we show in Section 3.2 how to solve parity games with weights. To this end, we show how to reduce the problem of solving parity games with weights to that of iteratively solving energy parity games, a well-known quantitative variant of parity games introduced by Chatterjee and Doyen [CD12]. This construction follows the structure of the proof of NP ∩ coNP-membership of solving parity games with costs by Fijalkow and Zimmermann [FZ14], which in turn follows the approach

of Chatterjee, Henzinger, and Horn [CHH09] to showing PTime-membership of the problem of solving finitary parity games. Due to recent advances by Daviaud, Jurdziński, and Lazić [DJL18], this reduction furthermore implies that parity games with weights can be solved in pseudo-quasi-polynomial time. Moreover, the associated decision problem is in NP ∩ coNP due to results by Chatterjee and Doyen [CD12].

Furthermore, in Section 3.3 we show that the same relation as shown in the previous section also holds in the opposite direction, i.e., we show how to solve energy parity games by iteratively solving parity games with weights. As a consequence, we obtain that solving parity games with weights is as hard as solving energy parity games.

Having thus determined the complexity of solving parity games with weights, we then turn our attention to the memory required by both players to implement winning strategies in Section 3.4. Since, in contrast to parity games with costs and due to the occurrence of negative weights, we are now required to keep track of the weight of the play so far, Player 0 now requires memory exponential in the size of the game in order to implement winning strategies. Clearly, Player 1 inherits the necessity of infinite memory in order to implement a winning strategy from the special case of finitary parity games (cf. ➔Proposition 2.25.2). ➔Sec. 2.4, Page 23

Finally, in Section 3.5 we then discuss the upper bounds on the costs incurred during a play that Player 0 can ensure if she wins at all. Recall that we denote the number of vertices, the number of odd colors, and the largest weight occurring in a parity game with costs by n, d, and W, respectively. In such a game, Player 0 can ensure that almost all requests are answered with cost at most $\mathcal{O}(nW)$ if she wins at all, due to Fijalkow and Zimmermann [FZ14] (cf. ➔Proposition 2.35). We show that a similar argument to that of Fijalkow and Zimmermann yields an upper bound of $\mathcal{O}((ndW)^2)$ in parity games with weights, where W now denotes the largest absolute weight occurring in the game. This increased upper bound on the maximal cost is in large parts due to the increased memory requirements for winning strategies for her. ➔Sec. 2.4, Page 27

This chapter is based on work published at CSL 2018 [SWZ18].

3.1 Definitions

In this section, we first extend the notions introduced in Chapter 2 to account for negative weights. Afterwards, we formally introduce the parity condition with weights and show that games with this condition are determined.

First, we extend the edge-labeling to include negative weights. To this end, let $\mathcal{A}$ be an arena with vertex set V and edge set E. We redefine a WEIGHT FUNCTION of $\mathcal{A}$ to be a function $\mathrm{Weight}\colon E \to \mathbb{Z}$. Analogously to the prior definition of weight functions, we extend the weight function to finite play prefixes via $\mathrm{Weight}(\epsilon) = \mathrm{Weight}(v) = 0$ for empty play prefixes and play prefixes of length one and $\mathrm{Weight}(\pi v v') = \mathrm{Weight}(\pi v') + \mathrm{Weight}(v, v')$ for all longer play prefixes. For the sake of readability, we use the term weight function interchangeably with the term weighting. For the remainder of this section, fix some arena $\mathcal{A} = (V, V_0, V_1, E)$, as well as a coloring $\Omega\colon V \to \mathbb{N}$ and a weighting $\mathrm{Weight}\colon E \to \mathbb{Z}$ of $\mathcal{A}$. **Def.** weight function

We aim to define the parity condition with weights as a generalization of the parity condition with costs and thus, in turn, of the finitary parity condition. To this end, given some play ρ, recall that we interpret each visit to an odd color in ρ as a request for that color, and each subsequent visit to a larger even color as an answer to that request. In parity games with costs, the cost of answering a request is defined as the accumulated weight of the play infix between a request and its earliest answer. This definition relies on the weight function along a play prefix being monotonic. Due to the presence of negative weights in the setting of parity games with weights, however, this assumption does not hold true anymore.

Thus, the first challenge in this setting is to define the cost of answering requests, as the weight may both increase and decrease during the course of the play. We require the weight of the infix to be bounded from above and from below. It does not, however, suffice to consider the accumulated weight of the infix between a request and its response.

Example 3.1. Consider the arena $\mathcal{A}$ shown in Figure 3.1a together with a coloring and a weighting. This game is played in turns, each of which starts in v_{req} and ends in v_{ans}. In each turn, both players choose some number of iterations in their respective vertex. Consider the play in which, in the j-th turn, Player 1 opts to remain in his vertex v_1 for j cycles before yielding control to Player 0 by moving to v_0, where she remains for j cycles as well. We show the evolution of the weight along this play in Figure 3.1b.

This play witnesses that only considering the accumulated weight along an infix between a request and its earliest response is insufficient: Each π_j as indicated in Figure 3.1b has weight zero. Intuitively, however, Player 0 fails to bound the cost of requests along ρ, as Player 1 may remain in his vertex v_1 arbitrarily long and may thus incur arbitrary "costs" (for a generalized notion of costs) for the request for color 1 posed by visiting v_{req}. △

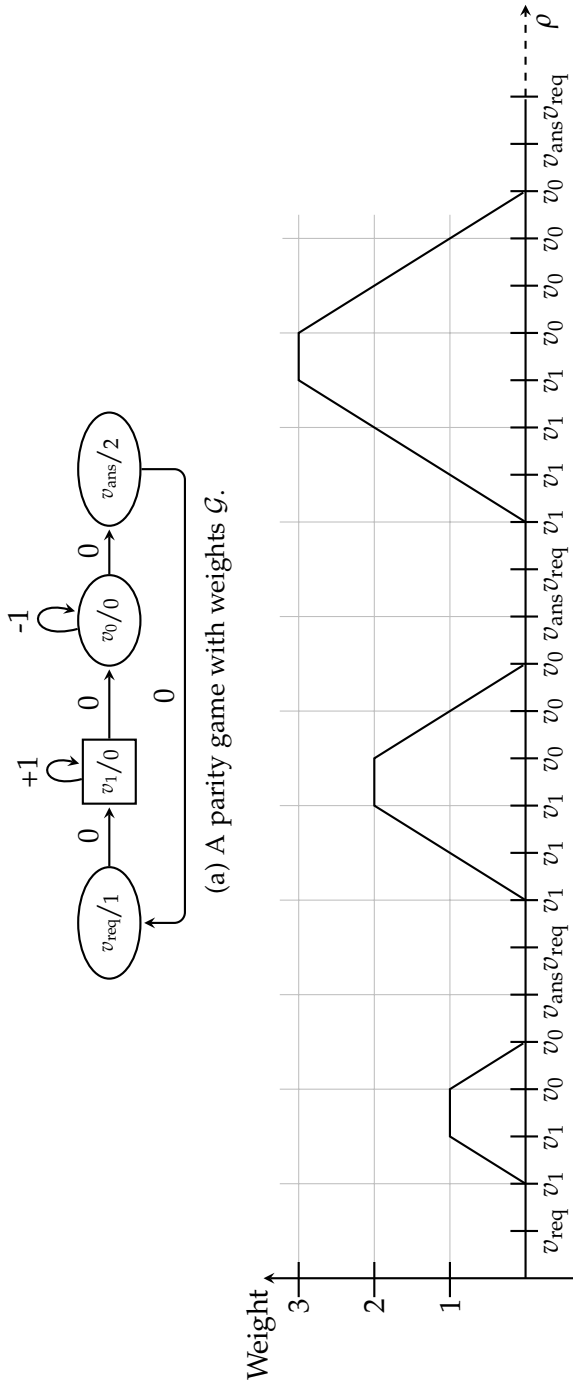
Def. amplitude

As argued in Example 3.1, we aim to define the cost of answering a request as the largest absolute cost that is incurred during the infix between the request and its answer. To this end, given a play (prefix) $\pi = v_0 v_1 v_2 \cdots$, we define the AMPLITUDE of π as

$$\text{Ampl}(\pi) = \sup_{j<|\pi|} \left|\text{Weight}(v_0 \cdots v_j)\right| \ .$$

This definition generalizes both the notion of distance of a play infix used by the finitary parity condition, as well as the notion of the weight of a play infix used by the parity condition with costs: If each edge has weight one, then we obtain $\text{Ampl}(\pi) = \text{Dist}(\pi)$, i.e., the measure of distance employed in the definition of the finitary parity condition. Similarly, if Weight only assigns nonnegative weights, then the amplitude of an infix is equal to its weight. Thus, the amplitude is indeed an intuitive and straightforward extension of the quality measure of the parity condition with costs to the setting of parity games with weights.

We now define the cost of answering a request in the setting of parity games with weights. Intuitively, we demand that Player 0 bounds the amplitude of each infix



(a) A parity game with weights $\mathcal{G}$.

(b) A play in $\mathcal{G}$ and the weight accumulated along it.

Figure 3.1: Motivating example for the definition of $\text{Cor}(\rho, j)$.

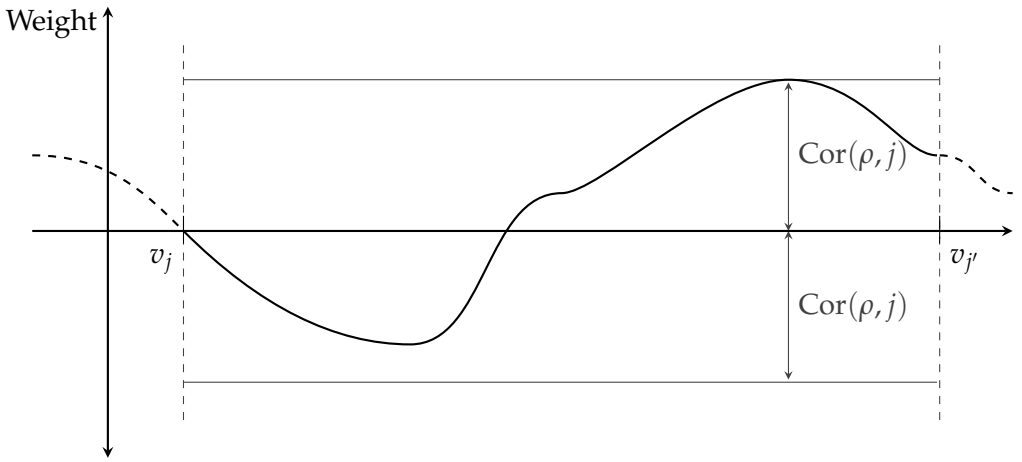


Figure 3.2: The cost-of-response of some request posed by visiting the vertex v_j, which is first answered by visiting the vertex $v_{j'}$.

between a request and any of its answers both from above and from below. Formally, we define the COST-OF-RESPONSE of the request posed at position $j \in \mathbb{N}$ of a play by

Def. cost-of-response

$$\mathrm{Cor}(v_0 v_1 v_2 \cdots, j) = \min\left\{\mathrm{Ampl}(v_j \cdots v_{j'}) \mid j' \geq j, \Omega(v_{j'}) \in \mathrm{Ans}(\Omega(v_j))\right\} \ ,$$

where $\min \emptyset = \infty$. Since, in contrast to the weight of an infix, the amplitude of an infix is monotonic, albeit no longer strictly monotonic, $\mathrm{Cor}(\rho, j)$ is the amplitude of the shortest infix that starts at position j and ends at an answer to the request posed at position j. We illustrate this notion in Figure 3.2.

Def. answered with cost b

We say that a request at position j of a play ρ is ANSWERED WITH COST b if $\mathrm{Cor}(\rho, j) = b$. Consequently, a request with even color is answered with cost zero. The cost-of-response of an unanswered request is infinite, even if the amplitude of the remaining play is finite. In particular, this means that an unanswered request at position j may be "unanswered with finite cost b" (if the amplitude of the remaining play is $b \in \mathbb{N}$) or "unanswered with infinite cost" (if the amplitude of the remaining play is infinite). In either case, however, we have $\mathrm{Cor}(\rho, j) = \infty$.

Example 3.2. Consider the game shown in Figure 3.3. We consider three plays showcasing answered and unanswered requests, the latter with finite and infinite cost, respectively.

First, consider $\rho_0 = v_\mathrm{I} v_\mathrm{II} v'_\mathrm{II} v_\mathrm{II} v'_\mathrm{II} v_\mathrm{II} (v_\mathrm{III})^\omega$. The request for color 1 posed by visiting v_I is answered with cost three, since $\mathrm{Weight}(v_\mathrm{I} v_\mathrm{II} v'_\mathrm{II} v_\mathrm{II} v'_\mathrm{II} v_\mathrm{II}) = -3$ and since all other prefixes of ρ_0 in which that request is open have smaller absolute weight. Now consider $\rho_1 = v_\mathrm{I} v_\mathrm{II} v'_\mathrm{II} v_\mathrm{II} v'_\mathrm{II} (v_\mathrm{II})^\omega$. In ρ_1, the request for color one posed by visiting v_I is unanswered with cost three: Even though the request is unanswered, the amplitude of the play is finite, as it only traverses the self-loop of v_II with weight zero ad infinitum. Finally, let $\rho_2 = v_\mathrm{I} (v_\mathrm{II} v'_\mathrm{II})^\omega$. Since the request for color one posed by visiting v_I is never answered, and since the weight of the prefixes of ρ_2 diverges towards $-\infty$, that request

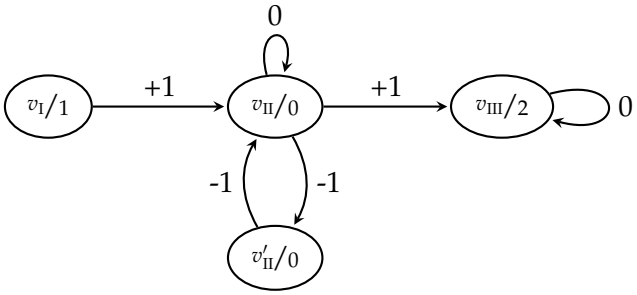


Figure 3.3: A parity game with weights exemplifying the nomenclature regarding answered and unanswered requests.

is unanswered with infinite cost. As the request posed at the initial position is unanswered in both ρ_1 and in ρ_2, we, in contrast, have $\mathrm{Cor}(\rho_1, 0) = \mathrm{Cor}(\rho_2, 0) = \infty$. △

We define the cost of a play ρ as $\mathrm{Cost}(\rho) = \limsup_{j\to\infty} \mathrm{Cor}(\rho, j)$ and we define the parity condition with weights as

Def. cost of a play

Def. parity condition with weights

$$\textsc{WeightParity}(\Omega, \mathrm{Weight}) = \{\rho \in V^\omega \mid \mathrm{Cost}(\rho) < \infty\} \quad .$$

Hence, a play ρ is winning for Player 0 according to parity condition with weights if and only if there exists a bound $b \in \mathbb{N}$ such that almost all requests in ρ are answered with cost at most b. In particular, only finitely many requests may be unanswered, even with finite cost.

We call a game $\mathcal{G} = (\mathcal{A}, \textsc{WeightParity}(\Omega, \mathrm{Weight}))$ a parity game with weights, where we again assume the function Weight to be given in binary encoding. Hence, we obtain $|\mathcal{G}| = |\mathcal{A}| \log(W)$, where W is the largest absolute weight assigned by Weight.

Def. parity game with weights

Moreover, we extend the notion of cost to strategies similarly to the special case of parity games with costs: Let σ be a strategy for Player 0 and let v be a vertex of $\mathcal{G}$. We define

$$\mathrm{Cost}_v(\sigma) = \sup_\rho(\mathrm{Cost}(\rho)) \ ,$$

where ρ ranges over all plays starting in v and consistent with σ. Thus, we directly obtain that the strategy σ is winning for Player 0 from some vertex v if we have $\mathrm{Cost}_v(\sigma) < \infty$.

Dually, we aim to define a measure of cost for strategies of Player 1 such that a strategy is winning for him from a vertex if and only if the cost of the strategy with respect to that vertex is infinite. Recall that, in order to win, Player 1 has to enforce infinite cost of the resulting play. Following this intuition, for a strategy τ of Player 1 we define

$$\mathrm{Cost}_v(\tau) = \min_\rho(\mathrm{Cost}(\rho)) \ ,$$

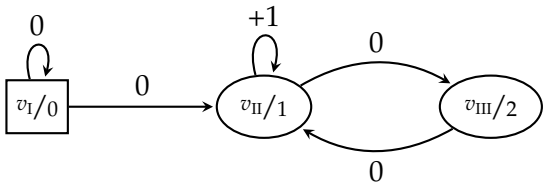


Figure 3.4: A parity game with weights

where ρ again ranges over all plays starting in v and consistent with τ. Note that we use the minimum over the cost of all consistent plays instead of the infimum. We are able to do so due to the cost of plays being trivially bounded from below by zero.

Our definition of $\mathrm{Cost}_v(\sigma)$ yields a uniform bound on the cost of the plays consistent with σ. If such a bound exists, then σ clearly is winning for Player 0 from v. This implication does, however, not hold true in the other direction.

Example 3.3. Consider the parity game with weights $\mathcal{G}$ shown in Figure 3.4. We define a strategy σ for Player 0 that is winning for her from v_{I}, but for which we have $\mathrm{Cost}_{v_{\mathrm{I}}}(\sigma) = \infty$.

To this end, let π be a play prefix in $\mathcal{G}$ that starts in v_{I} and ends in a vertex of Player 0. Then π begins with a finite prefix of the form $(v_{\mathrm{I}})^j$, after which v_{I} is never visited again. We define $\sigma(\pi)$ to prescribe to remain in v_{II} for j steps upon each visit to v_{II} before subsequently moving to v_{III}.

Let ρ be a play starting in v_{I} consistent with σ. Then ρ is either of the form $(v_{\mathrm{I}})^\omega$ or of the form $(v_{\mathrm{I}})^j((v_{\mathrm{II}})^{j+1}v_{\mathrm{III}})^\omega$. In the former case, ρ does not contain any request for an odd color and hence clearly satisfies the parity condition with weights. In the latter case, we have $\mathrm{Cost}(\rho) = j < \infty$, i.e., ρ again satisfies the parity condition with weights. Thus, σ is indeed winning for Player 0 from v_{I}.

For each $j \in \mathbb{N}$, however, the play $\rho_j = (v_{\mathrm{I}})^j((v_{\mathrm{II}})^{j+1}v_{\mathrm{III}})^\omega$ is consistent with σ and has $\mathrm{Cost}(\rho_j) = j$. Hence, we have $\mathrm{Cost}_{v_{\mathrm{I}}}(\sigma) = \infty$. △

We will later see that while the converse direction does not hold true a priori, it does indeed hold true for finite-state strategies, due to a pumping argument.

The remainder of this chapter is dedicated to analyzing the properties of parity games with weights, i.e., the complexity of solving them, the memory required by both players to implement winning strategies, and the maximal cost of a play that Player 0 is forced to allow if she wins.

We begin our analysis of parity games with weights by showing that they are determined. To this end, we show that the condition is Borel, which suffices due to
→ Sec. 2.3, Page 17
→ Proposition 2.8.

Lemma 3.4. *Parity games with weights are determined.*

Proof. Recall that we argued earlier that a winning condition is Borel if it is a Boolean combination of open sets, i.e., constructed from open sets via union, intersection, and

complementation. In order to show that the parity condition with weights is indeed determined, we first rewrite it as follows:

$$\begin{aligned}\textsc{WeightParity}(\Omega, \mathrm{Weight}) &= \left\{\rho \in V^\omega \mid \limsup_{j\to\infty} \mathrm{Cor}(\rho, j) < \infty\right\} \\ &= \bigcup_{b\in\mathbb{N}} \left\{\rho \in V^\omega \mid \limsup_{j\to\infty} \mathrm{Cor}(\rho, j) \leq b\right\} \\ &= \bigcup_{b\in\mathbb{N}} \bigcup_{j\in\mathbb{N}} \left\{\rho \in V^\omega \mid \sup_{j'\geq j} \mathrm{Cor}(\rho, j') \leq b\right\} \\ &= \bigcup_{b\in\mathbb{N}} \bigcup_{j\in\mathbb{N}} \bigcap_{j'\geq j} \left\{\rho \in V^\omega \mid \mathrm{Cor}(\rho, j') \leq b\right\} \quad .\end{aligned}$$

Since we furthermore have

$$\left\{\rho \in V^\omega \mid \mathrm{Cor}(\rho, j') \leq b\right\} = V^\omega \setminus \left\{\rho \in V^\omega \mid \mathrm{Cor}(\rho, j') > b\right\} \quad ,$$

and since, for fixed $j' \in \mathbb{N}$ and $b \in \mathbb{N}$, the set $\{\rho \in V^\omega \mid \mathrm{Cor}(\rho, j') > b\}$ is clearly open, the parity condition with weights is Borel. Hence, parity games with weights are indeed determined due to Proposition 2.8. □

The parity condition with weights subsumes several winning conditions considered previously.

Remark 3.5. *For each coloring Ω we have*

1. $\textsc{Parity}(\Omega) = \textsc{WeightParity}(\Omega, \mathit{Weight})$, *where Weight assigns weight zero to all edges,*
2. $\textsc{FinParity}(\Omega) = \textsc{WeightParity}(\Omega, \mathit{Weight})$, *where Weight assigns positive weight to all edges, and*
3. *if Weight assigns nonnegative cost to all edges, then we have* $\textsc{CostParity}(\Omega, \mathit{Weight}) = \textsc{WeightParity}(\Omega, \mathit{Weight})$.

Recall that the problem of solving parity games with costs is in UP ∩ coUP due to → Proposition 2.33. In the following section we show that the problem of solving parity games with weights is in NP ∩ coNP and that it can be solved in pseudo-quasi-polynomial time, i.e., the complexity of solving parity games with weights increases only slightly over the special case of parity games with costs.

→ Sec. 2.4, Page 26

3.2 Computational Complexity

We now show how to solve parity games with weights and prove the problem of solving them to be in NP ∩ coNP. Our approach is inspired by the PTime-membership proof for the problem of solving finitary parity games [CHH09] and by the NP ∩ coNP-membership proof for the problem of solving parity games with costs [FZ14].

First, in Section 3.2.1, we define a stricter variant of the parity condition with weights, which we call bounded parity games with weights, and show how to solve parity games with weights in polynomial time using an oracle solving bounded parity games with weights.

In a second step we repeat this approach in order to solve bounded parity games with weights. For bounded parity games with costs and finitary parity games, the respective authors could solve the bounded variants of the games by iteratively solving classical parity games. These approaches leveraged the monotonicity of the cost measures in both kinds of games, which allowed to only consider "abstract" weights, due to ➔Remark 2.29. Thus, the target of the reduction, i.e., classical parity games, was not required to support quantitative features for these reductions.

➔Sec. 2.4, Page 26

For parity games with weights, in contrast, the weights along a play infix need to be tracked precisely in order to determine whether or not a request has finite cost, as argued previously. Thus, we require quantitative games as the target of the reduction, i.e., games that allow such tracking.

Hence, we reduce the problem of solving bounded parity games with weights to that of iteratively solving energy parity games as introduced by Chatterjee and Doyen [CD12]. We formally define energy parity games and their properties required for our construction in Section 3.2.2 before subsequently showing how to solve bounded parity games with weights in polynomial time using oracles that solve energy parity games in Section 3.2.3. As the problem of solving energy parity games is known to be in NP ∩ coNP due to Chatterjee and Doyen [CD12], this construction yields membership of the problem of solving parity games with weights in the same complexity class.

Thus, in this section, we give an algorithm for solving parity games with weights that uses an oracle solving energy parity games. Later, in Section 3.3, we show a dual result to the one presented in this section: Given an oracle solving parity games with weights, we are able to solve energy parity games in polynomial time, again taking a detour via bounded parity games with weights. Hence, the problems of solving parity games with weights, that of solving their bounded variant, as well as the problem of solving energy parity games are polynomial-time equivalent and thus, all belong to the same complexity class.

3.2.1 Bounded Parity Games with Weights

Def. bounded parity condition with weights

We first introduce the BOUNDED PARITY CONDITION WITH WEIGHTS, which strengthens the parity condition with weights by additionally demanding that the amplitude of each suffix succeeding an unanswered request is finite. Following this intuition, this condition is also induced by a coloring Ω and a weighting Weight:

$$\textsc{BndWeightParity}(\Omega, \text{Weight}) = \textsc{WeightParity}(\Omega, \text{Weight}) \cap \{\rho \in V^\omega \mid \text{no request in } \rho \text{ is unanswered with infinite cost}\} \quad .$$

Def. bounded parity game with weights

We call a game $\mathcal{G} = (\mathcal{A}, \text{Win})$ a BOUNDED PARITY GAME WITH WEIGHTS if Win =

BndWeightParity(Ω, Weight) for some coloring Ω and some weighting Weight.

This condition allows for a finite number of unanswered requests, as long as they are unanswered with finite cost. Moreover, it is 1-extendable, but not 0-extendable: Prepending a play prefix may introduce a request that is unanswered with infinite cost, but it cannot bound the cost of a later request or answer that request. Finally, by showing that the bounded parity condition with weights is Borel, we obtain that games with this condition are determined.

Lemma 3.6. *Bounded parity games with weights are determined.*

Proof. Similarly to the proof of Lemma 3.4 we again leverage ➔Proposition 2.8 to obtain the desired result. ➔Sec. 2.3, Page 17

Recall that the parity condition with weights is Borel as argued in the proof of Lemma 3.4. We show that the set

$$\text{Bnd} = \{\rho \in V^\omega \mid \text{ no request in } \rho \text{ is unanswered with infinite cost}\}$$

is Borel as well. This then implies that the bounded parity condition with weights is Borel, which yields the desired result. To this end, we reformulate the set Bnd via

$$\begin{aligned}&\{v_0v_1v_2\cdots \in V^\omega \mid \text{no request in } \rho \text{ is unanswered with infinite cost}\} = \\ &\quad \{v_0v_1v_2\cdots \in V^\omega \mid \forall j \in \mathbb{N}.\, v_j \text{ is answered or } \text{Ampl}(v_jv_{j+1}v_{j+2}\cdots) \in \mathbb{N}\} \ ,\end{aligned}$$

where we write "v_j is answered" to denote that there exists some $j' \geq j$ such that $\Omega(v_{j'}) \in \text{Ans}(\Omega(v_j))$.

We now argue that the equality

$$\begin{aligned}&\{v_0v_1v_2\cdots \in V^\omega \mid \forall j \in \mathbb{N}.\, v_j \text{ is answered or } \text{Ampl}(v_jv_{j+1}v_{j+2}\cdots) \in \mathbb{N}\} = \\ &\quad \bigcup_{b\in\mathbb{N}} \{v_0v_1v_2\cdots \in V^\omega \mid \forall j \in \mathbb{N}.\, v_j \text{ is answered or } \text{Ampl}(v_jv_{j+1}v_{j+2}\cdots) \leq b\}\end{aligned}$$

holds true as well.

It is clear that the latter set is a subset of the former one. Thus, let ρ be an element of the former set. If all requests in ρ are answered, then ρ is clearly in the latter set as well. If, however, there exists an unanswered request in ρ let j be the position of the earliest such request. We then have $\text{Ampl}(v_jv_{j+1}v_{j+2}\cdots) = b < \infty$. Thus, for each position j' of an unanswered request in ρ, we obtain that $\text{Ampl}(v_{j'}v_{j'+1}v_{j'+2}\cdots)$ is bounded by $2b$. Hence ρ is a member of the latter set as well.

A straightforward rewriting of the latter set yields

$$\begin{aligned}&\bigcup_{b\in\mathbb{N}} \{v_0v_1v_2\cdots \in V^\omega \mid \forall j \in \mathbb{N}.\, v_j \text{ is answered or } \text{Ampl}(v_jv_{j+1}v_{j+2}\cdots) \leq b\} = \\ &\quad \bigcup_{b\in\mathbb{N}} \bigcap_{j\in\mathbb{N}} \{v_0v_1v_2\cdots \in V^\omega \mid v_j \text{ is answered or } \text{Ampl}(v_jv_{j+1}v_{j+2}\cdots) \leq b\} \ ,\end{aligned}$$

Clearly, the set $\{v_0v_1v_2\cdots \in V^\omega \mid v_j \text{ is answered}\}$ is open. Moreover, we have

$$\{v_0v_1v_2\cdots \in V^\omega \mid \mathrm{Ampl}(v_jv_{j+1}v_{j+2}\cdots) \leq b\} = V^\omega \setminus \{v_0v_1v_2\cdots \in V^\omega \mid \exists j' \geq j.\, \mathrm{Ampl}(v_j\cdots v_{j'}) > b\} \ ,$$

where $\{v_0v_1v_2\cdots \in V^\omega \mid \exists j' \geq j.\, \mathrm{Ampl}(v_j\cdots v_{j'}) > b\}$ again clearly is an open set.

Thus, the set Bnd is a Boolean combination of open and closed sets, i.e., it is Borel. This implies the desired result due to Proposition 2.8. □

We now show how to solve parity games with weights by repeatedly solving their bounded variant. To this end, we apply the observation that the bounded parity condition with weights strengthens its unbounded variant, i.e., that

$$\textsc{BndWeightParity}(\Omega, \mathrm{Weight}) \subseteq \textsc{WeightParity}(\Omega, \mathrm{Weight})$$

holds true. Moreover, we leverage that $\textsc{WeightParity}(\Omega, \mathrm{Weight})$ is 0-extendable.

Hence, if Player 0 has a strategy from a vertex v such that every consistent play has a suffix that satisfies the bounded parity condition with weights, then each play consistent with that strategy also satisfies the parity condition with weights.

Remark 3.7. *Let $\mathcal{A}$ be an arena and let Ω and Weight be a coloring and a weighting of $\mathcal{A}$, respectively. We then have*

$$\mathit{Attr}_0(W_0(\mathcal{A}, \textsc{BndWeightParity}(\Omega, \mathit{Weight}))) \subseteq W_0(\mathcal{A}, \textsc{WeightParity}(\Omega, \mathit{Weight})) \ .$$

In order to solve a parity game with weights, we repeatedly remove attractors of winning regions of the bounded parity game with weights on the same arena with identical coloring and weighting until the games thus obtained stabilize. Remark 3.7 then yields that the removed parts are subsets of Player 0's winning region in the parity game with weights.

It then remains to show that this procedure indeed computes the complete winning region of Player 0. To this end, we use the following lemma to show that the remaining vertices are a subset of Player 1's winning region in the parity game with weights. The proof is very similar to the corresponding one for finitary parity games and parity games with costs [CHH09, FZ14].

Lemma 3.8. *Given an arena $\mathcal{A}$ as well as a coloring Ω and a weighting Weight of $\mathcal{A}$, let $\mathcal{G} = (\mathcal{A}, \textsc{WeightParity}(\Omega, \mathit{Weight}))$ and let $\mathcal{G}' = (\mathcal{A}, \textsc{BndWeightParity}(\Omega, \mathit{Weight}))$. We have that $W_0(\mathcal{G}') = \emptyset$ implies $W_0(\mathcal{G}) = \emptyset$.*

Proof. Recall that Lemma 3.6 stated that bounded parity games with weights are determined. Hence, $W_0(\mathcal{G}') = \emptyset$ implies that, for every vertex v of $\mathcal{A}$, Player 1 has a winning strategy τ_v from v in $\mathcal{G}'$.

We combine these strategies into a single strategy τ for Player 1 that is winning in $\mathcal{G}$ from every vertex of $\mathcal{A}$. This strategy is controlled by a vertex v^* (initialized with

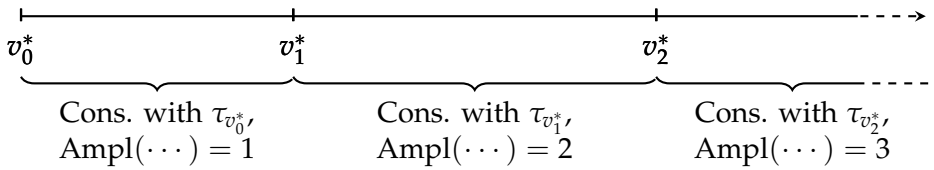


Figure 3.5: Construction of the strategy τ for Player 1 in the proof of Lemma 3.8.

the initial vertex of the play) and a counter κ ranging over $\mathbb{N}$ (initialized with zero). The strategy τ mimics the strategy τ_{v^*} from v^* until a request is followed by an infix that does not answer that request and has amplitude exceeding κ. This implies that the cost-of-response of this request is at least κ. If such a situation is encountered, then v^* is set to the current vertex of the play and κ is incremented. Furthermore, the history of the play is discarded at this point in the play, and τ henceforth behaves like τ_{v^*} when starting at v^*. We illustrate this construction in Figure 3.5.

Let ρ be a play consistent with this strategy τ. If κ is updated infinitely often along ρ, then ρ contains, for every $b \in \mathbb{N}$, a request that is answered or unanswered with cost greater than b. Hence, ρ violates the parity condition with weights.

If, however, κ is updated only finitely often along ρ, then ρ has a suffix ρ' that starts in some v and that is consistent with τ_v. As τ_v is winning for Player 1 from v in $\mathcal{G}'$, the suffix ρ' violates the bounded parity condition with weights. Also, as κ is updated only finitely often during ρ', the amplitude of every suffix of ρ' that starts at a request is bounded by κ. Hence, the only way for ρ' to violate the bounded parity condition with weights is to violate the parity condition. Since the parity condition is prefix-independent, the full play ρ also violates the parity condition. This in turn implies that ρ also violates the parity condition with weights, which strengthens the parity condition.

Therefore, τ is indeed winning for Player 1 from every vertex in $\mathcal{G}$, i.e., the winning region of Player 1 in $\mathcal{G}$ encompasses all vertices of $\mathcal{A}$. As the winning regions of the players are disjoint, this implies the desired result. □

We formalize our sketched algorithm for solving parity games with weights as Algorithm 3.1. In that algorithm, we write $\mathcal{A}_{k-1} \setminus \mathrm{Attr}_0^{\mathcal{A}_{k-1}}(X_k)$ to denote the arena resulting from removing the 0-attractor of X_k (computed in the arena $\mathcal{A}_{k-1}$) and all edges that are adjacent to vertices in that attractor from $\mathcal{A}_{k-1}$. Recall that, due to the definition of attractors, each vertex in $\mathcal{A}_{k-1}$ is either in the computed attractor, or it has at least one outgoing edge leading to a vertex that is not in the attractor. Hence, the resulting graph does not contain dead-end vertices, i.e., it is indeed an arena.

As part of Algorithm 3.1 we use an oracle for solving bounded parity games with weights. We provide a suitable algorithm to implement such an oracle in Section 3.2.3. Moreover, we illustrate the first two steps of the algorithm in Figure 3.6a.

Remark 3.7 together with Lemma 3.8 imply the correctness of Algorithm 3.1 for solving parity games with weights using an oracle that solves their bounded variant. The correctness proof is again similar to the corresponding one for finitary parity

Algorithm 3.1 A fixed-point algorithm computing $W_0(\mathcal{A}, \textsc{WeightParity}(\Omega, \text{Weight}))$.

Input: $\mathcal{G} = (\mathcal{A}, \textsc{WeightParity}(\Omega, \text{Weight}))$
  $k = 0$; $W_0^k = \emptyset$; $\mathcal{A}_k = \mathcal{A}$
  repeat
    $k = k + 1$
    $X_k = W_0(\mathcal{A}_{k-1}, \textsc{BndWeightParity}(\Omega, \text{Weight}))$
                /* Requires solving bounded parity game with weights */
    $W_0^k = W_0^{k-1} \cup \text{Attr}_0^{\mathcal{A}_{k-1}}(X_k)$
    $\mathcal{A}_k = \mathcal{A}_{k-1} \setminus \text{Attr}_0^{\mathcal{A}_{k-1}}(X_k)$
  until $X_k = \emptyset$
  return W_0^k
Output: $W_0(\mathcal{G})$

games [CHH09] and for parity games with costs [FZ14].

Lemma 3.9. *Algorithm 3.1 returns* $W_0(\mathcal{A}, \textsc{WeightParity}(\Omega, \mathit{Weight}))$.

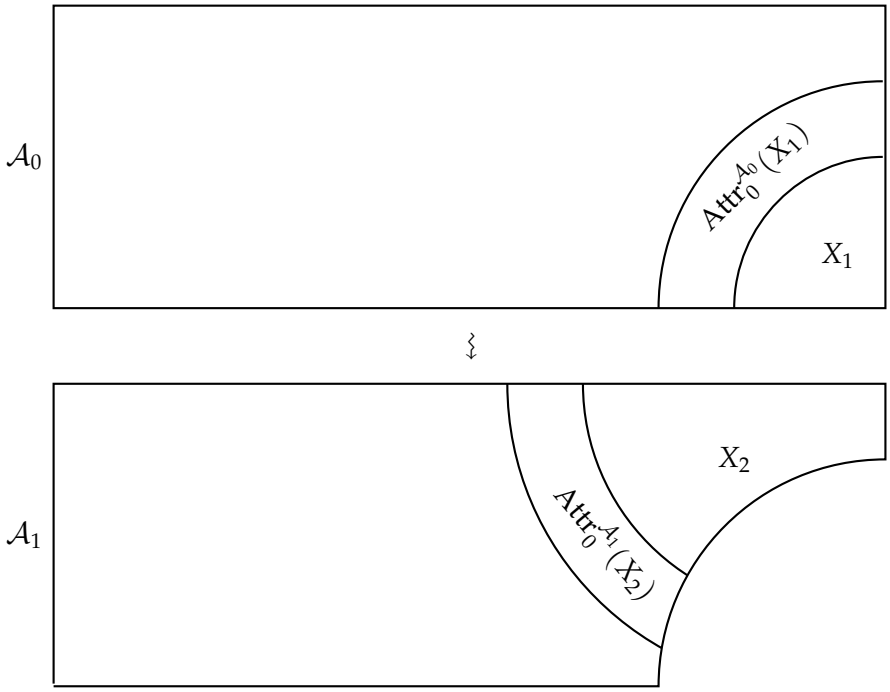
Proof. Let $\mathcal{G} = (\mathcal{A}, \textsc{WeightParity}(\Omega, \text{Weight}))$ and let k^* be the value of k during the final iteration of the main loop of the algorithm when running it on $\mathcal{G}$, i.e., its output is $W_0^{k^*} = \bigcup_{0<k<k^*} \text{Attr}_0^{\mathcal{A}_{k-1}}(X_k)$. During the execution of the algorithm, the arena is segmented into the sets X_k and their attractors, since $\text{Attr}_0^{\mathcal{A}_{k-1}}(X_k)$ is removed from $\mathcal{A}_{k-1}$ to obtain $\mathcal{A}_k$. All X_k are pairwise disjoint. This also holds true for the $\text{Attr}_0^{\mathcal{A}_k}(X_k)$. We illustrate this situation in Figure 3.6b. We now proceed to show $W_0^{k^*} = W_0(\mathcal{G})$ by showing the two set inclusions separately.

We first show $W_0^{k^*} \subseteq W_0(\mathcal{G})$. Recall $W_0^{k^*} = \bigcup_{0<k<k^*} \text{Attr}_0^{\mathcal{A}_{k-1}}(X_k)$, i.e., every vertex $v \in W_0^{k^*}$ is either a member of X_k or of $\text{Attr}_0^{\mathcal{A}_{k-1}}(X_k) \setminus X_k$ for some unique k. For every vertex v that is in some X_k, Player 0 has a strategy σ_v in the bounded parity game with weights $\mathcal{G}_k = (\mathcal{A}_{k-1}, \textsc{BndWeightParity}(\Omega, \text{Cost}))$ that is winning from v. Furthermore, for every k, she has a positional attractor strategy σ_k towards X_k from $\text{Attr}_0^{\mathcal{A}_{k-1}}(X_k)$ in $\mathcal{A}_k$. As both strategies are, however, computed in $\mathcal{A}_k$ which differs, in general, from $\mathcal{A}$, Player 1 may have the opportunity to perform "unexpected" moves by choosing to move to some vertex in $\mathcal{A}_{k'}$ where $k' \neq k$. We later see that such moves may only lead "down" the hierarchy of arenas, i.e., that we always have $k' < k$ in such a case.

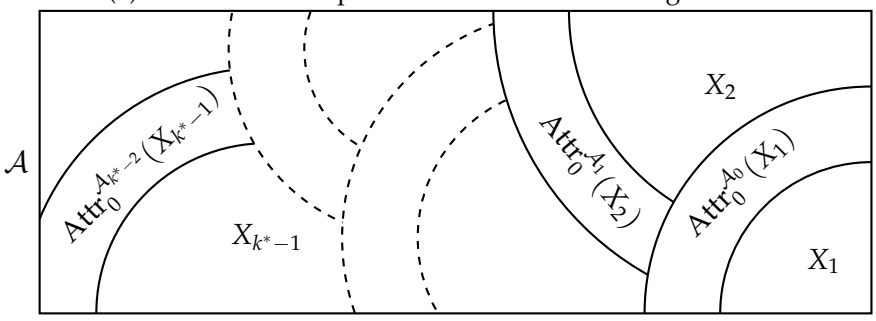
We show $W_0^{k^*} \subseteq W_0(\mathcal{G})$ by constructing a strategy σ for Player 0 in $\mathcal{A}$ that is winning for her from $W_0^{k^*}$. To this end, we define σ by compositing the strategies σ_v and σ_k as follows:

$$\sigma(v_0 \cdots v_j) = \begin{cases} \sigma_k(v_j) & \text{if } v_j \in \text{Attr}_0^{\mathcal{A}_{k-1}}(X_k) \setminus X_k \ , \\ \sigma_{v_{j'}}(v_{j'} \cdots v_j) & \text{if } v_j \in X_k, \text{ and} \\ & \quad \text{with } j' = \min\left\{ j' \mid \{v_{j'}, \ldots, v_j\} \subseteq X_k \right\} \ . \end{cases}$$

For the sake of readability, we omit the definition of σ for the irrelevant play prefixes not matching either of the cases above. Since for each $v \in W_0^{k^*}$ there exists a unique k



(a) The first two steps of the execution of the algorithm.



(b) After the termination of the algorithm.

Figure 3.6: Illustration of Algorithm 3.1.

such that either $v \in \mathrm{Attr}_0^{\mathcal{A}_{k-1}}(X_k) \setminus X_k$ or such that $v \in X_k$, the strategy σ is well-defined. In the first case of the definition, it suffices to consider the move prescribed by σ_k when starting at v_j, as the attractor strategies σ_k are positional by assumption. In the second case of the definition, $v_{j'} \cdots v_j$ is the longest suffix of $v_0 \cdots v_j$ that only contains vertices from $X_k = W_0(\mathcal{G}_k)$, i.e., from the set of vertices from which Player 0 has a winning strategy for $\mathcal{G}_k$. It remains to show that σ is winning for Player 0 from $W_0^{k^*}$.

To this end, consider a play $\rho = v_0 v_1 v_2 \cdots$ in $\mathcal{A}$ that starts in $W_0^{k^*}$ and is consistent with σ. For every j there is a unique k_j in the range $0 < k_j < k^*$ such that $v_j \in \mathrm{Attr}_0^{\mathcal{A}_{k_j-1}}(X_{k_j})$. As BndWeightParity$(\Omega, \mathrm{Weight})$ is 1-extendable, we obtain that each

$\mathrm{Attr}_0^{\mathcal{A}_{k-1}}(X_k)$ is a trap for Player 1 in $\mathcal{A}_{k-1}$: Otherwise, starting in $\mathrm{Attr}_0^{\mathcal{A}_{k-1}}(X_k)$, Player 1 could move to some vertex $v \notin \mathrm{Attr}_0^{\mathcal{A}_{k-1}}(X_k)$ and subsequently play consistently with a winning strategy for him from v. The resulting play would be winning for him due to 1-extendability of the bounded parity condition with weights and thus contradict the definition of $\mathrm{Attr}_0^{\mathcal{A}_{k-1}}(X_k)$.

Hence, we obtain $k_0 \geq k_1 \geq k_2 \geq \cdots$. As the k_j are bounded from below by zero, the sequence eventually stabilizes, say at position j'. This implies that ρ has a suffix $\rho' = v_{j'}v_{j'+1}v_{j'+2}\cdots$ that is consistent with $\sigma_{v_{j'}}$.

Hence, due to $\sigma_{v_{j'}}$ being a winning strategy for Player 0 in $\mathcal{G}_k$ from $v_{j'}$, we obtain $\rho' \in \textsc{BndWeightParity}(\Omega, \mathrm{Weight})$. Hence, $\rho \in \textsc{WeightParity}(\Omega, \mathrm{Weight})$ due to the bounded parity condition with weights strengthening of the parity condition with weights and due to 0-extendability of $\textsc{WeightParity}(\Omega, \mathrm{Weight})$. Hence, σ is indeed winning from $W_0^{k^*}$.

It remains to show $W_0(\mathcal{G}) \subseteq W_0^{k^*}$. To this end, we consider Player 1 and show $V \setminus W_0^{k^*} \subseteq W_1(\mathcal{G})$. Then, disjointness of winning regions yields $W_0^{k^*} = W_0(\mathcal{G})$ and, consequently, $V \setminus W_0^{k^*} = W_1(\mathcal{G})$.

Since $X_{k^*} = W_0(\mathcal{G}_{k^*-1})$ is empty and since bounded parity games with weights are determined due to Lemma 3.6, Player 1 wins the bounded parity game with weights $\mathcal{G}_{k^*}$ from every vertex. Hence, Lemma 3.8 implies that he also wins the parity game with weights $(\mathcal{A}_{k^*-1}, \textsc{WeightParity}(\Omega, \mathrm{Weight}))$ from every vertex in $\mathcal{A}_{k^*-1}$. Finally, as $V \setminus W_0^{k^*} = W_1(\mathcal{A}_{k^*-1}, \textsc{WeightParity}(\Omega, \mathrm{Weight}))$ is a trap for Player 0 in $\mathcal{A}$ by construction, Player 1 also wins $\mathcal{G} = (\mathcal{A}, \textsc{WeightParity}(\Omega, \mathrm{Weight}))$ from every vertex in $V \setminus W_0^{k^*}$. □

The loop of Algorithm 3.1 terminates after at most $|\mathcal{A}|$ iterations, as during each iteration at least one vertex is removed from $\mathcal{A}_{k-1}$ to obtain $\mathcal{A}_k$. Moreover, the runtime of each iteration is dominated by solving a bounded parity game with weights. Hence Algorithm 3.1 indeed only induces a polynomial overhead on the runtime of the underlying solver for bounded parity games with weights and requires at most polynomially many calls to it.

Finally, we consider the memory size required to implement the winning strategy σ constructed in the proof of Lemma 3.9. Since for each vertex $v \in W_0(\mathcal{G})$ there exists a unique k with $v \in \mathrm{Attr}_0^{\mathcal{A}_{k-1}}(X_k)$, and since the vertices in X_k are never revisited once left, the strategy σ_k to use is uniquely determined by the current vertex. Hence, the winning strategy σ can be implemented by "reusing" the memory states of its constituent strategies.

Corollary 3.10. *Let $\mathcal{G} = (\mathcal{A}, \textsc{WeightParity}(\Omega, \mathit{Weight}))$ be a parity game with weights, let k^* be the number of iterations of the main loop of Algorithm 3.1 when running on the input $\mathcal{G}$. Furthermore, for each k with $1 \leq k \leq k^*$, let σ_k be a winning strategy for Player 0 from her winning region in $(\mathcal{A}_{k-1}, \textsc{BndWeightParity}(\Omega, \mathit{Weight}))$.*

If the σ_k are finite-state strategies of size s_k, then σ is of size at most $\max_{k \leq k^*} s_k$.

3.2.2 Energy Parity Games

Our next goal is to solve bounded parity games with weights, thus providing the oracle required by ➔ Algorithm 3.1. To this end we employ a similar approach to that for solving parity games with weights, i.e., we reduce the problem to that of iteratively solving a number of conceptually simpler games, namely energy parity games in this case [CD12].

➔ Sec. 3.2, Page 42

An ENERGY PARITY GAME $\mathcal{G} = (\mathcal{A}, \Omega, \text{Weight})$ consists of an arena $\mathcal{A}$ with vertex set V and edge set E, a coloring $\Omega\colon V \to \mathbb{N}$ of V, and a weighting $\text{Weight}\colon E \to \mathbb{Z}$ of $\mathcal{A}$. This definition is not compatible with the framework presented in ➔ Section 2, as we have not (yet) defined the winner of the plays. This is because the winner depends on an initial credit, which is existentially quantified in the definition of winning the game $\mathcal{G}$. Formally, the set of winning plays with INITIAL CREDIT $c_I \in \mathbb{N}$ is defined as

Def. energy parity game

➔ Page 11

Def. initial credit

$$\textsc{EnergyParity}_{c_I}(\Omega, \text{Weight}) = \textsc{Parity}(\Omega) \cap \left\{ v_0 v_1 v_2 \cdots \in V^\omega \mid \forall j \in \mathbb{N}.\, c_I + \text{Weight}(v_0 \cdots v_j) \geq 0 \right\} .$$

Now, we say that Player 0 WINS the energy parity game $\mathcal{G} = (\mathcal{A}, \Omega, \text{Weight})$ from some vertex $v \in V$ if there exists some initial credit $c_I \in \mathbb{N}$ such that she wins $\mathcal{G}_{c_I} = (\mathcal{A}, \textsc{EnergyParity}_{c_I}(\Omega, \text{Weight}))$ from v (in the sense of the definitions in Section 2). If this is not the case, i.e., if Player 1 wins every $\mathcal{G}_{c_I}$ from v, then we say that Player 1 wins $\mathcal{G}$ from v. In energy parity games, the initial credit is uniform for all plays. This contrasts the bound on the cost-of-response in the parity condition with weights, which may depend on the play.

Def. winning an energy parity game

Unravelling these definitions shows that Player 0 wins $\mathcal{G}$ from v if there is an initial credit c_I and a strategy σ, such that every play that starts in v and is consistent with σ satisfies the parity condition *and* the accumulated weight over the play prefixes (the energy level) never drops below $-c_I$. We call such a strategy σ a WINNING STRATEGY for Player 0 in $\mathcal{G}$ from v. Dually, Player 1 wins $\mathcal{G}$ from v if, for every initial credit c_I, there is a strategy τ_{c_I}, such that every play that starts in v and is consistent with τ_{c_I} violates the parity condition *or* its energy level drops below $-c_I$ at least once. Hence, as the notation suggests, Player 1 may be required to use different strategies τ_{c_I} to win depending on the initial credit c_I. However, Chatterjee and Doyen [CD12] showed that this is, in fact, not necessary for him: There is a uniform strategy τ for Player 1 that is winning for him from v for every initial credit c_I.

Def. winning strategy for Player 0

Proposition 3.11 ([CD12])**.** *Let $\mathcal{G}$ be an energy parity game. If Player 1 wins $\mathcal{G}$ from v, then he has a positional strategy that is winning from v in $\mathcal{G}_{c_I}$ for every c_I.*

We call such a strategy τ as in Proposition 3.11 a WINNING STRATEGY for Player 1 from v. A play beginning in v that is consistent with τ either violates the parity condition, or the energy levels of its prefixes diverge towards $-\infty$, i.e., Player 1 is able to unbound the energy from below in the latter case.

Def. winning strategy for Player 1

Furthermore, Chatterjee and Doyen showed an analogous result characterizing the relevant space of strategies for Player 0: They obtained an upper bound on the initial

credit necessary for Player 0 to win an energy parity game, as well as an upper bound on the size of a corresponding finite-state winning strategy.

Proposition 3.12 ([CD12])**.** *Let $\mathcal{G}$ be an energy parity game with n vertices, d colors, and largest absolute weight W. Moreover, let v be a vertex of $\mathcal{G}$.*

If Player 0 wins $\mathcal{G}$ from v, then she wins $\mathcal{G}_{(n-1)W}$ from v with a finite-state strategy with at most ndW states.

This proposition yields that finite-state strategies of bounded size suffice for Player 0 to win, i.e., that they suffice for her to bound the energy level from below. A straightforward pumping argument yields the additional property that such strategies do not admit long expensive descents. This property later on allows us to reason about the structure of plays consistent with winning strategies for Player 0.

Lemma 3.13. *Let $\mathcal{G} = (\mathcal{A}, \Omega, \mathit{Weight})$ be an energy parity game with n vertices and largest absolute weight W. Further, let σ be a finite-state strategy of size s, and let ρ be a play that starts in some vertex from which σ is winning, and that is consistent with σ.*

Every infix π of ρ satisfies $\mathit{Weight}(\pi) > -Wns$.

Proof. Let σ be implemented by $\mathcal{M} = (M, \mathit{init}, \mathit{upd})$ and let $\rho = v_0 v_1 v_2 \cdots$. Towards a contradiction assume that there is an infix $\pi = v_j \cdots v_{j'}$ of ρ with $\mathrm{Weight}(\pi) \leq -Wns$.

Since W is the maximal absolute weight occurring in $\mathcal{G}$, the infix π attains at least $ns + 1$ different nonpositive energy levels. Hence, we obtain $ns + 1$ prefixes of π with increasing length and with strictly decreasing nonpositive energy levels.

Thus, there are positions j_0, j_1 with $j \leq j_0 < j_1 \leq j'$ with $v_{j_0} = v_{j_1}$, $\mathit{upd}^+(v_0 \cdots v_{j_0}) = \mathit{upd}^+(v_0 \cdots v_{j_1})$, and $\mathrm{Weight}(v_{j_0} \cdots v_{j_1}) < 0$. Hence, the play $v_0 \cdots v_{j_0-1}(v_{j_0} \cdots v_{j_1-1})^\omega$ obtained by repeating the loop between v_{j_0} and v_{j_1} ad infinitum begins in a vertex from which σ is winning and is consistent with σ. This play, however, violates the energy parity condition, which in turn contradicts σ being winning from v_0. □

As a final result, Chatterjee and Doyen gave an upper bound on the complexity of solving energy parity games. This bound was recently improved to pseudo-quasi-polynomial time by Daviaud, Jurdziński, and Lazić [DJL18].

Proposition 3.14 ([CD12, DJL18])**.** *The following problem is in* NP ∩ coNP*:*

> *"Given an energy parity game $\mathcal{G}$ with n vertices, d colors, and largest absolute weight W, and a vertex v in $\mathcal{G}$, does Player 0 win $\mathcal{G}$ from v?"*

Moreover, the problem can be solved in time $\mathcal{O}(dn^{\log(d/\log n)}(W + 1/n))$.

No non-trivial lower bound on the complexity of solving energy parity games was found since the initial presentation of the problem by Chatterjee and Doyen [CD12]. Thus, the problem belongs to a family of problems that are known to be in NP ∩ coNP, but for which no polynomial-time algorithm exists. It shares this property with, e.g., mean-payoff parity games [CHJ05]. Furthermore, solving energy parity games is polynomial-time equivalent to solving mean-payoff parity games [CD12].

3.2.3 Solving Bounded Parity Games with Weights

Recall that in ➔Section 3.2.1 we have solved parity games with weights by iteratively solving bounded parity games with weights. In that section we used a solver for the latter kind of games as a black box. We now show how to solve such games by iteratively solving energy parity games. ➔Page 38

For the remainder of this section, fix a bounded parity game with weights $\mathcal{G} = (\mathcal{A}, \textsc{BndWeightParity}(\Omega, \text{Weight}))$ with vertex set V and set E of edges. Without loss of generality, we assume $\Omega(v) \geq 2$ for all $v \in V$.

As a first step towards solving $\mathcal{G}$, for each vertex v^* of $\mathcal{A}$ we construct an energy parity game $\mathcal{G}_{v^*}$ with the following property:

> Player 1 wins $\mathcal{G}_{v^*}$ from some designated vertex induced by v^* if and only if, when playing in $\mathcal{G}$ and starting from v^*, he is able to unbound the amplitude for the request opened by the initial visit to v^*.

Solving the $\mathcal{G}_{v^*}$ then forms the technical core of a fixed-point algorithm that solves bounded parity games with weights via solving energy parity games. For each v^* with even color, the constructed $\mathcal{G}_{v^*}$ is trivial, as the visit to v^* immediately answers the opened request. For the remainder of this section, fix some vertex v^* of $\mathcal{G}$.

The main obstacle towards the construction of $\mathcal{G}_{v^*}$ is that, in the bounded parity game with weights $\mathcal{G}$, Player 1 may win by unbounding the amplitude for a request from above or from below, while he can only violate the energy condition of $\mathcal{G}_{v^*}$ by unbounding the costs from below. We model this in $\mathcal{G}_{v^*}$ by constructing two copies of $\mathcal{A}$. In one of these copies the edge weights are inherited from $\mathcal{G}$, while they are negated in the other copy. We allow Player 1 to switch between these copies arbitrarily. To compensate for Player 1's power to switch, Player 0 may increase the energy level in the resulting energy parity game during each switch, thus offsetting a previously incurred negative energy level.

First, we define the set of POLARITIES $P = \{+, -\}$ as well as the COMPLEMENTATION OF POLARITIES $\overline{+} = -$ and $\overline{-} = +$. Given a vertex v^* of $\mathcal{A}$, we then define the "POLARIZED" ARENA $\mathcal{A}_{v^*} = (V', V_0', V_1', E')$ of $\mathcal{A} = (V, V_0, V_1, E)$, where

Def. polarities

Def. complementation of polarities

Def. polarized arena

- $V' = (V \times P) \cup (E \times P \times \{0, 1\})$,
- $V_i' = (V_i \times P) \cup (E \times P \times \{i\})$ for $i \in \{0, 1\}$, and
- E' contains the following edges for every edge $e = (v, v') \in E$ with $\Omega(v) \notin \text{Ans}(\Omega(v^*))$ and every polarity $p \in P$:
  - $((v, p), (e, p, 1))$: The player owning the vertex v picks a successor v'. The edge $e = (v, v')$ is stored together with the current polarity p.
  - $((e, p, 1), (v', p))$: Then, Player 1 can either keep the polarity p unchanged and execute the move to v', or
  - $((e, p, 1), (e, p, 0))$: He decides to change the polarity, and the play reaches another auxiliary vertex belonging to Player 0.
  - $((e, p, 0), (e, p, 0))$: If the polarity is to be changed, then Player 0 is able to use a self-loop to increase the energy level (see the definition of the weight function for $\mathcal{A}_{v^*}$ below), before

  - $((e, p, 0), (v', \overline{p}))$: She can eventually complete the polarity switch by moving to v'.
- Furthermore, for every vertex v with $\Omega(v) \in \mathrm{Ans}(\Omega(v^*))$ and every polarity $p \in P$, E' contains the self-loop $((v, p), (v, p))$.

This definition of $\mathcal{A}_{v^*}$ introduces some terminal vertices, namely those of the form $((v, v'), p, i)$ with $\Omega(v) \in \mathrm{Ans}(\Omega(v^*))$. However, these vertices also have no incoming edges and thus are irrelevant for any play not starting in them. In the following, we only consider plays starting in some vertex from $V \times P$. Hence, to simplify the definition, we just ignore these terminal vertices.

Intuitively, a play in $\mathcal{A}_{v^*}$ simulates a play in $\mathcal{A}$ until either

1. Player 0 stops the simulation by using the self-loop at a vertex of the form $(e, p, 0)$ ad infinitum, or
2. until an answer to $\Omega(v^*)$ is reached.

If neither condition holds true during a play in $\mathcal{A}_{v^*}$, then that play simulates a complete infinite play in $\mathcal{A}$. We define the coloring and the weighting for $\mathcal{A}_{v^*}$ so that Player 0 loses if the first condition holds true, while she wins if the second condition holds true. Furthermore, we define the coloring so that all simulating plays that are not stopped induce the same color sequence as the simulated play (save for irrelevant colors on the auxiliary vertices in $E \times P \times \{0, 1\}$). Following this intuition, we define

$$\Omega_{v^*}(v) = \begin{cases} \Omega(v') & \text{if } v = (v', p) \text{ with } \Omega(v') \notin \mathrm{Ans}(\Omega(v^*)) \ , \\ 0 & \text{if } v = (v', p) \text{ with } \Omega(v') \in \mathrm{Ans}(\Omega(v^*)) \ , \\ 1 & \text{otherwise} \ . \end{cases}$$

As desired, due to our assumption that $\Omega(v) \geq 2$ for all $v \in V$, the vertices from $E \times P \times \{0, 1\}$ do not influence the maximal color visited infinitely often during a play, unless either Player 0 opts to remain in some $(e, p, 0)$ ad infinitum, thereby violating the parity condition induced by Ω_{v^*}, or an answer to the color of v^* is reached, thereby satisfying the parity condition induced by Ω_{v^*}.

Moreover, recall that our aim is to allow Player 1 to choose the polarity of edges by switching between the two copies of $\mathcal{A}$. Intuitively, Player 1 should opt for positive polarity in order to unbound the costs incurred by the request posed by v^* from above, while he should opt for negative polarity in order to unbound these costs from below. Since it is, broadly speaking, beneficial for Player 1 to move along edges of negative weight in an energy parity game, we negate the weights of edges in the copy of $\mathcal{A}$ with positive polarity. Thus, we provide an incentive for Player 1 to move along edges of positive weight (in $\mathcal{G}$) while the simulation has positive polarity. Following this intuition, we define

$$\mathrm{Weight}_{v^*}(e) = \begin{cases} -\mathrm{Weight}(v, v') & \text{if } e = ((v, +), ((v, v'), +, 1)) \ , \\ \mathrm{Weight}(v, v') & \text{if } e = ((v, -), ((v, v'), -, 1)) \ , \\ 1 & \text{if } e = ((e', p, 0), (e', p, 0)) \ , \\ 0 & \text{otherwise} \ . \end{cases}$$

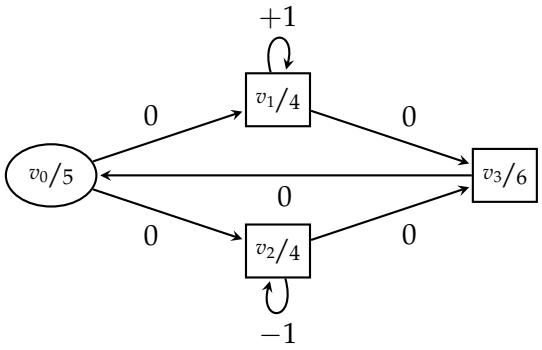


Figure 3.7: A bounded parity game with weights $\mathcal{G}$.

This definition implies that the self-loops at vertices of the form (v, p) with $\Omega(v) \in \mathrm{Ans}(\Omega(v^*))$ have weight zero. Since these vertices moreover have color zero, this construction allows Player 0 to win $\mathcal{G}_{v^*}$ by reaching such a vertex. Intuitively speaking, Player 0 can win $\mathcal{G}_{v^*}$ by answering the request posed at v^*. In particular, if $\Omega(v^*)$ is even, then Player 0 trivially wins $\mathcal{G}_{v^*}$ from (v^*, p), as we then have $\Omega(v^*) \in \mathrm{Ans}(\Omega(v^*))$.

Combining all the parts defined above, we define the energy parity game $\mathcal{G}_{v^*} = (\mathcal{A}_{v^*}, \Omega_{v^*}, \mathrm{Weight}_{v^*})$.

Example 3.15. Consider the bounded parity game with weights shown in Figure 3.7 and the polarized energy parity game $\mathcal{G}_{v_0}$ shown in Figure 3.8. All other $\mathcal{G}_v$ for $v \neq v_0$ are trivial in that each vertex (v, p) in such a game has even color and its only outgoing edge is a self-loop. Hence, Player 0 trivially wins each of these games from the respective designated initial vertex.

Player 1 wins $\mathcal{G}$ from v_0, where a request for color five is opened, which is then kept unanswered with infinite cost by using the self loop at v_1 or v_2 ad infinitum, depending on which successor Player 0 picks.

We argue that Player 1 wins $\mathcal{G}_{v_0}$ from $(v_0, +)$: The outgoing edges of $(v_0, +)$ correspond to Player 0 picking the successor v_1 or v_2 as in $\mathcal{G}$. Before the move to the successor that is thus picked is executed, however, Player 1 gets to pick the polarity of the successor: He should pick $+$ for v_1 and $-$ for v_2, respectively. After Player 1 has thus chosen the polarity going forward, Player 0 can use the self loop at her "tiny" vertices arbitrarily often before she moves to $(v_1, +)$ or $(v_2, -)$. If she opts to use the self loop ad infinitum, Player 1 wins the resulting play, as the "tiny" vertices have color one. If she eventually leaves the self loop, the play reaches the vertex $(v_1, +)$ or $(v_2, -)$. From both vertices, Player 1 can enforce a loop of negative weight, which allows him to win by violating the energy condition. △

We observe that the winning strategy for Player 1 for $\mathcal{G}$ from v is very similar to that for him for $\mathcal{G}_{v_0}$ from $(v_0, +)$. We show that this intuition holds true in general: A winning strategy for Player 1 for $\mathcal{G}_v$ from $(v, +)$ can be transformed into one for him in $\mathcal{G}$ from v.

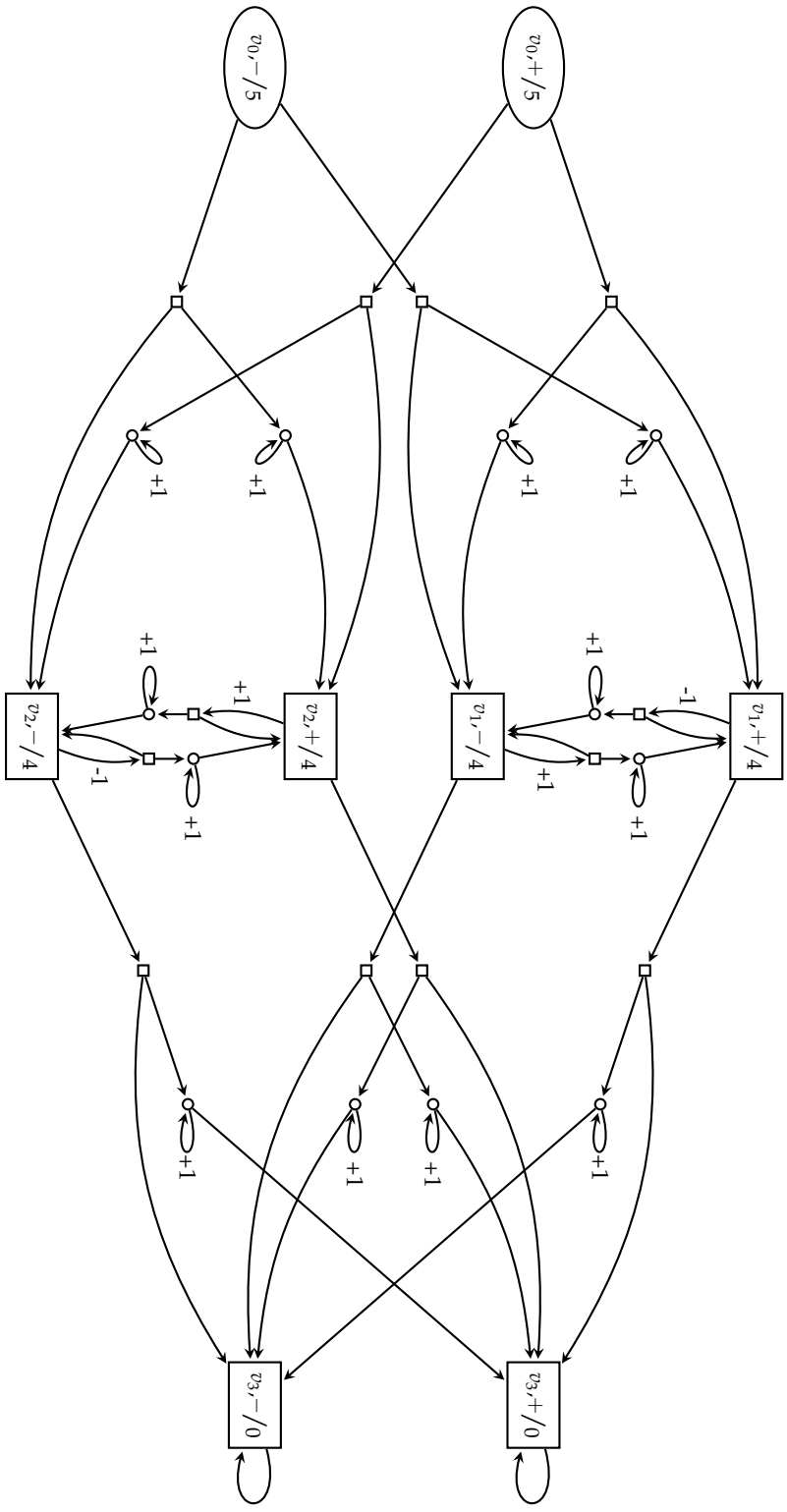


Figure 3.8: The associated energy parity game $\mathcal{G}_{v_0}$ of $\mathcal{G}$. The unnamed vertices of Player 1 (Player 0) are of the form $((v, v'), p, 1)$ (of the form $((v, v'), p, 0)$) when between the vertices (v, p) and (v', p'). For the sake of readability, we omit drawing the weight of edges with weight zero.

This observation does not, however, hold true for Player 0 in general, as shown in the next example.

Example 3.16. Consider the bounded parity game with weights discussed in Example 3.15, but with an added vertex v_{-1} of color three with a single edge to v_0. Then, vertices of the form (v_i, p) with $i \in \{1, 2\}$ in $\mathcal{G}_{v_{-1}}$ are winning sinks for Player 0. Hence, she wins $\mathcal{G}_{v_{-1}}$ from (v_{-1}) in spite of losing the bounded parity game with weights from v_{-1}. This is caused by $\mathcal{G}_{v_{-1}}$ only "considering" the request posed by visiting v_{-1}, but "disregarding" those made along the play. △

As witnessed by Example 3.16, the initial request posed by visiting the vertex v^* inducing $\mathcal{G}_{v^*}$ plays a special role in the construction: It is this request that Player 1 aims to keep unanswered with infinite cost. To "stitch together" the solutions for the individual $\mathcal{G}_{v^*}$ and to complete our construction, we show a statement reminiscent of ➔Lemma 3.8: If Player 0 wins $\mathcal{G}_v$ from $(v, +)$ for every v, then she also wins $\mathcal{G}$ from every vertex. With this relation at hand, one can again construct a fixed-point algorithm solving bounded parity games with weights using an oracle for solving energy parity games that is very similar to ➔Algorithm 3.1.

➔Sec. 3.2, Page 40

➔Sec. 3.2, Page 42

Lemma 3.17. *Let $\mathcal{G}$ be a bounded parity game with weights with vertex set V.*

1. *Let $v^* \in V$. If Player 1 wins $\mathcal{G}_{v^*}$ from $(v^*, +)$, then $v^* \in W_1(\mathcal{G})$.*
2. *If Player 0 wins $\mathcal{G}_{v^*}$ from $(v^*, +)$ for all $v^* \in V$, then $W_1(\mathcal{G}) = \emptyset$.*

Before proving this lemma, we first argue that it indeed provides us with an argument showing NP ∩ coNP-membership of the problem of solving parity games with weights. To show this membership, we provide an algorithm that solves bounded parity games with weights by repeatedly solving energy parity games, which is very similar to Algorithm 3.1. Indeed, we just swap the roles of the players: We compute 1-attractors instead of 0-attractors and we change the definition of X_k, thus obtaining Algorithm 3.2.

Algorithm 3.2
A fixed-point algorithm computing $W_1(\mathcal{A}, \textsc{BndWeightParity}(\Omega, \text{Weight}))$.

$k = 0$; $W_1^k = \emptyset$; $\mathcal{A}_k = \mathcal{A}$
repeat
  $k = k + 1$
  $X_k = \{v^* \mid$ Player 1 wins the
    energy parity game $((\mathcal{A}_{k-1})_{v^*}, \Omega_{v^*}, \text{Weight}_{v^*})$ from $(v^*, +)\}$
  $W_0^k = W_0^{k-1} \cup \text{Attr}_1^{\mathcal{A}_{k-1}}(X_k)$
  $\mathcal{A}_k = \mathcal{A}_{k-1} \setminus \text{Attr}_1^{\mathcal{A}_{k-1}}(X_k)$
until $X_k = \emptyset$
return W_1^k

In each iteration of its main loop Algorithm 3.2 solves one energy parity game for each vertex in $\mathcal{A}_{k-1}$. As in each iteration of this loop, save for the final one, at least

one vertex of $\mathcal{A}_{k-1}$ is slated for removal, the loop is iterated at most $|\mathcal{A}|$ times. Hence, Algorithm 3.2 terminates after solving at most a quadratic (in the number of vertices of $\mathcal{G}$) number of energy parity games. Furthermore, the proof of correctness is analogous to the one for Algorithm 3.1, relying on Lemma 3.17 instead of Lemma 3.8. We again only require two further properties: The fact that the bounded parity condition with weights is 1-extendable, and the assertion that $\mathrm{Attr}_1^{\mathcal{A}_{k-1}}(X_k)$ is a trap for Player 0 in $\mathcal{A}_{k-1}$. We have discussed the former assertion in ➔Section 3.2.1. The latter one is easy to verify.

➔Page 38

Plugging Algorithm 3.2 into ➔Algorithm 3.1 yields the main theorem of our chapter due to ➔Proposition 3.14. This theorem provides an upper bound on the complexity of solving parity games with weights.

➔Sec. 3.2, Page 42
➔Sec. 3.2, Page 46

Theorem 3.18. *The following problem is in* NP ∩ coNP*:*

> *"Given a parity game with weights $\mathcal{G}$ with n vertices, d colors, and largest absolute weight W, and a vertex v in $\mathcal{G}$, does Player 0 win $\mathcal{G}$ from v?"*

Furthermore, Daviaud, Jurdziński, and Lazić observed that, due to our reduction to the problem of solving energy parity games, the problem stated in Theorem 3.18 can be solved in time $\mathcal{O}(n^2(d(n')^{\log(d/\log n')+4.45}(W+1/n')))$, where $n' \in \mathcal{O}(n^2)$ [DJL18].

NP ∩ coNP-membership of the problem follows from NP ∩ coNP-membership of the problem of solving energy parity games due to Chatterjee and Doyen [CD12]. The pseudo-quasi-polynomial runtime of the algorithm solving parity games with costs, on the other hand, follows from a reduction of the problem of solving energy parity games to that of solving mean-payoff parity games due to Chatterjee and Doyen [CD12] and an algorithm solving the latter games in pseudo-quasi-polynomial time due to Daviaud, Jurdziński, and Lazić [DJL18].

The remainder of this section is dedicated to showing Lemma 3.17. In order to do so, we first introduce some notation. Recall that we fixed a bounded parity game with weights $\mathcal{G}$ with vertex set V and set E of edges. Let $v^* \in V$ and consider $\mathcal{G}_{v^*}$ with vertex set V'. We distinguish three types of plays in $\mathcal{G}_{v^*}$ starting in (v^*, p):

Type -1: Plays that have a suffix $(e, p, 0)^\omega$ for some $e \in E$ and some $p \in P$.

Type 0: Plays that visit infinitely many vertices from both $V \times P$ and $E \times P \times \{0, 1\}$.

Type 1: Plays that have a suffix $(v, p)^\omega$. This implies $\Omega(v) \in \mathrm{Ans}(\Omega(v^*))$.

Clearly, plays of type -1 are losing for Player 0 due to the coloring of $\mathcal{G}'_{v^*}$ labeling vertices of the form $(e, p, 0)$ with the odd color one. Dually, plays of type 1 are losing for Player 1, since $\Omega(v) \in \mathrm{Ans}(\Omega(v^*))$ implies that both v and (v, p) carry an even color. We formalize this observation in the following remark.

Remark 3.19. *Let ρ' be a play in $\mathcal{G}_{v^*}$ that starts in (v^*, p).*

1. *If ρ' is consistent with a winning strategy for Player 0 from (v^*, p), then ρ' is not a play of type -1.*

2. *If ρ' is consistent with a winning strategy for Player 1 from (v^*, p), then ρ' is not a play of type 1.*

In order to remove the added vertices of the form $E \times P \times \{0,1\}$ from plays in $\mathcal{G}_{v^*}$, we define the homomorphism unpol: $(V')^* \cup (V')^\omega \to \{\epsilon\} \cup V^* \cup V^\omega$ induced by $\text{unpol}(v,p) = v$ and $\text{unpol}(e,p,i) = \epsilon$ for $v \in V$, $e \in E$, $p \in P$, and $i \in \{0,1\}$. Let $\rho' \in (V')^* \cup (V')^\omega$. We call $\text{unpol}(\rho')$ the UNPOLARIZATION of ρ'. **Def.** unpolarization

Remark 3.20. *Let ρ' be a play of type 0 in $\mathcal{G}_{v^*}$. We have $\rho' \in \text{PARITY}(\Omega_{v^*})$ if and only if $unpol(\rho') \in \text{PARITY}(\Omega)$.*

Proof of Lemma 3.17.1

We now show the first item of Lemma 3.17. Recall that this item states that if Player 1 wins $\mathcal{G}_{v^*}$ from $(v^*, +)$, then he wins $\mathcal{G}$ from v^*.

Let τ_{v^*} be a positional winning strategy for Player 1 from $(v^*, +)$ in $\mathcal{G}_{v^*}$ such that each play beginning in $(v^*, +)$ and consistent with τ_{v^*} either violates the parity condition, or has the energy level along its prefixes diverge towards $-\infty$. Such a strategy exists for Player 1 if he wins $\mathcal{G}_{v^*}$ from $(v^*, +)$ due to ➔Proposition 3.11. We define a winning strategy τ for Player 1 from v^* in $\mathcal{G}$ that mimicks the moves made by τ_{v^*}. To this end, τ keeps track of a play prefix in $\mathcal{G}_{v^*}$. Formally, we define τ together with a simulation function h that satisfies the following invariant: ➔Sec. 3.2, Page 45

> If π is a non-empty play prefix in $\mathcal{A}$ beginning in v^*, is consistent with τ, and ends in some v, then $h(\pi)$ is a play prefix in $\mathcal{A}_{v^*}$ that starts in $(v^*, +)$, is consistent with τ_{v^*}, and ends in some (v, p). Further, $\text{unpol}(h(\pi)) = \pi$.

If h satisfies the above invariant, then we, intuitively, construct the strategy τ as follows: Let π be some non-empty play prefix in $\mathcal{A}$ beginning in v^* that is consistent with τ and recall that $\tau_{v^*}(h(\pi))$ prescribes a move to some vertex $((v,v'),p,1)$. We then define τ to move to v' in $\mathcal{G}$, thus mimicking the move made by the strategy τ_{v^*}.

We define h and τ inductively and begin with $h(v^*) = (v^*, +)$, which clearly satisfies the invariant. Now let $\pi = v_0 \cdots v_j$ be some non-empty play prefix in $\mathcal{A}$ beginning in v^* and consistent with τ such that $h(\pi)$ is defined. Due to the invariant, $h(\pi)$ ends in (v_j, p_j) for some $p_j \in P$. We first determine a successor of v_j, defining $\tau(\pi)$ along the way if v_j is a vertex of Player 1.

If $v_j \in V_1$, let v_{j+1} be the unique vertex of $\mathcal{A}$ such that $h(\pi) \cdot ((v_j, v_{j+1}), p_j, 1)$ is consistent with τ_{v^*} and define $\tau(\pi) = v_{j+1}$. Such a v_{j+1} exists, because (v_j, p_j), the last vertex of $h(\pi)$, satisfies $\Omega(v_j) \notin \text{Ans}(\Omega(v^*))$ due to the invariant and Remark 3.19.2. If, however, $v_j \in V_0$, then let v_{j+1} be an arbitrary successor of v_j in $\mathcal{A}$. In either case, it remains to define $h(\pi \cdot v_{j+1})$.

Since we aim to have $h(\pi \cdot v_{j+1})$ simulate the move from v_j to v_{j+1} in $\mathcal{A}_{v^*}$, we first move from (v_j, p_j) to $((v_j, v_{j+1}), p_j, 1)$. Moreover, as $\pi \cdot v_{j+1}$ is consistent with τ, we aim to simulate this play prefix such that $h(\pi \cdot v_{j+1})$ is consistent with τ_{v^*} in order to satisfy the invariant. The strategy τ_{v^*} may either prescribe for Player 1 to preserve the polarity p_j, or it may prescribe to switch it during the simulated move from v_j to v_{j+1}.

In the former case, i.e., if $\tau_{v^*}(h(\pi) \cdot ((v_j, v_{j+1}), p_j, 1)) = (v_{j+1}, p_j)$, we define

$$h(\pi \cdot v_{j+1}) = h(\pi) \cdot ((v_j, v_{j+1}), p_j, 1) \cdot (v_{j+1}, p_j) \ .$$

In the latter case, i.e., if Player 1 opts to switch polarities, the simulated play proceeds to $((v_j, v_{j+1}), p_j, 0)$, where Player 0 gets an opportunity to recharge the energy by taking the self-loop of that vertex finitely often. Recall that if she remains in that vertex ad infinitum, she loses the resulting play due to the coloring Ω_{v^*} assigning that vertex the odd color one. We opt to let her recover the energy lost so far in the play prefix, i.e., we pick

$$m_j = \max\left\{0, -\text{Weight}_{v^*}(h(\pi) \cdot ((v_j, v_{j+1}), p_j, 1) \cdot ((v_j, v_{j+1}), p_j, 0))\right\}$$

and define

$$h(\pi \cdot v_{j+1}) = h(\pi) \cdot ((v_j, v_{j+1}), p_j, 1) \cdot ((v_j, v_{j+1}), p_j, 0)^{m_j+1} \cdot (v_{j+1}, \overline{p_j})$$

in this case. In particular, since we traverse the self-loop of the vertex $((v_j, v_{j+1}), p_j, 0)$ m_j times, we visit the vertex itself $m_j + 1$ times. Since $h(\pi \cdot v_{j+1})$ is consistent with τ_{v^*} whether or not Player 1 switches the polarity, this definition satisfies the invariant in either case. This completes the definition of h.

It remains to show that τ is indeed winning for Player 1 from v^* in $\mathcal{G}$. To this end, let $\rho = v_0 v_1 v_2 \cdots$ be a play in $\mathcal{A}$ that starts in v^* and is consistent with τ. We show $\rho \notin \textsc{BndWeightParity}(\Omega, \text{Weight})$ by examining the play ρ' in $\mathcal{A}_{v^*}$, which is the limit of the $h(\pi)$ for increasing prefixes π of ρ. Due to the invariant, ρ' starts in $(v^*, +)$ and is consistent with τ_{v^*}. Moreover, due to the construction of h, we obtain $\text{unpol}(\rho') = \rho$. Finally, we have that ρ' is a play of type 0 in $\mathcal{G}_{v^*}$: Due to our definition of the simulation function h, Player 0 never remains in a vertex of the form $(e, p, 0)^\omega$ ad infinitum, i.e., ρ' is not of type -1. Dually, since each prefix of ρ' is consistent with the winning strategy τ_{v^*} from v^*, the play never encounters a vertex of the form (v, p), where $\Omega(v) \in \text{Ans}(\Omega(v^*))$, i.e., ρ is not of type 1. Hence, due to Remark 3.20, ρ satisfies the parity condition if and only if ρ' satisfies the parity condition.

As the play ρ' is consistent with the positional winning strategy τ_{v^*} for Player 1 in the energy parity game $\mathcal{G}_{v^*}$, it either violates the parity condition or the energy levels of its prefixes diverge towards $-\infty$. In the former case ρ violates the parity condition as well, as argued above, i.e., ρ is indeed winning for Player 1 in this case.

Hence assume that ρ' violates the energy condition. Due to the structure of $\mathcal{A}_{v^*}$ and the construction of h we have

$$\rho' = \Pi_{j=0,1,2,\ldots}\left[(v_j, p_j) \cdot ((v_j, v_{j+1}), p_j, 1) \cdot ((v_j, v_{j+1}), p_j, 0)^{m_j}\right]$$

for some $m_j \in \mathbb{N}$. Since ρ' violates the energy condition, we have

$$\inf_{j \in \mathbb{N}}\left[\text{Weight}_{v^*}((v_0, p_0) \cdots (v_j, p_j) \cdot ((v_j, v_{j+1}), p_j, 1))\right] = -\infty \ .$$

The restriction to play prefixes of this form suffices due to the structure of $\mathcal{A}_{v^*}$ and, in turn, the structure of ρ'. Moreover, since Player 1 wins $\mathcal{G}_{v^*}$ from $(v^*, +)$, the initial

vertices v^* and $(v^*, +)$ of ρ and ρ', respectively, have the same odd color. Also, as ρ' is a play of type 0, the request for the color $\Omega(v^*) = \Omega(v^*, +)$ is never answered in ρ nor ρ'. We show that the weight along the prefixes of ρ diverges, which implies that the request for $\Omega(v^*)$ in ρ is unanswered with infinite cost and thus concludes the proof of the first item of Lemma 3.17.

To this end, we split ρ' into infixes in which Player 1 does not switch the polarity of the simulation. Given a vertex $v = (v', p)$ or $v = ((v', v''), p, i)$, we call p the POLARITY of v. Let $\rho' = \mu'_0\mu'_1\mu'_2\cdots$, where each μ'_j is a maximal finite (or infinite) infix (or suffix) of ρ', such that all vertices in μ'_j have the same polarity. We call an infix μ'_j of ρ' an EQUIPOLAR INFIX (EPI) of ρ'.

Def. polarity

Def. equipolar infix (EPI)

If vertices of the form $(e, p, 0)$ occur at all, then they only appear at the end of such an infix, since the polarity remains constant throughout each infix μ'_j. Hence, Player 0 can only recover lost energy via repeatedly traversing a self-loop of a vertex in $\mathcal{A}_{v^*}$ at the last vertex visited in μ'_j, if at all. Thus, the weight accumulated along μ'_j and the energy level attained during $\text{unpol}(\mu'_j)$ are closely related, as formally stated in the following remark.

Remark 3.21. *Let μ' be an EPI beginning in (v_j, p_j) and let $\mu = unpol(\mu') = v_j v_{j+1} v_{j+2} \cdots$ be the unpolarization of μ'. For each j' with $j \leq j' < j + |\mu|$, we have $\left|Weight(v_j \cdots v_{j'})\right| = \left|Weight_{v^*}((v_j, p_j) \cdots (v_{j'}, p_{j'}))\right|$.*

Since the amplitude of a play prefix π only depends on the absolute value of the accumulated weights of the edges traversed during π, we obtain $\text{Ampl}(\text{unpol}(\mu')) = \text{Ampl}(\mu')$ for all EPIs μ' of ρ' due to Remark 3.21, the structure of $\mathcal{A}_{v^*}$, and the definition of h. Thus, if there exist only finitely many EPIs of ρ', let μ' be the final EPI of ρ', which is of infinite length, let $\mu = \text{unpol}(\mu')$, and note that, due to $\text{Ampl}(\rho') = \infty$, we have $\text{Ampl}(\mu') = \infty$. As argued above, we obtain $\text{Ampl}(\mu) = \infty$, which implies $\text{Ampl}(\rho) = \infty$, as μ is a suffix of ρ. This concludes the proof in the case that there are only finitely many EPIs of ρ'.

Now consider the case that there exist infinitely many EPIs of ρ'. By construction of ρ', the energy level is nonnegative at the end of each EPI. By assumption on ρ', for each bound $b \in \mathbb{N}$ there exists an EPI μ' of ρ' with a prefix of weight less than than $-b$. We obtain $\text{Ampl}(\text{unpol}(\mu')) > b$ via Remark 3.21.

As argued above, the initial vertex of ρ opens a request that is never answered throughout ρ. Moreover, the weight accumulated along ρ diverges, i.e., the request posed by visiting the initial vertex of ρ is unanswered with infinite cost. Hence, ρ is winning for Player 1, which concludes the proof of the first item of Lemma 3.17 for the case that there are infinitely many EPIs of ρ'.

Proof of Lemma 3.17.2

We now proceed to prove the second item of Lemma 3.17. Recall that this item asserts that if, for all $v \in V$, Player 0 wins $\mathcal{G}_v$ from $(v, +)$, then the winning region of Player 1 in $\mathcal{G}$ is empty. To prove this lemma, we construct a strategy σ for Player 0 in $\mathcal{G}$ that is

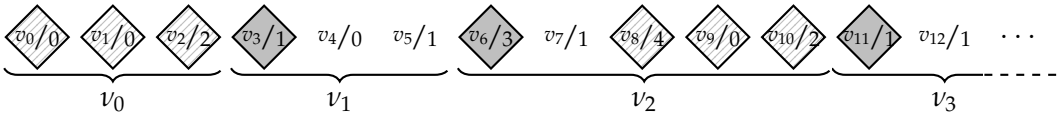


Figure 3.9: A play ρ, its induced color sequence, its most relevant requests, and the ESIs of ρ.

winning for her from every vertex in $\mathcal{G}$. As winning regions are disjoint, this implies the desired result.

For each energy parity game $\mathcal{G}_v = (\mathcal{A}_v, \Omega_v, \text{Weight}_v)$ we have $n'_v = |\mathcal{A}_v| \in \mathcal{O}(|\mathcal{A}|^2)$ as well as that d'_v is the number of odd colors assigned by Ω_v, which is bounded by the number of odd colors assigned by Ω plus one (due to our assumption of $\Omega(v) \geq 2$ for all vertices v of $\mathcal{G}$), and $W'_v = \max(\text{Weight}(E'_v)) = \max(\text{Weight}(E) \cup \{1\})$, where E and E'_v are the sets of edges in $\mathcal{A}$ and the $\mathcal{A}_v$, respectively. We define n', d', and W' as the maximum of the n'_v, d'_v, and W'_v, respectively, and observe that n', d', and W' are bounded from above by polynomials of the corresponding values of $\mathcal{A}$. Recall that the assumption of the second item of Lemma 3.17 states that we have that Player 0 wins each $\mathcal{G}_v$ from $(v, +)$. Hence, due to →Proposition 3.12, Page 46 for each $v \in V$, there exists a finite-state strategy σ_v with at most $n'd'W'$ states that is winning for Player 0 from $(v, +)$ in $\mathcal{G}_v$.

We construct the winning strategy σ for Player 0 in $\mathcal{G}$ by "stitching together" the individual σ_v. To this end, given a play prefix, we identify the request which should be answered most urgently. Say this request was first opened by visiting vertex v. The strategy σ then mimics the moves made by σ_v when starting in $(v, +)$. Once the request for $\Omega(v)$ is answered, σ prescribes arbitrary moves until a new request is opened, say by visiting the vertex v'. The request for $\Omega(v')$ becomes the new most relevant request and the strategy σ mimicks the moves prescribed by $\sigma_{v'}$ when starting in $(v', +)$. As Player 0 wins every $\mathcal{G}_v$ from $(v, +)$, we can iterate this process ad infinitum.

Def. open

Recall that, given a play prefix $\pi = v_0 \cdots v_j$, we say that a request for color c is OPEN in π if there exists a position j' with $0 \leq j' \leq j$ such that $\Omega(v_{j'}) = c$ and if for all positions j'' with $j' \leq j'' \leq j$, we have $\Omega(v_{j''}) \notin \text{Ans}(\Omega(v'_j))$. There is never an open request for an even color. Intuitively, we define the position of the most relevant request as the position of the earliest occurrence of the largest open request in π which has not yet been answered.

Def. posMRR

If there is no open request in π, the POSITION OF THE MOST RELEVANT REQUEST is undefined and we write $\text{posMRR}(\pi) = \bot$. Otherwise, let c be the maximal color for which there is an open request in π. We define $\text{posMRR}(\pi)$ as the smallest position j' such that the request for color c is open in all prefixes of π of length at least j'.

Example 3.22. Consider the play prefix shown in Figure 3.9. We mark a position j with solid background if we have $\text{posMRR}(v_0 \cdots v_j) = j$ and with dashed background if we have $\text{posMRR}(v_0 \cdots v_j) = \bot$. Otherwise, i.e., if $\bot \neq \text{posMRR}(v_0 \cdots v_j) < j$, we leave j unmarked. For those positions, $\text{posMRR}(v_0 \cdots v_j)$ is equal to the largest (i.e., last visited) earlier position marked with solid background. △

We aim to define the strategy σ for Player 0 in $\mathcal{G}$ by simulating moves made by the strategies σ_v in $\mathcal{G}_v$. To this end, we need to simulate play prefixes in the former game in the latter ones. For this, we use an approach reminiscent of that of the proof of Lemma 3.17.1 and define σ together with a simulation function h mapping plays in $\mathcal{G}$ that are consistent with σ to a sequence of vertices from V', i.e., from the set of vertices shared by all $\mathcal{G}_v$ (but not necessarily to a play prefix in any $\mathcal{G}_v$). We define the simulation function h such that we are able to leverage the choices made by the σ_v in order to define σ. Our aim is to define h such that it satisfies the following invariant:

> Let $\pi = v_0 \cdots v_j$ be a play in $\mathcal{A}$ consistent with σ. Then $h(\pi)$ ends in some (v_j, p_j). Moreover, if $\text{posMRR}(\pi) = j' \neq \bot$, then $h(\pi)$ has a (unique) suffix $(v_{j'}, +) \cdots (v_j, p_j)$ that is consistent with $\sigma_{v_{j'}}$ and that satisfies $\text{unpol}((v_{j'}, +) \cdots (v_j, p_j)) = v_{j'} \cdots v_j$.

We define h and σ inductively and begin with $h(v) = (v, +)$ for each $v \in V$, which clearly satisfies the invariant. Now let $\pi = v_0 \cdots v_j$ be a nonempty play prefix in $\mathcal{A}$ and consistent with σ such that $h(\pi)$ is defined. We again first determine a successor of v_j, defining $\sigma(\pi)$ along the way if v_j is a vertex of Player 0.

If $v_j \in V_1$, let v_{j+1} be an arbitrary successor of v_j in $\mathcal{A}$. If, however, $v_j \in V_0$, we distinguish two cases based on whether or not $\text{posMRR}(\pi)$ is defined. If $\text{posMRR}(\pi) = \bot$, again let v_{j+1} be an arbitrary successor of v_j. If, however, $\text{posMRR}(\pi) = j' \neq \bot$, then the invariant of h yields a suffix $(v_{j'}, +) \cdots (v_j, p_j)$ of $h(\pi)$ that is consistent with $\sigma_{v_{j'}}$. Let v_{j+1} be the unique vertex such that $(v_{j'}, +) \cdots (v_j, p_j) \cdot ((v_j, v_{j+1}), p_j, 1)$ is consistent with $\sigma_{v_{j'}}$. Such a vertex v_{j+1} exists, because the request posed by visiting $v_{j'}$ is open in π due to $\text{posMRR}(\pi) = j'$ and since the color sequences induced by $v_{j'} \cdots v_j$ and $(v_{j'}, +) \cdots (v_j, p_j)$ coincide, save for the irrelevant intermediate vertices of colors zero and one. Hence, (v_j, p_j) is not an accepting sink in $\mathcal{G}_{v_{j'}}$. Since $v_j \in V_0$, the vertex v_{j+1} is unique and we define $\sigma(\pi) = v_{j+1}$. This concludes the definition of σ.

It remains to define $h(\pi \cdot v_{j+1})$ such that it satisfies the above invariant. To this end, we use one of two operations. Firstly, we define the DISCONTINUOUS EXTENSION OF $h(\pi)$ WITH v_{j+1} as $h(\pi) \cdot (v_{j+1}, +)$. Recall that in none of the $\mathcal{A}_v$ there exists an edge between the final vertex of $h(\pi)$ and $(v_{j+1}, +)$, since we omit the intermediate vertices of the form (e, p, i). Thus, the resulting sequence of vertices is not a play infix in any of the $\mathcal{A}_v$.

Def. discontinuous extension

Secondly, we define a simulated extension of $h(\pi)$ such that we obtain $h(\pi \cdot v_{j+1})$ by simulating the move from v_j to v_{j+1} in some $\mathcal{G}_v$. Formally, the SIMULATED EXTENSION OF $h(\pi)$ WITH v_{j+1} AND CHARGE m is $h(\pi) \cdot ((v_j, v_{j+1}), p_j, 1) \cdot ((v_j, v_{j+1}), p_j, 0)^m \cdot (v_{j+1}, p_{j+1})$, where $p_{j+1} = p_j$ if $m = 0$ and $p_{j+1} = \overline{p_j}$ otherwise. This ensures that a suffix of the simulated extension is indeed a play infix in some $\mathcal{G}_v$.

Def. simulated extension

In order to define $h(\pi \cdot v_{j+1})$ we again distinguish whether or not $\text{posMRR}(\pi)$ is defined. If $\text{posMRR}(\pi) = \bot$, we define $h(\pi \cdot v_{j+1})$ to be the discontinuous extension of $h(\pi)$ with v_{j+1}. This clearly satisfies the first condition of the invariant. Moreover, the second condition of the invariant is satisfied as well: If $\text{posMRR}(\pi \cdot v_{j+1}) = \bot$, this condition holds true vacuously. Otherwise we obtain $\text{posMRR}(\pi \cdot v_{j+1}) = j + 1$ due to

our assumption $\text{posMRR}(\pi) = \bot$ and observe that the suffix $(v_{j+1}, +)$ of $h(\pi \cdot v_{j+1})$ satisfies the second condition of the invariant.

If, however, $\text{posMRR}(\pi) \neq \bot$, then let $\text{posMRR}(\pi) = j'$. By definition of posMRR we have $\text{posMRR}(\pi \cdot v_{j+1}) \in \{\bot, j', j+1\}$. We distinguish two sub-cases and first define $h(\pi \cdot v_{j+1})$ for the case that $\text{posMRR}(\pi \cdot v_{j+1}) \in \{\bot, j+1\}$. In this case, the move to v_{j+1} either answers the most relevant request in π, or the request posed by visiting v_{j+1} is itself the most relevant request of $\pi \cdot v_{j+1}$: We have $\text{posMRR}(\pi \cdot v_{j+1}) = \bot$ in the former case and $\text{posMRR}(\pi \cdot v_{j+1}) = j+1$ in the latter one. In either case, we define $h(\pi \cdot v_{j+1})$ to be the discontinuous extension of $h(\pi)$ with v_{j+1} and observe that the first condition of the invariant holds true. If $\text{posMRR}(\pi \cdot v_{j+1}) = \bot$, the second condition of the invariant holds true vacuously. If, however, $\text{posMRR}(\pi \cdot v_{j+1}) = j+1$, then the suffix $(v_{j+1}, +)$ witnesses that the second condition of the invariant holds true.

Now assume that the move to v_{j+1} neither opens a new most relevant request, nor answers the existing one, i.e., that we have $\text{posMRR}(\pi \cdot v_{j+1}) = j'$. In this case, we extend the suffix of $h(\pi)$ that is consistent with $\sigma_{v_{j'}}$ by simulating the move from v_j to v_{j+1}. Recall that we picked the vertex v_{j+1} such that $(v_{j'}, +) \cdots (v_j, p_j) \cdot ((v_j, v_{j+1}), p_j, 1)$ is consistent with $\sigma_{v_{j'}}$. As we can freely choose whether or not Player 1 switches the polarity in the simulation, we follow the intuition stated during the construction of the polarized arena: Recall that both players currently play "with respect to" the request for $\Omega(v_{j'})$ opened by visiting $v_{j'}$. Hence, we opt to let Player 1 move to positive polarity if the cost of the request for $\Omega(v_{j'})$ so far is nonnegative, and let him move to negative polarity otherwise. To this end, we use the sign function Sgn which is defined as

$$\text{Sgn}(n) = \begin{cases} + & \text{if } n \geq 0 \ , \\ - & \text{otherwise} \ . \end{cases}$$

If $\text{Sgn}(\text{Weight}(v_{j'} \cdots v_{j+1})) = p_j$, we define $h(\pi \cdot v_{j+1})$ to be the simulated extension of $h(\pi)$ with v_{j+1} and charge 0, thus implementing the polarity switch as described above. Otherwise, i.e., if $\text{Sgn}(\text{Weight}(v_{j'} \cdots v_{j+1})) = \overline{p_j}$, let $m \in \mathbb{N}$ such that $(v_{j'}, +) \cdots (v_j, p_j) \cdot ((v_j, v_{j+1}), p_j, 1) \cdot ((v_j, v_{j+1}), p_j, 0)^m \cdot (v_{j+1}, \overline{p_j})$ is consistent with $\sigma_{v_{j'}}$. Such an m exists, as otherwise the play $(v_{j'}, +) \cdots (v_j, p_j) \cdot ((v_j, v_{j+1}), p_j, 1) \cdot ((v_j, v_{j+1}), p_j, 0)^\omega$ of type -1 that starts in $(v_{j'}, +)$ would be consistent with the winning strategy $\sigma_{v_{j'}}$ from $(v_{j'}, +)$ for Player 0, contradicting ➔Remark 3.19.1. We define $h(\pi \cdot v_{j+1})$ to be the simulated extension of $h(\pi)$ with v_{j+1} and charge m. Since we have $\text{posMRR}(v_0 \cdots v_{j+1}) = j'$ by assumption, either definition of $h(\pi \cdot v_{j+1})$ satisfies the invariant. This completes the definition of h.

➔Sec. 3.2, Page 52

It remains to show that the strategy σ is indeed winning for Player 0 from every vertex in $\mathcal{G}$. To this end, fix some $v^* \in V$ as well as some play $\rho = v_0 v_1 v_2 \cdots$ starting in v^* that is consistent with σ, and let ρ' be the limit of the $h(\pi)$ for increasing prefixes π of ρ. By construction of h, we obtain $\text{unpol}(\rho') = \rho$. Hence, the play ρ' is of the form $(v_0, p_0) \cdots (v_1, p_1) \cdots (v_2, p_2) \cdots$. We call a position j of ρ a DISCONTINUITY of ρ if either $j = 0$ or if $h(v_0 \cdots v_j)$ is the discontinuous extension of $h(v_0 \cdots v_{j-1})$ with v_j.

Def. discontinuity

Let j and j' be adjacent discontinuities of ρ with $j < j'$. We call the infix $v_j \cdots v_{j'-1}$ of ρ an EQUISTRATEGIC INFIX (ESI) of ρ since it was constructed by mimicking the strategy σ_{v_j}. Moreover, if there only exist finitely many discontinuities of ρ, let j^* be its final discontinuity. We call the suffix $v_{j^*}v_{j^*+1}v_{j^*+2}\cdots$ of ρ the TERMINAL ESI of ρ.

Def. equistrategic infix (ESI)

Def. terminal ESI

Remark 3.23. *Let $\mu = v_j v_{j+1} v_{j+2} \cdots$ be an ESI of ρ.*

1. *If μ is finite, then the infix μ' of ρ' starting at position $\left|h(v_0 \cdots v_j)\right| - 1$ and ending at position $\left|h(v_0 \cdots v_{j+|\mu|-1})\right| - 1$ starts in $(v_j, +)$, is consistent with σ_{v_j}, and ends in $(v_{j+|\mu|-1}, p)$ for some $p \in P$.*
2. *If μ is infinite, then the suffix μ' of ρ' starting at position $\left|h(v_0 \cdots v_j)\right| - 1$ is a play in $\mathcal{A}_v$ that starts in $(v_j, +)$ and is consistent with σ_{v_j}.*

For each position j of ρ we define $\mathrm{ESI}(j) = k$ if the k-th ESI of ρ contains j. Furthermore, if $\mu = v_j v_{j+1} v_{j+2} \cdots$ is an ESI of ρ, then we call $\Omega(v_j)$ the CHARACTERISTIC COLOR of μ. By construction of h, if the characteristic color of an ESI μ is even, then μ consists only of a single vertex. If, however, the characteristic color c of an ESI μ is odd, then we have $\Omega(v) \leq c$ for all vertices v in μ. Moreover, let c' be the characteristic color of the ESI succeeding μ, if μ is not the terminal ESI of ρ. If c is odd, then we have $c' > c$ due to the construction of h. If c' is even, this observation implies $c' \in \mathrm{Ans}(\Omega(v))$ for all vertices v in μ. As the number of colors in $\mathcal{G}$ is finite, this in turn implies that the number of ESIs between a request and its response (if a response exists at all) is bounded.

Def. characteristic color

Remark 3.24. *Let j be some position in ρ and let $k = \mathrm{ESI}(j)$. Moreover, let d be the number of odd colors in $\mathcal{G}$.*

1. *If the request at position j is first answered at position j', then $\mathrm{ESI}(j') < k + d$*
2. *If the request at position j is unanswered in ρ, then ρ contains less than $k + d$ many ESIs.*

We now show that the play ρ satisfies the bounded parity condition with weights. Recall that this condition requires

1. that the play ρ satisfies the parity condition, as well as
2. that the cost of almost all requests in ρ is bounded and that there exists no unanswered request with infinite cost in ρ.

We first show that ρ satisfies the qualitative parity condition. In a second step, we then show that there exists a bound on the cost of each (answered or unanswered) request in ρ, which implies the latter condition. The former condition, i.e., that ρ satisfies the parity condition, is in large parts implied by Remark 3.24.

Lemma 3.25. *The play ρ satisfies the parity condition.*

Proof. If ρ contains no unanswered requests, then it vacuously satisfies the parity condition. Hence, let j be the position of an unanswered request in ρ. Due to Remark 3.24, ρ contains only finitely many ESIs in ρ. Let $\mu = v_{j^*}v_{j^*+1}v_{j^*+2}\cdots$ be the terminal ESI of ρ. By construction of h, there exists a suffix μ' of ρ' with $\mathrm{unpol}(\mu') = \mu$.

Due to Remark 3.23.2, the suffix μ' is a play in $\mathcal{A}_{v_j^*}$ that begins in $(v_{j^*}, +)$ and is consistent with the winning strategy $\sigma_{v_{j^*}}$ for Player 0 from $(v_{j^*}, +)$ in $\mathcal{G}_{v_{j^*}}$. Moreover, μ' is a play of type 0 due to being consistent with $\sigma_{v_{j^*}}$ and due to construction of h. Hence, we obtain that μ satisfies the parity condition via ➔Remark 3.20, which in turn implies that ρ satisfies the parity condition. □

➔Sec. 3.2, Page 53

It remains to show that the costs of requests in ρ are indeed bounded. Recall that we defined $n' = |\mathcal{A}_v|$, d' as the number of odd colors in the $\mathcal{G}_v$, and W' as the largest absolute weight of an edge. We claim that the costs of the requests posed at the beginning of the ESIs of ρ are bounded by $(n'd'W')^2$. This implies that the cost of all requests are bounded: Due to Remark 3.24 we obtain that the number of ESIs between a request and its response, if one exists, is bounded by d. Hence, it suffices to show that each ESI contributes at most a bounded amount to the cost of answering a request.

Lemma 3.26. *Let $\mu = v_j v_{j+1} v_{j+2} \cdots$ be an ESI of ρ. For each j' with $j \leq j' < j + |\mu|$ we have $\left|\mathit{Weight}(v_j \cdots v_{j'})\right| \leq d'(n'W')^2$.*

Proof. Towards a contradiction let j' be a position with $j \leq j' < j + |\mu|$, such that we have $\left|\mathrm{Weight}(v_j \cdots v_{j'})\right| > d'(n'W')^2$. We assume $\mathrm{Weight}(v_j \cdots v_{j'}) > d'(n'W')^2$, i.e., that the infix $v_j \cdots v_{j'}$ violates the claimed upper bound. The other case is dual. Since each traversed edge adds a cost of at most W', there exists a minimal position j'' such that, for all k with $j'' \leq k \leq j'$, we have $\mathrm{Weight}(v_j \cdots v_k) > 0$. Let $\pi' = (v_{j''}, p_{j''}) \cdots (v_{j'}, p_{j'})$ be an infix of ρ' such that $\mathrm{unpol}(\pi') = v_{j''} \cdots v_{j'}$. The polarity remains positive throughout π' due to the construction of h. Furthermore, since the weights of the edges in the copy of $\mathcal{A}$ with positive polarity are inverted, we have $\mathrm{Weight}_{v_j}(\pi') < -d'(n'W')^2$. Finally, by definition of ESIs, the infix π' is an infix of a play in an energy parity game that starts in $(v_j, +)$ and is consistent with a winning strategy σ_{v_j} for Player 0 from $(v_j, +)$. This, however, contradicts ➔ Lemma 3.13. □

➔Sec. 3.2, Page 46

Due to Lemma 3.26, each ESI strictly in-between a request and its response contributes at most $d'(n'W')^2$ to the cost incurred by the request. Similarly, the ESIs containing the request and its response also contribute at most $d'(n'W')^2$ each to the cost of answering the given request. Hence we obtain that each (answered or unanswered) request in ρ incurs a cost of at most $(d'n'W')^2$ via Remark 3.24. We illustrate this argument in Figure 3.10.

In that figure, the labels μ_j through μ_{j+4} indicate the ESIs of the play ρ. The solid black line denotes the weight accumulated by the play prefix starting at the initial vertex of μ_j. Lemma 3.26 yields that during no ESI, that line can deviate from the height at which it entered the current ESI by more than $\Delta = d'(n'W')^2$, as indicated by the bounds in Figure 3.10.

Hence, σ is a winning strategy for Player 0 from v^* in $\mathcal{G}$, as each play that starts in v^* and is consistent with σ satisfies the parity condition due to Lemma 3.25 and because no such play contains a request that is unanswered with infinite cost, which concludes the proof of the second item of Lemma 3.17.

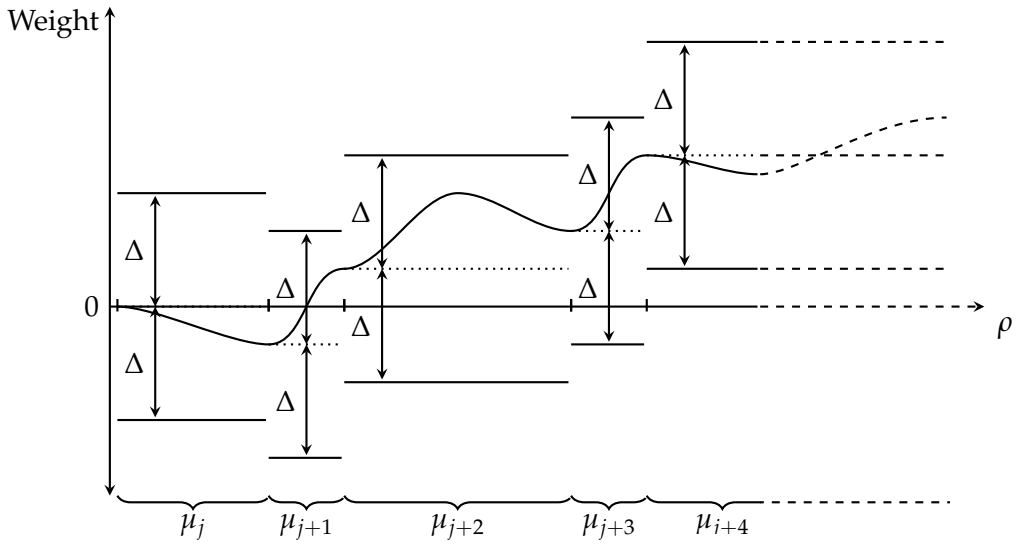


Figure 3.10: Bounds on the amplitude of an infix following a request given by Lemma 3.26. The μ_j denote the ESIs of ρ. The weight accumulated along each ESI is bounded by $\Delta = d'(n'W')^2$.

Before we conclude this section, we formalize the above observation about the winning strategy for Player 0 uniformly bounding the costs of requests in the following corollary.

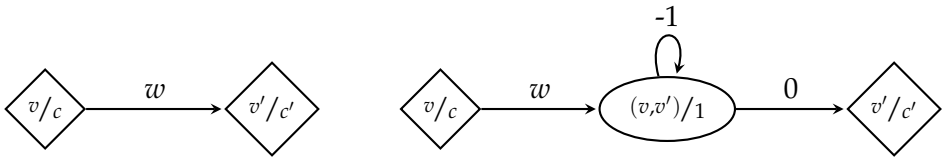
Corollary 3.27. *Let $\mathcal{G}$ be a bounded parity game with weights with n vertices, d odd colors, and largest absolute weight W. There exists a strategy σ for Player 0 that is winning from $W_0(\mathcal{G})$, such that in each play ρ that starts in $W_0(\mathcal{G})$ and is consistent with σ, each request is answered or unanswered with cost at most $((d+1)(2n+4n^2)(W+1))^2$.*

We later improve this bound in ➔Section 3.5 to $((d+1)(6n)(W+1))^2$. However, we require Corollary 3.27 in the following section to show that the above relation also holds in the opposite direction, i.e., that we are able to solve energy parity games by repeatedly solving parity games with weights. ➔Page 75

3.3 Polynomial-Time Equivalence to Energy Parity Games

In the previous sections we have shown how to solve parity games with weights via iteratively solving energy parity games with only a polynomial overhead. In this section, we consider the reduction in the other direction, i.e., whether one can also solve energy parity games by solving parity games with weights. We answer this question affirmatively, which yields PTime-equivalence of the problem of solving energy parity games and the problem of solving parity games with weights.

We first show how to transform an energy parity game into a bounded parity game with weights such that solving the latter also solves the former in Section 3.3.1. Sub-



(a) An edge with weight w in the original game $\mathcal{G}$.

(b) The associated split edge in $\mathcal{G}'$.

Figure 3.11: Splitting edges in $\mathcal{G}$ to obtain $\mathcal{G}'$.

sequently, in Section 3.3.2, we show how to solve bounded parity games with weights by iteratively solving parity games with weights. Both constructions increase the size of the arenas only linearly. Hence, all three types of games considered in this chapter are interreducible with at most polynomial overhead, making the problems of solving them polynomial-time equivalent.

3.3.1 Energy Parity Games to Bounded Parity Games with Weights

We begin by showing that the problem of solving energy parity games reduces to that of iteratively solving bounded parity games with weights. Hence, for the remainder of this section, fix some energy parity game $\mathcal{G} = (\mathcal{A}, \Omega, \text{Weight})$ with $\mathcal{A} = (V, V_0, V_1, E)$.

Recall that, in an energy parity game, Player 0 wins if the energy increases without a bound, as long as the energy level is bounded from below and as long as the resulting play satisfies the parity condition. In a bounded parity game with weights, in contrast, she has to ensure both an upper and a lower bound on the cost incurred by requests of the play in addition to the parity condition.

Thus, we show in a first step how to modify $\mathcal{G}$ such that Player 0 still has to ensure a lower bound on the energy while allowing her to "discard" unnecessary energy during each transition. The resulting game is still an energy parity game. We then show that if Player 0 wins the original game, she also wins in the modified game while ensuring an upper bound on the energy level. The proof relies on ➔Lemma 3.13, Page 46, which allows us to identify when energy becomes "unnecessary" to ensure a lower bound.

Intuitively, we subdivide every edge of $\mathcal{A}$ and add a new vertex for Player 0, where she can decrease the energy level. The odd color of the vertices thus added ensures that she eventually leaves this vertex in order to satisfy the parity condition.

W.l.o.g. we assume that the minimal color occurring in $\mathcal{G}$ is strictly greater than one. We define the energy parity game $\mathcal{G}' = (\mathcal{A}', \Omega', \text{Weight}')$ with $\mathcal{A} = (V', V_0', V_1', E')$, where

- $V' = V \cup E$, $V_0' = V_0 \cup E$, and $V_1' = V_1$, and
- $E' = \{(v, e), (e, e), (e, v') \mid e = (v, v') \in E\}$.

Moreover, we define the coloring Ω' via $\Omega'(v) = \Omega(v)$ and $\Omega'(e) = 1$, as well as the weighting Weight′ via $\text{Weight}'(v, e) = \text{Weight}(e)$, $\text{Weight}'(e, e) = -1$, and $\text{Weight}(e, v') = 0$ for every $e = (v, v') \in E$. We illustrate this construction in Figure 3.11.

We say that a strategy σ for Player 0 in $\mathcal{G}'$ is CORRIDOR-WINNING for her from some vertex $v \in V$ if there is a $b \in \mathbb{N}$ such that every play ρ that starts in v and is consistent with σ satisfies the parity condition and satisfies $\mathrm{Ampl}(\rho) \leq b$. Hence, instead of just requiring Player 0 to ensure a lower bound on the energy level as is the case in the energy parity condition, we also require her to ensure a uniform upper bound on the energy level. W.l.o.g. we assume the upper and lower bound to coincide in the remainder of this section.

Def. corridor-winning

Lemma 3.28. *Let $v \in V$. Player 0 has a winning strategy in $\mathcal{G}$ from v if and only if she has a corridor-winning strategy from v in $\mathcal{G}'$.*

Proof. We first show the direction from left to right, i.e., that Player 0 having a winning strategy in $\mathcal{G}$ implies her having a corridor-winning strategy in $\mathcal{G}'$. Since Player 0 wins $\mathcal{G}$ from v, she also has a finite-state winning strategy σ for $\mathcal{G}$ from v, due to →Proposition 3.12. Let s be the size of σ. Furthermore, there exists an initial credit c_{I} such that every play prefix π that starts in v and is consistent with σ satisfies $\mathrm{Weight}(\pi) \geq -c_{\mathrm{I}}$. Finally, define $b = Wns$, where n and W are the number of states of $\mathcal{G}$ and the largest absolute weight occurring in $\mathcal{G}$, respectively.

→Sec. 3.2, Page 46

We define a strategy σ' for Player 0 in $\mathcal{G}'$ that mimics the behavior of σ and additionally ensures that the energy level of a play prefix never exceeds $b + W$. To this end, during the simulation of each move made by σ in $\mathcal{G}'$, σ' discards energy exceeding b. By definition of W, the move from some v to some vertex of the form (v, v') in $\mathcal{G}'$ incurs an increase in energy of at most W. Hence, the energy level of any play consistent with σ' is bounded from above by b in vertices from V, while it is bounded from above by $b + W$ in vertices from E.

To formally define σ', we first define the homomorphism $f\colon (V \cup E)^* \cup (V \cup E)^\omega \to \{\epsilon\} \cup V^* \cup V^\omega$ that removes the additional vertices occurring in $\mathcal{G}'$ via $f(v) = v$ for $v \in V$ and $f(e) = \epsilon$ for $e \in E$. Now, let π' be a play prefix in $\mathcal{G}'$. If π' ends in some $v' \in V_0$, then we define $\sigma'(\pi') = (v', \sigma(f(\pi')))$. If, however, π' ends in some $e = (v', v'') \in E$, then we define $\sigma'(\pi') = v''$ if $\mathrm{Weight}'(\pi') \leq b$, and $\sigma'(\pi') = e$ otherwise. Thus, σ' prescribes using the self loop of e until the energy level of the play prefix is at most b. This implements the above intuition and completes the definition of σ'.

It remains to show that σ' is indeed corridor-winning from v for Player 0. To this end, let ρ' be a play in $\mathcal{G}'$ that starts in v and is consistent with σ'. By definition of σ', the play ρ' never remains in some self loop of an edge ad infinitum, but instead visits infinitely many vertices from V. Hence, by construction of $\mathcal{A}$, the play $\rho = f(\rho')$ is a play in $\mathcal{G}$ that starts in v. Furthermore, ρ is consistent with σ, as σ' mimics σ on vertices from V. Thus, ρ satisfies the parity condition. As the vertices removed from ρ' to obtain ρ all have color one, and as all colors occurring in $\mathcal{G}$ are greater than one by assumption, we conclude that ρ' satisfies the parity condition as well.

Recall that σ bounds the energy level along the play by $-c_{\mathrm{I}}$ from below. We now show that every prefix π' of ρ' satisfies $-c_{\mathrm{I}} \leq \mathrm{Weight}'(\pi') \leq b + W$. This then implies that σ' is indeed corridor-winning from v and thus concludes the proof of this direction of the claim. The upper bound $b + W$ is satisfied by construction of σ' as argued above. To prove the lower bound of $-c_{\mathrm{I}}$ on the energy level along ρ', we consider two cases:

Firstly, assume ρ' has no prefix whose energy level exceeds b. Since σ' mimicks the moves made by σ without ever prescribing to take a self loop in this case, we obtain $\text{Weight}'(\pi') = \text{Weight}(f(\pi'))$ for every prefix π' of ρ' ending in a vertex from V. Since edges of the form (e, v) have weight zero, since $f(\pi)$ is consistent with σ as argued above, and since σ provides a lower bound of $-c_{\mathrm{I}}$ on the energy level attained during ρ by assumption, this yields $\text{Weight}'(\pi') \geq -c_{\mathrm{I}}$.

Secondly, assume ρ' has at least one prefix whose weight exceeds b. The energy level of shorter prefixes is bounded from below by $-c_{\mathrm{I}}$ as argued above. We show that every longer prefix has nonnegative weight, which concludes the proof. Towards a contradiction, assume there is a longer suffix with negative weight. Then, there also is an infix π' of ρ' of weight strictly smaller than $-b$, during which Player 0 never uses a self-loop in $\mathcal{A}'$ to discard energy during π'. Hence, $f(\rho)$ also has an infix with weight strictly smaller than $-b$. As $f(\rho)$ is consistent with σ, however, this contradicts

→ Sec. 3.2, Page 46

→ Lemma 3.13 due to our choice of $b = Wns$, where $s = |\sigma|$, and thus concludes the proof for this direction.

We now show the other direction of the claim, i.e., that Player 0 having a corridor-winning strategy from some v in $\mathcal{G}'$ indeed implies her having a winning strategy from v in $\mathcal{G}$. To this end, let σ' be such a corridor-winning strategy for Player 0 in $\mathcal{G}'$ from v. Further, let the homomorphism $f\colon (V \cup E)^* \cup (V \cup E)^\omega \to \{\epsilon\} \cup V^* \cup V^\omega$ be defined as above.

We again define a strategy σ for Player 0 from v in $\mathcal{G}$ that is obtained by simulating play prefixes in $\mathcal{G}'$. To this end, we use a simulation function h that satisfies the following invariant:

> Let $v_0 \cdots v_j$ be a play prefix in $\mathcal{G}$ that starts in v and is consistent with σ. Then $h(v_0 \cdots v_j)$ is a play prefix in $\mathcal{G}'$ that starts in v, is consistent with σ', and ends in v_j.

We define h inductively starting with $h(v) = v$. Now, assume we have a play prefix $\pi = v_0 \cdots v_j$ in $\mathcal{G}$ that starts in v, is consistent with σ and such that $\pi' = h(\pi)$ is defined. Due to the invariant, we obtain that the play prefix π' in $\mathcal{G}'$ starts in v, is consistent with σ', and ends in v_j. If $v_j \in V_0 \subseteq V_0'$, then let $v_{j+1} \in V$ be the unique vertex such that $\pi' \cdot (v_j, v_{j+1})$ is consistent with σ'. We define $\sigma(\pi) = v_{j+1}$, which is a legal move in $\mathcal{A}$ due to the construction of $\mathcal{A}'$. This concludes the definition of σ. If $v_j \in V_1$, however, then let v_{j+1} be an arbitrary successor of v_j in $\mathcal{A}$.

In both cases it remains to define $h(\pi \cdot v_{j+1})$. As σ' is a corridor-winning strategy for Player 0 from $v = v_0$ in $\mathcal{G}'$ and due to our choice of v_{j+1}, there exists a unique m such that the play $h(\pi) \cdot (v_j, v_{j+1})^m \cdot v_{j+1}$ is consistent with σ'. We define

$$h(\pi \cdot v_{j+1}) = h(\pi) \cdot (v_j, v_{j+1})^m \cdot v_{j+1} \ ,$$

which clearly satisfies the invariant and concludes the definition of h.

It remains to show that σ is indeed winning for Player 0 from v in $\mathcal{G}$. To this end, let ρ be a play in $\mathcal{G}$ that starts in v and is consistent with σ. Moreover, let b be the uniform bound on the amplitude of plays in $\mathcal{G}'$ consistent with σ' starting in v, i.e.,

no such play has a prefix whose weight exceeds b nor $-b$. Furthermore, let ρ' be the limit of the $h(\pi)$ for increasing prefixes of ρ. By construction of h, ρ' starts in v as well and is consistent with σ'. Hence, ρ' does not remain in some self loop ad infinitum, as this would contradict σ' being corridor-winning for Player 0, but visits infinitely many vertices from V. This in turn implies $f(\rho') = \rho$. Hence, as ρ' satisfies the parity condition, ρ does so as well: The colors of the vertices from E removed by applying f are inconsequential by assumption.

It remains to show that the energy level along ρ is bounded from below. To this end, let π_j be the prefix of length j of ρ. A straightforward induction shows that the energy level of π_j is bounded from below by that of $h(\pi_j)$, as the additional edges of the latter only have nonpositive weight. Since the energy level of $h(\pi_j)$ is bounded from below by b, we conclude that σ is winning for Player 0 in $\mathcal{G}$ from v with initial credit b. □

As a second step, we now show how to use bounded parity games with weights to determine whether or not Player 0 has a corridor-winning strategy in $\mathcal{G}'$. Recall that in a bounded parity game with weights, the cost-of-response of requests has to be bounded, but the overall energy level of the play may still diverge to $-\infty$. To rule this out, we construct the bounded parity game with weights such that every play beginning in a designated vertex starts with an unanswerable request. Then, in order to satisfy the bounded parity condition with costs, Player 0 has to ensure that this request only incurs finite cost. If this is the case, then the accumulated weight along the play is in a bounded corridor, i.e., we obtain a corridor-winning strategy.

Formally, let c^* be some odd color larger than any color occurring in $\mathcal{G}'$. For every vertex $v \in V'$, we add a vertex $\overline{v}$ of color c^* to $\mathcal{A}'$. Thus, visiting $\overline{v}$ ensures that the play contains a request that can never be answered. Furthermore, $\overline{v}$ has a single outgoing edge to v of weight zero, i.e., it is irrelevant whose turn it is in that vertex. Hence, we arbitrarily give $\overline{v}$ to Player 1. We call the resulting arena $\mathcal{A}''$, the resulting coloring Ω'', and the resulting weighting Weight$''$, and define $\mathcal{G}'' = (\mathcal{A}'', \textsc{BndWeightParity}(\Omega'', \text{Weight}''))$.

Lemma 3.29. *Let $v \in V$. Player 0 has a corridor-winning strategy for $\mathcal{G}'$ from v if and only if $\overline{v} \in W_0(\mathcal{G}'')$.*

Proof. We first show the direction from left to right, i.e., if Player 0 has a corridor-winning strategy in $\mathcal{G}'$ from v, then she wins $\mathcal{G}''$ from $\overline{v}$. To this end, let σ' be such a corridor-winning strategy for her in $\mathcal{G}'$ from v. Further, let b be the corresponding uniform bound on the amplitude of plays that start in v and are consistent with σ'. We define a strategy σ'' for Player 0 in $\mathcal{G}''$ from $\overline{v}$ via $\sigma''(\overline{v}\pi) = \sigma'(\pi)$ for all play prefixes π starting in v and ending in a vertex of Player 0.

Let $\overline{v}\rho$ be a play that is consistent with σ''. By construction of $\mathcal{A}''$ and σ'', the play ρ starts in v and is consistent with σ'. Hence, it satisfies the parity condition and its amplitude is bounded by b. Thus, almost all requests in ρ are answered with cost at most $2b$ and there is no unanswered request of infinite cost. This implies that $\overline{v}\rho$ satisfies the bounded parity condition with weights. Hence, σ'' witnesses $\overline{v} \in W_0(\mathcal{G}'')$.

In order to show the direction from right to left, let σ'' be a winning strategy for Player 0 in $\mathcal{G}''$ from $\overline{v}$ and let $b \in \mathbb{N}$ be a bound such that every request in a play starting in $\overline{v}$ and consistent with σ'' is answered or unanswered with cost at most b. Due to ➔Corollary 3.27, such a strategy σ'' and such a bound b exist. We define a strategy σ' for Player 0 from v in $\mathcal{G}'$ via $\sigma'(\pi) = \sigma''(\overline{v}\pi)$ for all play prefixes π starting in v and ending in a vertex of Player 0.

➔Sec. 3.2, Page 61

Let ρ be a play starting in v that is consistent with σ'. By construction, $\overline{v}\rho$ is consistent with σ''. Hence, $\overline{v}\rho$ satisfies the parity condition and every request is answered or unanswered with cost at most b. This holds true in particular for the unanswered request posed by visiting $\overline{v}$. Hence, the amplitude of $\overline{v}\rho$ (and thus also that of ρ) is bounded by b.

Thus, ρ satisfies the parity condition and the energy level of all its prefixes is bounded by b from above and by $-b$ from below. Hence, σ' is corridor-winning from v. □

➔Page 47

Together with the results from ➔Section 3.2.3, we obtain that the problem of solving energy parity games and that of solving bounded parity games with weights are polynomial-time equivalent. We now further show that the latter problem and that of solving parity games with weights are polynomial-time equivalent as well.

3.3.2 Bounded Parity Games with Weights to Unbounded Ones

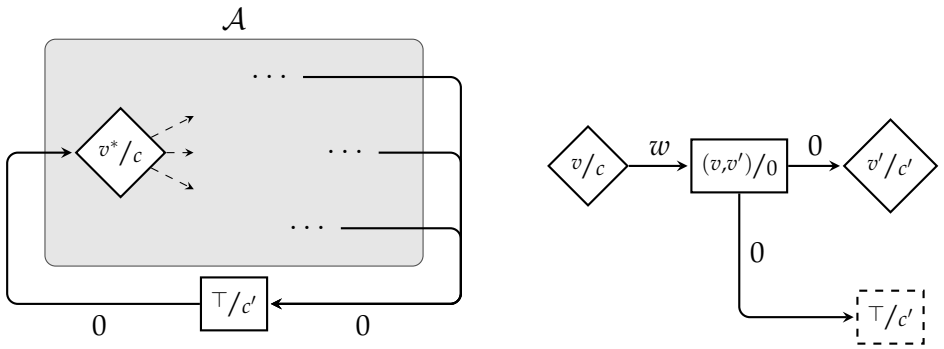
Next, we show how to solve bounded parity games with weights via repeatedly solving parity games with weights. This construction uses the same restarting mechanism that underlies the proof of ➔Lemma 3.8: Starting with $\kappa = 0$, Player 1 plays according to his winning strategy in the bounded parity game with weights until the cost incurred by some request exceeds κ. Subsequently, Player 1 may restart the play in order to enforce a request of cost $\kappa + 1$ and he may repeat this process ad infinitum. The overall structure of the algorithm, however, is different: Instead of iteratively solving parity games with weights of decreasing size, we instead construct for each vertex v^* of the bounded parity game with weights $\mathcal{G}$ an unbounded parity game with weights $\mathcal{G}'_{v^*}$ such that Player 1 wins $\mathcal{G}$ from v^* if and only if he wins $\mathcal{G}'_{v^*}$ from v^*.

➔Sec. 3.2, Page 40

To this end, we adapt the restart mechanic used in the proof of Lemma 3.8 such that, instead of being able to restart the play at any vertex and continuing from that vertex, Player 1 always returns to v^* upon restarting the play in $\mathcal{G}'_{v^*}$. During a restart, we moreover answer all requests in order to prevent Player 1 from using the reset to prevent requests from being answered. Then, we subdivide every edge in the arena of $\mathcal{G}'_{v^*}$ in order to allow Player 1 to restart the play during each simulated move.

For the remainder of this section, fix a bounded parity game with weights $\mathcal{G} = (\mathcal{A}, \textsc{BndWeightParity}(\Omega, \text{Weight}))$ with $\mathcal{A} = (V, V_0, V_1, E)$ and a vertex $v^* \in V$. We define the parity game with weights $\mathcal{G}'_{v^*} = (\mathcal{A}'_{v^*}, \textsc{WeightParity}(\Omega_{v^*}, \text{Weight}_{v^*}))$ with $\mathcal{A}'_{v^*} = (V', V'_0, V'_1, E'_{v^*})$ where

- $V' = V \cup E \cup \{\top\}$, $V'_0 = V_0$, and $V'_1 = V_1 \cup E \cup \{\top\}$, as well as
- $E'_{v^*} = \{(v, e), (e, \top), (e, v') \mid e = (v, v') \in E\} \cup \{(\top, v^*)\}$.



(a) The overall construction of $\mathcal{A}'_{v^*}$ from $\mathcal{A}$. (b) Splitting an edge in $\mathcal{A}'_{v^*}$.

Figure 3.12: The construction of $\mathcal{G}'_{v^*}$ from $\mathcal{G}$. We write $c' = 2\max(\Omega(V))$.

Moreover, we define the coloring

$$\Omega_{v^*}(v) = \begin{cases} \Omega(v) & \text{if } v \in V \ , \\ 0 & \text{if } v \in E \ , \\ 2\max(\Omega(V)) & \text{if } v = \top \ , \end{cases}$$

and the weight function

$$\text{Weight}_{v^*}(e') = \begin{cases} \text{Weight}(e) & \text{if } e' = (v,e) \in V \times E \ , \\ 0 & \text{otherwise} \ . \end{cases}$$

We illustrate this construction in Figure 3.12. Figure 3.12a shows the adaptations to the overall structure of the arena $\mathcal{A}$ of $\mathcal{G}$, while Figure 3.12b details the splitting of individual edges.

Lemma 3.30. *We have $v^* \in W_0(\mathcal{G})$ if and only if $v^* \in W_0(\mathcal{G}'_{v^*})$.*

Proof. We first show the direction from left to right, i.e., we show that $v^* \in W_0(\mathcal{G})$ implies $v^* \in W_0(\mathcal{G}'_{v^*})$. To this end, let σ be a strategy for Player 0 for $\mathcal{G}$ with the following property: There exists a $b \in \mathbb{N}$ such that every request in a play that starts in v^* and is consistent with σ is answered or unanswered with cost at most b. Since $v^* \in W_0(\mathcal{G})$, such a strategy σ exists due to ➔Corollary 3.27. ➔Sec. 3.2, Page 61

We define a winning strategy σ' for Player 0 from v^* in $\mathcal{G}'_{v^*}$ as follows: Given a play (prefix) π' in $\mathcal{A}'_{v^*}$ that does not end in vertex $\top$, let $\text{sfx}(\pi')$ be the longest suffix of π' that does not contain $\top$. Hence, if π' starts in v^*, then $\text{sfx}(\pi')$ starts in v^* as well, as v^* is the unique successor of $\top$. Further, let $f\colon (V \cup E)^* \cup (V \cup E)^\omega \to V^* \cup V^\omega$ be the homomorphism induced by $f(v) = v$ for $v \in V$ and $f(e) = \epsilon$ for $e \in E$. Now, if π' is a play (prefix) in $\mathcal{A}'_{v^*}$ that does not visit $\top$, then $f(\pi')$ is a play (prefix) in $\mathcal{A}$ of the same weight that induces the same sequence of colors (save for the occurrences of the "irrelevant" minimal color zero at the vertices from E that are deleted by f).

Let π' be a play prefix in $\mathcal{A}'_{v^*}$ that ends in a vertex $v \in V'_0 = V_0$. We define $\sigma'(\pi') = (v, \sigma(f(\text{sfx}(\pi'))))$ and show that σ' is winning for Player 0 in $\mathcal{G}_{v^*}$ from v^*. To this end, let ρ' be a play in $\mathcal{A}'_{v^*}$ starting in v^* that is consistent with σ'. We consider two cases, depending on whether or not the play ρ' visits the vertex $\top$ infinitely often.

If ρ' visits $\top$ only finitely often, then $\text{sfx}(\rho')$ is an infinite play in $\mathcal{G}'_{v^*}$ starting in v^*. By definition of f and construction of σ', the infinite play $f(\text{sfx}(\rho'))$ in $\mathcal{A}$ starts in v^* and is consistent with σ. Hence, $f(\text{sfx}(\rho'))$ satisfies the bounded parity condition with weights. Since this condition strengthens the parity condition with weights and since the latter condition is 0-extendable, we conclude that $f(\rho')$ satisfies the parity condition with weights as well. This, in turn, implies that the complete play ρ' satisfies the parity condition with weights due to the construction of $\mathcal{A}'_{v^*}$.

Now, assume that the play ρ' visits $\top$ infinitely often. Then, ρ' is of the form $\pi'_0 \top \pi'_1 \top \pi'_2 \top \cdots$, where none of the π'_j visits $\top$. Hence, by definition of $\mathcal{A}'_{v^*}$, f, and σ', each play prefix $f(\pi'_j)$ in $\mathcal{A}$ starts in v^* and is consistent with σ. Furthermore, every request in each π'_j is answered by the next visit to the vertex $\top$ at the latest, i.e., ρ' satisfies the parity condition. Thus, it suffices to show that the cost-of-response of all requests in ρ' is bounded. This follows immediately from the fact that σ only admits answered or unanswered requests of cost at most b when starting in v^* and that each $f(\pi'_j)$ starts in v^* and is consistent with σ. This property is inherited by the π'_j due to the construction of $\mathcal{A}'_{v^*}$. Thus, ρ' satisfies the parity condition with weights, i.e., σ' is indeed winning for Player 0 from v^*.

It remains to show the other direction of the claim, i.e., that $v^* \in W_0(\mathcal{G}'_{v^*})$ implies $v^* \in W_0(\mathcal{G})$. We proceed by contraposition. Due to determinacy of both $\mathcal{G}$ and $\mathcal{G}'_{v^*}$, it suffices to show that $v^* \in W_1(\mathcal{G})$ implies $v^* \in W_1(\mathcal{G}'_{v^*})$. To this end, let τ be a winning strategy for Player 1 in $\mathcal{G}$ from v^*. Furthermore, let sfx and f be defined as above.

We construct a strategy τ' for Player 1 from v^* in $\mathcal{G}'_{v^*}$ that is controlled by a counter κ, which is initialized with zero, and which is incremented during a play every time the costs of some request exceed the current value of κ. Every time κ is updated, the strategy τ' prescribes moving to $\top$ and subsequently to v^*, thus "restarting" the $\mathcal{G}'_{v^*}$.

Let π' be a play prefix in $\mathcal{A}'_{v^*}$ that ends in a vertex of Player 1. In order to define $\tau'(\pi')$ we consider several cases depending on the last vertex of π'. If π' ends in $\top$, we define $\tau'(\pi') = v^*$, which is the unique successor of $\top$.

If π' ends in $v \in V_1 \subseteq V'_1$, then we define $\tau'(\pi') = (v, \tau(f(\text{sfx}(\pi'))))$, i.e., we discard the prefix up to and including the last occurrence of $\top$ and mimick the decision of τ given the remaining suffix of π'. Finally, if π' ends in $e = (v, v') \in E \subseteq V'_1$, then we consider two cases. Let κ be the current counter value. If $\text{sfx}(\pi')$ contains a request such that the suffix of π' that starts at this request has an amplitude exceeding κ, then we define $\tau'(\pi') = \top$ and increment κ. Otherwise, we define $\tau'(\pi') = v'$ and leave κ unchanged.

It remains to show that τ' is winning in $\mathcal{G}'_{v^*}$ from v^*. To this end, let ρ' be a play in $\mathcal{G}'_{v^*}$ that starts in v^* and that is consistent with τ'. If ρ' visits $\top$ infinitely often, then all requests in ρ' are answered, but ρ' contains, for every $b \in \mathbb{N}$, a (different) request

that is answered with cost at least b. Hence, the costs of responses along ρ' diverge, causing ρ' to violate the parity condition with costs.

Finally, if ρ' visits $\top$ only finitely often, then the counter κ is incremented only finitely often. Similarly to the previous case, $f(\mathrm{sfx}(\rho'))$ is a play in $\mathcal{A}$ that starts in v^* and is consistent with τ. Let b be the final value of κ. Every request in $\mathrm{sfx}(\rho')$ is answered or unanswered with cost at most b, as otherwise the strategy τ' would prescribe another move to $\top$. As $\mathrm{sfx}(\rho')$ and $f(\mathrm{sfx}(\rho'))$ have essentially the same evolution of the weights (save for the removed edges of weight zero) and the same color sequence (save for the removed vertices of color zero), every request in $f(\mathrm{sfx}(\rho'))$ is answered or unanswered with cost at most b. Since $f(\mathrm{sfx}(\rho'))$ is, however, consistent with τ, it violates the bounded parity condition with weights. As all requests are answered or unanswered with cost at most b, this is only possible by violating the parity condition. Hence $f(\mathrm{sfx}(\rho'))$ violates the parity condition. This implies that both $\mathrm{sfx}(\rho')$ and ρ' violate the parity condition as well due to construction of f and $\mathcal{A}'_{v^*}$ and the prefix-independence of that condition. Since the parity condition with weights strengthens the parity condition, ρ' violates the former condition as well.

In both cases, ρ' is winning for Player 1, i.e., τ' is indeed winning for Player 1 in $\mathcal{G}'_{v^*}$ from v^*. □

Thus, we have reduced the problem of solving energy parity games first to the problem of solving bounded parity games with weights, which we have then further reduced to the problem of solving parity games with weights. All these reductions are polynomial. Combined with the results from ➔Section 3.2, we obtain that all three problems are polynomial-time equivalent. ➔Page 37

Theorem 3.31. *The following three decision problems are polynomial-time equivalent:*

> *"Given a parity game with weights $\mathcal{G}$ and a vertex v of $\mathcal{G}$, does Player 0 have a winning strategy from v in $\mathcal{G}$?"*
>
> *"Given an energy parity game $\mathcal{G}$ and a vertex v of $\mathcal{G}$, does Player 0 have a winning strategy from v in $\mathcal{G}$?"*
>
> *"Given a bounded parity game with weights $\mathcal{G}$ and a vertex v of $\mathcal{G}$, does Player 0 have a winning strategy from v in $\mathcal{G}$?"*

Proof. Polynomial-time equivalence of the first two problems follows directly from the results presented in ➔Section 3.2.1, Lemma 3.28, and Lemma 3.29. Analogously, polynomial-time equivalence of the latter two problems follows from the results presented in ➔Section 3.2.3 and Lemma 3.30. □ ➔Page 38 ➔Page 47

Via the above theorem we moreover obtain that the problem of solving parity games with weights is strongly connected to the problem of solving another important class of games as well, so-called mean-payoff parity games as introduced by Chatterjee, Henzinger, and Jurdziński [CHJ05].

A mean-payoff parity game is played on a colored arena with weights. In addition to satisfying the parity condition induced by the coloring, it is furthermore the task of Player 0 to ensure that the average weight of the traversed edges is nonnegative. Formally, given an arena with vertex set V and set of edges E, a coloring Ω of V, and a weight function Weight over E, the MEAN-PAYOFF PARITY CONDITION is defined as

Def. mean-payoff parity condition

$$\textsc{MeanPayoffParity}(\Omega, \text{Weight}) = \left\{ v_0 v_1 v_2 \cdots \in V^\omega \mid \liminf_{j \to \infty} \frac{1}{j} \text{Weight}(v_0 \cdots v_j) \geq 0 \right\} \cap \textsc{Parity}(\Omega) \ .$$

A game $(\mathcal{A}, \textsc{MeanPayoffParity}(\Omega, \text{Weight}))$ is called a MEAN-PAYOFF PARITY GAME.

Def. mean-payoff parity game

Chatterjee and Doyen [CD12] showed that energy parity games and mean-payoff parity games can be transformed into each other.

Proposition 3.32 ([CD12]). *Let $\mathcal{G} = (\mathcal{A}, \textsc{MeanPayoffParity}(\Omega, \mathit{Weight}))$ be a mean-payoff parity game with n vertices and let $\mathcal{G}' = (\mathcal{A}, \textsc{EnergyParity}_\Omega(\mathit{Weight}'))$, where $\mathit{Weight}'(e) = \mathit{Weight}(e) + \frac{1}{1+n}$ for all edges e of $\mathcal{A}$. Then $W_0(\mathcal{G}) = W_0(\mathcal{G}')$.*

Due to Proposition 3.32, the problem of solving energy parity games and the problem of solving mean-payoff parity games are logarithmic-space-equivalent. Hence, we can use the techniques underlying ➔ Theorem 3.18 to reduce the problem of solving parity games with weights to the problem of solving mean-payoff parity games. In fact, it is Proposition 3.32 that allows us to leverage the pseudo-quasi-polynomial algorithm for solving mean-payoff parity games due to Daviaud, Jurdziński, and Lazić [DJL18] to solve parity games with weights in pseudo-quasi polynomial time.

➔ Sec. 3.2, Page 52

Corollary 3.33. *The following decision problems are polynomial-time equivalent:*

> *"Given a parity game with weights $\mathcal{G}$ and a vertex v of $\mathcal{G}$, does Player 0 have a winning strategy from v in $\mathcal{G}$?"*
>
> *"Given a mean-payoff parity game $\mathcal{G}$ and a vertex v of $\mathcal{G}$, does Player 0 have a winning strategy from v in $\mathcal{G}$?"*

Solving parity games with weights by iteratively solving mean-payoff parity games as sketched above, however, does not allow us to obtain the memory bounds from ➔ Theorem 3.34. This is due to Player 0, in general, requiring infinite memory in order to win a mean-payoff parity game [CHJ05].

➔ Sec. 3.4, Page 71

3.4 Memory Requirements

In the previous sections we have shown how to solve parity games with weights and determined the computational complexity of this problem. We now turn our attention to the memory required by both players to implement their respective winning strategies.

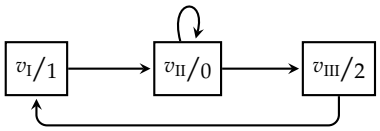


Figure 3.13: Player 1 wins this finitary parity game from every vertex, but he has no finite-state winning strategy from any vertex.

Recall that we use binary encoding to denote weights, i.e., the size of a game $\mathcal{G}$ only grows logarithmically in the largest absolute weight W occurring in $\mathcal{G}$. Moreover, recall that we denote the number of vertices of $\mathcal{G}$ by n and the number of odd colors of $\mathcal{G}$ by d. In this section we show polynomial bounds in n, d, and W on the necessary and sufficient memory for Player 0 to win parity games with weights. Due to the binary encoding of weights, these bounds are exponential in the size of the game. Player 1, in contrast, requires infinite memory, since parity games with weights subsume finitary parity games.

Theorem 3.34. *Let $\mathcal{G}$ be a parity game with weights with n vertices, d odd colors, and largest absolute weight W. Moreover, let v be a vertex of $\mathcal{G}$.*

1. *Player 0 has a strategy σ that is winning from each vertex in $W_0(\mathcal{G})$ with $|\sigma| \in \mathcal{O}(nd^2W)$.*
2. *Player 1 requires, in general, infinite memory to win from $W_1(\mathcal{G})$.*

Furthermore, for each $n > 0$ and each $W \geq 0$ there exists a parity game with weights $\mathcal{G}_{n,W}$ with $|\mathcal{G}_{n,W}| \in \mathcal{O}(n \log W)$ such that Player 0 wins $\mathcal{G}_{n,W}$ from every vertex, but she requires a strategy of size $\Omega(nW)$ to do so.

The proof of the second item of Theorem 3.34 follows directly from the necessity of infinite memory for winning strategies of Player 1 in finitary parity games due to Chatterjee and Henzinger [CH06], i.e., from ➔Proposition 2.25.2. Since parity games with weights subsume finitary parity games, this result carries over to our setting. The game shown in ➔Figure 2.6 witnesses this lower bound. We reprint that game in Figure 3.13 for the sake of completeness.

➔Sec. 2.4, Page 23
➔Sec. 2.4, Page 22

Remark 3.35. *There exists a parity game with weights $\mathcal{G}$ such that Player 1 has a winning strategy from each vertex v in $\mathcal{G}$, but he has no finite-state winning strategy from any v in $\mathcal{G}$.*

Having argued that no finite upper bound on the space requirements of winning strategies for Player 1 exists, we now show that, in contrast, exponential memory is sufficient, but also necessary, for Player 0 to implement a winning strategy. To this end, we first prove that the winning strategy for her in a bounded parity game with weights constructed in the proof of the second item of ➔Lemma 3.17 suffers at most a linear blowup in comparison to his winning strategies in the underlying energy parity games. The desired upper bound on the size of winning strategies for Player 0 in parity games with weights then follows directly from this, since

➔Sec. 3.2, Page 51

- Chatterjee and Doyen [CD12] showed that Player 0 has winning strategies of linear size in the number of vertices, the number of colors, and the largest absolute weight in energy parity games (cf. → Proposition 3.12), and since
- we can reuse memory states in order to implement winning strategies in parity games with weights, given winning strategies in the underlying bounded parity games with weights (cf. → Corollary 3.10).

→ Sec. 3.2, Page 46

→ Sec. 3.2, Page 44

Lemma 3.36. *Let $\mathcal{G}$ be a bounded parity game with weights with n vertices, d odd colors, and largest absolute weight W. Player 0 has a finite-state strategy of size at most $d(6n)(d+2)(W+1)$ that is winning from each vertex in $W_0(\mathcal{G})$.*

Proof. Let V and E be the vertex set and the set of edges of $\mathcal{G}$, respectively, and recall that we have defined $P = \{+, -\}$. Recall that in the proof of Lemma 3.17 we have constructed an energy parity game $\mathcal{G}_v$ with vertices $(V \times P) \cup (E \times P \times \{0,1\})$ for each vertex $v \in V$. We have then constructed a winning strategy σ for Player 0 for $\mathcal{G}$ by "stitching together" winning strategies for her in the $\mathcal{G}_v$. As only one of the constituent strategies is simulated at any given time, it is straightforward to implement σ via the disjoint union of memory structures implementing the constituent strategies. Hence, this approach yields an upper bound of $n(2n + 4n^2)(d+2)(W+1)$ on the size of σ due to the upper bound on the size of winning strategies for Player 0 in energy parity games stated in Proposition 3.12.

In the construction of the $\mathcal{G}_v$, however, we only store the edges chosen by the players in the vertices of the form $E \times P \times \{0,1\}$ for didactic purposes. In fact, it suffices to store the target vertex of an edge instead, resulting in each $\mathcal{G}_v$ only containing $6n$ vertices. Moreover, recall that the definition of the $\mathcal{G}_v$ only takes the color of v into account: If two vertices v and v' have the same color, then the games $\mathcal{G}_v$ and $\mathcal{G}_{v'}$ are isomorphic. Further, Chatterjee and Doyen have shown that, if Player 0 wins an energy parity game $\mathcal{G}'$ with n' vertices, d' colors, and largest absolute weight W', then she has a uniform strategy of size $n'd'W'$ that is winning from all vertices from which she wins $\mathcal{G}'$ [CD12]. Hence, it suffices to combine at most d strategies, each of size $(6n)(d+2)(W+1)$, in order to obtain a winning strategy for Player 0 in $\mathcal{G}$. □

Having established an upper bound on the memory required by Player 0, we now proceed to show a polynomial lower bound (in the number of vertices and the largest absolute weight) on the memory required by Player 0 to implement a winning strategy.

Lemma 3.37. *For each $n > 0$ and each $W \geq 0$ there exists a parity game with weights $\mathcal{G}_{n,W}$ with $|\mathcal{G}_{n,W}| \in \mathcal{O}(n \log W)$ such that Player 0 wins $\mathcal{G}_{n,W}$ from every vertex, but each winning strategy for her is of size at least $nW + 1$.*

Proof. We show the game $\mathcal{G}_{n,W}$ in Figure 3.14. This game has $n+4$ vertices and the largest absolute weight of an edge is W. Hence, we have $|\mathcal{G}_{n,W}| = (n+4)\log W \in \mathcal{O}(n \log W)$. Each vertex of $\mathcal{G}_{n,W}$ save for $v_{\text{del}} \in V_0$ and $v'_{\text{ans}} \in V_1$ only has a single successor. Thus, the only choice of Player 0 in $\mathcal{G}_{n,W}$ consists in determining how often to take the self-loop of vertex v_{del} upon each visit. Dually, the only choice of Player 1 consists of deciding whether or not to move from v'_{ans} to $v'_{\text{req},1}$, or to continue to v_{ans}.

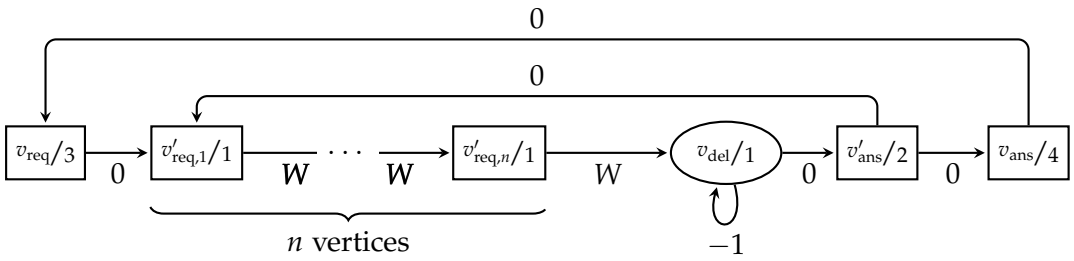


Figure 3.14: A game of size $\mathcal{O}(n \log W)$ in which Player 0 only wins with strategies of size at least $nW + 1$.

Player 0 wins $\mathcal{G}_{n,W}$ from each vertex by taking the self-loop of v_{del} nW times upon each visit to v_{del} and by subsequently moving to v'_{ans}. Each request in each play that is consistent with this strategy is answered or unanswered with cost at most nW, independently of the choices of Player 1 in v'_{ans}. Moreover, as the only way to visit v_{req} is to move there from v_{ans}, the play visits v_{ans} infinitely often if and only if it visits v_{req} infinitely often. Further, the play visits $v'_{\text{req},1}$ and v'_{ans} infinitely often. Hence, almost all requests are answered, i.e., this strategy is winning for Player 0 from all vertices. This strategy can be implemented by a counter that counts the number of self-loops of v_{del} taken so far, which is reset to zero upon leaving v_{del}. As this counter is bounded by nW, the strategy is of size $nW + 1$.

It remains to show that each finite-state winning strategy for Player 0 has at least $nW + 1$ memory states. Towards a contradiction, let σ be a winning strategy for Player 0 from some vertex v with less than $nW + 1$ memory states and let ρ be a play that starts in v and that is consistent with σ. We implement a strategy for Player 1 using a counter κ that is initialized with one if $v = v_{\text{req}}$ and with zero otherwise. Upon each visit to v_{req} we increment κ. After each visit to v_{req}, the strategy τ prescribes moving from v'_{ans} to $v'_{\text{req},1}$ for the first κ visits to v'_{ans}, and it prescribes moving from v'_{ans} to v_{ans} at the $\kappa + 1$-th visit to v'_{ans}. Hence, after the $\kappa + 1$-th visit to v'_{ans}, the vertex v_{req} is visited again, κ is incremented and the behavior of τ described above repeats with incremented κ.

Let ρ be a play consistent with σ and τ. Since σ is winning for Player 0, the play ρ does not remain in v_{del} from some point onwards ad infinitum, as this would violate the parity condition and thus contradict σ being winning from v for Player 0. Hence, playing consistently with τ, Player 1 enforces a play that starts with a (possibly empty) finite prefix that ends before the first visit to v_{req} and that continues with infinitely many rounds, each starting in v_{req}. The j-th round is of the form

$$v_{\text{req}} \cdot \Pi_{k=0,\dots,j}(v'_{\text{req},1} \cdots v'_{\text{req},n} \cdot (v_{\text{del}})^{\ell_{j,k}} \cdot v'_{\text{ans}}) \cdot v_{\text{ans}} \ .$$

We first show $\ell_{j,k} < nW + 1$ for all j, k. Towards a contradiction, assume $\ell_{j,k} \geq nW + 1$ for some $j, k \in \mathbb{N}$. Since σ is of size less than $nW + 1$, a straightforward pumping argument shows that the play ρ' consisting of the finite prefix of ρ ending

before the first visit to v_{req} concatenated with the first $j-1$ rounds of ρ, but ending with the infinite suffix

$$v_{\text{req}} \cdot \Pi_{k'=0,\dots,k-1}(v'_{\text{req},1} \cdots v'_{\text{req},n} \cdot (v_{\text{del}})^{\ell_{j,k'}} \cdot v'_{\text{ans}}) \cdot v'_{\text{req},1} \cdots v'_{\text{req},n} \cdot (v_{\text{del}})^{\omega}$$

is consistent with σ. This, however, contradicts σ being a winning strategy for Player 0 from v as v_{del} has the odd color one, i.e., as the resulting play violates the parity condition. Hence, we obtain $\ell_{j,k} < nW + 1$ for all j, k, which, in turn, yields $\text{Weight}((v_{\text{del}})^{\ell_{j,k}}) < -nW$.

Since each edge $(v'_{\text{req},n'}, v'_{\text{req},n'+1})$ for $1 \leq n' < n$ as well as the edge $(v'_{\text{req},n}, v'_{\text{del}})$ has weight W, we obtain

$$\text{Weight}(v'_{\text{req},1} \cdots v'_{\text{req},n} \cdot (v_{\text{del}})^{\ell_{j,k}} \cdot v'_{\text{ans}}) > 0$$

for all j, k. This, in turn, implies

$$\text{Weight}(v_{\text{req}} \cdot \Pi_{k=0,\dots,j}(v'_{\text{req},1} \cdots v'_{\text{req},n} \cdot (v_{\text{del}})^{\ell_{j,k'}} \cdot v'_{\text{ans}})) \geq j+1$$

for each j. Since, as argued above, the play ρ consistent with σ consists of infinitely many rounds, we obtain that for each $b \in \mathbb{N}$ there exist infinitely many requests in ρ that are answered with cost at least b. Hence, the costs-of-requests along ρ diverge, which contradicts σ being a winning strategy for Player 0. □

This concludes the study of memory requirements for both players in parity games with weights. For Player 0, the results from Theorem 3.34 also hold true for bounded parity games with weights: Lemma 3.36 directly yields an upper bound on the memory required by Player 0 in order to win in bounded parity games with weights, while it is easy to see that Player 0 also wins the games constructed in the proof of Lemma 3.37 when interpreting them as bounded parity games with weights, but only using a strategy of size $nW + 1$.

For Player 1, however, these results do not directly carry over, as he has, in fact, a positional winning strategy for the game witnessing necessity of infinite memory for him in parity games with weights shown in Figure 3.13. Recall that, in the proof of the first item of ➔Lemma 3.17, for a given parity game with weights $\mathcal{G}$, we construct a strategy τ that is winning for Player 1 from a vertex v out of a winning strategy τ_v for him in an induced energy parity game $\mathcal{G}_v$ from a designated vertex v'.

➔Sec. 3.2, Page 51

While τ_v can be assumed to be positional as shown by Chatterjee and Doyen [CD12] (cf. ➔Proposition 3.11), the strategy τ keeps track of play prefixes in $\mathcal{G}_v$ and thus requires potentially infinitely many memory states. In particular, in order to win $\mathcal{G}_v$ from v', recall that it may be necessary to switch between two copies of the arena of $\mathcal{G}$. Whether or not to perform this switch is governed by the accumulated weight of the play prefix in $\mathcal{G}$ thus far.

➔Sec. 3.2, Page 45

Hence, our construction from the proof of that lemma does not directly allow us to obtain positional or finite-state winning strategies for Player 1 in bounded parity games with weights. It remains open whether Player 1 requires infinite memory to win

in bounded parity games with weights. Since she, however, has finite-state winning strategies in the special case of bounded parity games with costs as shown by Fijalkow and Zimmermann [FZ14] we conjecture that she requires at most finite memory for winning strategies in bounded parity games with weights as well.

We now turn our attention to the quantitative properties of this winning condition. To this end, we provide tight bounds on the costs of requests that Player 0 can guarantee.

3.5 Bounds on Cost

We have shown in the previous section that finite-state strategies of exponential size suffice for Player 0 to win parity games with weights, while Player 1 in general requires infinite memory. As we are dealing with a quantitative winning condition, however, we are not only interested in the size of winning strategies, but also in their quality. More precisely, we are interested in an upper bound on the cost of requests that Player 0 can ensure.

In this section, we show that if Player 0 wins a game with n vertices, d odd colors, and largest absolute weight W, then she can ensure that almost all requests are answered with at most polynomial cost in n, d, and W, i.e., exponential cost in the size of the game. Moreover, we provide an example witnessing a polynomial bound in n and W.

The dual question is vacuous: In order to win a parity game with costs, Player 1 may have to cause the costs-of-response to diverge. Hence, there does not exist, in general, a finite lower bound on the costs-of-response of requests in plays consistent with a winning strategy for Player 1.

Theorem 3.38. *Let $\mathcal{G}$ be a parity game with weights with n vertices, d odd colors, and largest absolute weight W.*

There exists some $b \in \mathcal{O}((ndW)^2)$ and a strategy σ for Player 0 such that, for all plays ρ beginning in $W_0(\mathcal{G})$ and consistent with σ, we have $Cost(\rho) \leq b$.

Furthermore, for each $n > 0$ and each $W \geq 0$ there exists a parity game with weights $\mathcal{G}_{n,W}$ with $|\mathcal{G}_{n,W}| \in \mathcal{O}(n \log W)$ such that Player 0 wins $\mathcal{G}_{n,W}$ from each vertex, but can only enforce a cost of $\Omega(nW)$.

We first show that Player 0 can indeed ensure the upper bound as stated in Theorem 3.38 from her winning region in parity games with weights. We obtain this bound via a pumping argument leveraging the upper bound on the size of winning strategies obtained in ➔ Lemma 3.36.

➔ Sec. 3.4, Page 72

Lemma 3.39. *Let $\mathcal{G}$, n, d, and W be as in the statement of Theorem 3.38 and define $s = d(6n)(d+1)(W+1)$. Player 0 has a strategy σ such that, for each play ρ that starts in $W_0(\mathcal{G})$ and is consistent with σ, we have $Cost(\rho) \leq nsW$.*

Proof. Let σ be a winning strategy for Player 0 in $\mathcal{G}$ from $W_0(\mathcal{G})$ of size at most s. Such a strategy exists due to Lemma 3.36. Moreover, let $\rho = v_0 v_1 v_2 \cdots$ be a play that starts in $W_0(\mathcal{G})$ and is consistent with σ.

Def. sumptuous

We call a position $j \in \mathbb{N}$ of ρ SUMPTUOUS if $nsW < \mathrm{Cor}(\rho, j) < \infty$. Each sumptuous position j has some odd color c, and the request for c posed by visiting v_j is eventually answered due to $\mathrm{Cor}(\rho, j) < \infty$.

We show that ρ only contains finitely many sumptuous positions. Since ρ is consistent with the winning strategy σ for Player 0, almost all requests in ρ are eventually answered, i.e., there exist only finitely many positions j with $\mathrm{Cor}(\rho, j) = \infty$. Thus, ρ containing only finitely many sumptuous positions directly implies the desired statement.

Towards a contradiction, assume that there exist infinitely many sumptuous positions in ρ. We show that then there exists a play that is consistent with σ but which has diverging costs-of-responses, contradicting σ being winning for Player 0. To this end, we employ a pumping argument.

We define a sequence of positions that begins with the first sumptuous position j_0. Let j_0' be the minimal position that satisfies $\Omega(v_{j_0'}) \in \mathrm{Ans}(\Omega(v_{j_0}))$. This position exists since $\mathrm{Cor}(\rho, j_0) < \infty$ due to the definition of sumptuous positions. We continue by defining j_1 as the smallest sumptuous position greater than j_0' and by defining j_1' as the minimal position that satisfies $\Omega(v_{j_1'}) \in \mathrm{Ans}(\Omega(v_{j_1}))$. Continuing in this manner, we obtain a sequence $j_0 < j_0' < j_1 < j_1' < j_2 < j_2' < \cdots$, where j_0 is the first sumptuous position of ρ, each j_k' for $k \geq 0$ is the minimal position that satisfies both $j_k' > j_k$ and $\Omega(v_{j_k'}) \in \mathrm{Ans}(\Omega(v_{j_k}))$, and each j_k for $k > 0$ is the smallest sumptuous position greater than j_{k-1}'. Since there exist infinitely many sumptuous positions by assumption and since each request posed at a sumptuous position is answered by definition, the sequence $j_0 < j_0' < j_1 < j_1' < j_2 < j_2' < \cdots$ is indeed infinite.

Due to the definition of sumptuous positions and the j_k', we have $\mathrm{Ampl}(v_{j_k} \cdots v_{j_k'}) > nsW$ for each $k \in \mathbb{N}$. Since ρ is consistent with the finite-state strategy σ of size s, we claim that in each such $v_{j_k} \cdots v_{j_k'}$ there exists an infix that can be repeated arbitrarily often while retaining consistency with σ. To identify such infixes, we partition the sumptuous positions j_k: We call a position j_k POSITIVELY SUMPTUOUS if there exists a j' with $j_k \leq j' \leq j_k'$ such that $\mathrm{Weight}(v_{j_k} \cdots v_{j'}) > nsW$ and NEGATIVELY SUMPTUOUS otherwise.

Def. positively sumptuous

Def. negatively sumptuous

Let σ be implemented by the memory structure $(M, init, upd)$. As each edge contributes cost at most W to $\mathrm{Ampl}(v_{j_k} \cdots v_{j_k'})$, for each $k \in \mathbb{N}$ there exist positions ℓ_k and ℓ_k' with $j_k < \ell_k < \ell_k' < j_k'$ such that

- $v_{\ell_k} = v_{\ell_k'}$,
- $upd^+(init(v_0), v_0 \cdots v_{\ell_k}) = upd^+(init(v_0), v_0 \cdots v_{\ell_k'})$,
- $\mathrm{Weight}(v_{\ell_k} \cdots v_{\ell_k'-1}) > 0$, if j_k is positively sumptuous, and such that
- $\mathrm{Weight}(v_{\ell_k} \cdots v_{\ell_k'-1}) < 0$, if j_k is negatively sumptuous.

We illustrate the requirements towards the j_k, the j_k', the ℓ_k, and the ℓ_k' in Figure 3.15.

The positions j_k, ℓ_k, ℓ_k', and j_k' split ρ into infinitely many infixes

$$\rho = \pi \cdot \Pi_{k=0,1,2,\ldots} \pi_{k,\mathrm{I}} \cdot \pi_{k,\mathrm{II}} \cdot \pi_{k,\mathrm{III}} \cdot \pi_{k,\mathrm{IV}} \ ,$$

where π is some prefix of ρ and $\pi_{k,\mathrm{I}}$, $\pi_{k,\mathrm{II}}$, $\pi_{k,\mathrm{III}}$, and $\pi_{k,\mathrm{IV}}$ start at j_k, ℓ_k, ℓ_k', and j_k',

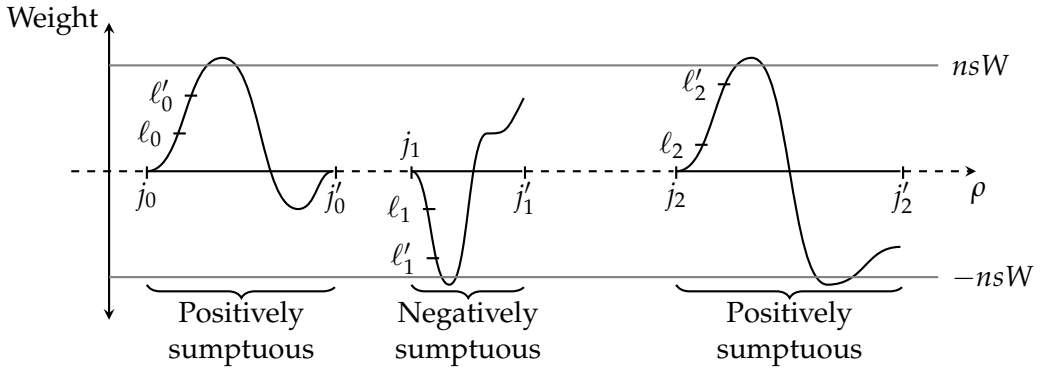


Figure 3.15: Sumptuous positions along some play ρ.

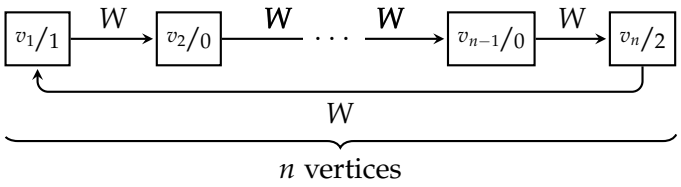


Figure 3.16: The game $\mathcal{G}_n$ witnessing an exponential lower bound on the cost that Player 0 can ensure.

respectively. Due to the definition of ℓ_k and ℓ'_k, the play

$$\rho' = \pi \cdot \Pi_{k=0,1,2,\ldots} \pi_{k,\mathrm{I}} \cdot (\pi_{k,\mathrm{II}})^k \cdot \pi_{k,\mathrm{III}} \cdot \pi_{k,\mathrm{IV}}$$

is consistent with σ. The costs-of-response of the requests opened by visiting the v_{j_k}, however, diverge due to $|\mathrm{Weight}(\pi_{k,\mathrm{II}})| = \left|\mathrm{Weight}(v_{\ell_k} \cdots v_{\ell'_k - 1})\right| > 0$. Hence, ρ' begins in the initial vertex of ρ and is consistent with σ, but violates the parity condition with weights, which contradicts σ being a winning strategy from the initial vertex of ρ. This yields the desired statement, as argued above. □

Having thus shown that Player 0 can indeed ensure an upper bound on the incurred cost that is polynomial in n, d, and W, i.e., exponential in the size of the game, we now proceed to show a polynomial (in n and W) lower bound. This is witnessed by a sequence of games $\mathcal{G}_n$ of size linear in both n and $\log W$, in which Player 0 wins from every vertex, but in which she cannot enforce a cost smaller than $(n-1)W$.

Lemma 3.40. *For each $n > 0$ and each $W \geq 0$ there exists a parity game with weights $\mathcal{G}_{n,W}$ with $|\mathcal{G}_{n,W}| \in \mathcal{O}(n \log W)$ containing a vertex v such that Player 0 wins $\mathcal{G}_{n,W}$ from v, but for each winning strategy for Player 0 from v there exists a play ρ starting in v and consistent with σ with $\mathit{Cost}(\rho) \geq (n-1)W$.*

Proof. We show the game $\mathcal{G}_{n,W}$ in Figure 3.16. The arena of $\mathcal{G}_{n,W}$ is a cycle with n vertices of Player 1, where each edge has weight W. Since the weights are encoded

in binary, the game $\mathcal{G}_{n,W}$ is of linear size in both n and $\log W$. Moreover, all vertices are labeled with color zero, save for one (arbitrary) vertex of color two and its directly succeeding vertex of color one.

Player 0 only has a single strategy in this game and there exist only n plays in $\mathcal{G}_{n,W}$, each starting in a different vertex of $\mathcal{G}_{n,W}$. In each play, each request for color one is only answered after $n-1$ steps, each contributing a cost of W. Hence, each request incurs a cost of $(n-1)W$. Moreover, as the request for color one is posed and answered infinitely often in each play, we obtain the desired result. □

While Player 0 is able to bound the costs from above by $d(6n^2)(d+1)W(W+1) \in \mathcal{O}((ndW)^2)$ due to Lemma 3.39, we only obtain a sequence of examples witnessing a lower bound of $(n-1)W \in \Theta(nW)$ use on these costs due to Lemma 3.40. Thus, while both the upper and the lower bound are exponential in the size of the game, there remains a polynomial gap between the two bounds.

The upper bound on the costs is due to a pumping argument leveraging the upper bound on the memory Player 0 requires to implement a winning strategy due to ➔Lemma 3.36. We have a lower bound of $(n-4)W \in \Theta(nW)$ on the memory required by Player 0 to implement a winning strategy due to ➔Lemma 3.37. Thus, even if we are able to improve the upper bounds on the memory requirements for Player 0, the pumping argument leveraged in the proof of Lemma 3.39 can only improve the upper bound on the cost incurred by Player 0 to $d(n-4)W^2 \in \mathcal{O}(ndW^2)$. Hence, the proof methods used in this section do not enable us to completely close this gap between the upper and the lower bound on the cost Player 0 is able to enforce.

➔Sec. 3.4, Page 72
➔Sec. 3.4, Page 72

All bounds obtained in this section also hold for bounded parity games with weights: The upper bound from Lemma 3.39 holds for bounded parity games with weights since the bounded parity condition with weights strengthens its unbounded variant. Analogously, the games constructed for the proof of the lower bound on the incurred cost in Lemma 3.40 yield the same lower bound for bounded parity games with weights.

This concludes our study of parity games with weights and their properties for the boundedness case, i.e., if we just require Player 0 to bound the costs-of-response from above. We summarize our findings in the following section.

3.6 Summary of Results

In this chapter we have introduced parity games with weights, an extension of parity games with costs [FZ14] that lifts the restriction to nonnegative costs and allows for integer weights. After lifting this restriction, we have extended the concepts and notions of parity games with costs to our setting in ➔Section 3.1 before showing that solving parity games with weights is in NP ∩ coNP in ➔Section 3.2. Our proof of NP ∩ coNP-membership of the problem also yields an algorithm for solving them in pseudo-quasi-polynomial time due to Daviaud, Jurdziński, and Lazić [DJL18], as observed by them. Subsequently, we have provided a lower bound of energy-parity-hardness on

➔Page 31
➔Page 37

	Complexity	Mem. Pl. 0/Pl. 1	Bounds
Parity Games	UP ∩ coUP quasi-poly.	pos./pos.	–
Energy Parity Games	NP ∩ coNP pseudo-quasi-poly.	$\mathcal{O}(ndW)$/pos.	$\mathcal{O}(nW)$
Finitary Parity Games	PTime	pos./inf.	$\mathcal{O}(nW)$
Parity Games with Costs	UP ∩ coUP quasi-poly.	pos./inf.	$\mathcal{O}(nW)$
Parity Games with Weights	NP ∩ coNP pseudo-quasi-poly.	$\mathcal{O}(nd^2W)$/inf.	$\mathcal{O}((ndW)^2)$

Table 3.17: Characteristic properties of variants of parity games.

the complexity of solving parity games with weights in ➔Section 3.3 and discussed the memory requirements of both players as well as the quality of plays that Player 0 can ensure in ➔Section 3.4 and ➔Section 3.5, respectively. We summarize our results and compare them to the properties of existing quantitative variants of parity games in Table 3.17.

➔Page 61
➔Page 70
➔Page 75

Throughout this chapter, we have assumed the weighting to be given in binary encoding, resulting in the size of a parity game with weights $\mathcal{G}$ being logarithmic in the largest absolute weight occurring in $\mathcal{G}$. All results obtained in this chapter, however, still hold true if the weight is given in unary encoding. In this case, subdividing the edges such that each edge has absolute weight at most one only incurs a polynomial blowup in the size of the game. Hence, we are able to discuss the complexity of solving parity games with weights given in unary encoding by discussing such games in which the largest absolute weight is bounded by one.

We first recall that we stated in ➔Theorem 3.31 that solving parity games with weights is polynomial-time equivalent to solving energy parity games. Furthermore, recall that, as discussed in ➔Section 3.4, in order to solve a parity game with weights $\mathcal{G}$ with largest absolute weight W, we solve a polynomial number of energy parity games with largest absolute weight $\max(1, W)$. Since the problem of solving energy parity games can still only be solved in quasi-polynomial time even in this special case, we do not obtain an improved runtime of Algorithm 3.2 due to the assumption of weights given in unary encoding.

➔Sec. 3.3, Page 69
➔Page 70

Remark 3.41. *The following problem is in* NP ∩ coNP*:*

> *"Given a parity game with weights $\mathcal{G}$ with n vertices, d colors, and largest absolute weight one, and a vertex v in $\mathcal{G}$, does Player* 0 *win $\mathcal{G}$ from v?"*

Furthermore, due to an observation by Daviaud, Jurdziński, and Lazić [DJL18], the problem stated in the above remark can be solved in time $\mathcal{O}(dn^{2\log(d/\log n^2)+4.45})$.

Regarding the memory requirements of winning strategies in parity games with weights, the results of ➔Sec. 3.4, Page 71 ➔Theorem 3.34 cannot be further improved for the special case of the weights being given in unary encoding. These results yield that Player 0 requires $\mathcal{O}(nd^2W)$ memory states to implement a winning strategy in parity games with weights. Hence, if $W = 1$, then polynomial memory space suffices for Player 0 to implement such strategies. ➔Sec. 3.4, Page 72 Moreover, the games $\mathcal{G}_{n,1}$ constructed in ➔Lemma 3.37 witness a polynomial lower bound. Player 1, in contrast, still requires infinite memory to win even in this special case, as this lower bound already holds true for the case of finitary parity games [CH06].

Remark 3.42. *Let $\mathcal{G}$ be a parity game with weights with n vertices, d odd colors, and largest absolute weight one. Moreover, let v be a vertex of $\mathcal{G}$.*

1. *Player 0 has a strategy σ that is winning from each vertex in $W_0(\mathcal{G})$ with $|\sigma| \in \mathcal{O}(nd^2)$.*
2. *Player 1 requires, in general, infinite memory to win from $W_1(\mathcal{G})$.*

Furthermore, for each $n > 0$, there exists a parity game with weights $\mathcal{G}_n$ such that Player 0 wins $\mathcal{G}_n$ from every vertex, but she requires a strategy of size $\Omega(n)$ to do so.

Analogously, the results of ➔Sec. 3.5, Page 75 ➔Theorem 3.38 can also not be further improved for the special case of $W = 1$. In particular, if Player 0 wins a parity game with weights $\mathcal{G}$, she can still guarantee cost at most polynomial in the number of vertices and the number of colors occurring in $\mathcal{G}$. Moreover, the tightness of this bound is again witnessed by the games $\mathcal{G}_{n,1}$ constructed in ➔Sec. 3.5, Page 77 ➔Lemma 3.40.

Remark 3.43. *Let $\mathcal{G}$ be a parity game with weights with n vertices, d odd colors, and largest absolute weight one.*

There exists some $b \in \mathcal{O}((nd)^2)$ and a strategy σ for Player 0 such that, for all plays ρ beginning in $W_0(\mathcal{G})$ and consistent with σ, we have $Cost(\rho) \leq b$.

Furthermore, for each $n > 0$, there exists a parity game with weights $\mathcal{G}_n$ with n vertices and largest absolute weight one such that Player 0 wins $\mathcal{G}_n$ from each vertex, but can only enforce a cost of $\Omega(n)$.

Finally, we remark that our definition of parity games with weights is not without alternatives. Recall that we defined the amplitude of a play infix as the largest absolute value of the accumulated weight attained along that infix, i.e., we defined

$$\mathrm{Ampl}(\pi) = \sup_{j<|\pi|} \left|\mathrm{Weight}(v_0 \cdots v_j)\right| \in \mathbb{N}_\infty$$

Three possible variations of this definition would

- use a one-sided definition (instead of the absolute value) for the amplitude of a play, i.e., define

$$\mathrm{Ampl}(\pi) = \sup_{j<|\pi|} \mathrm{Weight}(v_0 \cdots v_j) \in \mathbb{N}_\infty \ ,$$

- not allow Player 0 to "store" energy by letting the weight of a play infix below zero, i.e., define

$$\mathrm{Weight}(v_0 \cdots v_j) = \max\left\{0, \mathrm{Weight}(v_0 \cdots v_{j-1}) + \mathrm{Weight}(v_{j-1}, v_j)\right\} \ ,$$

and

- cause Player 0 to lose the game once the cost of an unanswered request drops below zero, i.e., declare all plays $v_0v_1v_2\cdots$ in which there exist positions j and j' such that the request for $\Omega(v_j)$ is not answered in $v_j\cdots v_{j'}$ and such that $\text{Weight}(v_j\cdots v_{j'}) < 0$ losing for Player 0,

respectively. We believe our definition, however, to be the most intuitive, straightforward, and practicable extension of the parity condition with costs to the setting of integer bounds. We leave it open whether these alternative definitions yield more or less complex decision problems.

Furthermore, Bruyère, Hautem, and Randour [BHR16] have studied a variant of finitary parity games with multiple colorings, each of which induces a finitary parity condition. Player 0 wins this multidimensional finitary parity game if she satisfies all induced finitary parity conditions. The authors have shown the boundedness problem for this setting to be ExpTime-complete. One could also consider a variant of parity games with weights that features multiple colorings, each with a respective weight function. Analogously to the multidimensional finitary parity games of Bruyère, Hautem, and Randour, it would be the aim of Player 0 to satisfy all induced parity conditions with weights. We leave the problem of solving such games open for future work.

Instead, in the next section, we focus on the optimality problem for parity games with weights. Recall that, in order to win a parity game with weights, Player 0 only has to bound the costs incurred by the requests in the play. In the optimality problem, it is our task to determine a minimal b such that Player 0 is able to enforce the cost of the resulting play to not exceed b. We will show that this changes the characteristics of the decision problems as well as of the strategies witnessing the solutions significantly.

CHAPTER 4

Playing Parity Games with Weights Optimally

In the previous section, we have defined parity games with weights as a generalization of parity games with costs. While parity games with costs only allow for nonnegative weights along their edges, parity games with weights drop this restriction and allow arbitrary integer weights. Moreover, we have lifted the concept of the cost of response from parity games with costs to parity games with weights. In the latter model, we have defined the cost of response via the amplitude of the play infix starting at some request and ending at its earliest answer. It is then the task of Player 0 to ensure a finite upper bound on the cost-of-response of almost all requests. We have investigated the properties of the problem of solving parity games with weights in the previous chapter.

In order to satisfy the parity condition with weights, however, Player 0 only has to ensure some finite upper bound on the cost of the resulting play. This, however, may not be sufficient for real-world applications: Recall that a major motivation for the use of parity games with weights is their ability to model reactive systems with an attached resource that may be charged or drained during the runtime of the system. A winning strategy for Player 0 in a parity game with weights then implements a controller for that system that provides an arbitrary upper bound on the fluctuation of that resource in-between requests and responses. This controller, however, may let the level of that resource fluctuate more than necessary, which is, in general, undesirable.

Instead, in order to obtain an "optimal" controller for the system, it is desirable to ask for the optimal bound b such that the system can ensure that the level of the resource fluctuates by at most b in-between requests and responses, as well as for a controller witnessing this bound b. Speaking in terms of parity games with weights, one asks for the minimal b such that Player 0 has a strategy of cost at most b from some initial vertex. The following example shows that this additional strengthening significantly changes the characteristics of the game, even in the special case of finitary parity games.

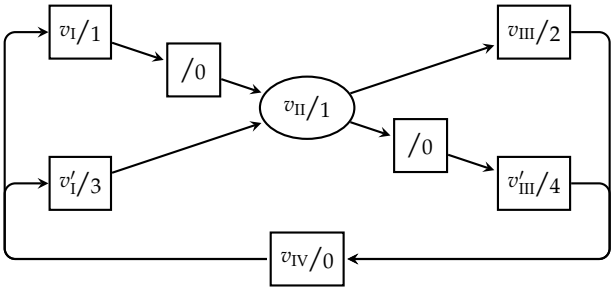


Figure 4.1: Example of a finitary parity game in which Player 0 has two strategies of differing qualities [CF13].

Example 4.1. Consider the finitary parity game $\mathcal{G}$ shown in Figure 4.1, which is due to Chatterjee and Fijalkow [CF13]. Clearly, Player 0 wins $\mathcal{G}$ from every vertex. As in all finitary parity games (see ➔ Proposition 2.25.1), she has a positional winning strategy σ which has to prescribe moving towards v'_{III} upon every visit to v_{II}. In particular, the play $\rho = (v_{\mathrm{I}} \cdots v_{\mathrm{II}} \cdots v'_{\mathrm{III}} v_{\mathrm{IV}})^\omega$ is consistent with σ and has $\mathrm{Cost}(\rho) = 4$.

➔ Sec. 2.4, Page 23

Now assume we aim to find a strategy for Player 0 that allows her to bound the cost of consistent plays by three. We define the strategy σ' that prescribes moving from v_{II} to v_{III} if the preceding named vertex of v_{II} is v_{I}, and that prescribes moving towards v'_{III}, otherwise. Every play ρ consistent with σ' has $\mathrm{Cost}(\rho) = 3$. In order to realize this lower cost, however, σ' requires two memory states: In $\mathcal{G}$, Player 0 only has two positional strategies. Clearly, neither of these strategies has cost three or smaller. Hence, every strategy that enforces a cost of consistent plays of at most three requires at least two memory states. △

Since finitary parity games are a special case of parity games with costs, which, in turn, are a special case of parity games with weights, the above example witnesses that even if Player 0 has a positional winning strategy in parity games with weights, she may require memory in order to enforce cost below a given bound. Our main aim in this chapter is to determine, given a parity game with weights $\mathcal{G}$ and a vertex v of $\mathcal{G}$, the minimal b such that Player 0 has a strategy σ with $\mathrm{Cost}_v(\sigma) = b$, as well as a strategy σ witnessing this bound.

We later show that, due to our results from ➔ Chapter 3, it suffices to determine for a given bound b, whether Player 0 has a strategy σ with $\mathrm{Cost}_v(\sigma) \leq b$. Thus, our main object of interest in this chapter is the following decision problem, the so-called THRESHOLD PROBLEM:

➔ Page 29

Def. threshold problem

> "Let $\mathcal{G}$ be a parity game with weights, let v be a vertex of $\mathcal{G}$, and let $b \in \mathbb{N}$. Does Player 0 have a strategy σ with $\mathrm{Cost}_v(\sigma) \leq b$?"

We show that this problem is ExpTime-complete for the general case of parity games with weights, and that it is PSpace-complete for the special case of parity games with costs as well as for the special case of finitary parity games.

Hence, the threshold problem for parity games with weights is harder than the problem of solving them, which is known to be in NP ∩ coNP for parity games with

weights due to ➔Theorem 3.18, in UP ∩ coUP for the special case of parity games with costs due to ➔Proposition 2.33, and in PTime for the special case of finitary parity games due to ➔Proposition 2.27. ➔Sec. 3.2, Page 52 ➔Sec. 2.4, Page 26 ➔Sec. 2.4, Page 24

Furthermore, we are interested in bounds on the memory required to implement such a strategy σ and show that, in order to keep the costs below a bound b in a parity game with weights with n vertices and with d odd colors, Player 0 requires memory of size $\mathcal{O}(b^d)$. This amount of memory furthermore suffices for her to keep the costs below b, if she is able to do so at all. This contrasts the upper bound of $\mathcal{O}(nd^2W)$ on the memory required for her to win a parity game with weights (where W denotes the largest absolute weight occurring in the game) due to ➔Theorem 3.34.1, as well as the existence of positional winning strategies for her in parity games with costs and finitary parity games due to ➔Proposition 2.34.1. Finally, we consider the memory requirements of strategies for Player 1 that ensure the cost of consistent plays to exceed a given b and show that he only requires $\mathcal{O}(nb^d)$ memory states in order to enforce a cost larger than b, which again contrasts his requirement for infinite memory in order to win even finitary parity games (see ➔Proposition 2.25.2). ➔Sec. 3.4, Page 71 ➔Sec. 2.4, Page 26 ➔Sec. 2.4, Page 23

The remainder of this section is structured as follows: First, we reduce the threshold problem for parity games with weights to that of solving parity games in Section 4.1. In order to do so, we construct so-called threshold games. We use these threshold games to show that we are able to solve the above problem in exponential time. Subsequently, we show that we are able to lower this bound for the special case of parity games with costs in Section 4.2: Here, we show that we can leverage the structure of the threshold games to solve them in polynomial space (in the size of the parity game with costs). Moreover, we argue that in both cases we can determine the optimal bound in exponential time and polynomial space, respectively. In Section 4.3 we then provide matching lower bounds on the complexity of the above problems, i.e., we show that the general problem for parity games with weights is ExpTime-hard and that the problem is still PSpace-hard even for finitary parity games.

After having thus precisely determined the complexity of the threshold problem for parity games with weights and their special cases, we turn our attention to the memory requirements of both players. We show in Section 4.4 that both players require exponential memory in order to ensure a given upper (in the case of Player 0) or lower (in the case of Player 1) bound on the cost of responses, even for finitary parity games.

Finally, recall that in parity games with costs, it sufficed to consider the costs as "abstract" values, i.e., to consider for a given edge whether it has costs zero, or greater than zero (see ➔Remark 2.29). When requiring Player 0 to keep the costs below a given bound, this is no longer the case. Thus, in Section 4.5, we discuss the influence of a unary encoding of the weights on the results obtained in the preceding sections for weights encoded in binary. We conclude this chapter in Section 4.6 with a summary of the obtained results. ➔Sec. 2.4, Page 26

The results on parity games with costs in this chapter are based on work published in LMCS [WZ17]. The results on parity games with weights constitute novel work.

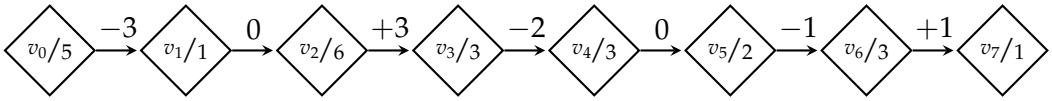


Figure 4.2: The play prefix discussed in Example 4.2 illustrating that not all information about all open requests needs to be stored.

4.1 Reduction to Threshold Games

As a first step in our analysis of the threshold problem, we reduce that problem to that of solving classical qualitative parity games as introduced in ➔Section 2.3.2. To this end, for a given parity game with weights $\mathcal{G}$ and a bound b, we construct a parity game $\mathcal{G}_b$ such that Player 0 has a strategy of cost at most b from v in $\mathcal{G}$ if and only if she wins $\mathcal{G}_b$ from a designated vertex induced by v. Hence, for the remainder of this section, fix some parity game with weights $\mathcal{G}$ with n vertices, d odd colors, and largest absolute weight W as well as a bound $b \in \mathbb{N}$.

➔Page 20

Recall that in parity games, positional strategies suffice for both players to win. Hence, intuitively, upon reaching a vertex v in a parity game, neither player has any information about the play that led to v, but they can only use their knowledge of the current vertex in order to determine the next move. Thus, in order to construct a parity game with the above property, we need to encode the information necessary to determine the next move in $\mathcal{G}$ in the vertices of the resulting parity game. We show that it suffices to store abstracted information about the requests that are currently open, as well as the number of times that the bound b has already been exceeded.

4.1.1 Request Functions and Overflow Counters

First, we consider the information required about the open requests at the end of a play prefix and observe that we do not require full information about all open requests and their incurred costs. Instead it suffices to store, for each color c, the minimal and maximal costs incurred by unanswered requests for c in the play prefix.

Example 4.2. Consider the play prefix shown in Figure 4.2. At the end of π there is an open request for color 1, which has incurred cost 0. Moreover, there are three open requests for color 3, which have incurred cost -2, cost 0 and cost 1, respectively. It is unnecessary to store the request for color 3 with cost 0: If this request ever violates some positive bound, then the request that has incurred cost 1 will have done so previously. Dually, if the request for color 3 with cost 0 violates some negative bound, then the request for color 3 that has incurred cost -2 will have done so before.

Finally, there is no open request for color 5, as this request has been answered by visiting the color 6 in the third step. △

Def. request functions

We formalize the above observation via REQUEST FUNCTIONS, which, for each odd color c for which there is an open request at the end of a play prefix store the maximal and minimal cost incurred by unanswered requests for color c, up to and including

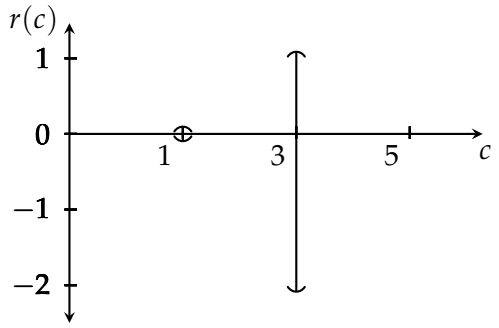


Figure 4.3: The request function denoting the requests posed at the end of π from Example 4.2.

the bound b. Towards the definition of request functions, we first define the set of intervals

$$I = \{(l,h) \mid -b \leq l \leq h \leq b\} \ .$$

Clearly, we have $|I| = (2b+1) + \cdots + 1 = (2b+1)(2b+2)/2 = (4b^2+4b+2b+2)/2 = 2b^2+3b+1$. We denote the set of odd colors occurring in $\mathcal{G}$ by D. A b-BOUNDED REQUEST FUNCTION $r\colon D \to \{\bot\} \cup I$ is a function mapping each odd color of $\mathcal{G}$ either to

Def. (b-bounded) request function

- $\bot$, denoting that currently no request for color c is open, or to
- some $(l,h) \in I$, denoting that
  - there exists an open request for color c that has accumulated weight l, that
  - there exists an open request for color c that has accumulated weight h, and that
  - all requests for color c have accumulated weight at least l and at most h.

Given some request function r, we define the LOWER AND UPPER RESIDUAL REQUEST FUNCTIONS $r_\downarrow$ and $r_\uparrow$ as $r_\downarrow(c) = l$ and $r_\uparrow(c) = h$, if $r(c) = (l,h)$, and as $r_\downarrow(c) = r_\uparrow(c) = \bot$ if $r(c) = \bot$.

Def. $r_\downarrow, r_\uparrow$

Example 4.3. Consider again the play π defined in Example 4.2. This play prefix induces the request function r defined as

$$r : \begin{cases} 1 \mapsto (0,0) \ , \\ 3 \mapsto (-2,1) \ , \\ 5 \mapsto \bot \ . \end{cases}$$

We show a graphical representation of r in Figure 4.3. Moreover, we obtain

$$r_\downarrow(c) = \begin{cases} 1 \mapsto 0 \ , \\ 3 \mapsto -2 \ , \\ 5 \mapsto \bot \ , \end{cases} \qquad \text{and} \qquad r_\uparrow(c) = \begin{cases} 1 \mapsto 0 \ , \\ 3 \mapsto 1 \ , \\ 5 \mapsto \bot \end{cases}$$

as the lower and upper residual request functions. △

Def. R

We define R to be the set of all request functions. We have $|R| = (|I|+1)^{|D|} = (2b^2+3b+2)^d$, i.e., there exist exponentially many request functions when measured in the size of the game $\mathcal{G}$, but only polynomially many when measured in the bound b.

4.1.2 Threshold Games

Request functions provide an explicit view of the open requests at the end of some play prefix. It is, however, insufficient to only track the request function along an infinite play ρ in order to decide whether or not ρ is winning for Player 0, as she may have to tolerate finitely many requests with costs greater than b. Hence, in order to construct the parity game $\mathcal{G}_b$ we augment the vertices of $\mathcal{G}$ not only with a request function, but also with a so-called OVERFLOW COUNTER that is incremented each time some request incurs a cost greater than b.

Def. overflow counter

We later show that it suffices for Player 1 to enforce requests with costs exceeding the bound b n times in order to witness that he is able to do so infinitely often. Hence, it suffices to have this counter bounded from above by the number of vertices of $\mathcal{G}$. We now define a memory structure that, together with the game $\mathcal{G}$, induces the desired parity game $\mathcal{G}_b$.

Recall that we fixed some parity game with weights $\mathcal{G}$ with n vertices and d odd colors as well as a bound $b \in \mathbb{N}$. Using the set R of request functions defined in the previous section, we define the set of memory states $M = \{0,\dots,n\} \times R$. As we aim to track the cost of open requests using the functions from R, we define the initial memory element $\mathit{init}(v) = (0, r_v)$, where r_v is defined as

Def. r_v

$$r_v(c) = \begin{cases} (0,0) & \text{if } \Omega(v) \text{ is odd and } c = \Omega(v), \text{ and} \\ \bot & \text{otherwise.} \end{cases}$$

We define the update function $\mathit{upd}\colon M \times E \to M$ implementing the above intuition. Let $m = (o, r) \in M$ and let $e = (v, v') \in E$. This update function updates the memory state via $\mathit{upd}(m, e)) = (o', r')$ by performing the following steps in order:

Weight First, we resolve the effect of traversing the edge e with weight w by defining r'_{I} as

$$r'_{\mathrm{I}}(c) = \begin{cases} (r_{\downarrow}(c) + \mathrm{Weight}(e), r_{\uparrow}(c) + \mathrm{Weight}(e)) & \text{if } r(c) \neq \bot, \text{ and} \\ \bot & \text{otherwise.} \end{cases}$$

Overflow In a second step, we check whether some request has violated the bound b during the move to v' and update the overflow counter if this is the case. Thus, if there exists a color c such that either $(r'_{\mathrm{I}})_{\downarrow}(c) < -b$ or $(r'_{\mathrm{I}})_{\uparrow}(c) > b$, then we define $r'_{\mathrm{II}}(c) = \bot$ for all $c \in D$ and set o' to the minimum of $o+1$ and n. Otherwise, we define $r'_{\mathrm{II}} = r'_{\mathrm{I}}$ and $o' = o$.

Request Finally, we resolve the effect of moving to a vertex with color $\Omega(v')$ as follows: If $\Omega(v')$ is even, then we define

$$r'_{\mathrm{III}}(c) = \begin{cases} \bot & \text{if } c \leq \Omega(v'), \text{ and} \\ r'_{\mathrm{II}}(c) & \text{otherwise.} \end{cases}$$

If, however, $\Omega(v')$ is odd, then we define

$$r'_{\mathrm{III}}(c) = \begin{cases} \left(\min\left\{(r'_{\mathrm{II}})_{\downarrow}(\Omega(v')), 0\right\}, \right. & \\ \qquad\qquad \left. \max\left\{(r'_{\mathrm{II}})_{\uparrow}(\Omega(v')), 0\right\}\right) & \text{if } c = \Omega(v'), \text{ and} \\ r'_{\mathrm{II}}(c) & \text{otherwise,} \end{cases}$$

where we use the convention $\min\{\bot, 0\} = \max\{\bot, 0\} = 0$.

In either case, we define $r' = r'_{\mathrm{III}}$, which concludes the definition of $m' = (o', r')$. The resulting o' is at most n and the resulting function r' is an element of R. We combine these elements in the memory structure $\mathcal{M} = (M, \mathit{init}, \mathit{upd})$.

Example 4.4. Let r be a request function defined as

$$r : \begin{cases} 1 \mapsto (-1, 2) \ , \\ 3 \mapsto \bot \ , \\ 5 \mapsto (1, 3) \ . \end{cases}$$

We illustrate this request function in Figure 4.4a.

Now assume that some play prefix that induces the request function r traverses two edges of weight -1 each, first reaching a vertex v of color 2 and subsequently reaching a vertex v' of color 3. Moreover, assume $b > 3$. Moving to v answers all requests for color 1. Subsequently, moving to v' opens a new request for color 3 which has not yet incurred any costs. This extended play prefix induces the request function

$$r' : \begin{cases} 1 \mapsto \bot \ , \\ 3 \mapsto (0, 0) \ , \\ 5 \mapsto (-1, 1) \ . \end{cases}$$

We illustrate the request function r' in Figure 4.4b. △

Recall that throughout this section we fixed a parity game with weights $\mathcal{G}$. Let $\mathcal{G} = (\mathcal{A}, \textsc{WeightParity}(\Omega, \mathrm{Weight}))$. We define the ($b$-)threshold game of $\mathcal{G}$

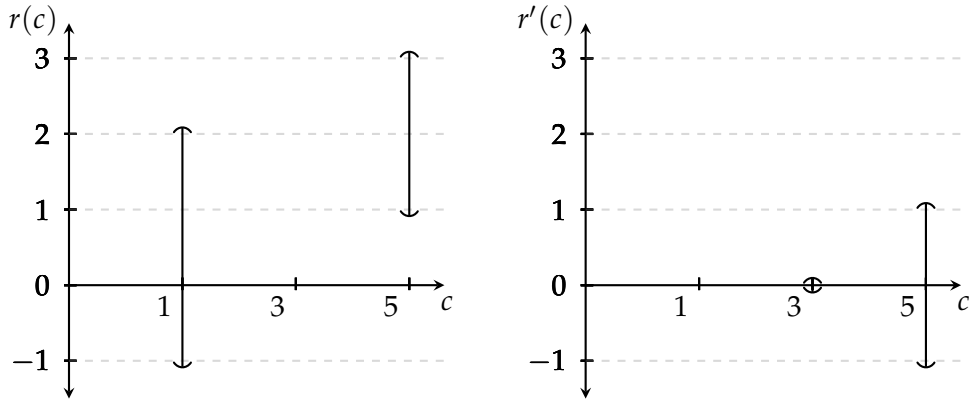
Def. (b-)threshold game

$$\mathcal{G}_b = (\mathcal{A}', \textsc{Parity}(\Omega')) \ ,$$

with $\mathcal{A}' = \mathcal{A} \times \mathcal{M}$ and with

$$\Omega'(v, o, r) = \begin{cases} \Omega(v) & \text{if } o < n, \text{ and} \\ 1 & \text{otherwise,} \end{cases}$$

which concludes the definition of $\mathcal{G}_b$. A straightforward induction shows that this construction indeed allows us to reason about the open requests along a play prefix.



(a) The request function r.

(b) The request function r'.

Figure 4.4: Illustration of the update of request functions from Example 4.4.

Remark 4.5. *Let $\rho = v_0 \cdots v_j$ be a play in the parity game with weights $\mathcal{G}$ and let $ext(\rho) = (v_0, o_0, r_0) \cdots (v_j, o_j, r_j)$ be its unique extended play in the threshold game $\mathcal{G}_b$ with $o_j < n$. Moreover, let $c \in D$.*

If $r_j(c) = (l, h) \neq \bot$, then there exist positions $j_\downarrow \leq j$ and $j_\uparrow \leq j$ such that

- *$\Omega(v_{j_\downarrow}) = \Omega(v_{j_\uparrow}) = c$, such that*
- *the request for color c is not answered in the infix $v_{j_\downarrow} \cdots v_j$ nor in the infix $v_{j_\uparrow} \cdots v_j$, and such that*
- *$Weight(v_{j_\downarrow} \cdots v_j) = l$, and $Weight(v_{j_\uparrow} \cdots v_j) = h$.*

Furthermore, if $r_j(c) = (l, h) \neq \bot$ let $j' \leq j$ be a position with $\Omega(v_j) = c$ such that the request for color c is not answered in the infix $v_{j'} \cdots v_j$. We then obtain $l \leq Weight(v_{j'} \cdots v_j) \leq h$.

The implication of Remark 4.5 does not hold true in the other direction: Since the request function is "reset" every time the costs of some request exceed the bound b, it does not provide information on all open requests.

In $\mathcal{G}_b$ we cap the amount of times the costs of some request may exceed the bound b by n: Due to the coloring of $\mathcal{A}'$, every play that at some point encounters a vertex of the form (v, n, r) is losing for Player 0.

Remark 4.6. *Let $\rho = (v_0, o_0, r_0)(v_1, o_1, r_1)(v_2, o_2, r_2) \cdots$ be a play in $\mathcal{G}_b$. If there exists a position $j \in \mathbb{N}$ such that $o_j = n$, then ρ violates the parity condition.*

The threshold game $\mathcal{G}_b$ is a classical qualitative parity game and thus has no weight function. In order to simplify notation, however, we lift this notion from the underlying game $\mathcal{G}$ to $\mathcal{G}_b$ and say that an edge $((v, m), (v', m'))$ of $\mathcal{G}_b$ has weight w if the edge (v, v') of $\mathcal{G}$ has weight w.

Def. overflow position

Furthermore, let $\rho = (v_0, o_0, r_0)(v_1, o_1, r_1)(v_2, o_2, r_2) \cdots$ be a play or a play prefix in $\mathcal{A}'$. We say that a position j is an OVERFLOW POSITION if either $j = 0$ or if $o_j = o_{j-1} + 1$. Due to the construction of $\mathcal{M}$ we have $r_j = r_{v_j}$ for every overflow position j, i.e., the request function is "reset" at each such position.

As a final piece of notation, we say that the overflow counter along the play ρ has SATURATED if there exists a position j with $o_j = n$. Due to the construction of $\mathcal{A}'$ this implies $o_{j'} = n$ for all $j' > j$. Moreover, the overflow positions indeed denote positions at which the costs of some request exceed the bound b.

Def. saturated overflow counter

Remark 4.7. *Let $\rho = v_0 v_1 v_2 \cdots$ be a play in the parity game with weights $\mathcal{G}$ and let $ext(\rho) = (v_0, o_0, r_0)(v_1, o_1, r_1)(v_2, o_2, r_2) \cdots$ be its unique extended play in the threshold game $\mathcal{G}_b$.*

1. *Let $j \in \mathbb{N}$ such that $b < Cor(\rho, j) < \infty$. If $o_j < n$, then there exists some $j' \geq j$ such that j' is an overflow position.*
2. *Let $j, j' \in \mathbb{N}$ be two adjacent overflow positions. There exists a j'' with $j \leq j'' < j'$ such that $Cor(\rho, j'') > b$.*

Our aim in this section was to construct the parity game $\mathcal{G}_b$ such that we can reduce the problem of solving $\mathcal{G}$ with respect to some bound b to solving $\mathcal{G}_b$. We show that this is indeed the case in the following section.

4.1.3 Correctness

In the previous section we constructed the threshold game $\mathcal{G}_b$ of $\mathcal{G}$, which we obtained by augmenting the parity game with weights $\mathcal{G}$ with the memory structure $\mathcal{M} = (M, init, upd)$. Each vertex (v, o, r) of $\mathcal{G}_b$ consists of a vertex v from $\mathcal{G}$, a counter value o that counts the number of times the cost of some request has exceeded the bound b, and a request function $r\colon D \to \{\bot\} \cup I$ that stores information about the open requests and the costs they have incurred so far.

Moreover, we claimed that the threshold problem reduces to solving the constructed parity game. In this section, we show that this is indeed the case, i.e., this section is dedicated to showing the following theorem. Recall that, in →Section 3.1, we defined $\mathrm{Cost}_{v^*}(\sigma) = \sup_\rho \mathrm{Cost}(\rho)$, where ρ ranges over all plays starting in v^* and consistent with σ.

→Page 31

Theorem 4.8. *Let v^* be a vertex in $\mathcal{G}$. Player 0 has a strategy σ in $\mathcal{G}$ with $Cost_{v^*}(\sigma) \leq b$ if and only if she wins $\mathcal{G}_b$ from $(v^*, init(v^*))$.*

We split the proof into several lemmas. To this end, recall that we defined $\mathcal{G}_b = (\mathcal{A}', \textsc{Parity}(\Omega'))$. Let $\mathcal{A}' = (V', V_0', V_1', E')$, i.e., in particular, let $V' = V \times M$ and $V_i' = V_i \times M$ for $i \in \{0, 1\}$. Moreover, for the remainder of this section, fix some vertex v^* of $\mathcal{G}$.

The direction from right to left is relatively straightforward: Since $\mathcal{G}_b$ is a parity game, if Player 0 wins $\mathcal{G}_b$ from $(v^*, init(v^*))$, then she does so with a positional strategy σ', due to →Proposition 2.18. This strategy assigns to each vertex (v, o, r) of Player 0 in $\mathcal{G}_b$ a unique successor (v', o', r'), where the values of o' and r' are deterministic updates of o and r via the update function upd. Hence, σ' can be interpreted as picking only a successor vertex v' of v with respect to the current memory state (o, r). Thus, the choices of σ' can be mimicked in $\mathcal{G}$.

→Sec. 2.3, Page 20

Lemma 4.9. *If Player 0 wins $\mathcal{G}_b$ from $(v^*, init(v^*))$, then she has a strategy σ in $\mathcal{G}$ with $Cost_{v^*}(\sigma) \leq b$.*

Proof. Let $\sigma' \colon V_0' \to V'$ be a positional winning strategy for Player 0 from $(v^*, \mathit{init}(v^*))$ in $\mathcal{G}_b$. Since $\mathcal{G}_b$ is a classical parity game, such a strategy exists due to Proposition 2.18. We define the finite-state strategy σ for Player 0 in $\mathcal{G}$ as the unique strategy induced by the memory structure $\mathcal{M}$ and the next-move function nxt defined as $\mathit{nxt}(v, m) = v'$, if $\sigma'(v, m) = (v', m')$ for some m'. It remains to show $\mathrm{Cost}_{v^*}(\sigma) \leq b$.

Let $\rho = v_0 v_1 v_2 \cdots$ be a play in $\mathcal{G}$ that begins in v^* and that is consistent with σ. Let

$$\rho' = \mathrm{ext}(\rho) = (v_0, o_0, r_0)(v_1, o_1, r_1)(v_2, o_2, r_2) \cdots$$

be the unique extended play in $\mathcal{G}_b$.

A simple induction shows that ρ' is consistent with σ'. As ρ' moreover starts in $(v^*, \mathit{init}(v^*))$ by definition, it is winning for Player 0, i.e., it satisfies the parity condition. This in particular implies that the overflow counter along ρ' never saturates, i.e., that we have $o_j < n$ for all $j \in \mathbb{N}$, due to Remark 4.6. Hence, the plays ρ and ρ' coincide on their color sequences. Since ρ' is winning for Player 0 in $\mathcal{G}_b$, it satisfies the parity condition, which in turn implies that ρ satisfies the parity condition. It remains to show that almost all requests in ρ are answered with cost at most b.

As argued above, the overflow counter along ρ' eventually stabilizes at some value less than n. Moreover, since ρ satisfies the parity condition, after some finite prefix, all requests are answered. Hence, there exists a position j such that $o_{j'} = o_j < n$ and such that $\mathrm{Cor}(\rho, j') < \infty$ for every $j' > j$. Assume towards a contradiction that there exists some $j' \geq j$ with $b < \mathrm{Cor}(\rho, j') < \infty$. Then, the play ρ' contains some overflow position at some point after j' due to Remark 4.7.1. This, however, contradicts the choice of j. Hence, we obtain that almost all requests in ρ are answered with cost at most b. □

We now turn our attention to the other direction of the statement, i.e., we aim to show that, if Player 0 has a strategy σ in $\mathcal{G}$ with $\mathrm{Cost}_{v^*}(\sigma) \leq b$, then she wins $\mathcal{G}_b$ from $(v^*, \mathit{init}(v^*))$. We show this claim via contraposition: Assume that Player 0 does not win $\mathcal{G}_b$ from $(v^*, \mathit{init}(v^*))$. Since $\mathcal{G}_b$ is a parity game, it is determined due to
→ Sec. 2.3, Page 20
→ Proposition 2.18. Hence, Player 1 wins $\mathcal{G}_b$ from $(v^*, \mathit{init}(v^*))$, say with the positional strategy τ'. We obtain that such a positional strategy for him exists due to Proposition 2.18. We construct a strategy τ for Player 1 in $\mathcal{G}$ that enforces $\mathrm{Cost}(\rho) > b$ for each play ρ starting in v^* and consistent with τ. Since this implies $\mathrm{Cost}_{v^*}(\sigma) > b$ for each strategy σ of Player 0, this suffices to show the desired statement.

Recall that the overflow counter along each play starting in $(v^*, \mathit{init}(v^*))$ is monotonically increasing and bounded from above by the number n of vertices in $\mathcal{G}$. Hence, the value of the overflow counter either stabilizes at some value less than n, or it eventually saturates at value n. In the former case, τ' has to ensure that the resulting play violates the parity condition. Hence, it suffices to mimic the moves made by τ' in $\mathcal{G}$ ad infinitum in this case, which results in a play with infinitely many unanswered requests in $\mathcal{G}$. In the latter case, however, mimicking τ' does not yield a strategy in $\mathcal{G}$ with the desired property, as τ' does not necessarily prescribe "meaningful" moves in $\mathcal{G}_b$ once the overflow counter saturates. In order to leverage τ' even after saturation of the overflow counter, we intervene whenever the overflow counter is incremented, by resetting it to the smallest possible value from which τ' is still winning.

Formally, we define the set $\mathcal{R}$ that contains all vertices (v, o, r) that are visited by some play that starts in $(v^*, \mathit{init}(v^*))$ and that is consistent with τ'. Recall that we defined r_v as the function denoting the requests opened by visiting the vertex v in ➔Section 4.1.2. Given a vertex v, we then define $o_v = \min(\{n\} \cup \{o \mid (v, o, r_v) \in \mathcal{R}\})$. ➔Page 88 In particular, we have $o_{v^*} = 0$, since $(v^*, \mathit{init}(v^*)) = (v^*, 0, r_v) \in \mathcal{R}$.

Def. $\mathcal{R}$

Lemma 4.10. *The strategy τ' is winning for Player 1 from (v, o_v, r_v) in $\mathcal{G}_b$ for all $v \in V$.*

Proof. If $o_v = n$, then all plays starting in (v, o_v, r_v) violate the parity condition by construction of $\mathcal{A}'$. Thus, for the remainder of this proof, assume $o_v < n$. Let $(v, o_v, r_v)\rho$ be a play starting in (v, o_v, r_v) that is consistent with τ'. Moreover, let π be a play prefix starting in $(v^*, \mathit{init}(v^*))$, consistent with τ', and ending in (v, o_v, r_v). Such a play prefix exists due to definition of o_v and due to the assumption of $o_v < n$.

Since τ' is positional, the play $\pi\rho$ starts in $(v^*, \mathit{init}(v^*))$ and is consistent with τ'. Thus, $\pi\rho$ violates the parity condition, which implies that ρ violates the parity condition due to prefix-independence. □

We now define a new memory structure $\mathcal{M}'$ implementing the strategy τ. Recall that we defined the memory structure $\mathcal{M} = (M, \mathit{init}, \mathit{upd})$ during the construction of the threshold game $\mathcal{G}_b$ in ➔Section 4.1.2. Using the components of that memory structure, we define $M' = M = [n+1] \times R$, $\mathit{init}' = \mathit{init}$, and ➔Page 88

$$\mathit{upd}'((o, r), (v, v')) = \begin{cases} (o, r') & \text{if } \mathit{upd}((o, r), (v, v')) = (o, r'), \text{ and} \\ (o_{v'}, r') & \text{if } \mathit{upd}((o, r), (v, v')) = (o+1, r'). \end{cases}$$

and combine these elements into the memory structure $\mathcal{M}' = (M', \mathit{init}', \mathit{upd}')$.

In the second case of the definition of upd', we have $r' = r_{v'}$ by definition of upd. Finally, we define the next-move function nxt' via $\mathit{nxt}'(v, m) = v'$, if $\tau'(v, m) = (v', m')$ for some $m' \in M$ and let τ be the finite-state strategy implemented by $\mathcal{M}'$ and nxt. We claim that for each play ρ starting in v^* that is consistent with τ we have $\mathrm{Cost}(\rho) > b$.

To show this claim, let $\rho = v_0 v_1 v_2 \cdots$ be some play in $\mathcal{G}$ that starts in v^* and that is consistent with τ. Moreover, let

$$\rho' = \mathrm{ext}_{\mathcal{M}'}(\rho) = (v_0, o_0, r_0)(v_1, o_1, r_1)(v_2, o_2, r_2) \cdots$$

be the extended play of ρ with respect to $\mathcal{M}'$. We say that j is a RESET POSITION if $j = 0$ or if

Def. reset position

$$\mathit{upd}((o_{j-1}, r_{j-1}), (v_{j-1}, v_j)) = (o_{j-1} + 1, r_j) \ ,$$

i.e., if the second case in the definition of upd' is applied.

The play ρ' is not necessarily a play in $\mathcal{G}_b$, since $\mathcal{G}_b$ is defined with respect to $\mathcal{M}$ instead of $\mathcal{M}'$, but every infix of ρ' that starts at a reset position and does not contain another one, is a play infix in $\mathcal{G}_b$ that is consistent with τ'. At every reset position, instead of incrementing the overflow counter, we set it to o_v.

Our first aim is to show $o_j < n$ for all j by analyzing the structure of ρ'. Intuitively, this then implies that the strategy τ always uses "meaningful" moves of τ' for its choice of move and thus allows us to subsequently argue that τ is indeed winning for Player 1.

To this end, we first show that, even though ρ' is, in general, not a play in $\mathcal{G}_b$, every vertex (v_j, o_j, r_j) of ρ' is in $\mathcal{R}$. This implies that, intuitively, the positional strategy τ' prescribes a "useful" move from every (v_j, o_j, r_j) in V_1'.

Lemma 4.11. *For each $j \in \mathbb{N}$ we have $o_j < n$.*

Proof. We first show that for each $j \in \mathbb{N}$ we have $(v_j, o_j, r_j) \in \mathcal{R}$ by induction over j. For $j = 0$, this is clear, as we obtain $(v_0, o_0, r_0) = (v^*, \mathit{init}(v^*)) \in \mathcal{R}$ due to the definition of $\mathcal{R}$.

For $j > 0$, the induction hypothesis yields $v'_{j-1} = (v_{j-1}, o_{j-1}, r_{j-1}) \in \mathcal{R}$. Thus, by definition of $\mathcal{R}$, there exists a play prefix π starting in $(v^*, \mathit{init}(v^*)) = (v_0, o_0, r_0)$, consistent with τ' and ending in v'_{j-1}. If $v'_{j-1} \in V'_0$, then $\pi \cdot (v_j, o, r_j)$ is consistent with τ' for some $o \in \{0, \dots, n\}$, which implies $(v_j, o, r_j) \in \mathcal{R}$. If $o = o_j$, then $(v_j, o_j, r_j) \in \mathcal{R}$ clearly holds true. If $o \neq o_j$, however, then $o_j = o_{v_j}$ and $r_j = r_{v_j}$, i.e., $(v_j, o_j, r_j) \in \mathcal{R}$ by definition of o_{v_j}. Otherwise, i.e., if $v'_{j-1} \in V'_1$, then we have $\tau(\pi) = (v_j, o, r_j)$ for some $o \in \{0, \dots, n\}$. Similar reasoning as in the previous case yields $(v_j, o_j, r_j) \in \mathcal{R}$ in this case as well.

In a second step, we show that we do not "skip" values of the overflow counter using $\mathcal{M}'$, i.e., we show $o_{j+1} \leq o_j + 1$ for all $j \in \mathbb{N}$. To this end, let $j \in \mathbb{N}$ and, towards a contradiction, assume $o_{j+1} > o_j + 1$. Since $o_{j+1} \neq o_j$, the second case in the definition of upd' is applied, which implies

- $o_{j+1} = o_{v_{j+1}}$, as well as
- $\mathit{upd}((o_j, r_j), (v_j, v_{j+1})) = (o_j + 1, r_{v_{j+1}})$.

We show $(v_{j+1}, o_j + 1, r_{v_{j+1}}) \in \mathcal{R}$, which implies $o_{j+1} = o_{v_{j+1}} \leq o_j + 1$, i.e., the desired contradiction, due to definition of $o_{v_{j+1}}$.

Recall that we showed $(v_j, o_j, r_j) \in \mathcal{R}$ above. If (v_j, o_j, r_j) is a vertex of Player 0, then we directly obtain $(v_{j+1}, o_j + 1, r_{v_{j+1}}) \in \mathcal{R}$ due to $\mathit{upd}((o_j, r_j), (v_j, v_{j+1})) = (o_j + 1, r_{v_{j+1}})$, which implies that there is an edge leading from (v_j, o_j, r_j) to $(v_{j+1}, o_j + 1, r_{v_{j+1}})$ in $\mathcal{A}'$.

If, however, the vertex (v_j, o_j, r_j) belongs to Player 1, then we have $\tau(v_j, o_j, r_j) = (v_{j+1}, o_{j+1}, r_{j+1})$. By definition of τ, this yields $\tau'(v_j, o_j, r_j) = (v_{j+1}, o, r)$ for some $o \in \mathbb{N}$ and some request function r. By definition of the arena $\mathcal{A}' = \mathcal{A} \times \mathcal{M}$ we have $(o, r) = \mathit{upd}((o_j, r_j), (v_j, v_{j+1})) = (o_j + 1, r_{v_{j+1}})$. Thus, we obtain $(v_{j+1}, o_j + 1, r_{v_{j+1}}) \in \mathcal{R}$, which yields the desired contradiction as argued above. This concludes the proof of $o_{j+1} \leq o_j + 1$ for all $j \in \mathbb{N}$.

Finally, we prove the claim of this lemma by showing the following property by induction over j:

> If $o_j = k$, then for every $k' \leq k$ there is a reset position $j_{k'} \leq j$ with $o_{j_{k'}} = k'$.

Let us first argue that this indeed implies $o_j < n$ for all $j \in \mathbb{N}$. Towards a contradiction, assume $o_j = n$ for some $j \in \mathbb{N}$. Then, there are $n + 1$ reset positions, one for each

value k in the range $0 \leq k \leq n$ for the overflow counter. Thus, two such positions j', j'' share the same vertex $v_{j'} = v_{j''}$, but have differing values of the overflow counter $o_{j'} \neq o_{j''}$. By construction of $\mathcal{M}'$, however, we obtain $o_{j'} = o_{v_{j'}} = o_{v_{j''}} = o_{j''}$, i.e., the desired contradiction. It remains to show that the property above indeed holds true.

For the induction start, we have $o_0 = 0$ and pick $j_0 = 0$, which is a reset position. Now, let $j > 0$ and let $o_j = k$. If $k \leq o_{j-1}$, then the induction hypothesis yields the necessary positions. Hence, assume we have $k > o_{j-1}$, which implies $k = o_{j-1} + 1$ due to $o_{j+1} \leq o_j + 1$, which we showed above. Then, j is a reset position. Hence, we define $j_k = j$ and obtain the remaining $j_{k'}$ for $k' < k$ via the induction hypothesis. □

It remains to show that we indeed have $\mathrm{Cost}(\rho) > b$. As argued above, this then directly implies that for each strategy σ of Player 0 we have $\mathrm{Cost}_{v^*}(\sigma) > b$, concluding the proof of the direction from left to right of Theorem 4.8.

Proof of Theorem 4.8. The direction from right to left is encapsulated in Lemma 4.9.

For the direction from left to right, recall that we defined a strategy τ for Player 1, that we picked ρ beginning in v^* and consistent with τ arbitrarily and that we defined $\rho' = \mathrm{ext}_{\mathcal{M}'}(\rho) = (v_0, o_0, r_0)(v_1, o_1, r_1)(v_2, o_2, r_2)\cdots$. First assume that the overflow counter of ρ' eventually stabilizes, i.e., there exists some $j \in \mathbb{N}$ such that $o_{j'} = o_j$ for all $j' > j$. Then, there exists a suffix of ρ' that is consistent with τ', which therefore violates the parity condition. Hence, it suffices to note that the color sequences induced by ρ' and by ρ coincide in this case due to Lemma 4.11 and due to the construction of $\mathcal{G}_b$, in which vertices of the form (v, o, r) inherit the coloring of the vertex v for $o < n$. Thus, ρ violates the parity condition and, in turn, also the parity condition with weights with respect to any bound.

Now assume that the overflow counter of ρ' does not stabilize. Then, there are infinitely many reset positions in ρ'. We obtain that there exist a request that incurs cost greater than b between any two adjacent such positions as a direct consequence of ➔Remark 4.7.2. Hence, we obtain $\mathrm{Cost}(\rho) > b$, which concludes this direction of the proof. □ ➔Sec. 4.1, Page 91

Having thus reduced the threshold problem for parity games with weights to that of solving classical parity games, we discuss the implications of this reduction on the complexity of the former problem in the following section.

4.1.4 Complexity of Solving Parity Games with Weights Optimally

In the previous sections we have first defined threshold games and subsequently shown that solving these games suffices to solve the threshold problem for parity games with weights and encapsulated this argument in Theorem 4.8. Thus, we obtain an algorithm for solving the latter problem via a reduction to solving threshold games, which we formalize as Algorithm 4.1.

We argue that this algorithm requires at most exponential time. To this end, recall that parity games can be solved in polynomial time in the number of vertices and in exponential time in the logarithm of the number of odd colors (cf. ➔Proposition 2.20). ➔Sec. 2.3, Page 20

Algorithm 4.1

Input: Parity game with weights $\mathcal{G}$ with n vertices, d odd colors, and largest absolute weight W, vertex v of $\mathcal{G}$, bound $b \in \mathbb{N}$.

1: **if** $b \geq nd(6n)(d+1)(W+1)W$ **then**
2: **return** $v \in W_0(\mathcal{G})$ /* Requires solving $\mathcal{G}$, e.g. via Algorithm 3.1 */
3: **else**
4: $\mathcal{G}_b = b$-threshold game of $\mathcal{G}$ /* $\mathcal{G}_b$ is explicitly constructed */
5: **return** $(v, 0, r_v) \in W_0(\mathcal{G}_b)$ /* Requires solving $\mathcal{G}_b$ */
6: **end if**

Output: **True** if Player 0 has a strategy σ in $\mathcal{G}$ with $\text{Cost}_v(\sigma) \leq b$, **False** otherwise.

Moreover, recall that the (b-)threshold game $\mathcal{G}_b$ has $\mathcal{O}(n^2 b^{2d})$ many vertices, where n and d are the number of vertices and the number of odd colors in $\mathcal{G}$, respectively. As we are able to bound the value of b from above by a polynomial in n, d, and the largest absolute weight W of $\mathcal{G}$ due to → Theorem 3.38, Page 75, we obtain that we are able to solve $\mathcal{G}_b$ in exponential time.

Theorem 4.12. *The following problem is in* ExpTime*:*

> *"Given a parity game with weights $\mathcal{G}$, a vertex v of $\mathcal{G}$, and a bound $b \in \mathbb{N}$, does Player 0 have a strategy σ with $\text{Cost}_v(\sigma) \leq b$?"*

Proof. We claim that Algorithm 4.1 witnesses the claimed membership in ExpTime. The correctness of this algorithm follows directly from → Lemma 3.39, Page 75 and → Theorem 4.8, Page 91. It remains to show that Algorithm 4.1 terminates in exponential time. Let n be the number of vertices of $\mathcal{G}$, let d be the number of odd colors of $\mathcal{G}$, and let W be the largest absolute weight in $\mathcal{G}$.

If $b \geq nd(6n)(d+1)(W+1)W$, then the dominating factor for the runtime of Algorithm 4.1 is the call to a solver for parity games with weights in Line 2. This solver only has to solve the given parity game with weights as discussed in → Chapter 3, Page 29. Due to → Theorem 3.18, Page 52, the problem of solving these games is in NP ∩ coNP. Since NP ⊆ ExpTime, we obtain the desired runtime in this case.

If, however, $b < nd(6n)(d+2)(W+1)W$, Algorithm 4.1 constructs and solves the b-threshold game $\mathcal{G}_b$ of $\mathcal{G}$ in Line 4 and Line 5, respectively. Let n' be the number of vertices of $\mathcal{G}_b$. By construction of $\mathcal{G}_b$ we obtain

$$n' = n\,|\{0, \dots n\} \times R| = n(n+1)(2b^2+3b+2)^d \in \mathcal{O}(n^2(2b)^{2d}) \ ,$$

where R denotes the set of request functions as defined in → Section 4.1.2, Page 88.

Due to our case analysis based on the cardinality of b in Line 1, we furthermore obtain $b \in \mathcal{O}((ndW)^2)$, which in turn implies

$$n' \in \mathcal{O}(n^2(2(ndW)^2)^{2d}) = \mathcal{O}(n^2 2^{2d}(ndW)^{4d}) \ .$$

As we assume weights to be given in binary encoding, we additionally obtain $W \in \mathcal{O}(2^{|\mathcal{G}|})$, which finally implies

$$n' \in \mathcal{O}(n^2 2^{2d}(nd2^{|\mathcal{G}|})^{4d}) = \mathcal{O}(n^{2+4d} 2^{2d} d^{4d} 2^{4d|\mathcal{G}|}) = \mathcal{O}(n^{2+4d} 2^{2d+4d|\mathcal{G}|} d^{4d}) \ ,$$

i.e., $\mathcal{G}_b$ contains only exponentially many vertices and $d' = \max\{1, d\} \in \mathcal{O}(d)$ many colors in terms of $|\mathcal{G}|$. Recall that parity games can be solved in polynomial time in the number of vertices and in exponential time in the logarithm of the number of colors due to ➔ Proposition 2.20. Hence, $\mathcal{G}_b$ can indeed be solved in exponential time in $|\mathcal{G}|$, which implies membership of the above problem in ExpTime. □

➔ Sec. 2.3, Page 20

The above algorithm solving the threshold problem yields an algorithm determining the optimal b such that Player 0 has a strategy of cost at most b from a given v: Given a parity game with weights $\mathcal{G}$ with n vertices, d odd colors, and largest absolute weight W, and a vertex v of $\mathcal{G}$, we first solve $\mathcal{G}$ and determine whether or not $v \in W_0(\mathcal{G})$. If this is not the case, then no such bound b exists. Otherwise, the minimal b with the above property can be determined with a binary search over the range $0, \ldots, nd(6n)(d+1)(W+1)W$. This binary search solves at most $\log(nd(6n)(d+1)(W+1)W)$, i.e., polynomially many instances of the threshold problem, each of which can be solved in exponential time. Hence, the optimal b such that Player 0 has a strategy of cost at most b can be determined in exponential time in the size of $\mathcal{G}$.

At this point, there is a gap in the classification of the complexity of the threshold problem: We know the problem to be in ExpTime due to Theorem 4.12. On the other hand, we only know the problem to be at least as hard as the problem of solving energy parity games: The latter problem can be reduced to that of solving parity games with weights due to ➔ Theorem 3.31. That problem can, in turn, be reduced to the problem of solving parity games with weights optimally due to ➔ Lemma 3.39.

➔ Sec. 3.3, Page 69

➔ Sec. 3.5, Page 75

Our aim for the following sections is to close this gap by providing tight bounds on the complexity of this problem. To this end, we proceed in two directions. Firstly, we show the threshold problem for the special case of parity games with costs is in PSpace. Secondly, we provide lower bounds on the complexity of the problem by showing that even the threshold problem for finitary parity games is PSpace-hard, while the threshold problem for parity games with weights is ExpTime-hard.

These results complete the picture of the complexity of optimal play in parity games with weights: Playing optimally in finitary parity games or parity games with costs is PSpace-complete, while playing optimally in parity games with weights is ExpTime-complete.

4.2 Playing Parity Games with Costs Optimally in Polynomial Space

In the previous section we have reduced the threshold problem for parity games with weights to solving the b-threshold game $\mathcal{G}_b$. Moreover, we have argued that the former problem can be solved in exponential time.

In this section, we show that this upper bound on the complexity of solving parity games with weights optimally can be reduced to polynomial space in the special case of parity games with costs, i.e., in the case of parity games with weights in which all weights are nonnegative. Hence, for the remainder of this section, fix some parity

game with costs $\mathcal{G}$ with n vertices, d odd colors, and largest weight W, as well as some bound $b \in \mathbb{N}$. If $b \geq nW$, then the problem of solving $\mathcal{G}$ with respect to b reduces to that of solving $\mathcal{G}$ due to ➔Proposition 2.35. As the latter problem is in NP ∩ coNP, and since both complexity classes are subsumed by PSpace, the problem of solving $\mathcal{G}$ with respect to b can clearly be solved in polynomial space if $b \geq nW$. Hence, for the remainder of this section, we assume $b < nW$. Finally, let $\mathcal{G}_b$ be the b-threshold game of $\mathcal{G}$.

➔Sec. 2.4, Page 27

Recall that $\mathcal{G}_b$ is a classical parity game, albeit one of exponential size in d. We aim to solve $\mathcal{G}_b$ in polynomial space (in $|\mathcal{G}|$ and b) by reducing it to a finite variant, i.e., by declaring the winner of a play after only finitely many moves.

It is easy to see that the existence of positional winning strategies for both players in parity games allows to declare the winner of a play in a parity game after only polynomially many moves when measured in the size of $\mathcal{A}'$: When playing consistently with a positional winning strategy, neither player may "allow" traversing a cycle with odd (in the case of Player 0), or even (in the case of Player 1) maximal color, respectively: Repeating such a cycle would be consistent with the currently played positional strategy, but losing for the respective player. Hence, one can solve a parity game by solving a stricter variant in which the polarity of the first traversed cycle determines the winner of the play.

Furthermore, that stricter variant can easily be solved using an alternating Turing machine, where the existential and universal vertices take the roles of Player 0 and Player 1, respectively. Each infinite sequence of configurations of such a Turing machine then corresponds to a play of the parity game.

Thus, one can construct the alternating Turing machine above such that it accepts a run if and only if the first closed cycle has even maximal color. The Turing machine thus solves the parity game by simulating it for at most as many steps as the game contains vertices. Since APTime = PSpace due to results by Chandra, Kozen, and Stockmeyer [CKS81], one can solve parity games in polynomial space using this approach.

However, although we have reduced the problem of solving the threshold problem of $\mathcal{G}$ to that of solving the threshold game $\mathcal{G}_b$, the latter game is of exponential size in the size of the bound. Hence, using the above approach as a black box would yield a solution to the threshold problem using exponential space. Thus, in this section we show how to leverage the structure of $\mathcal{G}_b$ in order to declare a winner after only polynomially many moves.

To this end, we retain the general idea behind the above approach: We play $\mathcal{G}_b$ until one player is able to prove that they are able to enforce a cycle in $\mathcal{G}$ that is clearly beneficial to them. Thus, we first define in Section 4.2.1 what it means for a play infix in $\mathcal{G}_b$ to be beneficial for one player.

At that point, however, such an infix may yet be of exponential size in $\mathcal{G}$, as it may be necessary to traverse a smaller cycle exponentially often in order to prove that it is indeed beneficial. Hence, in Section 4.2.2, we identify so-called shortcuts in plays in $\mathcal{G}_b$ that allow players to provide a beneficial cycle faster and formally define the

finite variant $\mathcal{G}_b^f$ of $\mathcal{G}_b$.

We subsequently show in Section 4.2.3 that this shortcut mechanism indeed allows for players to win $\mathcal{G}_b^f$ quickly, i.e., after at most polynomially many moves. We moreover show that either player wins $\mathcal{G}_b$ if and only if they are able to traverse a beneficial cycle in $\mathcal{G}_b^f$.

Finally, we argue in Section 4.2.4 that solving $\mathcal{G}_b^f$ not only is equivalent to solving $\mathcal{G}_b$, but also that we are able to solve $\mathcal{G}_b^f$ using an alternating Turing machine whose runtime is polynomially bounded in the size of $\mathcal{G}$. Thus, we obtain the desired PSPACE upper bound on the complexity of playing parity games with costs optimally.

4.2.1 Dominating Cycles

We begin by defining what it means for either player to prove that they are able to enforce a beneficial cycle in $\mathcal{G}$ via a play infix in $\mathcal{G}_b$. To this end, recall that we distilled the information about the open requests in a play prefix and the costs these requests have incurred into request functions in ➔ Section 4.1.1. ➔ Page 86

Recall that in the special case of parity games with costs, we only have nonnegative weights. Thus, the weight along a play prefix is monotonically increasing. Hence, we first observe that not all open requests are "relevant" for the remainder of the play: If there are open requests for colors c and c' with $c < c'$ such that $r_\uparrow(c) \leq r_\uparrow(c')$, then every play infix that causes the cost incurred by the request for color c to violate the upper bound b has a prefix that causes the cost incurred by the request for c' to violate the upper bound. Moreover, as $\mathcal{G}$ is a parity game with costs, no edge in $\mathcal{G}$ has negative weight. Thus, the costs incurred by the requests for color c never violate the lower bound of $-b$. Hence, the request for color c is irrelevant for both players and can be ignored in this case.

This observation allows us to simplify our notation for the remainder in this section, as we only need to consider upper residual request functions. Hence, for the remainder of this section, a REQUEST FUNCTION is a function $r\colon D \to \{\bot, 0, \ldots, b\}$, where D denotes the set of odd colors of $\mathcal{G}$. The threshold game $\mathcal{G}_b$ of $\mathcal{G}$ is defined analogously to the more general case of parity games with weights, where we replace every request function r in that earlier definition with its upper residual request function $r_\uparrow$. Since the update rule used for the construction of $\mathcal{G}_b$ updates the upper and lower bounds stored by the request functions independently, it can easily be restricted to operate only on the upper bound, yielding the restricted update function used in this section.

Def. request function

Thus, we obtain the arena of the threshold game $\mathcal{G}_b$ by augmenting the arena of $\mathcal{G}$ with memory states that record the maximal cost incurred by open requests so far as well as the number of times some request has violated that bound. Each such memory element is of the form (o, r), where $o \in \{0, \ldots, n\}$ is the overflow counter and r is a request function.

As we aim to determine the situations in which either player was able to improve their situation by enforcing a cycle in $\mathcal{G}$, we first define a preorder $\sqsupseteq$ on these memory elements such that $m \sqsupseteq m'$ if m is "better" for Player 1 than m'. As the overflow counter

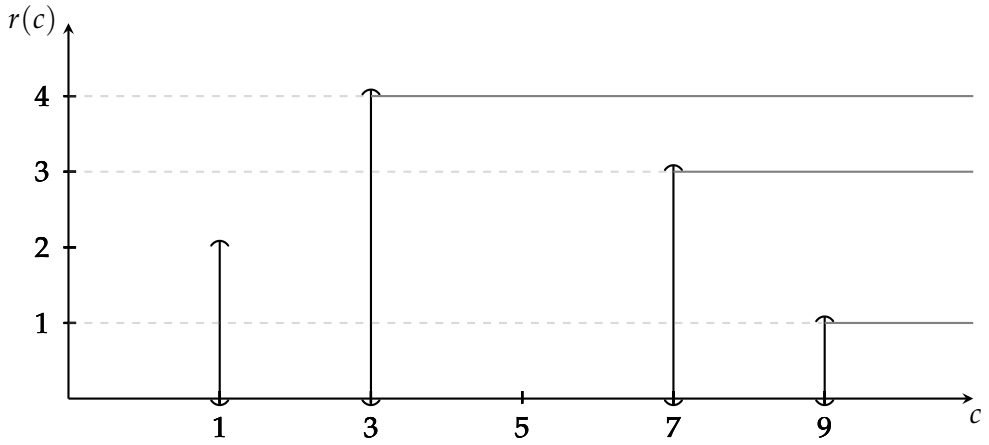


Figure 4.5: A visual representation of the request function r from Example 4.13.

is only incremented during a play and since each increment brings the play closer to the winning sink for Player 1, we directly obtain that increasing the overflow counter is beneficial for Player 1. It remains to define a preorder on the request functions.

To this end, let r be a request function. Recall that we observed that not all open requests are relevant for the players. We now formalize this observation.

Def. relevant
Def. $\textsc{RelReq}(r)$

Formally, we say that a request for color c is RELEVANT in r if for all $c' \geq c$ we have $r(c') < r(c)$, where we use $\bot < n$ for all $n \in \{0, \dots, b\}$. We write $\textsc{RelReq}(r)$ to denote the set of relevant requests of r.

Example 4.13. Let r be the request function defined as $r(1) = 2$, $r(3) = 4$, $r(5) = \bot$, $r(7) = 3$, and $r(9) = 1$. We illustrate this request function in Figure 4.5. The set of relevant requests of r is $\textsc{RelReq}(r) = \{3, 7, 9\}$. This illustration demonstrates the intuitive characterization of relevant requests as those requests for which there exists no request for a larger color that has incurred higher cost. △

Intuitively, Player 1 can improve his situation by either opening new relevant requests, or by increasing the cost incurred by already opened relevant requests. Formally, let r and r' be two request functions. We say that r IS DOMINATED BY r' if for each $c \in \textsc{RelReq}(r)$ there exists some $c' \geq c$ such that $r'(c') \geq r(c)$. If r' is dominated by r, we write $r' \sqsubseteq r$. If $r' \sqsubseteq r$ and $r' \sqsupseteq r$ both hold true, we write $r \approx r'$.

Def. is dominated by

Def. $\sqsubseteq$

Remark 4.14.

1. *The relation $\sqsubseteq$ is reflexive.*
2. *The relation $\sqsubseteq$ is transitive.*
3. *If a vertex (v, o, r) of $\mathcal{G}_b$ has incoming edges, then $r_v \sqsubseteq r$.*

The requirement for incoming edges in the third item of the above remark follows from our definition of $\mathcal{G}_b$: For the sake of simplicity, the game $\mathcal{G}_b$ contains, e.g., vertices (v, o, r), where v carries an odd color, but r maps all colors to $\bot$. Such vertices

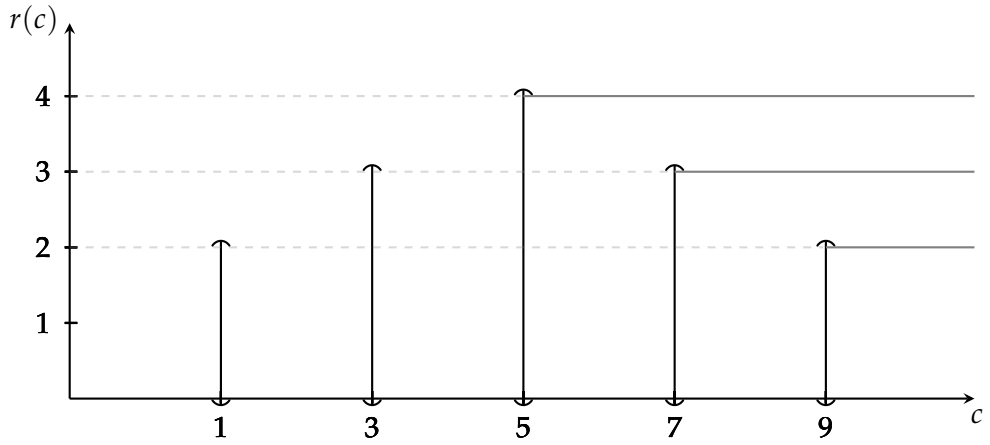


Figure 4.6: A visual representation of the request function r' from Example 4.15.

are, however, not reachable, as each play prefix beginning in some vertex $(v, init(v))$ and ending in a vertex of odd color has an open request for that color.

Example 4.15. Let r be the request function as defined in Example 4.13 and let r' be the request function defined as $r'(1) = 1$, $r'(3) = 4$, $r'(5) = 2$, $r'(7) = 3$, and $r'(9) = 2$. We illustrate the function r' in Figure 4.6.

Although the requests for color 3 have incurred lower cost in r' than they have in r, there are open requests for color 5 that have incurred as much cost as the requests for color 3 in r. Moreover, the costs for requests for color 9 in r' exceed those for the same color in r. Hence, we obtain $r \sqsubseteq r'$. △

Following the above intuition, we extend the definition of the relation $\sqsupseteq$ to memory elements: The memory element (o, r) dominates (o', r') if either $o > o'$, or if $o = o'$ and $r \sqsupseteq r'$. Similarly, we extend the notation of $\approx$ such that $(o, r) \approx (o', r')$ if and only if both $(o, r) \sqsubseteq (o', r')$ and $(o, r) \sqsupseteq (o', r')$ hold true.

We first show that the relation $\sqsupseteq$ over memory states is preserved under concatenation of vertices from $\mathcal{A}$ and the deterministic memory update defining the arena $\mathcal{A}'$ of the threshold game.

Lemma 4.16. *Let $(v, o_1, r_1), (v, o_2, r_2)$ be vertices in $\mathcal{G}_b$, let $(v, v') \in E$, and, for $j \in \{1, 2\}$, let $upd((o_j, r_j), (v, v')) = (o'_j, r'_j)$. If $(o_1, r_1) \sqsubseteq (o_2, r_2)$ and $o_2 < n$, then $(o'_1, r'_1) \sqsubseteq (o'_2, r'_2)$.*

Proof. First assume $o_1 < o_2$. Due to the construction of $\mathcal{G}_b$, this implies $o'_1 \leq o'_2$. If $o'_1 < o'_2$, the statement clearly holds true. If $o'_1 = o'_2$, however, then $r'_1 = r_{v'}$ and thus $(o'_1, r'_1) \sqsubseteq (o'_2, r'_2)$, due to Remark 4.14.3.

Now, assume $o_1 = o_2$ and $r_1 \sqsubseteq r_2$ for the remainder of this proof. First, we consider the case $o'_1 = o_1 + 1$ and show that this assumption implies $o'_2 = o_2 + 1$. This then implies $r'_1 = r'_2 = r_{v'}$ and hence, $(o'_1, r'_1) = (o'_2, r'_2)$, which suffices due to Remark 4.14.1. Since the move from v to v' causes an overflow when starting in (v, o_1, r_1), i.e., since we have $o'_1 = o_1 + 1$, we have $\text{Weight}(v, v') = w > 0$. Let c_1 be some color that causes the

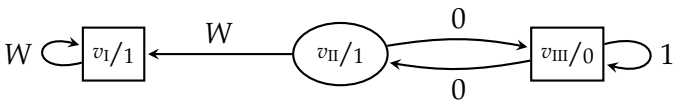


Figure 4.7: A parity game with costs witnessing necessity of shortcuts.

overflow in r_1, i.e., let $c_1 \in \{c \mid r_1(c) > b - w\}$. Since $r_2 \sqsupseteq r_1$, there exists a color $c_2 \geq c_1$ with $r_2(c_2) \geq r_1(c_1)$.

Furthermore, since the range of r_2 is bounded by b from above, this implies $b \geq r_2(c_2) > b - w$. Hence, the move from v to v' also causes an overflow in r_2, i.e., we have $o_2' = o_2 + 1$. This completes the proof in the case $o_1' = o_1 + 1$ as argued above.

Now consider the case $o_1' = o_1$. We distinguish two subcases: If $o_2' = o_2 + 1$, then $o_1' < o_2'$ due to the assumption $o_1 = o_2$ and hence, $(o_1', r_1') \sqsubseteq (o_2', r_2')$. On the other hand, if $o_2' = o_2$, then let c_1 be a relevant request in r_1', i.e., pick $c_1 \in \textsc{RelReq}(r_1')$ arbitrarily. We show that there exists a color $c_2 \geq c_1$ with $r_2'(c_2) \geq r_1'(c_1)$, which concludes the proof: If $c_2 \notin \textsc{RelReq}(r_2')$, then, by the definition of relevant requests, there exists some color $c_3 \in \textsc{RelReq}(r_2')$ such that $c_3 > c_2$ and such that $r_2'(c_3) \geq r_2'(c_2)$, which in turn implies $r_2'(c_3) \geq r_1'(c_1)$. Since this holds true for all relevant requests of r_1', this then concludes the proof.

First assume that a request for c_1 was already open in r_1, i.e., assume that $r_1(c_1) \neq \bot$. Let $c_2 \geq c_1$ such that $r_2(c_2) \geq r_1(c_1)$. Such a color c_2 exists due to $r_2 \sqsupseteq r_1$. Since the request for c_1 was not answered during the move to v', and since $o_2 = o_2' < n$, neither was the request for c_2 during the same move. Hence, we have $r_2'(c_2) \geq r_1'(c_1)$. If, however, a request for c_1 was not already open in r_1, then the request for c_1 was opened by moving to v', i.e., we have $\Omega(v') = c_1$. Thus, we directly obtain $r_1'(c_1) = 0$ and $r_2'(c_1) \geq 0$. Picking $c_2 = c_1$ concludes the proof in this case. □

We now leverage the preorder $\sqsubseteq$ in order to define what it means for either player to enforce a witness of a beneficial cycle (in $\mathcal{A}$) by playing in $\mathcal{A}'$. To this end, let $\pi = (v_0, o_0, r_0) \cdots (v_j, o_j, r_j)$ be a sequence of vertices in $\mathcal{G}_b$. We say π is a DOMINATING CYCLE if $v_0 = v_j$, $o_0 = o_j < n$, and either

Def. dominating cycle

- the maximal color occurring on π' is even and $r_0 \sqsupseteq r_j$, or
- the maximal color occurring on π' is odd and $r_0 \sqsubseteq r_j$.

We call the former and latter type of dominating cycles even and odd, respectively. A dominating cycle is only a cycle when projected to a play in $\mathcal{A}$, i.e., it is a cycle in the original parity game with costs $\mathcal{G}$.

We aim to define the finite variant $\mathcal{G}_b^f$ of $\mathcal{G}_b$ such that the goal of Player 0 is to enforce an even dominating cycle, while Player 1 aims to enforce an odd dominating cycle. Even if both players play consistently with positional strategies, however, it may take exponentially many moves until such a cycle is closed.

Example 4.17. For some $W \in \mathbb{N}$, let $\mathcal{G}_W$ be the parity game with costs shown in Figure 4.7 and let $b \in \mathbb{N}$. Player 1 wins $\mathcal{G}_W$ from v_{II} with respect to b: If Player 0 at some point opts to move to v_{I}, then Player 1 wins, as the resulting play violates the

parity condition. If Player 0, however, always moves to v_{III} from v_{II}, then Player 1 can take the self-loop of v_{III} $b+1$ often, thus enforcing costs of answering the requests for color one posed when visiting vertex v_{II} to exceed b.

For $0 \leq j \leq b$ define r_j as the unique request function that satisfies $r_j(1) = j$ and consider the play prefix $(v_{\text{II}}, r_0)(v_{\text{III}}, r_0)(v_{\text{III}}, r_1) \cdots (v_{\text{III}}, r_b)(v_{\text{II}}, b)$ in the threshold game $\mathcal{G}_{b,W}$. This play starts in $(v_{\text{II}}, \mathit{init}(v_{\text{II}}))$, is consistent with a positional strategy for Player 1 in $\mathcal{G}_{b,W}$, comprises an odd dominating cycle of length $b+3$, but contains no dominating cycle of length less than $b+3$. As b is only bounded from above by $\mathcal{O}(W)$, this dominating cycle may be exponential in the size of the problem description. Hence, even playing consistently with positional strategies, Player 1 may take exponentially many moves in $\mathcal{G}_{b,W}$ before producing a dominating cycle, i.e., a witness of him winning $\mathcal{G}_W$ from v_{II} with respect to b. △

In order to allow both players to produce dominating cycles in alternating polynomial time, in the next section we develop a way to skip over long infixes in $\mathcal{G}_b$ whose projection to the arena of $\mathcal{G}$ is a cycle.

4.2.2 Taking Shortcuts in Threshold Games

In the previous section we have defined even and odd dominating cycles and informally argued that traversing an even dominating cycle is beneficial for Player 0, while traversing an odd such cycle is beneficial for Player 1. Hence, intuitively, Player 0 wins $\mathcal{G}_b$ if she is able to enforce traversing an even dominating cycle before an odd such cycle is traversed. We have, however, also seen in Example 4.17 that enforcing a dominating cycle may require exponentially many steps.

Recall that it is, in contrast, our aim to witness dominating cycles in at most polynomially many steps. Thus, we now define a finite variant $\mathcal{G}_b^f$ of $\mathcal{G}_b$ that skips infixes during which the costs incurred by the relevant requests increase, but the set of these requests is stable. We show that this enforces dominating cycles to occur after at most polynomially many steps. Moreover, the game ends once a dominating cycle occurs and the player for whom that cycle is beneficial is declared the winner. Thus, the winner of a play in $\mathcal{G}_b^f$ is determined after polynomially many steps, which allows us to solve it in polynomial space.

Formally, we say that a sequence of vertices $\pi = (v_0, o_0, r_0) \cdots (v_j, o_j, r_j)$ in $\mathcal{A}'$ satisfies the SHORTCUT CRITERION if

Def. shortcut criterion

- $v_0 = v_j$, if
- $o_0 = o_j$, if
- $\textsc{RelReq}(r_0) = \textsc{RelReq}(r_{j'}) \neq \emptyset$ for all j' with $0 \leq j' \leq j$, if
- $\text{Weight}(\pi) > 0$, and if
- $r_j(c^*) + \text{Weight}(\pi) \leq b$ for $c^* = \arg\max_c r_j(c)$.

Since the overflow counter remains constant during π, we have that the condition $r_j(c^*) + \text{Weight}(\pi) \leq b$ is equivalent to $\text{Weight}(\pi) \leq \frac{b - r_0(c^*)}{2}$: Intuitively, we demand that traversing π does not "close more than half the distance" between the cost incurred by the request for the color that has incurred the largest cost at the end of π

and the upper bound b.

For the sake of readability, we refrain from defining the arena underlying $\mathcal{G}_b^f$ formally. We rather define the set of play prefixes of $\mathcal{G}_b^f$ inductively, which are subsequences of plays in $\mathcal{A}'$. In particular, the vertices in $\mathcal{G}_b^f$ inherit the coloring from $\mathcal{G}_b$. First, for each vertex v of $\mathcal{A}$, the play prefix of length one $(v, \mathit{init}(v)) = (v, 0, r_v)$ is a play prefix of $\mathcal{G}_b^f$. Now, let

$$\pi = (v_0, o_0, r_0) \cdots (v_j, o_j, r_j)$$

be a play prefix of $\mathcal{G}_b^f$ and let (v', o', r') be a successor of (v_j, o_j, r_j) in $\mathcal{A}'$. If there exists no j' such that the infix

$$\pi' = (v_{j'}, o_{j'}, r_{j'}) \cdots (v_j, o_j, r_j) \cdot (v', o', r')$$

of π satisfies the shortcut criterion, then $\pi \cdot (v', o', r')$ is a play prefix of $\mathcal{G}_b^f$. If, however, such a j' exists, let it be the maximal one, let

- $c^* = \arg\max_c r'(c)$, let
- $s = \text{Weight}(\pi')$, and let
- $t = \max\{t' > 0 \mid r'(c^*) + s \cdot t' \leq b\}$.

The set used in the definition of t is not empty due to the fifth condition in the definition of the shortcut condition. Hence, we have $t \geq 1$. We then define the request function r^* via

$$r^*(c) = \begin{cases} r'(c) + s \cdot t & \text{if } r'(c) \neq \bot, \text{ and} \\ \bot & \text{otherwise.} \end{cases}$$

Then, the sequence of vertices from $\mathcal{A}'$

$$(v_0, o_0, r_0) \cdots (v_j, o_j, r_j)(v', o', r^*)$$

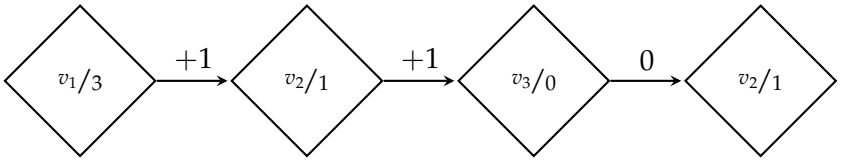
is a play prefix of $\mathcal{G}_b^f$. Intuitively, moving from (v_j, o_j, r_j) directly to (v', o', r^*) instead of (v', o', r') summarizes t traversals of the cycle $v_j \cdots v'$, each of which incurs weight s. Hence, we define the weight of the transition from (v_j, o_j, r_j) to (v', o', r^*) as $\text{Weight}(v_j, v') + s \cdot t$ in $\mathcal{G}_b^f$. Furthermore, we redefine the notion of the weight of a play prefix accordingly such that we obtain uniform notation.

Example 4.18. Let $b = 5$ and consider the play prefix shown in Figure 4.8a. We show the corresponding play prefix in $\mathcal{G}_b$ in Figure 4.8b, where the request functions r_1 through r_4 are defined as

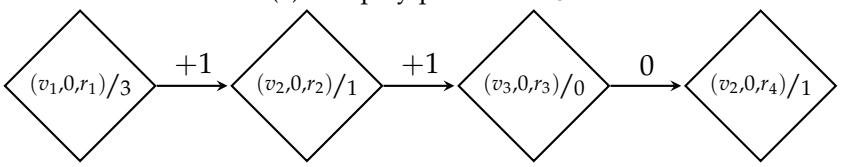
$$r_1 = (1 \mapsto \bot, 3 \mapsto 0) \qquad r_2 = (1 \mapsto 0, 3 \mapsto 1)$$
$$r_3 = (1 \mapsto 1, 3 \mapsto 2) \qquad r_4' = (1 \mapsto 1, 3 \mapsto 2) \ .$$

The infix $(v_2, 0, r_2), (v_3, 0, r_3), (v_2, 0, r_4)$ of π' satisfies the shortcut condition. Thus, the play prefix in $\mathcal{G}_b^f$ corresponding to the play prefix π in $\mathcal{G}$ is the one shown in Figure 4.8c, where

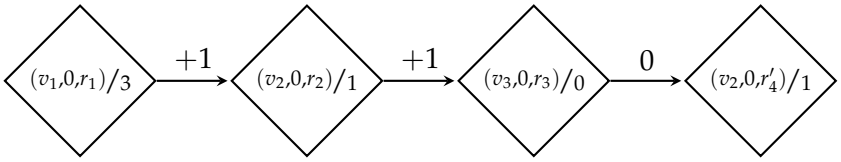
$$r_4' = (1 \mapsto 4, 3 \mapsto 5) \ . \qquad \triangle$$



(a) The play prefix π in $\mathcal{G}$.



(b) The play prefix π' corresponding to π in $\mathcal{G}_b$.



(c) The play prefix π'_f corresponding to π' in $\mathcal{G}_b^f$.

Figure 4.8: The play prefixes discussed in Example 4.18.

When talking about plays in $\mathcal{G}_b^f$, in order to concisely talk about the influence of the shortcuts in the construction of the play, we use the following notions:

- The transition from (v_j, o_j, r_j) to (v', o', r^*) is a SHORTCUT. **Def.** shortcut
- The infix $(v_{j'}, o_{j'}, r_{j'}) \cdots (v_j, o_j, r_j)(v', o', r^*)$ is a SHORTCUT CYCLE, with the DESTINATION (v', o', r^*). **Def.** shortcut cycle **Def.** destination
- The infix $(v_{j'}, o_{j'}, r_{j'}) \cdots (v_j, o_j, r_j)(v', o', r')$ is the DETOUR corresponding to the shortcut cycle, with DESTINATION (v', o', r'). **Def.** detour **Def.** destination

Similarly to dominating cycles, a shortcut cycle is only a cycle when projected to its first component, i.e., when projected to the arena $\mathcal{A}$ of the original parity game with costs $\mathcal{G}$.

Example 4.19. Consider again the play prefixes π' and π'_f shown in Example 4.18. The transition from $(v_3, 0, r_3)$ to $(v_4, 0, r'_4)$ is a shortcut. Furthermore, the play infix $(v_2, 0, r_2)(v_3, 0, r_3)(v_2, 0, r'_4)$ is a shortcut cycle with destination $(v_4, 0, r'_4)$, while the play infix $(v_2, 0, r_2)(v_3, 0, r_3)(v_2, 0, r_4)$ is the detour corresponding to the above shortcut cycle and has destination $(v_2, 0, r_4)$. △

As taking a shortcut only increases the costs of already open requests, doing so does not influence the relevant requests. Moreover, intuitively, taking a shortcut is beneficial for Player 1.

Remark 4.20. *Let $(v_j, o_j, r_j)(v', o', r^*)$ be a shortcut with (v', o', r') as the destination of its corresponding detour. Then, we have*

1. $\textsc{RelReq}(r') = \textsc{RelReq}(r^*)$, *as well as*
2. $r' \sqsubseteq r^*$.

Moreover, if c^* is the open request that has incurred the highest cost in $r_{j'}$, taking this shortcut "closes at least half the distance" between $r_{j'}(c^*)$, i.e., the largest cost incurred by any open request at the beginning of the shortcut cycle, and b. Formally, we have

$$r^*(c^*) \geq r_{j'}(c^*) + \frac{b - r_{j'}(c^*)}{2} \ .$$

Hence, no infix π containing a shortcut π' satisfies the shortcut criterion, as the cost of π' is already larger than half the cost that would cause an overflow. Thus, π violates the fifth condition in the definition of the shortcut criterion.

While a shortcut may thus not be part of another shortcut, it may, however, be part of a dominating cycle. In fact, if the maximal color along the detour associated with some shortcut is odd, then the shortcut cycle is an odd dominating cycle.

Having thus defined the play prefixes occurring in $\mathcal{G}_b^f$, it remains to define its winning condition: To this end, let $\rho = (v_0, o_0, r_0)(v_1, o_1, r_1)(v_2, o_2, r_2)\cdots$. We say that ρ is

Def. settled

SETTLED if either $o_j = n$ for some position $j \in \mathbb{N}$, or if there exist some $j \leq j'$ such that $(v_j, o_j, r_j)\cdots(v_{j'}, o_{j'}, r_{j'})$ is a dominating cycle. As argued before, this dominating cycle may include shortcuts.

Intuitively, both players aim to settle the play in $\mathcal{G}_b^f$: Player 0 aims to settle the play due to an even dominating cycle, while Player 1 aims to either saturate the overflow counter, or to traverse an odd dominating cycle. In order to formally define the winning condition of $\mathcal{G}_b^f$, we first show in the next section that every play in $\mathcal{G}_b^f$ is settled after at most polynomially many moves.

4.2.3 Settling Plays in Polynomial Time

We now show that there exists a polynomial upper bound on the length of unsettled play prefixes in $\mathcal{G}_b^f$. Recall that we defined n and W as the number of vertices of $\mathcal{G}$ and as the largest absolute weight occurring in $\mathcal{G}$. Moreover, fix $\ell = (\log(nW)+1)(n+1)^6$ and note that the value of ℓ is polynomial in the size of $\mathcal{G}$.

Lemma 4.21. *Let π be a play prefix of $\mathcal{G}_b^f$. If $|\pi| > \ell$, then π is settled.*

Proof. Let $\pi = (v_0, o_0, r_0)\cdots(v_j, o_j, r_j)$ be an unsettled play prefix of $\mathcal{G}_b^f$. We show $|\pi| \leq \ell$, which implies the given statement. First, recall that the relation $\sqsubseteq$ is reflexive, hence a vertex repetition in π implies that π contains a dominating cycle. As the occurrence of such a cycle causes π to be settled, we obtain that π does not contain a vertex repetition.

We sketch the structure of our argument in Figure 4.9, where we draw the following vertices of interest in gray: We recall the definition of overflow positions, define debt-free, request-adding, relevance-reducing, and halving positions, and show

1. that there are at most n overflow positions in π,
2. that there are at most n debt-free positions between any two adjacent overflow positions,

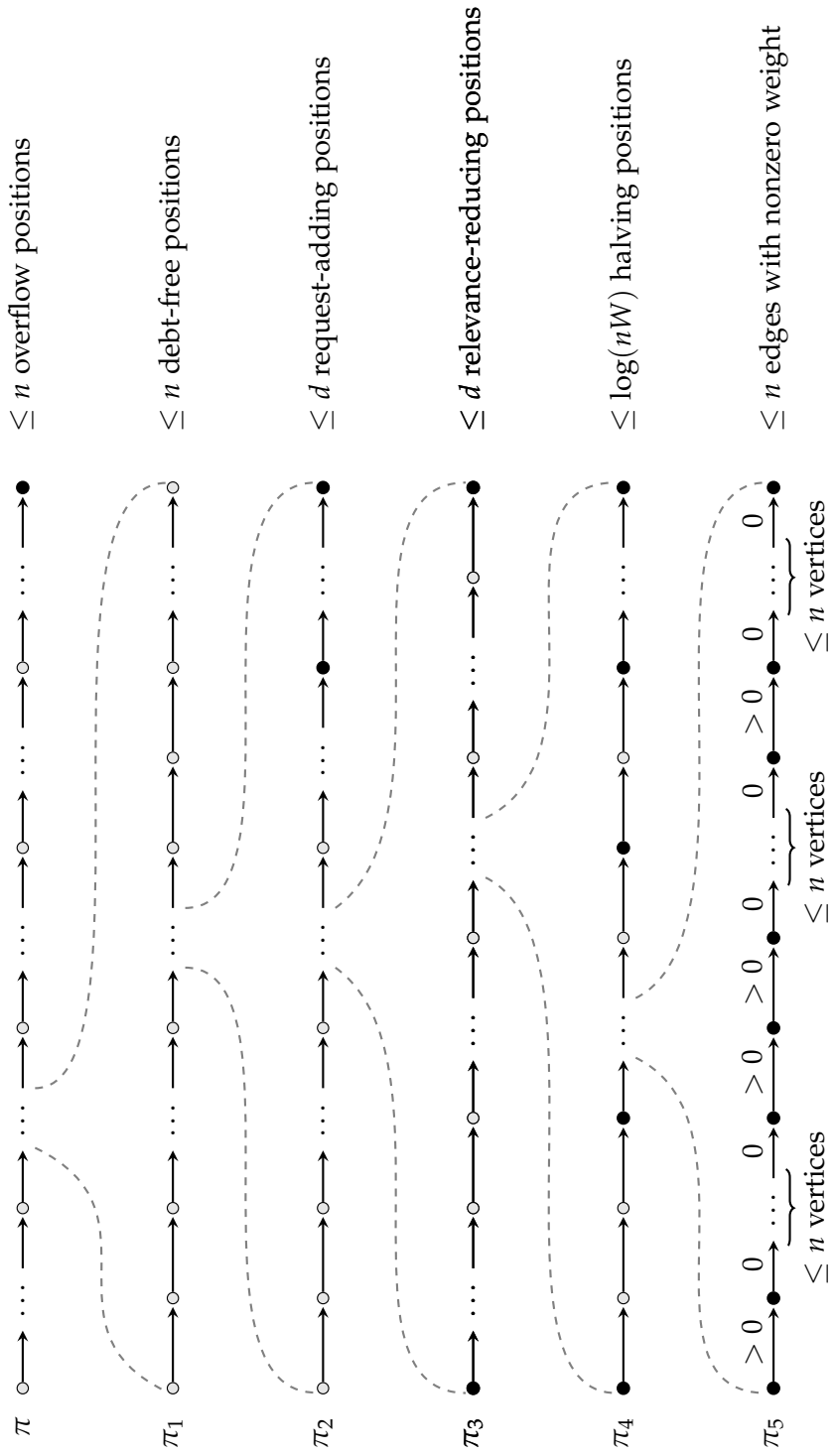


Figure 4.9: The structure of the argument for the proof of Lemma 4.21.

3. that there are at most d request-adding positions between any two adjacent debt-free positions, where d is the number of odd colors occurring in $\mathcal{G}$,
4. that there are at most d relevance-reducing positions between any two adjacent request-adding positions,
5. that there are at most $\log(nW)$ halving positions between any two adjacent relevance-reducing positions,
6. that there are at most n edges of nonzero weight between any two adjacent halving positions, and
7. that there are at most n vertices between two such edges of nonzero weight.

First, recall that an overflow position of π is a k with $k = 0$ or with $o_k = o_{k-1} + 1$. As π is unsettled and the o_k are non-decreasing, π has at most n overflow positions, $n - 1$ real increments and the initial position. Hence, by splitting π at the overflow positions we obtain at most n non-empty infixes of π, each without overflow positions. We say such an infix has type 1.

Def. debt-free position

Fix a non-empty type 1 infix π_1. A DEBT-FREE POSITION of π is a k with $r_k = r_{v_k}$, i.e., a position that has no other costs than those incurred by visiting v_k. As all vertices of π_1 share the same overflow counter value, there are at most n debt-free positions in π_1: $n + 1$ such positions would induce a vertex repetition, which we have ruled out above. Hence, by splitting π_1 at the debt-free positions we obtain at most $n + 1$ non-empty infixes of π_1, each without debt-free and overflow positions. We say such an infix has type 2.

Def. request-adding position

Fix a non-empty type 2 infix π_2. A REQUEST-ADDING POSITION of π is a k with odd $\Omega(v_k)$ such that $r_{k-1}(c) = \bot$ for all $c \geq \Omega(v_k)$. Recall that we defined d as the number of odd colors assigned by Ω. We claim that there are at most d request-adding positions in π_2. Assume towards a contradiction there are $d + 1$ such positions. Then, two request-adding positions $k < k'$ share a color, call it c. As k' is request-adding, only requests strictly smaller than c are open at position $k' - 1$, i.e., c and all larger requests have to be answered in between k and k'. Hence, there is a debt-free position between k and k', which contradicts π_2 being of type 2. Hence, by splitting π_2 at the request-adding positions we obtain at most $d + 1$ non-empty infixes of π_2, each without request-adding, debt-free, and overflow positions. We say such an infix has type 3.

Def. relevance-reducing position

Fix a non-empty type 3 infix π_3. A RELEVANCE-REDUCING POSITION of π is a k such that $\textsc{RelReq}(r_{k-1}) \supset \textsc{RelReq}(r_k)$. We show that π_3 contains at most d relevance-reducing positions. To this end, we first argue that there is at least one request that is open throughout π_3. We first observe that some request must be open at the beginning of π_3, as otherwise the first vertex of π_3 would be at a debt-free position, which do not occur in π_3. Let c^* be the maximal color for which there is an open request at the beginning of π_3. Due to π_3 not containing debt-free nor request-adding positions, all colors c visited during π_3 satisfy $c \leq c^*$. Hence, the request for c^* remains open throughout π_3. We now show that the sets of relevant requests along π_3 form a descending chain in the subset-relation. Assume towards a contradiction that an infix $(v, o, r) \cdot (v', o', r')$ of π_3 and a color c exist such that $c \notin \textsc{RelReq}(r)$, but $c \in \textsc{RelReq}(r')$. This directly implies $\Omega(v') = c$. Then, $c > c^*$, i.e., there exists a request-adding

position in π_3, a contradiction. Hence, the sets of relevant requests indeed form a descending chain. Since at the beginning of π_3 at most d requests are relevant, there are at most d relevance-reducing positions. Thus, by splitting π_3 at its relevance-reducing positions, we obtain at most $d+1$ non-empty infixes, each without relevance-reducing, request-adding, debt-free, and overflow positions. We say such an infix has type 4.

Fix a non-empty type 4 infix π_4 and recall that there is at least one request continuously open throughout π_4, as each type 4 infix is also of type 3. Let c^* be the request that has incurred the largest cost at the beginning of π_4 and observe that the request for c^* is relevant. Hence, that request is continuously open throughout π_4, as π_4 contains no relevance-reducing positions. We denote the cost incurred by the request for c^* at the beginning of π_4 by s and define the first HALVING POSITION as the minimal position k where $r_k(c^*) \geq s + \frac{b-s}{2}$ holds true, if such a position exists at all. Inductively, if k is a halving position, then the minimal $k' > k$ with $r_{k'}(c^*) \geq r_k(c^*) + \frac{b-r_k(c^*)}{2}$ is a halving position as well, if it exists. Since at each halving position the difference between the currently incurred cost of c^* and the bound b has been halved when compared to the previous halving position, there exist at most $\log(b) \leq \log(nW)$ many such positions. Hence, splitting π_4 at its halving positions yields at most $\log(nW)+1$ many infixes without overflow, debt-free, request-adding, relevance-reducing or halving positions. We say such an infix has type 5.

Def. halving position

Fix a non-empty type 5 infix π_5. We show that π_5 contains at most n edges of nonzero weight. Towards a contradiction assume that it contains $n+1$ such edges and let c^* be a request that has incurred the largest cost s at the beginning of π_5. Since some request must be open at the beginning of π_5 due to similar reasoning as above, such a color c^* exists. Since there exist more than n edges of nonzero weight, there exist two such edges leading to the vertices (v_k, o_k, r_k) and $(v_{k'}, o_{k'}, r_{k'})$ with $v_k = v_{k'}$ and $k < k'$. If the weight of the infix $(v_k, o_k, r_k) \cdots (v_{k'}, o_{k'}, r_{k'})$ is at least $\frac{b-s}{2}$, then π_5 contains a halving position, which yields the desired contradiction. If the weight is lower, however, then this infix satisfies the shortcut condition, since $\pi_{4'}$ does not contain overflow positions, the relevant requests are stable throughout $\pi_{4'}$, and since the edge leading to $(v_{k'}, o_{k'}, r_{k'})$ has nonzero weight. Hence, the vertex $(v_{k'}, o_{k'}, r_{k'})$ is the destination of a shortcut cycle, which contradicts the infix $(v_k, o_k, r_k) \cdots (v_{k'}, o_{k'}, r_{k'})$ having a weight of less than $\frac{b-s}{2}$. Thus, π_5 contains at most n edges of nonzero weight and, by splitting π_5 at these edges, we obtain a decomposition of π_5 into at most $n+1$ infixes, each without edges of nonzero weight and without halving, request-adding, debt-free, and overflow positions. We say such an infix has type 6.

Fix a non-empty type 6 infix π_6. We show that π_6 is of length at most n. Assume towards a contradiction that π_6 contains at least $n+1$ vertices. Then there exists an infix $\pi' = (v, o, r) \cdots (v, o, r')$ of π_6, since π_6 does not contain overflow positions. As π_6 is of type 6, it does not contain request-adding nor relevance-reducing positions, hence we have $\textsc{RelReq}(r) = \textsc{RelReq}(r')$ as argued above. Moreover, as π_6 only contains edges of weight zero, we furthermore obtain $r \approx r'$. Thus, π' is a dominating cycle, which contradicts π being unsettled. Hence, π_6 is of length at most n.

Aggregating all these bounds yields an upper bound of $(\log(nW)+1)(n+1)^6$ on

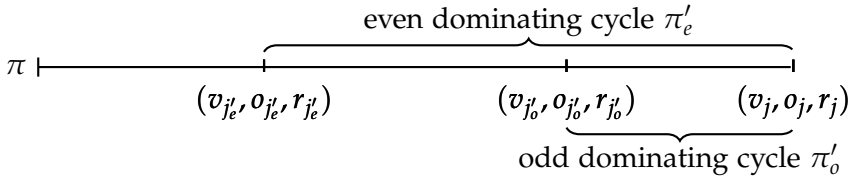


Figure 4.10: The situation occurring in the proof of Lemma 4.22.

the length of the unsettled play prefix π, as we have $d \leq n$. □

Due to Lemma 4.21 every infinite play in $\mathcal{G}_b^f$ is settled. Thus, we can now formally define the winning condition Win_f of $\mathcal{G}_b^f$. Let $\rho = (v_0, o_0, r_0)(v_1, o_1, r_1)(v_2, o_2, r_2) \cdots$ be a play of $\mathcal{G}_b^f$ and let j be the minimal position such that $\pi = (v_0, o_0, r_0) \cdots (v_j, o_j, r_j)$ is settled. Then π is either settled due to $o_j = n$, or because it ends in an even or odd dominating cycle. These three cases are mutually exclusive.

Lemma 4.22. *Exactly one of the following holds true:*
1. *We have $o_j = n$,*
2. *there exists $j' < j$ such that $(v_{j'}, o_{j'}, r_{j'}) \cdots (v_j, o_j, r_j)$ is an odd dominating cycle, or*
3. *there exists $j' < j$ such that $(v_{j'}, o_{j'}, r_{j'}) \cdots (v_j, o_j, r_j)$ is an even dominating cycle.*

Proof. As π is the minimal settled prefix of ρ, clearly at least one of the above conditions holds true. It remains to show that at most one of the conditions holds true.

To this end, we first observe that the first condition holding true implies that neither the second nor the third condition hold true. This is due to the fact that $o_j = n$ and π being the minimal settled prefix of ρ, we obtain $o_{j'} < n$ for all $j' < n$. Hence, π does not contain a dominating cycle, as this would contradict its minimality.

As a second step, we now show that the conjunction of the second and the third condition cannot hold true, which concludes the proof. Towards a contradiction, assume that both the second and the third condition hold true and let j'_e and j'_o be positions such that $\pi'_e = (v_{j'_e}, o_{j'_e}, r_{j'_e}) \cdots (v_j, o_j, r_j)$ is an even dominating cycle and such that $\pi'_o = (v_{j'_o}, o_{j'_o}, r_{j'_o}) \cdots (v_j, o_j, r_j)$ is an odd dominating cycle. Clearly, we have $j'_e \neq j'_o$. Assume $j'_e < j'_o$. The other case, i.e., $j'_e > j'_o$, is dual. We illustrate this situation in Figure 4.10.

Since π'_e and π'_o are even and odd dominating cycles, respectively, we obtain $r_{j'_e} \sqsupseteq r_j$ and $r_{j'_o} \sqsubseteq r_j$. As the relation $\sqsubseteq$ is transitive due to ➔Remark 4.14.1, Page 100 this implies $r_{j'_e} \sqsupseteq r_{j'_o}$. We show that the largest color in the infix $(v_{j'_e}, o_{j'_e}, r_{j'_e}) \cdots (v_{j'_o}, o_{j'_o}, r_{j'_o})$ is even, which implies that this infix is an even dominating cycle and thus contradicts π being minimally settled.

Let j_{c_e} and j_{c_o} with $j'_e \leq j_{c_e} \leq j$ and $j'_o \leq j_{c_o} \leq j$ be positions of vertices that carry the largest color in π'_e and π'_o, respectively. By assumption, $\Omega(v_{j_{c_e}}) = c_e$ and $\Omega(v_{j_{c_o}}) = c_o$ are even and odd, respectively. Towards a contradiction, assume $j_{c_e} \geq j'_o$, i.e., assume that there exists a vertex in π'_o that carries the largest (even) color in π'_e. If $c_e > c_o$, then the largest color in π'_o is even, which contradicts π'_o being an odd dominating cycle. If, however, $c_e < c_o$, then the color c_o is odd and larger than the largest even color

occurring in π'_e. As π'_o is a suffix of π'_e, this contradicts π'_e being an even dominating cycle.

Thus, we obtain $j_{c_e} \leq j'_o$, i.e. the largest color in the infix $(v_{j'_e}, o_{j'_e}, r_{j'_e}) \cdots (v_{j'_o}, o_{j'_o}, r_{j'_o})$ is even. Hence, this infix is an even dominating cycle, as argued above, which contradicts π being the minimal settled prefix of ρ and concludes the proof. □

If π is settled due to ending in an even dominating cycle, we proclaim ρ to be winning for Player 0 in $\mathcal{G}_b^f$. Otherwise, i.e., if π is settled due to saturating the overflow counter or due to ending in an odd dominating cycle, then we proclaim ρ to be winning for Player 1 in $\mathcal{G}_b^f$. This concludes the definition of $\mathcal{G}_b^f$.

Due to Lemma 4.21 the winner in $\mathcal{G}_b^f$ is decided after at most ℓ many moves. Hence $\mathcal{G}_b^f$ is indeed a game of finite duration. This directly implies that $\mathcal{G}_b^f$ is determined due to Zermelo [Zer13].

Corollary 4.23. *The game $\mathcal{G}_b^f$ is determined.*

Having thus defined $\mathcal{G}_b^f$, it remains to show that solving $\mathcal{G}_b^f$ is indeed equivalent to solving $\mathcal{G}_b$ and that we can solve $\mathcal{G}_b^f$ using only polynomial space in the size of $\mathcal{G}$. We do so in the following section.

4.2.4 Solving Threshold Games in Polynomial Space

In this section we conclude the proof of PSPACE-membership of the threshold problem for parity games with costs. To this end, we first show that solving $\mathcal{G}_b$ is indeed equivalent to solving $\mathcal{G}_b^f$. Subsequently, we show how to solve $\mathcal{G}_b^f$ on-the-fly given only $\mathcal{G}$ and b, i.e., without explicitly constructing it. This on-the-fly technique requires only polynomial space in the size of $\mathcal{G}$ and thus yields the desired result.

To show equivalence between $\mathcal{G}_b$ and $\mathcal{G}_b^f$, we employ a technique similar to that used in the proof of →Theorem 4.8: We provide strategies for both players in $\mathcal{G}_b$ by simulating play prefixes in $\mathcal{G}_b^f$. As strategies in the latter game, however, only prescribe "useful" moves as long as the play prefix is not settled, we ensure that the play prefixes in $\mathcal{G}_b^f$ remain unsettled.

→Sec. 4.1, Page 91

We again split the proof of equivalence between $\mathcal{G}_b$ and $\mathcal{G}_b^f$ into two lemmas. First, in Lemma 4.24, we show that Player 0 winning $\mathcal{G}_b^f$ from some designated vertex implies her winning $\mathcal{G}_b$ from the same vertex. We then show the analogous result for Player 1 in Lemma 4.25. This suffices due to determinacy of $\mathcal{G}_b^f$.

Lemma 4.24. *Let v^* be a vertex of $\mathcal{G}$. If Player 0 wins $\mathcal{G}_b^f$ from $(v^*, init(v^*))$, then she wins $\mathcal{G}_b$ from $(v^*, init(v^*))$.*

Proof. Let σ_f be a winning strategy for Player 0 in $\mathcal{G}_b^f$ from $(v^*, init(v^*))$. We construct a winning strategy σ for her from $(v^*, init(v^*))$ in $\mathcal{G}_b$ by mimicking the moves made

in $\mathcal{G}_b$ in $\mathcal{G}_b^f$ using a simulation function h mapping play prefixes in $\mathcal{G}_b$ to play prefixes in $\mathcal{G}_b^f$. We construct h to satisfy the following invariant:

> Let π be consistent with σ and end in (v, o, r). Then, $h(\pi)$ is consistent with σ_f, is unsettled, and ends in (v, o_f, r_f) with $(o, r) \sqsubseteq (o_f, r_f)$.

To this end, recall that $\mathcal{G}_b$ and $\mathcal{G}_b^f$ share the set of vertices V'. We define h and σ inductively and simultaneously, starting with $h(v^*, init(v^*)) = (v^*, init(v^*))$, which clearly satisfies the invariant. Now let π be a play prefix of $\mathcal{G}_b$ consistent with σ, ending in (v, o, r) such that $h(\pi)$ is defined. If (v, o, r) is a vertex of Player 1, then let (v^*, o^*, r^*) be an arbitrary successor of (v, o, r) in $\mathcal{A}'$. Otherwise, i.e., if (v, o, r) is a vertex of Player 0, then, due to the invariant, $h(\pi)$ ends in some (v, o_f, r_f) with $(o, r) \sqsubseteq (o_f, r_f)$. Let (v^*, o_f^*, r_f^*) be the unique vertex such that $h(\pi) \cdot (v^*, o_f^*, r_f^*)$ is consistent with σ_f and define $\sigma(\pi) = (v^*, o^*, r^*)$, where $(o^*, r^*) = upd((o, r), (v, v^*))$. This concludes the definition of σ. In either case, let $\pi^* = \pi \cdot (v^*, o^*, r^*)$. It remains to define $h(\pi^*)$.

To this end, let (o_f^*, r_f^*) be the unique memory state such that $\pi_f^* = h(\pi) \cdot (v^*, o_f^*, r_f^*)$ is a play prefix of $\mathcal{G}_b^f$. If π_f^* is unsettled, we define $h(\pi^*) = \pi_f^*$. This choice satisfies the invariant: If the vertex (v^*, o_f^*, r_f^*) is the destination of a shortcut, then let $(v^*, o_\rightarrow^*, r_\rightarrow^*)$ be the destination of its corresponding detour. We obtain $(o_f^*, r_f^*) \sqsupseteq (o_\rightarrow^*, r_\rightarrow^*) \sqsupseteq (o^*, r^*)$ due to ➔Lemma 4.16 and due to ➔Remark 4.20.1. Otherwise, i.e., if (v^*, o_f^*, r_f^*) is not the destination of a shortcut, then Lemma 4.16 yields the invariant directly.

➔Sec. 4.2, Page 101
➔Sec. 4.2, Page 105

Now consider the case that π_f^* is settled. Then it is settled due to containing an even dominating cycle as a suffix, due to the invariant and due to π_f^* being consistent with the winning strategy σ_f for Player 0. We define $h(\pi^*)$ by removing the settling dominating cycle as follows: Since $h(\pi)$ is not settled, the dominating cycle is a suffix of π_f^*. Thus, the cycle starts in a vertex $(v_{j'}, o_{j'}, r_{j'})$ with $v_{j'} = v^*$ and $r_{j'} \sqsupseteq r_f^*$. We define

$$h(\pi \cdot (v^*, o^*, r^*)) = (v_0, o_0, r_0) \cdots (v_{j'}, o_{j'}, r_{j'}) \ ,$$

which satisfies the invariant due to transitivity of $\sqsubseteq$, as stated in ➔Remark 4.14.2.

➔Sec. 4.2, Page 100

It remains to show that σ is winning for Player 0 from $(v^*, init(v^*))$ in $\mathcal{G}_b$. To this end, consider a play ρ starting in $(v^*, init(v^*))$ and consistent with σ and let π_{j+1} be the prefix of length j of ρ.

As all π_j start in (v_0, o_0, r_0) and are consistent with σ, all $h(\pi_j)$ are consistent with σ_f due to the invariant of h. Since σ_f is winning for Player 0 from (v_0, o_0, r_0) in $\mathcal{G}_b^f$, this implies that the overflow counter of the $h(\pi_j)$ never reaches n. Thus, again due to the invariant of h, neither does the overflow counter of the π_j. Hence, the colors of the last vertices of π_j and $h(\pi_j)$ coincide for all $j \in \mathbb{N}$.

Towards a contradiction, assume that the maximal color occurring infinitely often along ρ is odd, call it c. After some finite prefix, c cannot occur on even dominating cycles in the $h(\pi_j)$ anymore, since each occurrence on such a cycle implies at least one occurrence of an even higher even color in ρ. Hence, after this prefix, each time a vertex of color c is visited, say at the end of the prefix π_j, a vertex of the same

color is appended to the simulated play $h(\pi_j)$. Moreover, this vertex is never removed from the simulated play, since only vertices occurring on even dominating cycles are removed from the simulated play. Hence, the simulated play becomes longer with each visit to a vertex of color c after a finite prefix. This contradicts the $h(\pi_j)$ being unsettled, as every play of length $\ell + 1$ is settled due to ➔Lemma 4.21. Thus, the maximal color occurring infinitely often in ρ is even, i.e., σ is winning for Player 0 in $\mathcal{G}_b$ from $(v^*, \mathit{init}(v^*))$. □

➔Sec. 4.2, Page 106

Having shown that Player 0 can leverage a winning strategy from $(v, \mathit{init}(v))$ in $\mathcal{G}_b^f$ in order to obtain a winning strategy from the same vertex in $\mathcal{G}_b$, we now show the analogous statement for Player 1. This then implies equivalence of $\mathcal{G}_b^f$ and $\mathcal{G}_b$ due to determinacy of $\mathcal{G}_b^f$ (cf. Corollary 4.23).

Lemma 4.25. *Let v^* be a vertex of $\mathcal{G}$. If Player 1 wins $\mathcal{G}_b^f$ from $(v^*, \mathit{init}(v^*))$, then he wins $\mathcal{G}_b$ from $(v^*, \mathit{init}(v^*))$.*

Proof. Let τ_f be a winning strategy from $(v^*, \mathit{init}(v^*))$ for Player 1 in $\mathcal{G}_b^f$. We construct a winning strategy τ for him from $(v^*, \mathit{init}(v^*))$ in $\mathcal{G}_b$ by simulating play prefixes in $\mathcal{G}_b$ by unsettled prefixes in $\mathcal{G}_b^f$ from which we remove shortcut- and dominating cycles. We again define a simulation function h that maintains the following invariant:

> Let π be consistent with τ and end in (v, o, r) with $o < n$. Then, $h(\pi)$ is consistent with τ_f, is unsettled, and ends in (v, o_f, r_f) with $(o_f, r_f) \sqsubseteq (o, r)$.

We define h and τ inductively and simultaneously, starting with $h((v^*, \mathit{init}(v^*))) = (v^*, \mathit{init}(v^*))$, which clearly satisfies the invariant. Now let π be a play prefix of $\mathcal{G}_b$ consistent with τ and ending in (v, o, r). If (v, o, r) is a vertex of Player 0 then let (v^*, o^*, r^*) be an arbitrary successor of (v, o, r) in $\mathcal{A}'$. Otherwise, if (v, o, r) is a vertex of Player 1, then, due to the invariant, $h(\pi) = \pi'$ ends in some (v, o_f, r_f) with $(o_f, r_f) \sqsubseteq (o, r)$. Let (v^*, o_f^*, r_f^*) be the unique vertex such that $h(\pi) \cdot (v^*, o_f^*, r_f^*)$ is consistent with τ_f and define $\tau(\pi) = (v^*, o^*, r^*)$, where $(o^*, r^*) = \mathit{upd}((o, r), (v, v^*))$. This concludes the definition of τ.

It remains to define the simulation function h. To this end, let $\pi^* = \pi \cdot (v^*, o^*, r^*)$ and let (o_f^*, r_f^*) be the unique memory state such that $\pi_f^* = h(\pi) \cdot (v^*, o_f^*, r_f^*)$ is a play prefix of $\mathcal{G}_b^f$. First consider the case that π_f^* is unsettled. If (v^*, o_f^*, r_f^*) is not the destination of a shortcut, we define $h(\pi^*) = \pi_f^*$, which satisfies the invariant due to ➔Lemma 4.16. If, however, (v^*, o_f^*, r_f^*) is the destination of a shortcut, let $(v^*, o_\rightarrow^*, r_\rightarrow^*)$ be the destination of the corresponding detour. We differentiate whether taking the shortcut to (v^*, o_f^*, r_f^*) merely allows Player 1 to "catch up" to the play prefix constructed in $\mathcal{G}_b$, or whether it is more advantageous for him than the position (v^*, o^*, r^*) actually reached in $\mathcal{G}_b$. In the former case, i.e., if $(o^*, r^*) \sqsupseteq (o_f^*, r_f^*)$, we define $h(\pi^*) = \pi_f^*$, which satisfies the invariant by assumption. In the latter case, however, i.e., if $(o^*, r^*) \sqsupseteq (o_f^*, r_f^*)$ does not hold true, we remove the shortcut cycle similarly to the removal of a settling dominating cycle in the proof of Lemma 4.24, obtaining π_f, and define $h(\pi_f^*) = \pi_f$. This

➔Sec. 4.2, Page 101

→Sec. 4.2, Page 101
→Sec. 4.2, Page 100

satisfies the invariant due to $(o^*, r^*) \sqsupseteq (o^*_{\to}, r^*_{\to})$, which we obtain via →Lemma 4.16 and →Remark 4.14.2.

Now consider the case that π^*_f is settled. In this case, we distinguish two cases: If π^*_f is settled due to $o^*_f = n$, then, due to the invariant and Lemma 4.16, we obtain $o^* = n$. Thus, the invariant of h is vacuously true and we define $h(\pi^*)$ arbitrarily. If, however, π^*_f is settled due to reaching a dominating cycle, we remove this cycle from π^*_f similarly to the removal of dominating cycles in the proof of Lemma 4.24.

It remains to show that τ is indeed winning for Player 1 from $(v^*, \mathit{init}(v^*))$ in $\mathcal{G}_b$. To this end, consider a play ρ consistent with τ and let π_j be the prefix of length $j+1$ of ρ. If the overflow counter along ρ eventually saturates, ρ is clearly winning for Player 1. Hence, assume that the overflow counter along ρ does not saturate. Then, due to the invariant of h, the colors of the last vertices of π_j and $h(\pi_j)$ coincide for all $j \in \mathbb{N}$.

Let c be the largest color occurring infinitely often along ρ and assume towards a contradiction that c is even. Similarly to the argument in the proof of Lemma 4.24, after some finite prefix, the color c may only occur on odd dominating cycles and on removed shortcuts, as these are the only play infixes that are removed from the simulation: If this is not the case, then a vertex with color c would be appended to the $h(\pi_j)$ without ever being removed from the simulation. As the $h(\pi_j)$ are unsettled due to the invariant of h, this unbounded growth contradicts the bounded length of unsettled play prefixes due to Lemma 4.21. Moreover, again analogously to the proof of Lemma 4.24, the color c can only occur finitely often on odd dominating cycles, as each such occurrence implies one occurrence of some larger, odd color. Hence, it remains to show that the color c does not occur infinitely often on removed shortcut cycles.

Towards a contradiction, assume that the color c occurs infinitely often on removed shortcut cycles. Since, by assumption, the overflow counter along ρ never saturates, none of the $h(\pi_j)$ contains a saturated overflow counter either due to the invariant of h. Moreover, as both the removal of an odd dominating cycle and that of a shortcut retain the value of the overflow counter, the values of the overflow counter of the $h(\pi_j)$ eventually stabilize. For all $j \in \mathbb{N}$ let

$$\pi_j = (v_0, o_0, r_0) \cdots (v_j, o_j, r_j) \text{ as well as } h(\pi_j) = (v^j_0, o^j_0, r^j_0) \cdots (v^j_{k_j}, o^j_{k_j}, r^j_{k_j}) \ .$$

Furthermore, pick the position p such that the overflow counter in both the play ρ as well as in the simulations $h(\pi_j)$ has stabilized and such that no color larger than c occurs after position p. Formally, we pick p such that for all $j > p$ we have $o_p = o_j$ and $o^p_{k_p} = o^j_{k_j}$ and such that c is the largest color occurring on the suffix of ρ starting at position p.

We show $o^p_{k_p} = o_p$ by contradiction, i.e., we assume $o^p_{k_p} \neq o_p$. Due to the invariant, we obtain $o^p_{k_p} \leq o_p$, i.e., $o^p_{k_p} < o_p$. We claim that $o^p_{k_p} < o_p$ implies that $h(\pi_j)$ results from $h(\pi_{j-1})$ by removing a shortcut cycle only finitely often. In fact, after the position p, no shortcut cycle is removed anymore in this case: If, for some $j > p$ a shortcut is used in the move from $h(\pi_{j-1})$ to $h(\pi_j)$, then $(o_j, r_j) \sqsupseteq (o^j_{k_j}, r^j_{k_j})$, i.e., the shortcut

cycle is not removed. Hence, only finitely many shortcut cycles are removed, which contradicts the assumption of c occurring on infinitely many such cycles. Since we have $o^p_{k_p} \leq o_p$ due to the invariant of h, we obtain $o^p_{k_p} = o_p$, which implies $r^p_{k_p} \sqsubseteq r_p$, again due to the invariant of h. In particular, for each $j > p$, for each relevant request for some color c' that is open in $r^j_{k_j}$, a request for some color $c'' \geq c'$ is open in r_j.

Whenever c occurs on a removed shortcut cycle, then c must be smaller than the smallest relevant request that is open during that cycle: Otherwise it would answer that relevant request, due to c being even and thus cause the detour corresponding to the infix to violate the shortcut condition. While there may be some requests for colors $c' < c$ in the infix corresponding to the shortcut cycle in ρ, visiting c does not answer all relevant requests in that corresponding infix in ρ, as argued above. This implies that traversing the shortcut cycle increases the cost of some request in ρ. Furthermore, since c is the maximal color visited in the considered suffix, one such request eventually causes an overflow after traversing at most $b + 1$ many edges of nonzero weight. This contradicts the choice of p such that no overflows occur after π_p. If less than $b + 1$ edges of nonzero weight occur during the remainder of the play, then also at most $b + 1$ shortcuts occur, since each shortcut requires the traversal of at least one such edge. This in turn contradicts c occurring on infinitely many removed shortcut cycles.

Hence, we conclude that vertices of color c occur only finitely often on odd dominating cycles and on removed shortcut cycles. As these cycles are the only cycles that are removed from the simulation, almost all visited vertices of color c are added to the simulated play and are never removed. Thus, the $h(\pi_j)$ grow increasingly longer. Such unbounded growth contradicts them being unsettled due to Lemma 4.21. This, in turn, contradicts the invariant of h.

Thus, the maximal color visited infinitely often during ρ is odd. Hence, ρ is winning for Player 1, i.e., τ' is winning for him in $\mathcal{G}_b$ from $(v^*, \mathit{init}(v^*))$. □

The combination of the above two lemmas together with determinacy of $\mathcal{G}^f_b$ due to ➔Corollary 4.23 yields the desired equivalence of $\mathcal{G}_b$ and $\mathcal{G}^f_b$. Moreover, as the winner of a play in $\mathcal{G}^f_b$ is determined after at most polynomially many moves, $\mathcal{G}^f_b$ can easily be solved by simulating it on an alternating Turing machine whose runtime is polynomially bounded. As such machines can be simulated using deterministic Turing machines with polynomially bounded space due to Chandra, Kozen, and Stockmeyer [CKS81], this yields PSpace membership of the threshold problem for parity games with costs.

➔Sec. 4.2, Page 111

Theorem 4.26. *The following problem is in* PSpace*:*

> *"Given a parity game with costs $\mathcal{G}$, a vertex v of $\mathcal{G}$, and a bound $b \in \mathbb{N}$, does Player 0 have a strategy σ with $\mathit{Cost}_v(\sigma) \leq b$?"*

Proof. Let n be the number of vertices of $\mathcal{G}$ and let W be the largest weight occurring in $\mathcal{G}$. First, recall that if $b \geq nW$, then ➔Proposition 2.35 yields that the problem

➔Sec. 2.4, Page 27

reduces to solving $\mathcal{G}$. As the problem of solving parity games with costs is in UP ∩ coUP due to ➔Proposition 2.33, and since PSpace subsumes both UP and coUP, this concludes the proof for the case $b \geq nW$.

➔Sec. 2.4, Page 26

If, however, $b < nW$, we show how to simulate the finite-duration game $\mathcal{G}_b^f$ on an alternating Turing machine using the game semantics of such machines, i.e., two players construct a single path of a run of the machine. The existential and universal player take the roles of Player 0 and Player 1, respectively. The Turing machine keeps track of the complete prefix of the simulated play of $\mathcal{G}_b^f$.

Every vertex of the underlying arena of $\mathcal{G}_b^f$ can be represented in polynomial size. Moreover, the length of the play is bounded from above by $(\log(nW) + 1)(n + 1)^6$ due to ➔Lemma 4.21. Thus, the Turing machine can keep track of the play prefix constructed thus far explicitly and check whether a vertex picked by either player is a valid continuation of the play prefix of $\mathcal{G}_b^f$ constructed thus far. Moreover, the Turing machine can check whether a dominating cycle has occurred after each step in polynomial time. If the play is settled due to an even dominating cycle, the machine accepts, if it is settled otherwise, the machine rejects.

➔Sec. 4.2, Page 106

This algorithm involves neither the explicit construction of $\mathcal{G}_b$ nor that of $\mathcal{G}_b^f$. Due to this construction, the Turing machine accepts $\mathcal{G}$ and b if and only if Player 0 wins $\mathcal{G}_b^f$ and, due to Lemma 4.21, this machine terminates after polynomially many steps. Since polynomially time-bounded alternating Turing machines are equivalent to polynomially space-bounded classical Turing machines due to Chandra, Kozen and Stockmeyer [CKS81], we obtain the desired result. □

This concludes our work on upper bounds on the complexity of the threshold problem for parity games with weights. We have argued that the general case is in ExpTime in ➔Theorem 4.12 and we have shown that its complexity drops to PSpace when only considering parity games with costs in Theorem 4.26. In the following section, we provide matching lower bounds.

➔Sec. 4.1, Page 96

4.3 Hardness of Playing Optimally

In the previous section we have shown that the threshold problem for parity games with costs is in PSpace, while the more general threshold problem for parity games with weights is in ExpTime. In this section, we show that these bounds are tight.

In fact, we first show in Section 4.3.1 that the threshold problem for finitary parity games already is PSpace-hard. As parity games with costs subsume finitary parity games, this yields PSpace-completeness for both the threshold problem for finitary parity games, as well as for the threshold problem for parity games with costs.

Subsequently, we show that the threshold problem for parity games with weights is ExpTime-hard in Section 4.3.2. This then implies ExpTime-completeness of the problem.

4.3.1 Playing Finitary Parity Games Optimally is PSpace-hard

In this section we show that the threshold problem for finitary parity games is PSPACE-hard. To this end, we reduce the problem of evaluating a quantified Boolean formula to the threshold problem for finitary parity games. As the former problem is known to be PSPACE-hard, this yields the desired lower bound on the complexity of the latter problem.

We first define the problem of evaluating a quantified Boolean formula formally. A quantified Boolean formula (QBF) is of the form

$$\varphi = Q_1 x_1 Q_2 x_2 \dots Q_n x_n \psi \ ,$$

where $Q_j \in \{\exists, \forall\}$ for all j with $1 \leq j \leq n$, and where ψ is a Boolean formula over the variables x_1 through x_n. Similarly to the satisfiability problem for propositional logic for the complexity class NP, the problem of deciding whether a given QBF evaluates to true is the canonical PSPACE-hard problem.

Proposition 4.27 ([SM73])**.** *The following problem is* PSPACE*-complete:*

"Given a QBF φ, does φ evaluate to true?"

For the remainder of this section fix some quantified Boolean formula

$$\varphi = Q_1 x_1 Q_2 x_2 \dots Q_n x_n \psi \ .$$

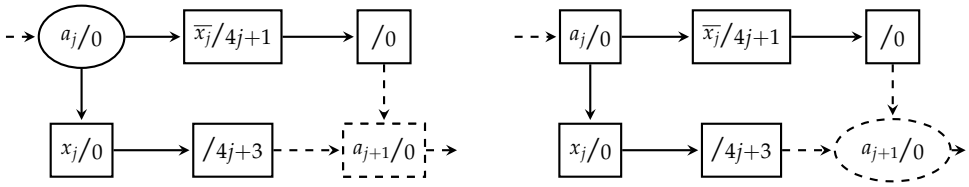
We assume w.l.o.g. that ψ is in conjunctive normal form, i.e., that it consists of a top-level conjunction and that every conjunct is a disjunction over three literals. Thus, we obtain

$$\psi = \bigwedge_{j=1}^{m} (\ell_{j,1} \vee \ell_{j,2} \vee \ell_{j,3}) \ ,$$

where every $\ell_{j,k}$ is either x or $\overline{x}$ for some $x \in \{x_1, \dots, x_n\}$. We call each $\ell_{j,k}$ for $k \in \{1,2,3\}$ a LITERAL and each conjunct of three literals a CLAUSE. Furthermore, we assume w.l.o.g. that the quantifiers Q_j are alternating with $Q_1 = Q_n = \exists$. We construct a finitary parity game $\mathcal{G}_\varphi$ containing a designated vertex v such that Player 0 has a strategy σ for $\mathcal{G}_\varphi$ with $\mathrm{Cost}_v(\sigma) = 3n + 5$ if and only if the formula φ evaluates to true, which implies PSPACE-hardness of the problem of playing optimally in finitary parity games due to Proposition 4.27.

Def. literal
Def. clause

This construction uses the standard framework for reducing QBF to infinite two-player games of polynomial size: Player 0 implements existential choices, i.e., existential quantification and disjunctions. Dually, Player 1 implements universal quantification and conjunctions. Intuitively, the players pick truth values for the variables in the order of their appearance in the quantifier prefix. Then, Player 1 picks a clause, followed by Player 0 picking a literal from that clause. Player 0 wins if and only if the literal evaluates to true under the assignment constructed earlier.



(a) Gadget for existentially quantified variables. If $j = n$, then $a_{j+1} = \psi$.

(b) Gadget for universally quantified variables.

Figure 4.11: The gadgets for assigning truth values to x_j.

Since, in order to obtain a polynomial-time reduction, the state space of the constructed game must be polynomially bounded, we are unable to store the assignment constructed during a play in the state space. Consequently, we encode the above requirement using the winning condition of the constructed game. We begin by constructing the arena $\mathcal{A}_\varphi$ of $\mathcal{G}_\varphi$ together with its coloring. As described above, $\mathcal{A}_\varphi$ consists of three types of gadgets that implement constructing a variable assignment, picking a literal, and asserting satisfaction of that literal, respectively.

Constructing the Assignment Recall that we reduce to the threshold problem for finitary parity games, i.e., we do not only have to construct a finitary parity game, but also some bound. To encode the above intuition, we encode assigning a truth value to a variable by opening requests: We model setting x_j to false by requesting color $4j + 1$ and setting it to true by requesting color $4j + 3$. Crucially, only one of the two colors can be requested. In a classical, qualitative parity game, Player 0 always prefers requesting color $4j + 1$ over requesting $4j + 3$, and vice versa for Player 1. In order to offset this monotonicity, we construct $\mathcal{A}_\varphi$ such that requesting the smaller color $4j + 1$ incurs a larger cost before leaving the gadget than requesting the larger color $4j + 3$.

We show the gadgets that assign a truth value to variable x_j in Figure 4.11. The vertex a_j belongs to Player 0 if x_j is existentially quantified, and to Player 1 if x_j is universally quantified. Otherwise, the two gadgets are identical. The dashed edges indicate the connections to the pre- and succeeding gadget, respectively. We construct the first part of $\mathcal{A}_\varphi$ out of n copies of this gadget. Moreover, the vertex a_1 has an incoming edge from the "end" of $\mathcal{A}_\varphi$, in order to enable infinite plays. Furthermore, the gadget assigning a truth value to x_n has an outgoing edge to the next part of the arena, in which a literal is picked. In the remainder of this section, let $c_{x_j} = 4j + 3$ and $c_{\overline{x_j}} = 4j + 1$ be the colors associated with assigning true or false to x_j, respectively.

Picking a Literal The second part of the arena starts with a vertex ψ belonging to Player 1, from which he picks a clause by moving to a vertex C_j belonging to Player 0. Each vertex C_j is connected to three gadgets, one for each of the three literals contained in C_j. We show this construction in Figure 4.12. As both paths through each gadget from the first part of the arena have uniform length, moving from the unique vertex

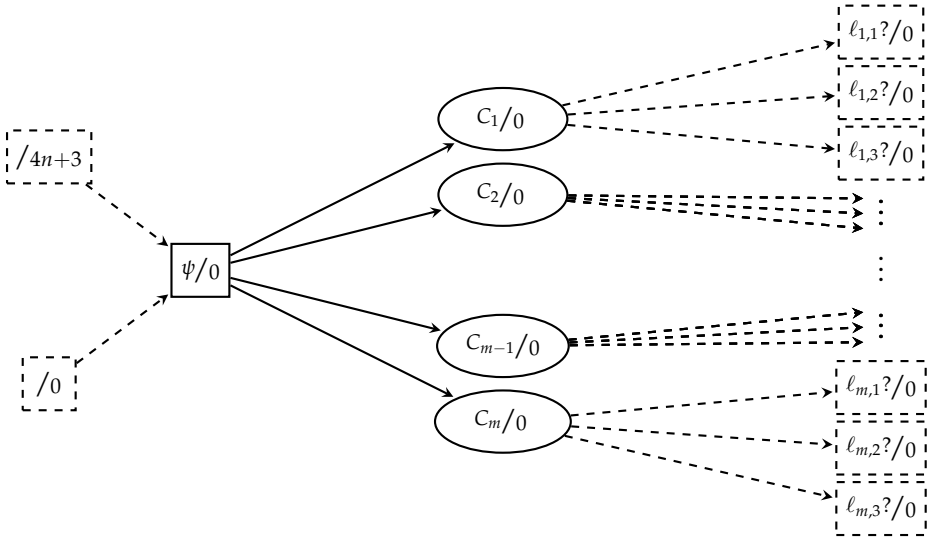


Figure 4.12: The gadget for the middle part of the arena.

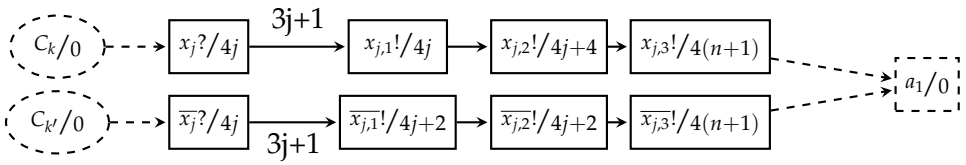


Figure 4.13: Gadgets checking the assignment of true to x_j (top) or to $\overline{x_j}$ (bottom).

of color c_{x_j} or of color $c_{\overline{x_j}}$ to the vertex ψ takes $3(n-j)+1$ and $3(n-j)+2$ steps, respectively. Upon leaving this middle gadget, the players have chosen some literal ℓ.

Satisfaction of the Literal We check whether or not the literal ℓ is satisfied by the assignment constructed in the former part of the arena by answering the corresponding request, e.g., if $\ell = \overline{x_j}$, then the color $4j+2$ occurs, and if $\ell = x_j$, then the color $4j+4$ occurs. Again due to monotonicity, moving to a vertex of color $4j+4$ answers both the request corresponding to setting x_j to false and the one setting it to true. Similarly to our previous construction, however, we construct the game such that moving to a vertex of color $4j+4$ incurs greater cost than moving to a vertex of color $4j+2$.

Following this intuition, the last part of the arena consists of one gadget for each literal x_1, $\overline{x_1}$ through x_n, $\overline{x_n}$ occurring in φ. In these gadgets, neither player has a non-trivial choice of the next vertex. Thus, the play "automatically" proceeds by first answering requests for all colors $c_{x_{j'}}$ and $c_{\overline{x_{j'}}}$ for $j' < j$ with cost $3(n-j')+3$ and cost $3(n-j')+4$ respectively. Thus, all such requests are answered with cost less than $3n+5$. It then either grants the request for color c_{x_j} after $3j+2$ steps, or the request for color $c_{\overline{x_j}}$ after $3j+1$ steps, both counted from the vertices $x_j?$ and $\overline{x_j}?$, respectively. Before leaving the gadget for the literal x_j or $\overline{x_j}$, all requests for the colors $c_{x_{j'}}$ and $c_{\overline{x_{j'}}}$

for $j' > j$ are answered after $3j + 3$ steps, measured from the vertices $x_j?$ and $\overline{x_j}?$, respectively. Since these requests have incurred cost $3(n - j') + 3$ and $3(n - j') + 4$ upon visiting those vertices, they are answered with cost at most $3(n - j') + 4 + 3j + 3 < 3(n - 1) + 7 < 3n + 5$.

We show these gadgets in Figure 4.13, which concludes the construction of $\mathcal{A}_\varphi$. In that drawing, we denote weights along the edges of the gadget in spite of the game under construction being a finitary parity game. These edges can easily be subdivided into edges of weight one with only a linear blowup and thus only serve as shorthands.

Example 4.28. Let $\varphi = \forall x_1 \exists x_2 (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2)$. In order to ease presentation, we disregard the assumption of $Q_n = \forall$ as well as that each clause shall contain three literals. It is, however, trivial to extend this example to satisfy these assumption by adding an unused universally quantified variable x_3 and by repeating some literal in each close.

We show the arena $\mathcal{A}_\varphi$ encoding the evaluation of φ in Figure 4.14. We have $c_{\overline{x_1}} = 5$, $c_{x_1} = 7$, $c_{\overline{x_2}} = 9$, and $c_{x_2} = 11$. △

The arena $\mathcal{A}_\varphi$ is of polynomial size in $|\varphi|$: The first part consists of one constant-size gadget per variable, while the second part is of linear size in the number of clauses in φ. The final part contains a gadget of size $\mathcal{O}(n)$ for each literal occurring in φ.

Remark 4.29. *The arena $\mathcal{A}_\varphi$ is of size $\mathcal{O}(n^2 + m)$.*

It remains to argue that this construction indeed implements our intuition, i.e., that a strategy σ for Player 0 in $\mathcal{G}_\varphi$ with $\mathrm{Cost}_{a_1}(\sigma) \leq 3n + 5$ indeed witnesses φ evaluating to true. Before we do so, however, we first observe that Player 0 is able to satisfy the classical parity condition from each vertex in the constructed arena: Before returning to the vertex a_1, each play visits a vertex $x_{j,3}!$ or $\overline{x_{j,3}}!$ for some j with $1 \leq j \leq n$, that is colored with the even color $4(n + 1)$, which is also the largest color occurring in the game. Hence, every strategy for Player 0 is winning for her with respect to the classical parity condition due to the structure of the constructed arena.

By carefully choosing the bound $3n + 5$, however, we force Player 0 to witness the evaluation of φ to true in order to satisfy the finitary parity condition with respect to that bound. The construction of $\mathcal{A}_\varphi$ causes the request for c_{x_j} or $c_{\overline{x_j}}$ to be answered only after its associated costs have violated that bound if the vertex $\overline{x_j}$ or x_j, respectively, has been visited beforehand: Since traversing the middle part of the arena incurs a constant cost of two, a request for color c_{x_j} has incurred a cost of $3(n - j) + 3$ upon reaching $x_j?$ and $\overline{x_j}?$, respectively, while a request for color $c_{\overline{x_j}}$ has incurred a cost of $3(n - j) + 4$ at these vertices. Hence, the total cost incurred by the request for color c_{x_j} is $(3(n - j) + 3) + (3j + 2) = 3n + 5$ in the gadget corresponding to x_j, and $(3(n - j) + 3) + (3j + 3) = 3n + 6$ in the gadget corresponding to $\overline{x_j}$. The dual reasoning holds true for requests for color $c_{\overline{x_j}}$. Hence, the bound of $3n + 5$ is only achieved if the request corresponding to the chosen literal was posed in the initial part of the arena.

Thus, the construction relies heavily on the bound on the cost that Player 0 has to enforce in order to break the monotonicity of the underlying parity condition. This

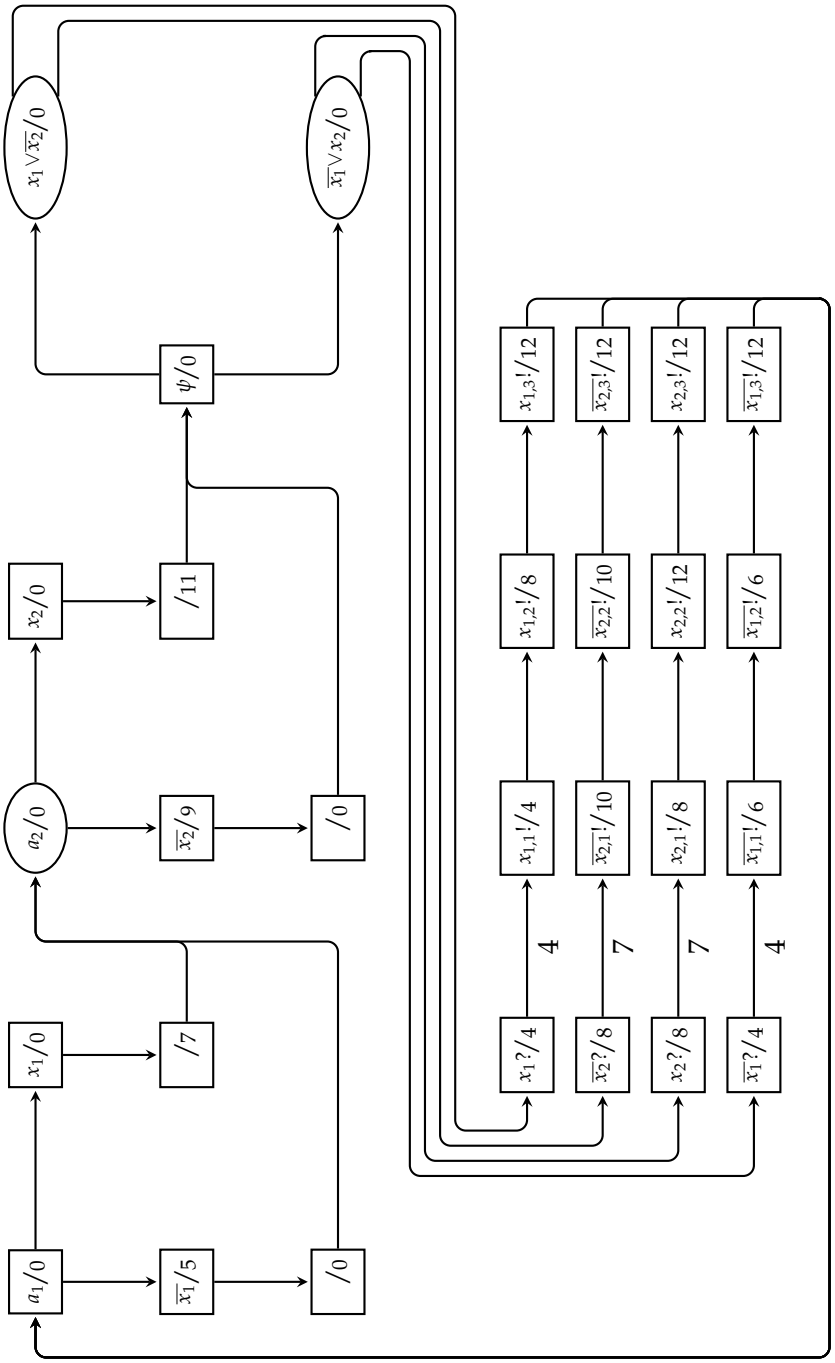


Figure 4.14: The arena $\mathcal{A}_\varphi$ from Example 4.28.

explains the increase in complexity from UP ∩ coUP and NP ∩ coNP to PSPACE when compared to both the classical parity condition and the parity condition with weights if the bound is quantified existentially.

Lemma 4.30. *Player 0 has a strategy σ with $Cost_{a_1}(\sigma) = 3n + 5$ in $\mathcal{G}_\varphi$ if and only if φ evaluates to true.*

Proof. In order to reason about the assignment of truth values to variables constructed during a play, we first introduce some notation. Let ψ be a quantifier-free Boolean formula over variables x_1 through x_n. Moreover, let $\alpha\colon \{x_1, \ldots, x_j\} \dashrightarrow \{\textbf{true}, \textbf{false}\}$ be a partial assignment of truth values to the variables from ψ. We denote the formula resulting from replacing the variables in the domain of α with their respective truth values by $\alpha(\psi)$. Moreover, given some partial mapping α, we write $\alpha[x_j \mapsto t]$ to denote the mapping α augmented by the assignment $x_j \mapsto t$.

Def. $\alpha(\psi)$

For the remainder of this proof, we only consider a finite play infix π that starts in a_1 and does not visit that vertex again. This suffices as all plays start in a_1, visit a_1 infinitely often, and all requests are answered before moving back to a_1.

First assume that φ evaluates to true. We construct a strategy σ for Player 0 with the properties given in the statement of the lemma inductively from assignments of truth variables to the existentially quantified variables of φ. Since the induction start and the induction step use similar constructions, we argue about both in parallel.

Due to the structure of the arena, we only need to define σ for play prefixes ending in vertices a_j where $Q_j = \exists$ and for play prefixes ending in one of the C_j as those are the only vertices of Player 0. Hence, first let j such that $1 \leq j \leq n$ and such that $Q_j = \exists$ and let π be a play prefix starting in a_1, consistent with σ and ending in a_j. By analyzing π we construct the partial mapping $\alpha\colon \{x_1, \ldots, x_{j-1}\} \dashrightarrow \{\textbf{true}, \textbf{false}\}$, where

- $\alpha_{j-1}(x_k) = \textbf{true}$ if π' visits x_k, and
- $\alpha_{j-1}(x_k) = \textbf{false}$ if π' visits $\overline{x_k}$.

Due to the structure of the arena, exactly one of these cases holds true, hence α_{j-1} is well-defined.

The induction hypothesis yields that, due to our construction of σ, the quantified Boolean formula $\exists x_j \cdots Q_n x_n \alpha_{j-1}(\psi)$ evaluates to true. Hence, there exists some $t \in \{\textbf{true}, \textbf{false}\}$ such that $\forall x_{j+1} \cdots Q_n x_n (\alpha_{j-1}[x_j \mapsto t])(\psi)$ evaluates to true as well. We define $\sigma(\pi) = x_j$ if $t = \textbf{true}$, and $\sigma(\pi) = \overline{x_j}$ otherwise.

When the play eventually reaches the vertex ψ, Player 1 chooses to move to some C_j. Hence, let π be some play prefix starting in a_1, consistent with σ, and ending in some C_j. The analysis of π yields an assignment $\alpha\colon \{x_1, \ldots, x_n\} \to \{\textbf{true}, \textbf{false}\}$, such that $\alpha(\psi)$ evaluates to true. Since $\alpha(\psi)$ evaluates to true, there exists a $k \in \{1, 2, 3\}$ such that $\alpha(\ell_{j,k}) = \textbf{true}$. We define $\sigma(\pi) = \ell_{j,k}?$, which concludes the definition of σ. It remains to show $\mathrm{Cost}_{a_1}(\sigma) \leq 3n + 5$.

To this end, let π be some play prefix starting in a_1, consistent with σ, and ending in a_1 without visiting a_1 in-between. We show that all requests in π are answered with cost at most $3n + 5$.

Upon reaching vertex ψ, there exist n open requests. By analyzing the prefix of π up to and including ψ, we construct an assignment α as above: If that prefix contains the vertex x_j, then we define $\alpha(x_j) = \textbf{true}$. Otherwise, i.e., if that prefix contains the vertex $\overline{x_j}$, then we define $\alpha(x_j) = \textbf{false}$. As previously argued, for all j with $1 \leq j \leq n$ we have that if $\alpha(x_j) = \textbf{true}$, then upon reaching ψ there is an open request for c_{x_j} with cost $3(n-j)+1$. Otherwise, there is an open request for $c_{\overline{x_j}}$ with cost $3(n-j)+2$ at that point.

Pick $\ell \in \{x_j, \overline{x_j} \mid 1 \leq j \leq n\}$ as the unique literal such that π visits the vertex ℓ?. We then have $\ell = x$ or $\ell = \overline{x}$ for some variable x. If $\ell = x$, then $\alpha(x) = \textbf{true}$ and hence, there is an open request for c_x. As argued previously, this request is then answered with cost $3n+5$, since we picked the gadget corresponding to x. Similarly, if $\ell = \overline{x}$, then $\alpha(x) = \textbf{false}$ and thus there is an open request for $c_{\overline{x}}$, which is answered with cost $3n+5$ as well. All other open requests are answered with cost at most $3n+5$, as argued previously during the construction of $\mathcal{G}_\varphi$, which concludes the proof of this direction of the lemma.

Now assume that φ evaluates to false. Then, irrespective of the choices made by Player 0 when constructing α in the first part of the arena, Player 1 can pick truth values for the universally quantified variables such that $\alpha(\psi) = \textbf{false}$ and subsequently pick a clause C_j such that $\alpha(C_j)$ evaluates to false. Since we have $C_j = \ell_{j,1} \vee \ell_{j,2} \vee \ell_{j,3}$, Player 0 has to move to some $\ell_{j,k}$ with $\alpha(\ell_{j,k}) = \textbf{false}$. If $\ell_{j,k} = x_l$, then there is an open request for $c_{\overline{x_j}}$ at $x_{l,1}$!, which is answered with cost $3n+6$. Similarly, if $\ell_{j,k} = \overline{x_l}$, then $\alpha(x_l) = \textbf{true}$, hence there is an open request for c_{x_l}, which is also answered with cost $3n+6$. Thus, in each round Player 1 can open a request that is only answered with cost at least $3n+6$, i.e., Player 0 has no strategy with cost $3n+5$. □

The above lemma shows that the construction of $\mathcal{G}_\varphi$ indeed captures our intuition: A strategy σ for Player 0 in $\mathcal{G}_\varphi$ with $\mathrm{Cost}_{a_1}(\sigma) \leq 3n+5$ witnesses φ evaluating to true. It is easy to see that the correctness of this statement heavily relies on our choice of the bound $3n+5$. In fact, in addition to the above observation that every play in $\mathcal{G}_\varphi$ satisfies the classical parity condition, we furthermore obtain that each strategy σ for her has $\mathrm{Cost}_v(\sigma) \leq 3n+7$ for all vertices v of $\mathcal{G}_\varphi$.

Due to Lemma 4.30, PSPACE-hardness of the threshold problem for finitary parity games then follows directly from PSPACE-hardness of evaluating quantified Boolean formulas.

Theorem 4.31. *The following problem is* PSPACE*-hard:*

> *"Given a finitary parity game $\mathcal{G}$, some vertex v of $\mathcal{G}$, and a bound $b \in \mathbb{N}$, does Player 0 have a strategy σ with $\mathrm{Cost}_v(\sigma) \leq b$?"*

Proof. Let $\varphi = Q_1 x_1 Q_2 x_2 \dots Q_n x_n \psi$ be some quantified Boolean formula. We construct the finitary parity game $\mathcal{G}_\varphi$ as described above and obtain that Player 0 has a strategy σ with $\mathrm{Cost}_{a_1}(\sigma) \leq 3n+5$ in $\mathcal{G}_\varphi$ if and only if φ evaluates to true due to Lemma 4.30. Since the problem of evaluating quantified Boolean formulas is PSPACE-hard due to Proposition 4.27 and since the reduction to the above problem is indeed polynomial due to Remark 4.29, we obtain the desired result. □

Recall that in Section 4.2 we have shown that the threshold problem for parity games with costs to be in PSPACE. As parity games with costs subsume finitary parity games, we obtain tight complexity bounds on the two problems.

Theorem 4.32. *The following problems are* PSPACE*-complete:*

> *"Given a finitary parity game $\mathcal{G}$, some vertex v of $\mathcal{G}$, and a bound $b \in \mathbb{N}$, does Player 0 have a strategy σ with $Cost_v(\sigma) \leq b$?"*
>
> *"Given a parity game with costs $\mathcal{G}$, some vertex v of $\mathcal{G}$, and a bound $b \in \mathbb{N}$, does Player 0 have a strategy σ with $Cost_v(\sigma) \leq b$?"*

4.3.2 Playing Parity Games with Weights Optimally is ExpTime-hard

In the previous section we have shown that the threshold problems for finitary parity games and for parity games with costs are PSPACE-complete. Since parity games with weights subsume both finitary parity games and parity games with weights, this implies PSPACE-hardness of the threshold problem for parity games with weights. Furthermore, in Section 4.1.4 we have shown EXPTIME-membership of the threshold problem for parity games with weights.

In this section we close the remaining gap in the picture by showing that the latter problem is EXPTIME-complete. To this end, we reduce the EXPTIME-hard problem of solving countdown games to the threshold problem for parity games with weights. Countdown games were introduced by Jurdziński, Laroussinie, and Sproston [JLS08].

In a countdown game, some initial credit is fixed at the beginning of a play. Both players then move in alternation in an arena that only contains nonpositive weights. In each turn, first Player 0 announces some weight, before Player 1 has to move along some edge of that weight, reducing the initial credit by the weight of the traversed edge. If the credit at some point reaches zero, Player 0 wins. If, however, the credit at some point turns negative, Player 1 wins. Since each edge has strictly negative weight, each play is won by either player after finitely many moves.

Def. countdown game

When formulated in our framework of arenas and winning conditions, a COUNTDOWN GAME $\mathcal{G} = (\mathcal{A}, \textsc{Countdown}(\text{Weight}, c_I))$ [JLS08] consists of an arena $\mathcal{A} = (V, V_0, V_1, E)$, a weighting Weight of $\mathcal{A}$, and some initial credit $c_I \in \mathbb{N}$, which satisfy the following conditions:

1. There exists a designated sink vertex $v_\perp \in V_1$,
2. we have
   - $E \subseteq (V_0 \times V_1) \cup (V_1 \times V_0) \cup (\{v_\perp\} \times \{v_\perp\})$,
   - $V_0 \times \{v_\perp\} \subseteq E$, and
   - $(v_\perp, v_\perp) \in E$,
3. for each $e_1 = (v, v_1'), e_2 = (v, v_2') \in E$, with $v \in V_0$ we have that $e_1 \neq e_2$ implies $\text{Weight}(e_1) \neq \text{Weight}(e_2)$,
4. for each $e \in E \cap (V_0 \times V)$ we have $\text{Weight}(e) \begin{cases} = 0 & \text{if } e \in (V_0 \times \{v_\perp\}), \text{ and} \\ < 0 & \text{otherwise, and} \end{cases}$

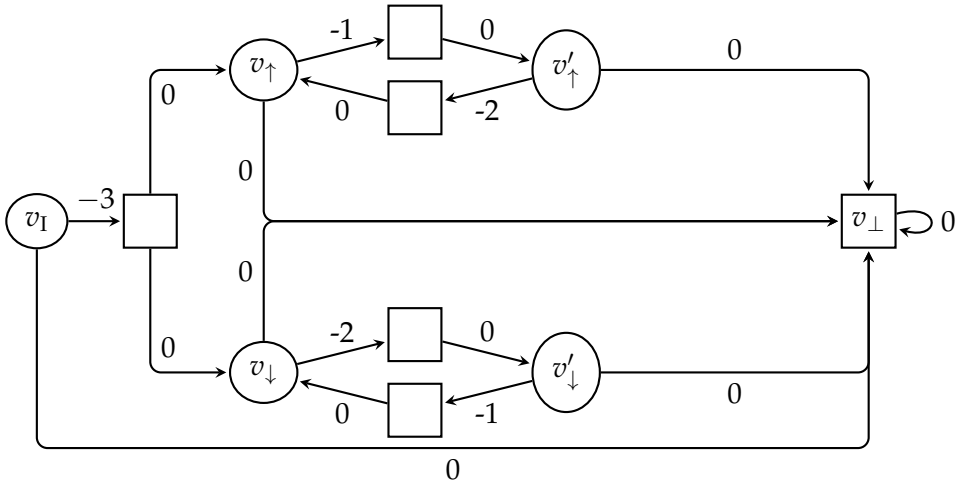


Figure 4.15: An example of the arena and the weighting of a countdown game.

5. for each $e \in E \cap (V_1 \times V)$ we have $\mathrm{Weight}(e) = 0$.

As discussed above, a countdown game is played in turns. Each turn starts at a vertex v of Player 0, from where Player 0 first picks some outgoing edge e leading to vertex v' of Player 1. That edge has unique weight among the outgoing edges of v due to the third requirement. Moreover, if Player 0 does not opt to end the play by moving to $v_\perp$, the weight of the edge is negative due to the fourth condition. Subsequently, Player 1 picks a successor of v' and moves to that successor along an edge of weight zero due to the fifth requirement, where the next turn of the play starts.

The COUNTDOWN CONDITION is defined as

Def. countdown condition

$$\textsc{Countdown}(\mathrm{Weight}, c_\mathrm{I}) = \\ \{\rho = v_0 v_1 v_2 \cdots \in V^\omega \mid \exists j.\, v_j = v_\perp \text{ and } c_\mathrm{I} + \mathrm{Weight}(v_0 \cdots v_j) = 0\} \quad .$$

Our definition of countdown games differs from the one given by Jurdziǹski, Laroussinie, and Sproston [JLS08], as we adapted it to fit our framework of games introduced in ➔Section 2. It is, however, easy to see that our definition and the one given by the authors are equivalent.

➔Page 11

Example 4.33. We show the arena of a countdown game together with a weighting in Figure 4.15. Player 0 wins this game from the vertex v_I if and only if the initial credit c_I is divisible by three: If $c_\mathrm{I} = 0$, then Player 0 can move directly to $v_\perp$. Otherwise, she moves to the only other successor vertex of v_I, from where Player 1 moves to either $v_\uparrow$ or to $v_\downarrow$. In either case, Player 0 can force the play to move between the vertices $v_\uparrow$ and $v'_\uparrow$, or $v_\downarrow$ and $v'_\downarrow$, respectively, incurring a weight of -3 in each iteration. Once the initial credit has been depleted, Player 0 can move to $v_\perp$, thus winning the play.

If, however, c_I is not divisible by three, first assume that $c_\mathrm{I} = 3j + 1$ for some $j \in \mathbb{N}$. If Player 0 moves to $v_\perp$ from v_I, she immediately loses. Thus, Player 1 can move to $v_\downarrow$, from where the only credit values occurring along the play are of the form $3j' - 1$

(upon visiting $v'_{\downarrow}$) and $3j' - 2$ (upon visiting $v_{\downarrow}$). Hence, whenever Player 0 opts to end the play by moving to $v_{\perp}$, the remaining credit is either greater or smaller than 0, i.e., she loses the resulting play. The argument for $c_{\mathrm{I}} = 3j + 2$ is analogous with Player 1 moving to $v_{\uparrow}$ at the beginning of the play. △

As countdown games are of finite duration, we obtain that they are determined due to Zermelo [Zer13].

Remark 4.34. *Countdown games are determined.*

Jurdziński, Laroussinie, and Sproston showed that solving countdown games is ExpTime-hard via a reduction from the word problem for Turing machines with an exponential time bound. The authors prove this via a reduction from the word problem for alternating Turing machines with polynomially bounded space. In order to concisely encode the exponential number of configurations attainable by the Turing machine during its run on the input word and the transitions between these configurations, this reduction requires the weights along the edges of the countdown game as well as the initial credit c_{I} to be given in binary encoding. Encoding the resulting countdown game in unary would entail an exponential blowup, i.e., the reduction would not be computable in polynomial time.

Proposition 4.35 ([JLS08])**.** *The following problem is* ExpTime*-hard:*

> *"Let $\mathcal{G}$ be a countdown game and let v be a vertex of $\mathcal{G}$. Does Player 0 have a winning strategy from v in $\mathcal{G}$?"*

We reduce the problem of solving countdown games to the threshold problem for parity games with weights. To this end, for the remainder of this section, fix some countdown game $\mathcal{G} = (\mathcal{A}, \textsc{Countdown}(\mathrm{Weight}, c_{\mathrm{I}}))$ where $\mathcal{A} = (V, V_0, V_1, E)$, as well as some initial vertex $v_{\mathrm{I}} \in V$.

Intuitively, we construct the parity game with weights $\mathcal{G}'$ such that at the beginning of the play a single request is opened, which is only answered upon visiting $v_{\perp}$. After visiting $v_{\perp}$, the play returns to the initial vertex v_{I} after reopening the unique request of $\mathcal{G}'$, enforcing infinitely many requests.

In a countdown game, only Player 0 may decide to move to $v_{\perp}$. Since she should, intuitively, only do so after traversing a play prefix of weight $-c_{\mathrm{I}}$, we equip the edges leading from her vertices to $v_{\perp}$ with weight $2c_{\mathrm{I}}$. Thus, she can enforce "tallying the score" by moving to $v_{\perp}$.

In order to afford Player 1 the same possibility, we add edges that allow him to move from his vertices to $v_{\perp}$. Furthermore, in order to incentivize him to only take these edges once he has exceeded the lower bound of $-c_{\mathrm{I}}$, these edges have weight zero. All remaining weights remain unchanged, thus the costs incurred by the unique request in $\mathcal{G}'$ model the remaining credit in the corresponding play in $\mathcal{G}$.

Formally, let $v_{\top}$ be some vertex not occurring in V. We define the parity game with weights $\mathcal{G}' = (\mathcal{A}', \textsc{WeightParity}(\Omega, \mathrm{Weight}'))$, where $\mathcal{A}' = (V', V'_0, V'_1, E')$, with

- $V' = V \cup \{v_{\top}\}$, $V'_0 = V_0 \cup \{v_{\top}\}$, $V'_1 = V_1$, and

- $E' = (E \setminus \{(v_\perp, v_\perp)\}) \cup (V_1 \times \{v_\perp\}) \cup \{(v_\perp, v_\top), (v_\top, v_\mathrm{I})\}$.

Since there exists a unique outgoing edge of $v_\perp$ leading back to the initial vertex v_I of the countdown game via $v_\top$, the play is restarted after visiting the sink vertex.

Furthermore, we define the weight function

$$\mathrm{Weight}'(e) = \begin{cases} \mathrm{Weight}(e) & \text{if } (v, v') \in E \setminus (V_0 \times \{v_\perp\}) \\ 2c_\mathrm{I} & \text{if } e \in V_0 \times \{v_\perp\} \\ 0 & \text{otherwise} \end{cases}$$

as well as the coloring

$$\Omega'(v) = \begin{cases} 1 & \text{if } v = v_\top, \\ 2 & \text{if } v = v_\perp, \\ 0 & \text{otherwise} \end{cases}$$

and claim that Player 0 has a strategy σ with $\mathrm{Cost}_{v_\top}(\sigma) \leq c_\mathrm{I}$ if and only if she wins $\mathcal{G}$ from v_I. We illustrate this construction in Figure 4.16.

Example 4.36. Let $\mathcal{A}$ and Weight be the arena and the weight function from Example 4.33. Furthermore, let $c_\mathrm{I} = 5$. We show the energy parity game $\mathcal{G}'$ resulting from the reduction above for $\mathcal{G} = (\mathcal{A}, \textsc{Countdown}(\Omega, c_\mathrm{I}))$ in Figure 4.17. △

We claim that this construction implements our intuition, i.e., that solving $\mathcal{G}'$ with respect to the bound c_I is indeed equivalent to solving $\mathcal{G}$.

Lemma 4.37. *Player 0 wins $\mathcal{G}$ from v_I if and only if she has a strategy σ with $\mathrm{Cost}_{v_\top}(\sigma) \leq c_\mathrm{I}$ in $\mathcal{G}'$.*

Proof. We first show the direction from left to right. To this end, let σ be a winning strategy for Player 0 in $\mathcal{G}$ from v_I. Moreover, let $\pi = v_0 \cdots v_j$ be a play prefix in $\mathcal{G}'$ starting in $v_\top$ and let j' be the largest position in π with $v_{j'} = v_\top$. We define the strategy σ' for Player 0 in $\mathcal{G}'$ via $\sigma'(\pi) = \sigma(v_{j'+1} \cdots v_j)$ as well as $\sigma'(\pi v_\perp) = v_\top$ and claim $\mathrm{Cost}_{v_\top}(\sigma') \leq b$.

To prove this claim, let $\rho' = v_0 v_1 v_2 \cdots$ be a play in $\mathcal{G}'$ starting in $v_\top$ that is consistent with σ'. First assume towards a contradiction that ρ' only visits $v_\top$ finitely often. Then, due to the structure of the arena, ρ' only visits $v_\perp$ finitely often. By construction of σ', this implies that ρ' contains a suffix that begins in v_I, is consistent with σ, but never visits $v_\perp$. This contradicts σ being a winning strategy for her in $\mathcal{G}$ from v_I. Hence, ρ' visits $v_\perp$ infinitely often.

Thus, ρ' is of the form

$$\rho' = v_\top \pi_0 v_\perp \cdot v_\top \pi_1 v_\perp \cdot v_\top \pi_2 v_\perp \cdots \ ,$$

where each π_j starts in v_I and is consistent with σ. We first argue that, if π_j ends in a vertex of Player 1, then we have $0 \geq \mathrm{Weight}'(\pi_j) \geq -c_\mathrm{I}$: All weights in $\mathcal{G}'$ except

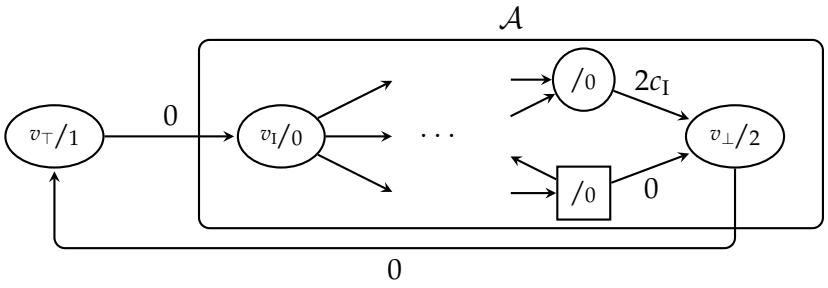


Figure 4.16: Construction of the parity game with weights $\mathcal{G}'$ from a given countdown game $\mathcal{G}$. We omit unchanged weights for the sake of readability.

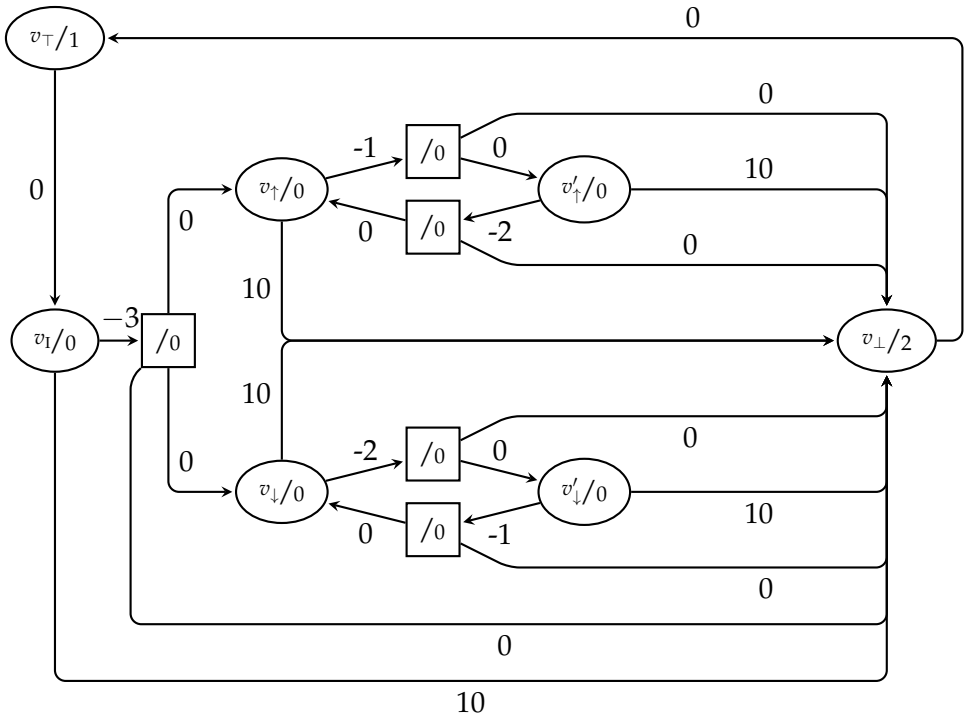


Figure 4.17: The parity game with weights corresponding to the arena and the weighting from Figure 4.15 for $c_\mathrm{I} = 5$.

for those from $V_0 \times \{v_\perp\}$ are nonpositive. Hence, $0 \geq \text{Weight}'(\pi_j)$ follows directly from the construction of $\mathcal{G}'$. Moreover, $\text{Weight}'(\pi_j) < -c_\text{I}$ would contradict π_j being consistent with the winning strategy σ for Player 0 in $\mathcal{G}$, since she would be unable to continue the play prefix π_j such that the resulting play is winning for her. Hence, we have $0 \geq \text{Weight}'(\pi_j) \geq -c_\text{I}$. Moreover, since all edges leading from $v_\top$ and all edges leading to $v_\perp$ have weight zero, and since $\text{Weight}'(\pi)$ is decreasing for increasing prefixes π of π_j due to construction of $\mathcal{G}'$, we obtain $\text{Ampl}(v_\top \pi_j v_\perp) < c_\text{I}$.

If, however, π_j ends in a vertex of Player 0, then we have $\text{Weight}'(\pi_j) = -c_\text{I}$ and $\text{Ampl}(v_\top \pi_j v_\perp) = c_\text{I}$, as π_j is consistent with the winning strategy σ for Player 0 in $\mathcal{G}$. In either case, we obtain that the unique request in $v_\top \pi_j v_\perp$ posed by visiting $v_\top$ is answered with cost at most c_I. Hence, ρ' has cost at most c_I, which concludes this direction of the proof.

We show the other direction of the statement via contraposition: Assume Player 0 does not win $\mathcal{G}$ from v_I. Since $\mathcal{G}$ is determined due to Remark 4.34, Player 1 wins $\mathcal{G}$ from v_I, say with strategy τ. We define a strategy τ' for Player 1 in $\mathcal{G}'$ that is winning for him from $v_\top$ via mimicking moves made by τ until the initial credit is used up. At that point, τ' prescribes moving to $v_\perp$ in order to witness exceeding the initial credit and to restart the play.

Formally, let $\pi' = v_0 \cdots v_j$ be a play prefix in $\mathcal{G}'$ that starts in $v_\top$ and ends in some vertex of Player 1. Moreover, let $j' \leq j$ be the largest position such that $v_{j'} = v_\top$. If $\text{Weight}(v_{j'} \cdots v_j) \geq -c_\text{I}$, we define $\tau'(\pi') = \tau(v_{j'+1} \cdots v_j)$. Otherwise, we define $\tau'(\pi') = v_\perp$. It remains to show $\text{Cost}_{v_\top}(\tau') > c_\text{I}$.

To this end, let ρ' be a play in $\mathcal{G}'$ starting in $v_\top$ consistent with τ'. Due to the structure of $\mathcal{A}'$, every infix of ρ' that does not visit $v_\top$ nor $v_\perp$ traverses edges of weight zero and negative weight in alternation. Moreover, since τ' prescribes moving to $v_\perp$ once the play infix since the last visit to $v_\top$ has incurred weight exceeding $-c_\text{I}$, the play ρ' is of the form

$$\rho' = v_\top \pi_0 v_\perp \cdot v_\top \pi_1 v_\perp \cdot v_\top \pi_2 v_\perp \cdots \ ,$$

where each π_j starts in v_I and is consistent with τ.

We aim to show $\text{Ampl}(v_\top \pi_j v_\perp) > c_\text{I}$ for all j, which implies $\text{Cost}(\rho') > c_\text{I}$ due to the construction of $\mathcal{G}'$ and thus suffices to show the desired statement. To this end, fix some $j \in \mathbb{N}$ and first consider the case that π_j ends in a vertex of Player 1. Then we obtain $\text{Weight}(\pi_j) < -c_\text{I}$ by definition of τ'. This directly implies the desired statement.

Now consider the case that π_j ends in a vertex of Player 0. We first argue that $\text{Weight}'(\pi_j) \neq -c_\text{I}$ holds true. Towards a contradiction, assume $\text{Weight}'(\pi_j) = -c_\text{I}$ and recall that π_j starts in v_I and is consistent with the winning strategy τ for Player 1 from v_I. Thus the play $\pi_j (v_\perp)^\omega$ in $\mathcal{G}$ is consistent with τ. Hence, $\text{Weight}'(\pi_j) = -c_\text{I}$ contradicts τ being a winning strategy for Player 1 from v_I.

It remains to show $\text{Ampl}(v_\top \pi_j v_\perp) > c_\text{I}$ for the case that π_j ends in a vertex of Player 0. If $\text{Weight}'(\pi_j) < -c_\text{I}$, we directly obtain $\text{Ampl}(v_\top \pi_j v_\perp) > c_\text{I}$. If, however, $\text{Weight}'(\pi_j) > -c_\text{I}$, we have $\text{Weight}'(\pi_j v_\perp) > c_\text{I}$, since we defined $\text{Weight}'(v, v_\perp) = 2c_\text{I}$

for each vertex v of Player 0. Thus, we obtain $\mathrm{Ampl}(v_\top \pi_j v_\bot) > c_\mathrm{I}$ for each infix $v_\top \pi_j v_\bot$ of ρ'. Hence, the unique request posed by visiting $v_\top$ is answered with cost greater than c_I, i.e., we obtain $\mathrm{Cost}(\rho') > c_\mathrm{I}$. □

Due to Lemma 4.37 we obtain a polynomial reduction from the problem of solving countdown games to the threshold problem for parity games with weights. As the former problem is ExpTime-hard due to Proposition 4.35, this implies ExpTime-hardness of the latter problem.

Theorem 4.38. *The following decision problem is* ExpTime*-hard:*

> *"Given a parity game with weights $\mathcal{G}$, some vertex v_I of $\mathcal{G}$, and a bound $b \in \mathbb{N}$, does Player 0 have a strategy σ with $\mathrm{Cost}_{v_\mathrm{I}}(\sigma) \leq b$ in $\mathcal{G}$?"*

Proof. We reduce the problem of solving countdown games to the given problem. To this end, let $\mathcal{G} = (\mathcal{A}, \textsc{Countdown}(\mathrm{Weight}, c_\mathrm{I}))$ be a countdown game and let v_I be a vertex of $\mathcal{A}$. We construct the parity game with weights $\mathcal{G}'$ as described above. Due to Lemma 4.37, Player 0 wins $\mathcal{G}$ from v_I if and only if she has a strategy σ with $\mathrm{Cost}_{v_\top} \leq c_\mathrm{I}$. Moreover, as the problem of solving countdown games is known to be ExpTime-hard due to Proposition 4.35, this implies the desired result. □

The result of Theorem 4.38 closes the remaining gap in the characterization of the complexity of the threshold problem for parity games with weights, parity games with costs, and finitary parity games. Having thus settled the computational complexity of these problems, we now turn our attention to the memory requirements of playing with respect to a fixed bound in these games. In the following section we consider the memory requirements of both players when playing optimally.

4.4 Memory Requirements of Playing Optimally

If Player 0 just aims to win a finitary parity game or a parity game with costs, she can do so using a positional winning strategy, due to ➔Proposition 2.25 (➔Sec. 2.4, Page 23) and ➔Proposition 2.34 (➔Sec. 2.4, Page 26). In parity games with weights, however, she requires linear memory in the number of vertices and in the largest absolute weight occurring in the game, as well as quadratic memory in the number of odd colors, due to ➔Theorem 3.34 (➔Sec. 3.4, Page 71). Dually, in all such games, Player 1 requires, in general, infinite memory in order to implement a winning strategy for him, due to the propositions and the theorem referenced above.

In this section we show that positional strategies no longer suffice for Player 0 in order to play optimally even in finitary parity games, but that she requires exponential memory in the number of odd colors occurring in the game. Dually, as Player 1 is no longer required to let the costs of the play diverge when playing optimally, exponential strategies suffice for him to win with respect to a fixed bound. Furthermore, we show that these bounds are tight for both players.

As the upper bounds on the memory required by both players result from the proof of ➔Theorem 4.8 (➔Sec. 4.1, Page 91), we provide both upper bounds in Section 4.4.1. Subsequently, we

prove the lower bound for Player 0 in Section 4.4.2, while we prove the lower bound for Player 1 in Section 4.4.3.

4.4.1 Exponential Memory Suffices for both Players

We first argue that exponential memory indeed suffices for both players to play optimally in parity games with weights. As such games subsume parity games with costs and finitary parity games, this implies the same result for the two latter kinds of games as well.

To this end, fix some parity game with weights $\mathcal{G}$ with n vertices, d odd colors, and largest absolute W as well as some bound $b \in \mathbb{N}$. Moreover, recall that in →Section 4.1 we defined the b-threshold game $\mathcal{G}_b$ of $\mathcal{G}$. This threshold game is a classical parity game whose arena consists of the arena of $\mathcal{G}$ augmented with request functions and an overflow counter that is bounded from above by n. →Page 86

Furthermore recall that in the proof of →Theorem 4.8 we showed how to leverage a winning strategy for either player in $\mathcal{G}_b$ in order to construct a winning strategy for them in $\mathcal{G}$. To this end, we implemented the strategy for Player 0 in $\mathcal{G}$ using the set $\{0, \dots, n\} \times R$ as memory states, where the set $\{0, \dots, n\}$ implements the overflow counter and where R denotes the set of request functions. →Sec. 4.1, Page 91

The first component of that memory structure is, however, irrelevant: Player 0 can always play assuming the largest possible value of that counter that still allows her to win.

Lemma 4.39. *Let $\mathcal{G}$ be a parity game with weights containing d odd colors and let $b \in \mathbb{N}$. Moreover, let v^* be a vertex of $\mathcal{G}$. If Player 0 has a strategy σ in $\mathcal{G}$ with $Cost_{v^*}(\sigma) = b$, then she also has a strategy σ' with $Cost_{v^*}(\sigma') \leq b$ and $|\sigma'| = (2b^2 + 3b + 2)^d$.*

Proof. Recall that we showed in the proof of →Theorem 4.8 that if Player 0 has a strategy σ in $\mathcal{G}$ with $\mathrm{Cost}_{v^*}(\sigma) \leq b$, then she also has a winning strategy from $(v^*, init(v^*))$ in the threshold game $\mathcal{G}_b$ as defined in →Section 4.1.2. Since Player 0 wins $\mathcal{G}_b$ from $(v^*, init(v^*))$, and since $\mathcal{G}_b$ is a parity game, there exists a positional strategy that is winning for her from that vertex due to →Proposition 2.18. Let σ_b be such a strategy. →Sec. 4.1, Page 91 →Page 88 →Sec. 2.3, Page 20

Let $\mathcal{R}$ be the set of vertices reached by plays starting in $(v^*, init(v^*))$ and consistent with σ_b. Note that, since σ_b is positional, that strategy is winning from all vertices in $\mathcal{R}$. Furthermore, for each vertex v and each request function r, let $o_{v,r} = \max(\{0\} \cup \{o \mid (v, o, r) \in \mathcal{R}\})$, i.e., $o_{v,r}$ is the maximal value such that Player 0 wins $\mathcal{G}_b$ from $(v, o_{v,r}, r)$ using σ_b, or zero, if no such value exists.

We now define a strategy σ' for Player 0 in $\mathcal{G}$ that only uses the set of request functions as memory states. To this end, recall that we defined the memory structure $\mathcal{M} = (M, init, upd)$ for the construction of $\mathcal{G}_b$, where $M = \{0, \dots, n\} \times R$, and where R is the set of request functions. We define $M' = R$, the update function $upd'(r, (v, v')) = r'$, if $upd((o_{v,r}, r), (v, v')) = (o', r')$, as well as the initialization function $init'(v) = init(v)$. Finally, we define the next-move function $nxt'(v, r) = v'$, where v' is the unique vertex that satisfies $\sigma_b(v, o_{v,r}, r) = (v', o', r')$, and claim that the strategy σ' implemented

by $\mathcal{M}' = (M', init', upd')$ and nxt' has $\text{Cost}_{v^*}(\sigma') \leq b$, which suffices to show the desired statement.

To prove this claim, let $\rho = v_0 v_1 v_2 \cdots$ be a play starting in v^* and consistent with σ' and let $\text{ext}_{\mathcal{M}'}(\rho) = (v_0, r_0)(v_1, r_1)(v_2, r_2) \cdots$ be the extension of ρ with respect to the memory structure $\mathcal{M}'$.

A straightforward induction yields $(v_j, o_{v_j,r_j}, r_j) \in \mathcal{R}$ for all $j \in \mathbb{N}$. We first argue that we have $o_{v_j,r_j} \leq o_{v_{j+1},r_{j+1}}$ for all $j \in \mathbb{N}$. To this end, let $(o, r) = upd((o_{v_j,r_j}, r_j), (v_j, v_{j+1}))$. By construction of $\mathcal{A}'$ we have $o \geq o_{v_j,r_j}$ and $r = r_{j+1}$. Moreover, since $(v_j, o_{v_j,r_j}, r_j) \in \mathcal{R}$ and due to our definition of σ', we obtain $(v_{j+1}, o, r) = (v_{j+1}, o, r_{j+1}) \in \mathcal{R}$. Hence, $o \leq o_{v_{j+1},r_{j+1}}$, which implies $o_{v_{j+1},r_{j+1}} \geq o_{v_j,r_j}$.

Thus, the o_{v_j,r_j} are monotonically increasing. Furthermore, we easily obtain $o_{v_j,r_j} < n$ due to all (v_j, o_{v_j,r_j}, r_j) being in $\mathcal{R}$, the definition of $\mathcal{R}$, and due to σ_b being winning for Player 0 from $(v^*, init(v^*))$. Hence, the sequence of o_{v_j,r_j} eventually stabilizes, i.e., there exists a $j \in \mathbb{N}$ such that $o_{v_{j'},r_{j'}} = o_{v_j,r_j}$ for all $j' \geq j$.

We argue that the play $(v_j, o_{v_j,r_j}, r_j)(v_{j+1}, o_{v_{j+1},r_{j+1}}, r_{j+1})(v_{j+2}, o_{v_{j+2},r_{j+2}}, r_{j+2}) \cdots$ is consistent with σ_b: Let $j' \geq j$ such that $(v_{j'}, o_{v_{j'},r_{j'}}, r_{j'}) \in V_0'$ and let $\sigma_b(v_{j'}, o_{v_{j'},r_{j'}}, r_{j'}) = (v_{j'+1}, o, r_{j'+1})$. We then clearly obtain $o \leq o_{v_{j'+1},r_{j'+1}}$ by definition of the latter. Furthermore, we have $o_{v_{j'},r_{j'}} \leq o$ due to the construction of $\mathcal{A}'$, which yields $o = o_{v_{j'+1},r_{j'+1}}$ due to our assumption $o_{v_{j'},r_{j'}} = o_{v_{j'+1},r_{j'+1}}$.

Thus, the play $(v_j, o_{v_j,r_j}, r_j)(v_{j+1}, o_{v_{j+1},r_{j+1}}, r_{j+1})(v_{j+2}, o_{v_{j+2},r_{j+2}}, r_{j+2}) \cdots$ starts in a vertex from $\mathcal{R}$, is consistent with σ_b, and shares a color sequence with a suffix of ρ due to $o_j \leq o_{v_j,r_j} < n$. The strategy σ_b being winning for Player 0 from $(v^*, init(v^*))$, the construction of $\mathcal{G}_b$ and prefix-independence of the parity condition with weights then yield $\text{Cost}(\rho) \leq b$. □

For Player 1, in contrast, it is open whether one can omit the overflow counter when implementing a strategy with cost at least b. Hence, we have to include it in the resulting memory structure. The following upper bound thus results directly from the proof of Theorem 4.8.

Corollary 4.40. *Let $\mathcal{G}$ be a parity game with weights with n vertices and d odd colors and let $b \in \mathbb{N}$. Moreover, let v^* be a vertex of $\mathcal{G}$. If Player 1 has a strategy τ in $\mathcal{G}$ with $Cost_{v^*}(\tau) = b$, then he also has a strategy τ' with $Cost_{v^*}(\tau') \geq b$ and $|\tau'| = n(2b^2 + 3b + 2)^d$.*

The factor of $2b^2 + 3b + 2$ in the upper bounds for both players stems from the construction of $\mathcal{G}_b$, which is of polynomial size in the number of request functions. We have, however, argued in Section 4.2 that in the special case of parity games with costs we are able to simplify the definition of request functions: While in parity games with weights we have to store an interval of costs incurred by open requests, it suffices to store the upper bound of that interval in parity games with costs. Thus, in that special case we only obtain $(b + 2)^d$ request functions, which yields the following improved upper bounds on memory requirements for both players.

Corollary 4.41. *Let $\mathcal{G}$ be a parity game with costs with n vertices and d odd colors and let $b \in \mathbb{N}$. Moreover, let v^* be a vertex of $\mathcal{G}$.*

1. *If Player 0 has a strategy σ in $\mathcal{G}$ with $Cost_{v^*}(\sigma) = b$, then she also has a strategy σ' with $Cost_{v^*}(\sigma') \leq b$ and $|\sigma'| = (b+2)^d$.*
2. *If Player 1 has a strategy τ in $\mathcal{G}$ with $Cost_{v^*}(\tau) = b$, then he also has a strategy τ' with $Cost_{v^*}(\tau') \geq b$ and $|\tau'| = n(b+2)^d$.*

Having argued that exponential memory suffices for both players to implement optimal strategies, we now turn our attention to providing matching lower bounds. Again, our matching lower bounds already hold for finitary parity games. These proofs rely on ideas that we already used to show ➔Theorem 4.31: We leverage the fact that fixing some bound requires a player, in the worst case, to store all open requests in order to answer them within the given bound or to exceed that bound.

➔Sec. 4.3, Page 123

4.4.2 Player 0 Requires Exponential Memory

We now show that Player 0 requires exponential memory to play optimally even in the special case of finitary parity games. To this end, we construct for each $d \geq 1$ a finitary parity game $\mathcal{G}_d$ that is of polynomial size in d and fix a bound b that is polynomial in d. We subsequently show that, for a designated vertex v_I of $\mathcal{G}_d$, Player 0 has a strategy σ_d with $\mathrm{Cost}_{v_\mathrm{I}}(\sigma_d) = b$ and that she has no strategy with smaller cost from that vertex. Furthermore, we show that the strategy σ_d is of size 2^{d-1} and that there exists no strategy of smaller size that witnesses the bound b.

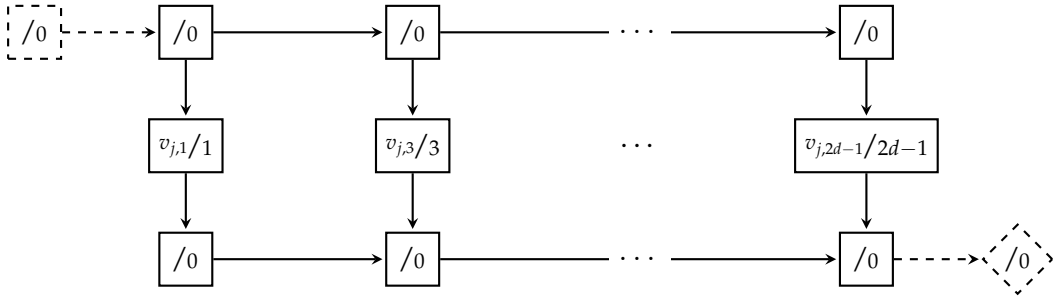
Thus, in order to play optimally in $\mathcal{G}_d$, Player 0 indeed requires exponential memory. Since $\mathcal{G}_d$ is a finitary parity game, this stands in stark contrast to the situation in which Player 0 merely aims to win $\mathcal{G}_d$, i.e., provide some finite upper bound on the cost along the play. In that situation, Chatterjee and Henzinger [CH06] showed that positional strategies, i.e., strategies of constant size one suffice for Player 0 (cf. ➔Proposition 2.25.1).

➔Sec. 2.4, Page 23

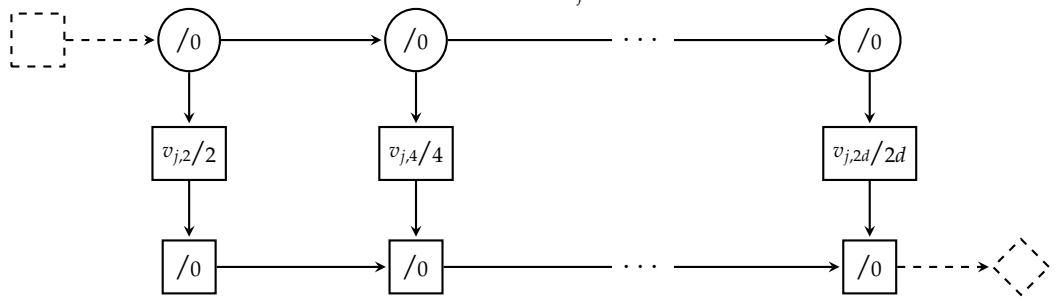
For the remainder of this section, fix some $d \geq 1$. We begin by constructing the parity game with costs $\mathcal{G}_d$. To this end, we generalize a construction by Chatterjee and Fijalkow [CF13]. While the construction by Chatterjee and Fijalkow yields a linear lower bound, our generalization indeed realizes the claimed exponential lower bound.

The game $\mathcal{G}_d$ is played in rounds. In each round, which starts at the designated vertex v_I, Player 1 poses d requests for odd colors in the range 1 through $2d-1$. Subsequently, Player 0 gives d answers using even colors in the range 2 through $2d$. If she recalls the choices made by Player 1 in the first part of the round, she is able to answer each request with respect to the fixed bound b. Otherwise, we show that Player 1 can exploit her insufficient memory in order to force requests that are only answered with cost greater than b. After each round, the play returns to the initial vertex v_I in order to allow for infinite plays.

The arena $\mathcal{A}_d$ of $\mathcal{G}_d$ consists of gadgets G_j^1 for Player 1 and gadgets G_j^0 that each allow exactly one request (for Player 1) or response (for Player 0) to be made. Moreover, each path through a gadget has the same length $d+2$, including the edge connecting a gadget to its successor. However, low-priority requests and responses are made earlier than high-priority ones when traversing such a gadget, due to its structure. We



(a) The j-th gadget G_j^1 for Player 1.



(b) The j-th gadget G_j^0 for Player 0.

Figure 4.18: The gadgets constituting the arena of $\mathcal{G}_d$.

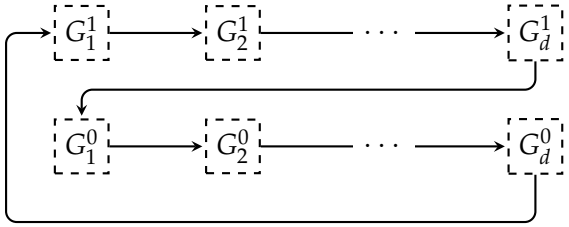


Figure 4.19: The finitary parity game $\mathcal{G}_d$ witnessing exponential memory requirements for Player 0.

show both gadgets in Figure 4.18. The dashed lines denote the edges to the pre- and succeeding gadget and the edge between the final and the initial gadget.

More precisely, the arena $\mathcal{A}_d$ comprises d repetitions of the gadget G_j^1 for Player 1, followed by d repetitions of the gadget G_j^0 for Player 0. We designate the top-left vertex of the initial gadget for Player 1 as the initial vertex v_I. Moreover, the final gadget of Player 0 has a single back-edge to the initial vertex. We illustrate the overall construction of the arena $\mathcal{A}_d$ in Figure 4.19.

As $\mathcal{A}_d$ consists of $2d$ gadgets, each of which contains $3d$ vertices, the arena $\mathcal{A}_d$ is of polynomial size in d. The largest odd color occurring in $\mathcal{A}_d$ is $2d-1$. Hence, $\mathcal{A}_d$ contains only d odd colors.

Remark 4.42. *The game $\mathcal{G}_d$ contains $6d^2$ many vertices and d odd colors.*

We now show that exponential memory is both necessary and sufficient for Player 0 to play optimally in $\mathcal{G}_d$. To this end, for the remainder of this section, fix $b = d^2 + 2d$. We first construct a strategy σ_d of size 2^{d-1} such that $\mathrm{Cost}_{v_\mathrm{I}}(\sigma_d) = b$. To this end, we define a memory structure together with a next-move-function that implements our strategy.

To construct such a strategy, it suffices to consider finite plays infixes similarly to the proof of ➔ Lemma 4.30. Even though the requests are not necessarily all answered after each round, we argue that Player 0 can always do so while playing optimally.

➔ Sec. 4.3, Page 122

Recall that, in parity games with costs, and thus also in finitary parity games, we call a request relevant if either there exists no request for a larger color that has incurred greater cost. We gave a formal definition of this notion in ➔ Section 4.2.1.

➔ Page 99

Intuitively, in order to play optimally, Player 0 tracks the requests made by Player 1 in the first part of each round. Instead of tracking each request precisely, however, it suffices to store the relevant ones. As we are dealing with finitary parity games, and due to the structure of the arena, relevant requests can only be opened by visiting some larger color than all currently open requests. In order to use memory efficiently, we do not initialize our memory structure with the empty sequence of relevant requests, but rather with the "worst-case assumption", i.e., with the memory element encoding that in each gadget of Player 1 a new relevant request was opened.

Following this intuition, we define the set of strictly increasing odd sequences

$$\mathrm{IncSeq}_d = \{(c_1, \ldots, c_d) \mid 1 \leq c_1 \leq \cdots \leq c_d = 2d-1, \\ c_j \neq 2d-1 \text{ implies } c_j < c_{j+1}, \text{ all } c_j \text{ are odd}\}$$

and use them as the set of memory states $M_d = \mathrm{IncSeq}_d$. We observe $|M_d| = 2^{d-1}$, as each increasing sequence is isomorphic to a subset of $\{1, 3, 5, \ldots, 2d-3\}$. We moreover define the initialization function

$$\mathit{init}_d(v) = (1, 3, \ldots, 2d-3, 2d-1) \text{ for all vertices } v \text{ in } \mathcal{A}_d,$$

which suffices as we are later only interested in plays starting in the designated initial vertex v_I.

It remains to define upd_d. To this end, let $m \in M_d$ and let $e = (v, v')$ be an edge of $\mathcal{A}_d$. If $v' = v_\mathrm{I}$, then we define $upd_d(m, e) = init(v_\mathrm{I})$. If, however, $v' \neq v_\mathrm{I}$, but v' is of even color, we define $upd_d(m, e) = m$. It remains to define $upd_d(m, e)$ for the case that v' has odd color. In this case we have $v' = v_{j,c}$ for some j with $1 \leq j \leq d$. Hence, v' is the unique vertex of odd color c in the j-th gadget of Player 1. In this case, let $m = (c_1, \ldots, c_d)$. We differentiate two cases: If the j-th entry of m denotes some color that is at least as large than the currently visited one, i.e., if $c_j \geq c$, then we keep the memory state unaltered and define $upd_d(m, e) = m$. Otherwise, we store the newly visited color in the memory state and define

$$upd_d(m, e) = (c_1, \ldots, c_{j-1}, c, c+2, c+4, \ldots, 2d-1, 2d-1, \ldots, 2d-1) \ .$$

Since upd_d only alters the memory state upon visiting either the initial vertex v_I or some vertex of odd color, the memory state

- is fixed once a partial play leaves the gadgets of Player 1,
- remains unchanged throughout the traversal of the gadgets of Player 0,
- and is only reset upon moving to the initial vertex of $\mathcal{G}_d$.

Towards the definition of σ_d, we define the next-move function nxt_d such that, if the play leaves the gadgets of Player 1 with memory state $m = (c_1, \ldots, c_d)$, then Player 0 moves to color $c_j + 1$ in her j-th gadget. Finally, we define the strategy σ_d as the strategy implemented by $\mathcal{M}_d = (M_d, upd_d, init_d)$ and nxt_d.

Lemma 4.43. *We have* $Cost_{v_\mathrm{I}}(\sigma_d) = d^2 + 2d$.

Proof. Let ρ be a play starting in v_I and consistent with σ_d. Moreover, consider the request for color c made by Player 1 in his initial visit to his j-th gadget. Due to the definition of σ_d, this request is answered before the next turn of the play, i.e., before the next visit to v_I: The final element of the memory state is fixed to be $2d-1$, i.e., Player 0 will move to visit the maximal even color $2d$ during her move through her final gadget.

Assume that Player 0 answers the request for color c in her j'-th gadget. The costs of answering this request for color c consists of three components: First, the play has to leave Player 1's j-th gadget, traversing $d - (c+1)/2 + 2$ many edges. Then, the play passes through $d - j + j'$ many gadgets, traversing $d + 2$ many edges in each. Finally, upon moving to color $c + 1$ in Player 0's j'-th gadget again traverses $(c+1)/2$ many edges. Thus, this request is answered after traversing

$$d - \frac{c+1}{2} + 2 + (d - j + j')(d+2) + \frac{c+1}{2} = d + 2 + (d - j + j')(d+2)$$

many edges. Due to the construction of σ_d, we obtain $(d - j + j') \leq d - 1$, which yields a maximal total cost of answering the request for color c of

$$d + 2 + (d-1)(d+2) = d^2 + 2d \ .$$

Thus, we obtain $\mathrm{Cost}_{v_\mathrm{I}}(\sigma_d) \leq d^2 + 2d$. It remains to argue $\mathrm{Cost}_{v_\mathrm{I}}(\sigma_d) \geq d^2 + 2d$. To this end, however, it suffices to note that the play in which Player 1 requests color $2d-1$

in his first gadget witnesses this lower bound: Player 0 answers this request in her first gadget by moving to $v_{1,2d}$ which takes $d^2 + 2d$ moves. Thus, we obtain $\mathrm{Cost}(\rho) \geq d^2 + 2d$ via the above reasoning. □

From the proof of Lemma 4.43 we furthermore obtain that each play in which Player 1 repeatedly requests color $2d - 1$ in his first gadget has cost at least $d^2 + 2d$. Thus, the strategy σ_d indeed witnesses the optimal bound with respect to which Player 0 wins $\mathcal{G}_d$ from v_I.

Corollary 4.44. *There exists no strategy σ for Player 0 with $Cost_{v_\mathrm{I}}(\sigma) < d^2 + 2d$.*

It remains to argue that σ_d is indeed a minimal strategy for Player 0 from v_I that witnesses this lower bound on the cost for her.

Lemma 4.45. *For each strategy σ for Player 0 we have that $Cost_{v_\mathrm{I}}(\sigma) \leq d^2 + 2d$ implies $|\sigma| \geq 2^{d-1}$.*

Proof. In order to simplify notation, we associate with each memory element $m \in M_d$ a play prefix which starts in the initial vertex v_I, where Player 1 requests the colors occurring in m in order. We denote this partial play by $\mathrm{req}(m)$. Clearly, for any two memory elements $m, m' \in M_d$, we have that $m \neq m'$ implies $\mathrm{req}(m) \neq \mathrm{req}(m')$.

Let σ be a finite-state strategy for Player 0 that is implemented by a memory structure $(M, init, upd)$ with $|M| < 2^{d-1}$. We inductively construct a play that starts in v_I and that is consistent with σ, but that has cost greater than $d^2 + 2d$. To this end, we start with the play $\pi = v_\mathrm{I}$, which is clearly consistent with σ.

Now let $\pi = v_0 \cdots v_j$ with $v_0 = v_j = v_\mathrm{I}$ be the play constructed so far and let m be the memory element attained after traversing π, i.e., let $m = upd^+(init(v_0), v_0 \cdots v_j)$. Due to the pigeon-hole principle, there exist $m'_1 \neq m'_2 \in M_d$, such that $upd^+(m, \mathrm{req}(m'_1)) = upd^+(m, \mathrm{req}(m'_2))$, i.e., Player 0 cannot differentiate between the play infixes $\mathrm{req}(m'_1)$ and $\mathrm{req}(m'_2)$ when starting in memory state m. Moreover, since $\mathrm{req}(m'_1) \neq \mathrm{req}(m'_2)$, as argued above, there exists a gadget of Player 1 in which the requests posed during $\mathrm{req}(m'_1)$ and $\mathrm{req}(m'_2)$ differ. Pick k as the minimal index of such a gadget and assume that in his k-th gadget, Player 1 requests color c during $\mathrm{req}(m'_1)$, and color c' during $\mathrm{req}(m'_2)$, where, w.l.o.g., $c < c'$.

If, upon reacting to $\mathrm{req}(m'_1)$ and $\mathrm{req}(m'_2)$, Player 0 has already answered the request for c' upon entering her k-th gadget, then some earlier request is not answered optimally when reacting to $\mathrm{req}(m'_2)$, as requests are posed in strictly increasing order in $\mathrm{req}(m'_2)$. Thus, we extend the play π by $\mathrm{req}(m'_2)$ and continue consistently with σ until we encounter v_I again.

If, on the other hand, Player 0 does not answer the request for color c' upon entering her k-th gadget, then first assume that she visits some color $c'' < c'$ in this gadget. Then she will only answer c' in some later gadget, thereby incurring a cost greater than $d(d+2) = d^2 + 2d$ when reacting to $\mathrm{req}(m'_2)$. We again extend π by $\mathrm{req}(m'_2)$ and its continuation consistent with σ up to and including the next visit to v_I.

If she instead visits some color $c'' \geq c'$, then she does not answer the request for c optimally, thus incurring a cost of at least $d^2 + 2d + (c' - c)/2 > d^2 + 2d$ when reacting

to $\text{req}(m_1')$. Thus, we extend π by $\text{req}(m_1')$ and by playing consistently with σ until the next encounter with v_I.

The resulting play ρ starts in v_I and is, by construction, consistent with σ. Moreover, in each round of ρ, Player 1 opens a request that is answered with cost greater than $d^2 + 2d$, i.e., we obtain $\text{Cost}(\rho) > d^2 + 2d$. □

Combining the lemmas proved in this section shows that Player 0 indeed requires exponential memory in order to play optimally. We summarize our findings from this section in the following theorem.

Theorem 4.46. *For every $d \geq 1$ there exists a finitary parity game $\mathcal{G}_d$ with a vertex v_I such that*

- *$\mathcal{G}_d$ has d odd colors and $|\mathcal{G}_d| \in \mathcal{O}(d^2)$,*
- *Player 0 has a strategy σ in $\mathcal{G}_d$ with $\text{Cost}_{v_\text{I}}(\sigma) = d^2 + 2d$,*
- *there exists no strategy σ' for Player 0 with $\text{Cost}_{v_\text{I}}(\sigma') < d^2 + 2d$, and*
- *for every strategy σ for Player 0 in $\mathcal{G}_d$, $\text{Cost}_{v_\text{I}}(\sigma) \leq d^2 + 2d$ implies $|\sigma| \geq 2^{d-1}$.*

Proof. The first item of the theorem is encapsulated in Remark 4.42, while the second and third item follow directly from Lemma 4.43 and Corollary 4.44, respectively. Finally, the last item of the theorem is encapsulated in Lemma 4.45. □

We have thus shown that Player 0 requires exponential memory to play optimally in finitary parity games. This contrasts the existence of positional winning strategies for her due to ➔Proposition 2.25.1. Since parity games with costs subsume finitary parity games, and since Player 0 still has positional winning strategies in the former games (see ➔Proposition 2.34.1), the same result holds true for such games as well.

➔Sec. 2.4, Page 23

➔Sec. 2.4, Page 26

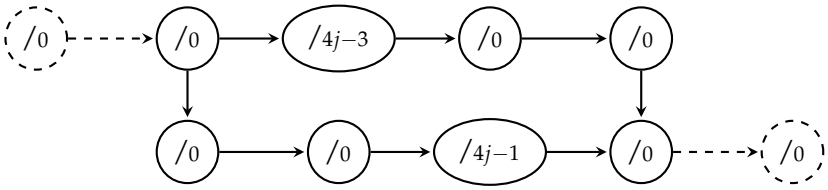
For parity games with weights, however, we do not obtain such a large gap between the memory required by winning strategies and that required by optimal ones, since Player 0 already requires exponential memory in parity games with weights (cf. ➔Theorem 3.34). Here, however, the characteristics of the upper bound change as follows. Recall that she requires polynomial memory in the number of vertices, the number of odd colors, and the largest absolute weights occurring in a parity game with weights in order to win, where the latter value is, in general, exponential in the size of the game due to binary encoding. In order to keep the costs below a given bound b, however, she requires memory that is polynomial in b (for a fixed number of odd colors) and exponential in the number of odd colors (for a fixed bound b).

➔Sec. 3.4, Page 71

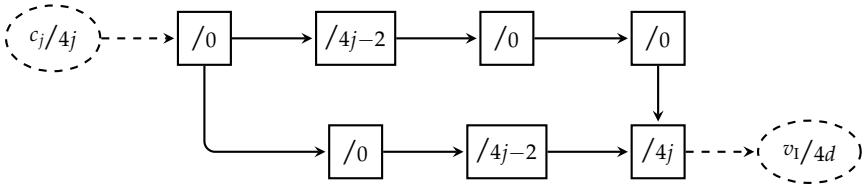
Having thus shown that Player 0 indeed requires exponential memory in order to play optimally even in finitary parity games, we now show an analogous result for Player 1.

4.4.3 Player 1 Requires Exponential Memory

To show that Player 1 requires exponential memory to exceed a given lower bound on the incurred cost in a finitary parity game, we use a similar idea to the proof of the analogous result for Player 0 in the previous section: For each $d > 1$ we first



(a) The j-th gadget G_j^0 for Player 0. If $j = 1$, then the top left vertex has label v_I and color $4d$. If $j = d$, then the bottom right vertex has label c_1 and color zero.



(b) The j-th gadget G_j^1 for Player 1.

Figure 4.20: The gadgets constituting the arena $\mathcal{A}_d$ of $\mathcal{G}_{d,W}$.

construct a finitary parity game $\mathcal{G}_d$ with polynomially many vertices in d together with a bound b that is polynomial in d. We subsequently show that Player 1 can enforce cost b from some designated vertex v_I in $\mathcal{G}_d$, but that he cannot enforce any larger bound. Furthermore, we show that he requires memory exponential in d in order to implement a strategy witnessing this fact.

Recall that the aim of Player 1 is dual to that of Player 0, i.e., Player 1 aims to ensure that the resulting play has infinite cost. Thus, Player 1 winning $\mathcal{G}_d$ with respect to the bound b amounts to him enforcing a cost of the resulting play exceeding b.

For the remainder of this section, fix some $d > 1$. Similarly to the previous proof, we construct the arena $\mathcal{A}_d$ using two kinds of gadgets, one for each player, each of which occurs d times. In $\mathcal{A}_d$, first Player 0 opens d requests and subsequently picks one of these requests to be answered. If Player 1 recalls the requests, then he can cause the request to be answered with cost $b = 5(d-1)+7$. Otherwise, Player 0 can construct a sequence of requests that is answered with cost less than $5(d-1)+7$.

We show the gadgets in Figure 4.20 together with their coloring, and call them G_j^0 and G_j^1 for the j-th gadget of Player 0 and Player 1, respectively. The gadget G_j^0 contains the colors 0, $4j-3$, and $4j-1$, while the gadget G_j^1 contains the colors 0, $4j-2$, and $4j$. Moreover, we show the complete arena $\mathcal{A}_d$ in Figure 4.21. In that figure, we label some edges with weight five for the sake of readability. These edges can, however, be subdivided into five edges each, thus obtaining a finitary parity game with only linear blowup. We fix the initial vertex v_I to be the vertex drawn in the top-left corner of Figure 4.21. Furthermore, we observe that $\mathcal{A}_d$ consists of $2d$ gadgets, each of which contains eight vertices (counting the vertices c_j towards the gadgets of Player 1) plus an initial vertex, and that the largest odd color occurring in $\mathcal{G}_d$ is $4d-1$.

Remark 4.47. *The game $\mathcal{G}_d$ contains $20d-3$ vertices, and $2d$ odd colors.*

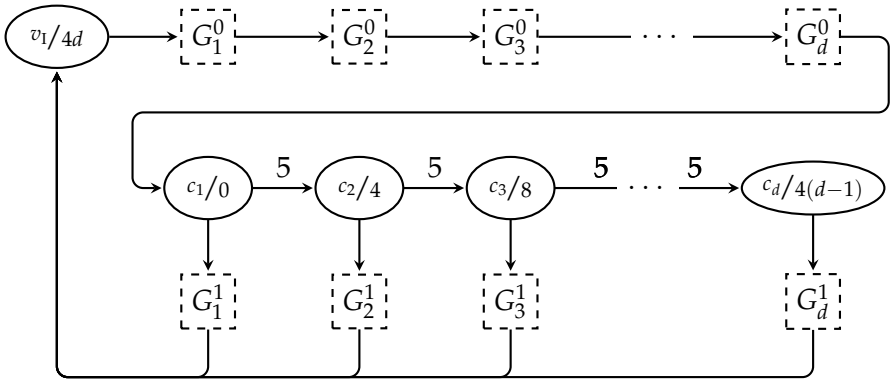


Figure 4.21: The arena $\mathcal{A}_d$ witnessing exponential memory requirements for Player 1.

The game $\mathcal{G}_d$ is played in rounds. Each such round starts and ends in the initial vertex v_I that answers every request. Thus, the rounds are independent of each other and it suffices to analyze them in isolation: In each round, Player 0 first poses d requests, one in each of her gadgets. She may use her j-th request to either request the color $4j-3$ and traverse three edges before leaving G_j^0, or she may request the color $4j-1$ and traverse one edge before leaving G_j^0. After posing d requests, Player 0 then picks some j with $1 \leq j \leq d$ and moves to the j-th gadget G_j^1 of Player 1 while answering all requests for colors $c \leq 4(j-1)$ with cost less than $5(d-1)+7$ along the way.

In his j-th gadget G_j^1, Player 1 then answers the request posed in the j-th gadget G_j^0 of Player 0. To this end, he either opts to answer the request for color $4j-3$ and $4j-1$ after taking two transitions and after three transitions, respectively, or after taking one transition and four transitions, respectively. After he has done so, all requests are reset and the next round begins by moving to v_I.

The cost for a request posed in $G_{j'}^0$ for $j' < j$ consist of the cost of leaving that gadget, traversing $d-j'$ gadgets of Player 0, and moving to $c_{j'}$. Thus, this request incurs a cost of at most

$$4 + (d-j')5 + (j'-1)5 = 5(d-1) + 4 < 5(d-1) + 7 \ .$$

Similarly, all requests posed in $G_{j'}^0$ for $j' > j$ are answered with cost at most

$$4 + 5(d-j') + 5(j-1) + 5W = 4 + 5d + 5(j'-j) < 5d - 1 < 5(d-1) + 7$$

immediately after leaving G_j^1. We first show that by recalling all requests, which requires 2^d many memory states, Player 1 can ensure that one request is only answered with cost $5(d-1)+7$.

To this end, we define the strategy τ_d for Player 1 as follows: During Player 0's part of the round, Player 1 stores the requests that she makes using 2^d memory states. Assume that Player 0 then opts to move to G_j^1. If Player 0 requested color $4j-3$ in

her j-th gadget during her part of the round, Player 1 picks the upper branch shown in Figure 4.20b, while he chooses the lower branch in case Player 0 requested color $4j-1$.

Lemma 4.48. *We have* $Cost_{v_I}(\tau_d) = 5(d-1)+7$.

Proof. Consider a play prefix that starts in v_I, is consistent with τ_d, ends in v_I and pick j such that the play has moved through G_j^1 in the last turn. We first show that all requests for colors $c < 4j-3$ and for colors $c > 4j-1$ posed during the last turn have been answered with cost less than $5(d-1)+7$ before showing that the request for either color $4j-3$ or for color $4j-1$ has been answered with cost $5(d-1)+7$. Since this holds true for all play prefixes starting and ending in v_I and consistent with τ_d, we then obtain $\mathrm{Cost}(\rho) = 5(d-1)+7$ for all plays starting in v_I and consistent with τ_d, which suffices to obtain the desired result.

Before moving through G_j^1 during the last turn, the play has passed through the vertex c_j due to the construction of $\mathcal{G}_d$. All requests for colors $c < 4j-3$ have already been answered by a visit to some vertex $c_{j'}$ with $j' < j$ due to the structure of the arena. As argued above, these requests have incurred cost at most $5(d-1)+4$.

Moreover, again as argued above, all requests for colors $c > 4j-1$ are unanswered and have incurred a cost of at most $5(d-1)-1$. These requests are answered upon visiting the vertex v_I, which is reached by traversing at most five edges from c_j, i.e., they are answered with cost at most $5(d-1)+4$.

It remains to show that either the request for color $4j-3$ or that for color $4j-1$ was answered with cost $5(d-1)+7$. By construction of $\mathcal{G}_d$, only one of these requests is open upon reaching c_j. First consider the case that there is an open request for color $4j-3$ upon entering gadget G_j^1. Then, according to τ_d, Player 1 moves through the lower branch of his gadget, answering this request with cost $5(d-1)+7$. If there is, however, an open request for color $4j-1$ upon entering G_j^1, then the strategy τ_d prescribes for Player 1 to move through the upper branch of his gadget, answering the open request with cost $5(d-1)+7$ as well. □

Similarly to the proof of the analogous result for Player 0, we again observe that in the proof of Lemma 4.48 we argue that Player 0 can enforce costs of at most $5(d-1)+7$ regardless of the choices made by Player 1.

Corollary 4.49. *There exists no strategy* τ *for Player* 1 *with* $Cost_{v_I}(\tau) > 5(d-1)+7$.

The above corollary shows that the strategy τ_d for Player 1 is indeed optimal for him. In order to conclude this section, we show that the strategy τ_d is not only optimal for Player 1, but also minimal.

Lemma 4.50. *For each strategy* τ *for Player* 1 *we have that* $Cost_{v_I}(\tau) \geq 5(d-1)+7$ *implies* $|\tau| \geq 2^d$.

Proof. Towards a contradiction, let τ be a finite-state strategy for Player 1 that is implemented by a memory structure (M, m_I, upd), where $|M| < 2^d$ and that satisfies $\mathrm{Cost}_{v_I}(\tau) \geq 5(d-1)+7$. We again inductively construct a play ρ consistent with τ

such that $\text{Cost}(\rho) < 5(d-1)+7$ and begin with the play prefix v_I. Now assume we have already defined a prefix π of ρ that starts in v_I, is consistent with τ, and ends in the initial vertex v_I. We determine a sequence of d requests and a choice of $1 \leq j \leq d$ and prolong π by letting Player 0 first pick the sequence of requests and then move into some gadget G_j^1. Then, Player 1 applies his strategy, which leads back to the initial vertex v_I.

To this end, let $m = upd^+(init(v_\text{I}), \pi)$. Since $|M| < 2^d$, and since there exist 2^d play infixes leading from the unique successor of v_I to c_1, there exist two such infixes π_1 and π_2, such that $upd^+(m, \pi_1) = upd^+(m, \pi_2)$. Let j be minimal such that the choices made in G_j^0 by Player 0 differ in π_1 and π_2, and w.l.o.g. assume that Player 0 poses a request for color $4j-3$ when playing π_1, while she poses a request for color $4j-1$ when playing π_2.

Now consider the response of Player 1 consistent with τ if Player 0 moves to G_j^1 after the play prefix $\pi\pi_1$ and observe that this response is the same as the one to the play prefix $\pi\pi_2$ due to $upd^+(m, \pi_1) = upd^+(m, \pi_2)$. If Player 1 traverses the upper branch of his gadget G_j^1 after witnessing $\pi\pi_1$ or $\pi\pi_2$, then he answers the request for $4j-3$ posed during the traversal of π_1 with cost $5(d-1)+6$. If he, however, traverses the lower branch of G_j^1 after witnessing $\pi\pi_1$ or $\pi\pi_2$, then he answers the request for $4j-1$ posed during π_2 with cost $5(d-1)+6$. In the former case, we continue π by letting Player 0 play according to π_1, while in the latter case we continue π by letting her play according to π_2. In either case, we move to G_j^1 afterwards. In G_j^1, Player 1 then plays consistently with τ.

In both cases all requests posed in gadgets $G_{j'}^0$ for $j' < j$ are answered after at most $5(d-1)+5$ steps, namely upon reaching the vertex $c_{j'+1}$. Also, the request posed in G_j^0 is answered after $5(d-1)+6$ steps. Finally, all requests posed in gadgets $G_{j'}^0$ for $j' > j$ are answered after at most $5(d-1)+5$ steps upon reaching the vertex v_I.

Since all requests are answered when reaching v_I, we extend the play prefix π as discussed above, obtaining an extension of π that again ends in the vertex v_I. By applying this construction inductively and since the reasoning above holds true for any memory state m reached at the end of any π as above, the play ρ thus resulting has $\text{Cost}(\rho) \leq 5(d-1)+6$. Since ρ is consistent with τ, this contradicts $\text{Cost}_{v_\text{I}}(\tau) \geq 5(d-1)+7$. □

We again summarize the results obtained in this section in the following theorem.

Theorem 4.51. *For every $d \geq 1$ there exists a finitary parity game $\mathcal{G}_d$ with a vertex v_I such that*

- *$\mathcal{G}_d$ has $\mathcal{O}(d)$ many vertices and $2d$ odd colors,*
- *Player 1 has a strategy τ in $\mathcal{G}_d$ with $Cost_{v_\text{I}}(\tau) = 5(d-1)+7$,*
- *there exists no strategy τ' for Player 1 with $Cost_{v_\text{I}}(\tau') > 5(d-1)+7$, and*
- *every strategy τ for Player 0 in $\mathcal{G}_d$ with $Cost_{v_\text{I}}(\tau) \geq 5(d-1)+7$ has size at least 2^d.*

Proof. The first item of the theorem is encapsulated in Remark 4.47, while the second and third item follow directly from Lemma 4.48 and Corollary 4.49, respectively.

Finally, the last item of the theorem is encapsulated in Lemma 4.50. □

Similarly to the case for Player 0, these results also hold true for the cases of parity games with costs and parity games with weights.

This concludes our investigation of the memory bounds for both players in parity games with weights. In the following section we discuss the influence of the encoding of the weight function on the complexity results obtained in this chapter.

4.5 Discussion: Unary Encoding of Weights

Similarly to the previous chapter, we have assumed the weighting to be given in binary encoding. Thus, the size of a parity game with weights $\mathcal{G}$ grows only logarithmically in the value of its largest absolute weight W: We have $|\mathcal{G}| \in \mathcal{O}(n \log W)$, where n is the number of vertices of $\mathcal{G}$. In this section, we discuss the implications of encoding the weights in unary. Hence, for the remainder of this section, we fix $W = 1$, i.e., all weights are in $\{-1, 0, 1\}$.

Since we showed ExpTime-membership of the threshold problem for parity games with weights in ➔Theorem 4.12, we directly obtain ExpTime-membership of the same problem for parity games with weights given in unary encoding. Analogously, PSpace-hardness of the latter problem follows directly from our proof of PSpace-hardness of the problem of solving finitary parity games optimally, i.e., from ➔Theorem 4.31.

➔Sec. 4.1, Page 96

➔Sec. 4.3, Page 123

Remark 4.52. *The following decision problem is in* ExpTime *and* PSpace*-hard:*

> *"Given a parity game with weights $\mathcal{G}$ with largest absolute weight 1, some vertex v^* of $\mathcal{G}$, and a bound $b \in \mathbb{N}$, does Player 0 have a strategy σ with $Cost_{v^*}(\sigma) \leq b$ in $\mathcal{G}$?"*

Recall that we showed ExpTime-hardness via a reduction from the problem of solving countdown games. The problem of solving countdown games is, in turn, known to be ExpTime-hard due to a reduction from the word problem for alternating Turing machines that use at most polynomial space (see ➔Proposition 4.35). In order to obtain a polynomial reduction, however, the authors require the weights of the countdown game as well as its initial credit to be given in binary encoding. As argued in ➔Section 4.3.2, encoding the weights in unary would entail an exponential blowup of the resulting countdown game. Thus, this approach cannot easily be adapted to show ExpTime-hardness of the threshold problem for parity games with weights given in unary encoding.

➔Sec. 4.3, Page 126

➔Page 124

The techniques used for showing PSpace-membership of the analogous problem for parity games with costs, on the other hand, also do not translate to this more general problem: Recall that for the proof of PSpace-membership in ➔Section 4.2.1 for that problem we defined dominating cycles for Player 0 and Player 1, which we called even and odd dominating cycles in the threshold game $\mathcal{G}_b$, respectively.

➔Page 99

This definition was based on the intuition that traversing a dominating cycle for Player i ad infinitum causes Player i to win the resulting play. We then showed that the

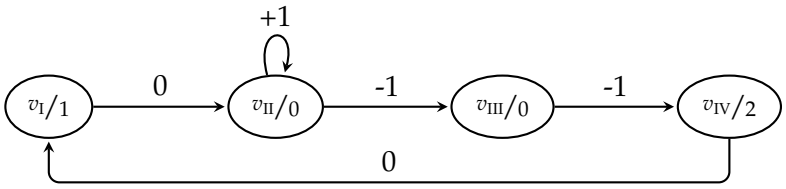


Figure 4.22: A parity game with weights witnessing traversal of edges of positive weight to be beneficial for Player 0.

game in which Player 0 is required to enforce an even dominating cycle before an odd one occurs has the same winning regions as the original threshold game. Furthermore, we defined a shortcut mechanism in ➔Section 4.2.2 that allowed dominating cycles to be reached in at most polynomially many steps.

➔ Page 103

In parity games with weights, however, our formalization of the notion of "beneficial" cycles does not capture the intuitively required properties anymore. On a basic level, our definition of beneficial cycles relied on the fact that it is in general beneficial for Player 1 to traverse edges of positive weight, while it is detrimental for Player 0 to do so. This does not hold true in parity games with weights anymore.

Example 4.53. Consider the parity game with weights shown in Figure 4.22. Player 0 has a strategy σ with $\mathrm{Cost}_{v_{\mathrm{I}}}(\sigma) = 1$ that prescribes moving from v_{I} to v_{II} and iterating the self-loop of v_{II} once before continuing to v_{III}. Furthermore, it can easily be verified that this is the unique strategy with cost one from v_{I}. △

Furthermore, beneficial cycles may not be witnessed using at most polynomially many steps even in the restricted setting of weights given in unary encoding, even if we only use the intuitive definition given above.

Example 4.54. Fix some $n \geq 1$. We construct a parity game with weights $\mathcal{G}_n$ with $\mathcal{O}(n)$ vertices and largest absolute weight n as well as a bound b_n such that Player 0 wins $\mathcal{G}_n$ from some designated vertex v_{I} with respect to the bound b_n, but such that she is only able to close a "beneficial cycle" after visiting exponentially many vertices. We only use weights whose absolute value is larger than one for the sake of readability. The resulting game can easily be transformed into one with largest absolute weight one with only a polynomial blowup.

Intuitively, we use n colors to implement a n-bit binary counter, which is initialized to $2^n - 1$. In each turn of $\mathcal{G}_n$, Player 0 can decrement the counter. We set up $\mathcal{G}_n$ such that each such decrement is, in isolation, beneficial for Player 1, as the largest color seen during such a decrement is odd and since the accumulated weight incurred by executing the decrement is zero. Once the counter has, however, reached zero, we incentivize Player 0 to restart the game. Thus, the play prefix consisting of exponentially many decrement-operations is beneficial for Player 0, while none of its subsequences is beneficial for her.

To encode the counter, we again employ a construction similar to the proof of PSpace-hardness of solving parity games with costs optimally in Theorem 4.31 and

ensure that requests for smaller colors incur larger cost than request for larger ones in order to force both players to keep track of all open requests. Following this intuition, for $j \in \{1, \dots, n\}$ an open request for color $4j - 3$ with cost $n - j + 1$ denotes that the j-th bit of the counter is set to one, while the absence of such a request denotes that bit being set to zero. Thus, we fix $b_n = n$. We use the "unused" colors $4j - 1$ as the odd colors making each individual decrement-operation beneficial for Player 1.

In order to allow Player 0 to decrement the counter, the arena of $\mathcal{G}_n$ contains a central vertex c that belongs to Player 0 and that is connected to n gadgets G_j. We show the construction of the j-th gadget G_j in Figure 4.23a. First, Player 1 may move to reset the play at vertex $g_{j,\to}$. If, upon visiting that vertex, a request for some color $4j' - 3$ with $j' < j$ is open, it is open with cost $n - j' + 1 = b_n - j' + 1$. Hence, moving along the edge with weight j causes that request to be answered with cost exceeding b_n.

If Player 1 does not reset the play at vertex $g_{j,\to}$, the play proceeds through the gadget automatically. First, the request for color $4j - 3$ and all smaller colors is answered at vertex a_j. Subsequently, the requests for all colors $4j' - 3$ for $j' < j$ are reopened with the "correct" costs as stated above.

Finally, the play visits a vertex of color $4j - 1$, which serves as a guard against repeated visits to G_j: The presence of this large odd color forces Player 0 to eventually reset the $(j + 1)$-th bit instead of repeating a sequence of gadgets with highest gadget G_j ad infinitum.

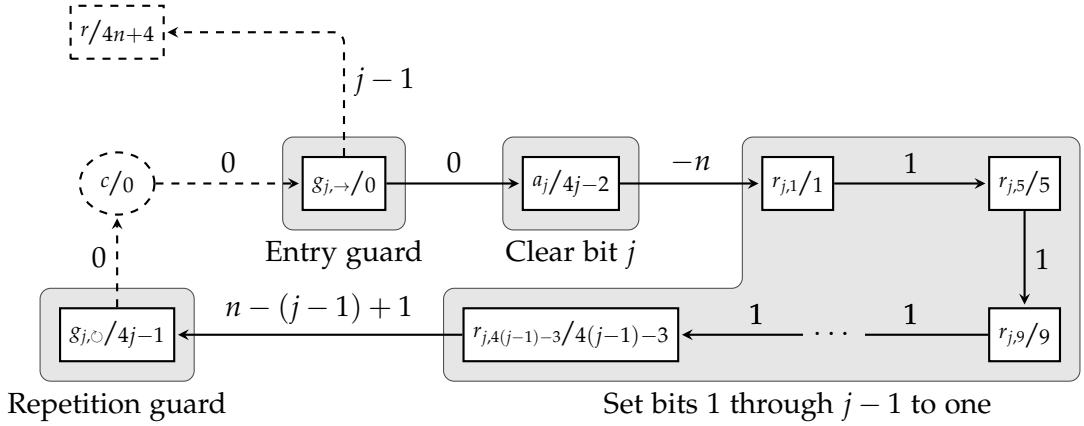
We arrange the gadgets around a single vertex c as described above and show the overall construction of the game $\mathcal{G}_n$ in Figure 4.23b. From the vertex c, there is also an edge of weight $n = b_n$ leading to the vertex r, which answers all requests and restarts the game. The vertex c is the only vertex belonging to Player 0, i.e., the only agency she has is that of choosing to either restart the game or to move to one of the $\mathcal{G}_j$. Due to the construction of the arena, however, all requests except for the repetition guards have cost at least one whenever the play encounters the vertex c. Hence, Player 0 can only reset the game once she has decreased the counter to zero and there are no open requests anymore, i.e., after $2^n - 1$ visits to the gadgets.

A strategy for Player 0 in $\mathcal{G}_n$ now essentially consists of assigning to each counter value in the range $0, \dots, 2^n - 1$ a gadget $G_1, \dots, G_n$ or choosing to reset the game. A strategy prescribing moving to the gadget j', where j' is the lowest bit that is set to one in the counter value represented by the open requests answers all requests with cost at most b_n, as described above. Otherwise, Player 1 may move to vertex r from some vertex $g_{j,\to}$, causing some requests to be answered with cost exceeding b_n, and restarting the game. Hence, the resulting play consists of visiting the gadgets in the sequence

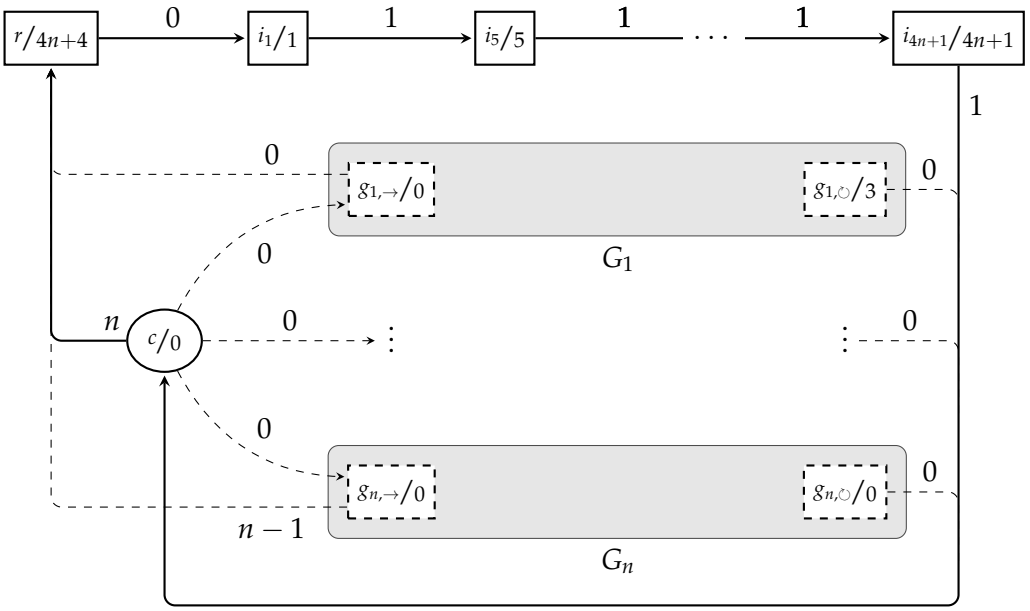
$$G_1c \cdot G_2cG_1c \cdot G_3cG_1cG_2cG_1c \cdot G_4cG_1cG_2cG_1cG_3cG_1cG_2cG_1c \cdots G_1c \cdot r \ ,$$

which visits $2^n - 1$ gadgets.

Now let $\pi = G_{j_0}c \cdots G_{j_k}c$ be a sequence of less than $2^n - 1$ gadgets that does not visit the vertex r and assume that there are open requests for all colors $4j - 3$ with cost $n - j + 1$ for all j with $1 \leq j \leq n$ at the beginning of π. If π at some point visits



(a) The j-th gadget G_j.



(b) The game $\mathcal{G}_d$.

Figure 4.23: Construction of $\mathcal{G}_n$.

a G_j although there is an open request for some color $4j' - 3$ with $j' < j$, then that request has incurred cost $n - j' + 1$ upon visiting the entry guard $g_{j,\to}$ of G_j. Hence, Player 1 can opt to move to the vertex r at the entry guard of G_j, causing the request for color $4j' - 3$ to be answered with cost $n - j' + 1 + j - 1 > n = b_n$. Thus, in order to bound the costs of the resulting play by b_n when starting in r, Player 0 indeed has to traverse the gadgets in the order described above, only moving to r after visiting $2^n - 1$ gadgets.

Moreover, in each such sequence π that starts and ends in c, but that does not visit r, the largest visited color is odd due to the repetition guards contained in each gadget. Thus, repeating any such sequence ad infinitum would be losing for Player 0. The only cycle that can be considered "beneficial" for her is one that starts and ends in vertex r and visits 2^n gadgets in-between. △

The above example illustrates that our approach used for showing membership of the threshold problem for parity games with costs in PSpace is unlikely to yield the same result for parity games with weights with unary encoding. It remains open whether the problem of solving parity games with weights is ExpTime-hard or in PSpace when the weights are given in unary encoding.

The results on the upper and lower bounds on the memory states required by either player to implement winning strategies given in ➔Section 4.4, however, still hold true ➔Page 130 if the weights are given in unary encoding. The upper bounds follow directly from the construction of threshold games presented in section ➔Section 4.1, which treats ➔Page 86 the more general case of parity games with weights given in binary encoding. The lower bounds, on the other hand, are witnessed by finitary parity games.

Remark 4.55. *Let $\mathcal{G}$ be a parity game with weights containing d odd colors with largest absolute weight one and let $b \in \mathbb{N}$.*

1. *If Player 0 has a strategy σ in $\mathcal{G}$ with $Cost(\sigma) = b$, then she also has a strategy σ' with $Cost(\sigma') \leq b$ and $|\sigma'| \in \mathcal{O}(b^d)$. This bound is tight.*
2. *If Player 1 has a strategy τ in $\mathcal{G}$ with $Cost(\tau) = b$, then he also has a strategy τ' with $Cost(\tau') \geq b$ and $|\tau'| \in \mathcal{O}(nb^d)$. This bound is tight.*

Having thus discussed the implications of the encoding of weights on the results obtained in this chapter, we now give a brief summary of these results.

4.6 Summary of Results

In this section we have investigated the threshold problem for parity games with weights and their special cases of parity games with costs and finitary parity games. We summarize our results in Table 4.24. First, we have reduced this problem to that of solving classical qualitative parity games of exponential size in ➔Section 4.1 before ➔Page 86 showing that we can solve the resulting parity games using only polynomial space in the special case of parity games with costs in ➔Section 4.2. These two results place the ➔Page 97 problem in the complexity class ExpTime for the general case and in the complexity class PSpace for the special case of parity games with weights.

	Complexity	Mem. Pl. 0/Pl. 1
Finitary Parity Games	PSpace-c.	exp./exp.
Parity Games with Costs	PSpace-c.	exp./exp.
Parity Games with Weights	ExpTime-c.	exp./exp.

Table 4.24: Characteristic properties of solving parity games with weights and their special cases optimally.

Subsequently, we have provided lower bounds in the form of hardness results for these decision problems in ➔Section 4.3, showing the two problems to be complete for their respective complexity classes. In fact, we have shown the lower bound of PSpace-hardness even for the case of finitary parity games, thus also showing PSpace-completeness of the latter problem.

➔Page 116

Having thus settled the complexity of this problem we have investigated the memory required by either player to implement a strategy that enforces a given upper (in the case of Player 0) or lower (in the case of Player 1) bound on the cost of the resulting play in ➔Section 4.4. We have shown that both players require exponential memory in this setting even in the special case of finitary parity games, but also that exponential memory suffices even for parity games with weights.

➔Page 130

The upper bound given by these results, however, increases with the bound b. Intuitively, however, the larger the bound b, the easier it should become for Player 0 to ensure cost below that bound. We later show in ➔Section 6.1 that this is indeed the case.

➔Page 174

Finally, we have discussed the implications of the encoding of the weights on the results obtained in this chapter in ➔Section 4.5. We have shown that all results still hold true for the special case of parity games with costs, but that there remains a gap in the characterization of the complexity of the problem of solving parity games with weights with respect to a given bound if the weights are given in unary encoding: This problem is in ExpTime, but it is only known to be PSpace-hard. The above results on the memory required by either player to implement a strategy that enforces a given bound, however, still remain valid, irrespective of the encoding of the weights.

➔Page 143

Recall that for the special case of finitary parity conditions, Bruyère, Hautem, and Randour [BHR16] also considered the setting of multiple colorings, each of which induces a finitary parity condition. It is then the goal of Player 0 to satisfy all induced conditions. The authors have shown the threshold problem for this setting to be ExpTime-complete. It remains for future work, however, to extend their model to the setting of parity games with costs or parity games with weights. Moreover, analogously to the case of the boundedness problem, one could consider the case of not only multiple colorings inducing parity conditions with weights, but also that of multiple weight functions for a single coloring function, as well as the combination of multiple coloring functions, each of which is associated with its respective weight function.

CHAPTER 5

Games with Disturbances

Recall that in ➔ Section 1.1 we have described how infinite games in general and parity games in particular are used routinely in the verification and synthesis of embedded systems: A wide range of specification languages for such systems can be compiled down into parity games such that solving the resulting game yields a strategy that can be turned into a controller for the system that satisfies the specification. Büchi and Landweber [BL69] showed that such games are determined, while Mostowski [Mos91] as well as Emerson and Jutla [EJ91] showed that positional strategies suffice for both players to win parity games. Thus, we obtain a solution to the synthesis problem as formulated by Church [Chu57] for the model of parity games. ➔ Page 2

In ➔ Section 1.3, we have argued that the canonical model of infinite games underlying both parity games as well as their quantitative extensions presented in the previous chapters implicitly induces an overly optimistic world view: Recall that, intuitively, in an infinite game modeling a reactive system, the vertices of the game denote possible states of the system and its environment. Moreover, each outgoing edge of a vertex for Player 0 represents an action that is available to the system in the corresponding state. ➔ Page 8

Thus, modeling a reactive system as an infinite game presumes not only that the designer of the game has complete knowledge about the interactions between the system and its environment, but also that an action taken by the system always has the intended effect. In real-world systems, however, this need not always be the case. The controller may prescribe taking an action, but the expected result does not manifest, e.g., due to a malfunctioning actuator or a bit flip, as illustrated in Section 1.3. Since embedded systems are often deployed in safety-critical roles it is paramount to have formal methods for the development and analysis of such systems.

Dallal, Neider, and Tabuada [DNT16] formalized the concept of unintended behavior under the name of "games with unmodeled intermittent disturbances". Following their notion, we summarize all events that introduce uncertainty about the results of

the actions of Player 0 under the generic term "disturbance." These disturbances manifest themselves in the framework of infinite games as special edges that are not under the control of either player, but that are traversed nondeterministically: Whenever it is the turn of Player 0, after she has picked an outgoing edge of the current vertex, a disturbance may occur. This causes her original move to be overridden and the token to move along a disturbance edge instead of the edge chosen by Player 0.

Such disturbances are rare events that do not occur antagonistically nor stochastically: If we assumed disturbances to be antagonistic, it would, in general, be impossible for Player 0 to satisfy the winning conditions. Moreover, as disturbances only occur rarely, there is, in general, not enough information to develop a stochastic model of their occurrence. Thus, it does not realistically model the system to give control over these edges to Player 1 or to a stochastic third player.

In their work, Dallal, Neider and Tabuada investigated the problem of constructing winning strategies that are still winning even under the occurrence of disturbances only for the special case of safety games, i.e., for a very restrictive winning condition. They showed that this problem is as complex as solving safety games without taking disturbances into account and that the resulting strategies are still positional. Hence, in safety games, the construction of strategies that are resilient against disturbances comes for free, both in terms of the complexity of computing such strategies, as well as in terms of the size of the resulting strategies.

In this chapter, we first extend the model of Dallal, Neider, and Tabuada to the variants of parity games discussed in this thesis, i.e., to parity games, finitary parity games, parity games with costs, and parity games with weights in Section 5.1. The extension of this model to the setting of parity games and their variants introduces a novel phenomenon: In safety games as considered by Dallal, Neider, and Tabuada, the authors showed that Player 0 can either win the game from a given vertex even if infinitely many disturbances occur, or the occurrence of less than n disturbances cause her to lose the game, where n denotes the number of vertices of the game. In parity games, in contrast, there exists a third possibility: There exist parity games with disturbances in which Player 0 can win as long as there are only finitely many disturbances, but she cannot win anymore if infinitely many disturbances occur.

We show a parity game witnessing this behavior in Figure 5.1, where the dotted edge indicates a disturbance edge. Here, each occurrence of a disturbance incurs a visit to a vertex of color one and, if no disturbance occurs, a vertex of color zero is visited. Hence, Player 0 wins a play from either vertex if and only if the number of disturbances is finite.

After having lifted the notion of disturbances and resiliences to the extended setting of parity games with weights, we subsequently show how to compute the maximal number of disturbances that still allow Player 0 to win in Section 5.2. In order to compute this value, we significantly generalize the approach of Dallal, Neider, and Tabuada to constructing resilient strategies for safety games with disturbances. Their solution was tailored specifically to safety games, as they adapted the standard attractor construction commonly used for solving safety games such that it also takes disturbances into account.

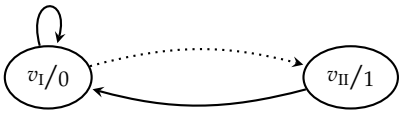


Figure 5.1: A parity game in which Player 0 wins from each vertex as long as only finitely many disturbances occur, but loses if infinitely many disturbances occur.

In this work, in contrast, we develop an approach to computing the resilience of vertices that is independent of the concrete method used to solve the underlying game without disturbances. Instead, our approach uses an algorithm solving the underlying games without disturbances as a blackbox. Using this approach, we are able to disentangle handling the disturbances occurring in the game from the actual winning condition of the underlying game.

Finally, still in Section 5.2, we show how to leverage our approach from the preceding section to compute strategies witnessing the maximal number of disturbances that still allows Player 0 to win the resulting play. To this end, we show that our algorithm computing the resilience of vertices can easily be adapted to yield such strategies as a by-product. We conclude this chapter by summarizing our results in Section 5.3.

This chapter is based on work published at CSL 2018 [NWZ18b].

5.1 Definitions

Throughout this work, we have defined the natural numbers $\mathbb{N}$ via their intuitive meaning. Moreover, we have used the special symbol ∞ with the property $\infty > n$ for all $n \in \mathbb{N}$. In this chapter, however, it is prudent to employ ordinal notation, i.e., to define the natural numbers via set inclusion for notational convenience.

To this end, we inductively define the nonnegative natural numbers as $0 = \emptyset$ and $n + 1 = n \cup \{n\}$. Now, the FIRST LIMIT ORDINAL is $\omega = \{0, 1, 2, \ldots\}$, i.e., the set of the nonnegative integers. The next two SUCCESSOR ORDINALS are $\omega + 1 = \omega \cup \{\omega\}$ and $\omega + 2 = \omega + 1 \cup \{\omega + 1\}$. These ordinals are ordered by set inclusion, i.e., we have $0 < 1 < 2 < \cdots < \omega < \omega + 1 < \omega + 2$. Furthermore, we also denote the cardinality of ω by ω.

Def. ω

Def. $\omega + 1$, $\omega + 2$

Following the intuition given in the introduction to this chapter, we first formalize the notion of disturbances in infinite games in Section 5.1.1. Subsequently, we define how we measure the resilience of strategies against such disturbances in Section 5.1.2.

5.1.1 Arenas and Games with Unmodeled Disturbances

As described above, we augment arenas with additional edges describing the possible disturbances. An ARENA WITH (UNMODELED) DISTURBANCES $\mathcal{A} = (V, V_0, V_1, E, D)$ consists of an arena (V, V_0, V_1, E), and a set of DISTURBANCE EDGES $D \subseteq V_0 \times V$. Since we are only interested in the response of Player 0 to the occurrence of disturbances, we

Def. arena with (unmodeled) disturbances

Def. disturbance edges

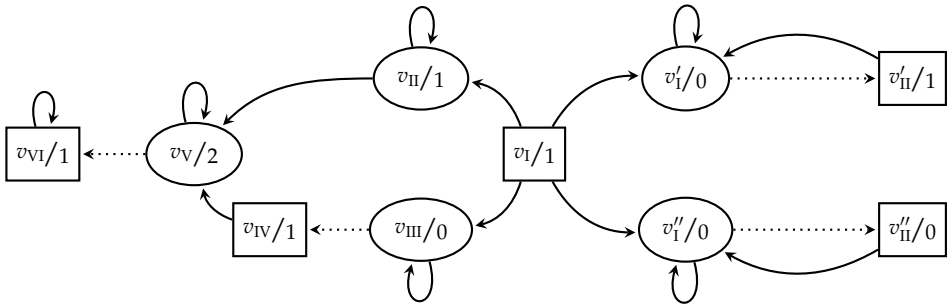


Figure 5.2: An arena with disturbances $\mathcal{A}$ and a coloring of $\mathcal{A}$. Disturbance edges are drawn as dotted arrows.

only allow outgoing disturbance edges from vertices of Player 0. This is, however, not an actual restriction, since outgoing disturbance edges from the vertices of Player 1 can easily be modeled by adding vertices of Player 0 that have only a single outgoing non-disturbance edge. In figures, we draw disturbance edges as dotted arrows.

Example 5.1. Consider the arena shown in Figure 5.2 and ignore the indicated coloring for the time being. This arena contains four disturbance edges, namely $(v_{\text{V}}, v_{\text{VI}})$, $(v_{\text{III}}, v_{\text{IV}})$, $(v'_{\text{I}}, v'_{\text{II}})$, and $(v''_{\text{I}}, v''_{\text{II}})$. Furthermore, it satisfies the general requirement towards arenas that even without the disturbance edges there exists no vertex without outgoing edges. △

Since plays in arenas with disturbances may now also traverse disturbance edges, we extend the notion of a play to denote whether a disturbance edge or a classical one has been traversed. A PLAY in $\mathcal{A}$ is an infinite sequence $\rho = (v_0, b_0)(v_1, b_1)(v_2, b_2)\cdots \in (V \times \{0,1\})^\omega$ such that $b_0 = 0$ and such that for all $j > 0$ we have that

Def. play

- $b_j = 0$ implies $(v_{j-1}, v_j) \in E$, and
- $b_j = 1$ implies $(v_{j-1}, v_j) \in D$.

Thus, the additional bits b_j for $j > 0$ denote whether a standard or a disturbance edge has been taken to move from v_{j-1} to v_j.

Example 5.2. Consider again the arena $\mathcal{A}$ discussed in Example 5.1. The infinite sequences over $V \times \{0,1\}$ $\rho_1 = (v_{\text{I}}, 0)(v_{\text{III}}, 0)(v_{\text{IV}}, 1)(v_{\text{V}}, 0)(v_{\text{V}}, 0)^\omega$ as well as $\rho_2 = (v_{\text{I}}, 0)((v'_{\text{I}}, 0)(v'_{\text{II}}, 1))^\omega$ are plays in $\mathcal{A}$. The infinite sequence of pairs of vertices and bits $(v_{\text{I}}, 0)(v'_{\text{I}}, 0)(v'_{\text{II}}, 0)((v'_{\text{I}}, 0)(v'_{\text{II}}, 1))^\omega$, however, is not a play, since the third bit wrongly indicates that no disturbance edge was taken for the move from v'_{I} to v'_{II}, which contradicts the structure of $\mathcal{A}$. △

We count the number of disturbances in a play $\rho = (v_0, b_0)(v_1, b_1)(v_2, b_2)\cdots$ by defining $\#_D(\rho) = \left|\{j \in \omega \mid b_j = 1\}\right|$, which is either some $k \in \omega$ if there are finitely many disturbances, namely k, or it is equal to ω if there are infinitely many disturbances. We say that a play ρ is DISTURBANCE-FREE, if $\#_D(\rho) = 0$.

Def. $\#_D(\rho)$

Def. disturbance-free

Example 5.3. Let ρ_1 and ρ_2 be the two plays defined in Example 5.2. We have $\#_D(\rho_1) = 1$ and $\#_D(\rho_2) = \omega$. △

A GAME WITH (INTERMITTENT UNMODELED) DISTURBANCES $\mathcal{G} = (\mathcal{A}, \text{Win})$ consists of an arena with unmodeled disturbances $\mathcal{A} = (V, V_0, V_1, E, D)$ and a winning condition $\text{Win} \subseteq V^\omega$. A play $\rho = (v_0, b_0)(v_1, b_1)(v_2, b_2) \cdots$ in $\mathcal{A}$ is WINNING for Player 0, if $v_0 v_1 v_2 \cdots \in \text{Win}$, otherwise it is winning for Player 1. Thus, winning is oblivious to occurrences of disturbances.

Def. game with (intermittent unmodeled) disturbances

Def. winning play

Example 5.4. Consider the arena with disturbances $\mathcal{A}$ and the coloring Ω shown in Figure 5.2. The game $\mathcal{G} = (\mathcal{A}, \text{PARITY}(\Omega))$ is a parity game with disturbances. Moreover, the play ρ_1 from Example 5.2 is winning for Player 0 in $\mathcal{G}$. Dually, the play ρ_2 from Example 5.2 is winning for Player 1 in $\mathcal{G}$. △

A STRATEGY σ for Player $i \in \{0, 1\}$ in an arena with disturbances (V, V_0, V_1, E, D) is a strategy for that player in the arena without disturbances (V, V_0, V_1, E). Thus, in arenas with disturbances, strategies for neither player may prescribe moving along a disturbance edge, i.e., neither player controls the occurrence of disturbances. A play $(v_0, b_0)(v_1, b_1)(v_2, b_2) \cdots$ is CONSISTENT with σ, if $v_{j+1} = \sigma(v_0 \cdots v_j)$ for every $j \in \omega$ that satisfies both $v_j \in V_i$ and $b_{j+1} = 0$. Intuitively, a play is consistent with a strategy σ if the next vertex is the one prescribed by σ unless a disturbance occurs.

Def. strategy

Def. consistent

We lift the notions of winning strategies to the setting of games with disturbances by saying that σ is WINNING for Player i from v in a game with disturbances $\mathcal{G} = (\mathcal{A}, \text{Win})$ if each disturbance-free play ρ starting in v and consistent with σ is winning for Player i. In particular, a winning strategy is only winning if no disturbances occur: We make no assumptions on the behavior of the strategy as soon as a single disturbance occurs. Furthermore, we say that Player i wins $\mathcal{G}$ from v if she has a winning strategy from v in $\mathcal{A}$ and we define $W_i(\mathcal{G})$ as the set of vertices from which Player i wins $\mathcal{G}$.

Def. winning strategy

Example 5.5. Consider again the parity game with disturbances $\mathcal{G}$ discussed in Example 5.4. Player 0 only has two positional strategies, σ_1 and σ_2, which differ in the move prescribed from v_{II}. Let $\sigma_1(v_{\text{II}}) = v_{\text{V}}$ and let $\sigma_2(v_{\text{II}}) = v_{\text{II}}$, where both strategies are defined by prescribing to move to the unique successor of the current vertex for all other vertices of Player 0.

No strategy for either player may prescribe moving from, e.g., v_{V} to v_{VI}, as doing so would prescribe taking a disturbance edge, which we have excluded above. Both plays ρ_1 and ρ_2 from Example 5.2 are consistent with both σ_1 and σ_2. △

Due to this definition, strategies do not have access to the additional bits occurring in plays in arenas with disturbances that denote whether or not an error occurred. This does not, however, restrict the "power" of strategies, as these bits can be reconstructed: Let $(v_0, b_0)(v_1, b_1)(v_2, b_2) \cdots$ be a play and let $j > 0$ such that $b_j = 1$, i.e., let j be a vertex that was reached via traversing a disturbance edge. We say that this disturbance is CONSEQUENTIAL (w.r.t. σ), if $v_j \neq \sigma(v_0 \cdots v_{j-1})$, i.e., if the disturbance edge (v_{j-1}, v_j) traversed by the play indeed leads to a different vertex than the one prescribed by σ. Such consequential disturbances can be detected by comparing the actual vertex v_j to σ's output $\sigma(v_0 \cdots v_{j-1})$. On the other hand, inconsequential disturbances can just

Def. consequential disturbance

be ignored by σ. In particular, the number of consequential disturbances is always bounded from above by the number of disturbances.

5.1.2 Resilience of Strategies against Disturbances

Def. α-resilience

Let $\mathcal{G}$ be a game with disturbances with vertex set V and let $\alpha \in \omega + 2$, i.e., α is either a natural number, it is ω, or it is $\omega + 1$. A strategy σ for Player 0 in $\mathcal{G}$ is α-RESILIENT from $v \in V$ if every play ρ that starts in v, that is consistent with σ, and that satisfies $\#_D(\rho) < \alpha$, is winning for Player 0. Thus, a k-resilient strategy with $k \in \omega$ is winning even if $k - 1$ disturbances (or less) occur, an ω-resilient strategy is winning even if an arbitrary, but finite number of disturbances occurs, and an $(\omega + 1)$-resilient strategy is winning even if infinitely many disturbances occur.

Since we demand the number of disturbances to be strictly less than α, the notion of resilience degenerates for $\alpha = 0$.

Remark 5.6. *Every strategy is* 0*-resilient.*

Furthermore, since setting α to one implies that we only consider plays without disturbances, we recover the classical definition of winning strategies for games without disturbances for this value of α.

Remark 5.7. *A strategy for Player* 0 *is* 1*-resilient from v if and only if it is winning for Player* 0 *from v.*

Finally, from the definition of resilience it directly follows that the property of being α-resilient is downwards-closed.

Remark 5.8. *Let $\mathcal{G}$ be a game with disturbances with vertex set V, let $\alpha, \alpha' \in \omega + 2$ with $\alpha \geq \alpha'$ and let σ be a strategy for Player* 0*. If σ is α-resilient from v, then σ is α'-resilient from v.*

Example 5.9. Consider again the strategies σ_1 and σ_2 defined in Example 5.5.

The strategy σ_1 is 2-resilient from v_{III}, as every play ρ that starts in v_{III}, is consistent with σ_1, and satisfies $\#_D(\rho) \leq 1$ either remains in v_{III} or in v_{V} ad infinitum. Both these vertices have even color, i.e., the play is winning for Player 0. The play $(v_{\text{III}}, 0)(v_{\text{IV}}, 1)(v_{\text{V}}, 0)(v_{\text{VI}}, 1)(v_{\text{VI}}, 0)^\omega$, however, begins in v_{III}, is consistent with σ_1, has two disturbances, but is not winning for Player 0. Thus, this play witnesses that σ_1 has resilience at most 2 from v_{III}.

Both the strategy σ_1 and σ_2 have resilience ω from v'_{I}: Any play that starts in v'_{I} and that has only finitely many disturbances eventually remains in vertex v'_{I} ad infinitum and thus is winning for Player 0. If, however, such a play has infinitely many disturbances, it infinitely often visits vertex v'_{II} and thus is losing for Player 0.

Finally, every strategy is $\omega + 1$-resilient from v''_{I}, as even a play with infinitely many disturbances only visits vertices of color zero, hence it is winning for Player 0. △

Def. resilience of a vertex

We define the RESILIENCE OF A VERTEX v of $\mathcal{G}$ as

$$r_{\mathcal{G}}(v) = \sup \{\alpha \in \omega + 2 \mid \text{Player 0 has an } \alpha\text{-resilient strategy for } \mathcal{G} \text{ from } v\}.$$

Example 5.10. Consider again the parity game with disturbances discussed in Example 5.4. The vertices v_{I}'', and v_{II}'' have resilience $\omega + 1$, while the vertices v_{I}' and v_{II}' have resilience ω. Furthermore, the vertex v_{III} has resilience two, while the vertices v_{I} v_{II}, v_{IV}, and v_{V} have resilience one. Finally, the vertex v_{VI} has resilience zero. △

Following our intuition of treating disturbances as rare events instead of antagonistic ones, this definition is not antagonistic, i.e., it is not defined via strategies of Player 1. Nevertheless, due to Remark 5.7, resilient strategies generalize winning strategies. Analogously, the notion of resilience of vertices generalizes the notion of winning regions.

Remark 5.11. *For each vertex v of G we have $r_{\mathcal{G}}(v) = 0$ if and only if $v \in W_1(\mathcal{G})$.*

A strategy σ for Player 0 is OPTIMALLY RESILIENT, if it is $r_{\mathcal{G}}(v)$-resilient from every vertex v. Every such strategy is a uniform winning strategy for Player 0, i.e., a strategy that is winning from every vertex in her winning region. Hence, positional optimally resilient strategies can only exist in games which have uniform positional winning strategies for Player 0. Our goal in this chapter is to, for a given game $\mathcal{G}$, determine the mapping $r_{\mathcal{G}}$ and to compute an optimally resilient strategy.

Def. optimally resilient

This concludes the formalization of disturbances in our existing framework of infinite games. In the following section, we show how to compute the resilience of the vertices of a game together with strategies witnessing this resilience.

5.2 Computing Resilience in Games with Disturbances

In this section, we show how to compute the resilience of the games discussed in this thesis in the presence of disturbances. In order to obtain a uniform approach handling all variants of the parity condition discussed previously, we only leverage very general properties of the winning conditions.

In fact, for a given game with disturbances $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$, we only require that

1. the winning condition is prefix-independent, and that
2. for all $V' \subseteq V$, the game $(\mathcal{A}, \mathrm{Win} \cap \textsc{Safety}(V'))$ is determined.

Recall that $\textsc{Safety}(V')$ contains those plays that never visit a vertex from V', i.e., the argument of the safety condition denotes those vertices that are "unsafe". These two conditions are satisfied by the parity condition with weights and thus also by the parity condition with costs, the finitary parity condition, and the parity condition.

Our aim in this section is to compute

1. the value $r_{\mathcal{G}}(v)$ for each vertex v of $\mathcal{G}$ and
2. a strategy σ that witnesses $r_{\mathcal{G}}(v)$ for all vertices v of $\mathcal{G}$.

To this end, recall that by definition we have $r_{\mathcal{G}}(v) \in \omega + 2 = \omega \cup \{\omega, \omega + 1\}$. Hence, to determine $r_{\mathcal{G}}(v)$, we proceed in three steps. First, we determine those vertices v with $r_{\mathcal{G}}(v) \in \omega$ as well as their exact resilience in Section 5.2.1. Subsequently, we determine the vertices with resilience $\omega + 1$ in Section 5.2.2.

We then directly obtain that the vertices of $\mathcal{G}$ that were characterized in neither Section 5.2.1 nor Section 5.2.2 have resilience ω. We conclude in Section 5.2.3 by showing how to effectively compute optimally resilient strategies, i.e., strategies that from every vertex v witness its resilience $r_{\mathcal{G}}(v)$.

5.2.1 Characterizing Finite Resilience

Our goal in this subsection is to characterize vertices with finite resilience in a game $\mathcal{G}$. Recall that, by definition, a vertex v has resilience $k \in \omega$ if and only if Player 0 can win from v even under $k-1$ disturbances, but she cannot win if k or more disturbances occur.

We first illustrate our approach using the game shown in Figure 5.2 in the following example.

Example 5.12. Recall that we stated in Example 5.10 that the vertices v_{I} through v_{VI} have finite resilience, while all other vertices either have resilience ω or $\omega+1$.

First, we observe that the winning region of Player 1 in that game consists only of the vertex v_{VI}. Thus, due to Remark 5.7 and Remark 5.8, the vertex v_{VI} is the only vertex with resilience zero: All other vertices have larger resilience.

Now, consider the vertex v_{V}, which has a disturbance edge leading into the winning region of Player 1, i.e., to v_{VI}. Due to this edge, v_{V} has resilience one, as a single disturbance can cause the resulting play to be losing for Player 0. The unique disturbance-free play starting in v_{VI} is consistent with every strategy for Player 0 and violates the winning condition. Due to prefix-independence, prepending the vertex v_{V} does not change the winner and consistency with every strategy for Player 0. Hence, this play witnesses that v_{V} has resilience at most one, while v_{V} being in Player 0's winning region yields the matching lower bound. However, v_{V} is the only vertex to which this reasoning applies. Now, consider v_{IV}: From here, Player 1 can force a play to visit v_{V} using a standard edge. Thus, v_{IV} has resilience one as well. Again, this is the only vertex to which this reasoning is applicable.

In particular, from v_{II}, Player 0 can avoid reaching the vertices for which we have determined the resilience by using the self loop. However, this comes at a steep price for her: Doing so results in a losing play, as the color of v_{II} is odd. Thus, if she wants to have a chance at winning, she has to take a risk by moving to v_{V}, from which she has a 1-resilient strategy, i.e., one that is only winning if no more disturbances occur. For this reason, v_{II} has resilience one as well. Similar reasoning applies to v_{I}: Player 1 can force the play to proceed to v_{II} and from there Player 0 has to take a risk by moving to v_{V}.

The vertices v_{V}, v_{II}, and v_{I} share the property that Player 1 can either enforce a play violating the winning condition or reach a vertex with already determined finite resilience. These three vertices are the only ones currently satisfying this property. They all have resilience one since Player 1 can enforce to reach a vertex of resilience one, but he cannot enforce reaching a vertex of resilience zero. Now, we can also determine the resilience of v_{III}: The disturbance edge from v_{III} to v_{IV} witnesses it

being two.

Afterwards, these two arguments no longer apply to new vertices: No disturbance edge leads from a $v \in \{v'_{\mathrm{I}}, v'_{\mathrm{II}}, v''_{\mathrm{I}}, v''_{\mathrm{II}}\}$ to some vertex whose resilience is already determined and Player 0 has a winning strategy from each v that additionally avoids vertices whose resilience is already determined. Thus, our reasoning cannot determine their resilience. This is consistent with our goal, as all four vertices have non-finite resilience: v'_{I} and v'_{II} have resilience ω and v''_{I} and v''_{II} have resilience $\omega + 1$, as stated in Example 5.10. Our reasoning here cannot distinguish these two values. We solve this problem later in Section 5.2.2. △

We now formalize the reasoning used in Example 5.12: Starting from the vertices in Player 1's winning region having resilience zero, we use so-called disturbance updates and risk updates of partial rankings $r\colon V \dashrightarrow \omega$ that assign a rank to a subset of the vertices of $\mathcal{G}$ to determine all vertices of finite resilience. A disturbance update computes the resilience of vertices having a disturbance edge to a vertex whose resilience is already known (such as vertices v_{V} and v_{III} in the example of Figure 5.2). A risk update, on the other hand, determines the resilience of vertices from which either Player 1 can force a visit to a vertex with known resilience (such as vertices v_{I} and v_{III}) or Player 0 needs to move to such a vertex in order to avoid losing (such as vertex v_{II}). To simplify our proofs, we describe both as monotone operators updating partial rankings mapping vertices to ω, which might update already defined values. Since we are only interested in vertices with finite resilience in this section, this domain suffices for the ranking. We show that applying these updates in alternation eventually yields a stable ranking that indeed characterizes the vertices of finite resilience.

As argued above, we aim to find a method for the computation of resilience that allows us to handle all winning conditions considered in this work. Thus, in order to obtain the most general approach possible, fix a game $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ with disturbances, where $\mathcal{A} = (V, V_0, V_1, E, D)$, and where Win is prefix-independent.

A RANKING for $\mathcal{G}$ is a partial mapping $r\colon V \dashrightarrow \omega$. The domain of r is denoted by $\mathrm{dom}(r)$, its image by $\mathrm{im}(r)$. Let r and r' be two rankings. We say that r' REFINES r if $\mathrm{dom}(r') \supseteq \mathrm{dom}(r)$ and if $r'(v) \leq r(v)$ for all $v \in \mathrm{dom}(r)$. A ranking r is SOUND, if for all vertices v, we have $r(v) = 0$ if and only if $v \in W_1(\mathcal{G})$, i.e., if r is consistent with Remark 5.7.

Def. ranking
Def. $\mathrm{dom}(r)$
Def. $\mathrm{im}(r)$
Def. refinement
Def. soundness

Let r be a ranking for $\mathcal{G}$. We define the DISTURBANCE UPDATE of r as the ranking r' defined via

Def. disturbance update

$$r'(v) = \min\left(\{r(v)\} \cup \{r(v') + 1 \mid v' \in \mathrm{dom}(r) \text{ and } (v, v') \in D\}\right),$$

where $\{r(v)\} = \emptyset$ if $v \notin \mathrm{dom}(r)$, and where $\min \emptyset$ is undefined (causing $r'(v)$ to be undefined).

Lemma 5.13. *The disturbance update r' of a sound ranking r is sound and refines r.*

Proof. As the minimization defining $r'(v)$ ranges over a superset of $\{r(v)\}$, we have $r'(v) \leq r(v)$ for every $v \in \mathrm{dom}(r)$. This immediately implies refinement.

Furthermore, since r is sound, we have $r(v) = 0$ for all $v \in W_1(\mathcal{G})$. The inequality $r'(v) \leq r(v)$ then yields $r'(v) = 0$ for all $v \in W_1(\mathcal{G})$. It remains to show $r'(v) > 0$ for all $v \notin W_1(\mathcal{G})$.

Let $v \in W_0(\mathcal{G})$. We directly obtain $r'(v) > 0$, since both $r(v)$ and $r(v') + 1$ are greater than zero. This suffices since we have $v \in W_0(\mathcal{G})$ if and only if $v \notin W_1(\mathcal{G})$ due to determinacy of Win. Altogether, r' is sound as well. □

The disturbance update formalizes the step in our intuitive determination of the resilience of the vertices v_{V} and v_{III} in Example 5.12. It remains to formalize the update that assigns resilience values to v_{II} and v_{IV}.

Again, let r be a ranking for $\mathcal{G}$. Furthermore, for every $k \in \mathrm{im}(r)$, define $U_k = \{v \in \mathrm{dom}(r) \mid r(v) \leq k\}$, $\mathcal{G}_k = (\mathcal{A}, \mathrm{Win} \cap \textsc{Safety}(U_k))$, and $A_k = W_1(\mathcal{G}_k)$. Intuitively, in $\mathcal{G}_k$, Player 1 wins by either reaching a vertex v with $r(v) \leq k$ or by violating the original winning condition. Now, define $r'(v) = \min\{k \mid v \in A_k\}$, where $\min \emptyset$ is again undefined. We call r' the RISK UPDATE of r.

Def. risk update

Lemma 5.14. *The risk update r' of a sound ranking r is sound and refines r.*

Proof. We will first show $r'(v) \leq r(v)$ for every $v \in \mathrm{dom}(r)$, which implies both refinement and $r'(v) = 0$ for every $v \in W_1(\mathcal{G})$, as argued in the proof of Lemma 5.13.

To this end, let $v \in \mathrm{dom}(r)$ and let $r(v) = k$. Trivially, $v \in U_k$. Thus, Player 1 wins the game $\mathcal{G}_k$ from v by violating the safety condition right away. Hence, $v \in A_k$ and thus $r'(v) \leq k = r(v)$.

To complete the proof of soundness of r', we just have to show $r'(v) > 0$ for every $v \in W_0(\mathcal{G})$, again due to determinacy of Win. Towards a contradiction, assume $r'(v) = 0$, i.e., $v \in A_0$. Thus, Player 1 has a strategy τ from v that ensures that either the winning condition is violated or that a vertex v' with $r(v') = 0$ is reached, i.e., $v' \in W_1(\mathcal{G})$ by soundness of r. Hence, Player 1 has a winning strategy $\tau_{v'}$ for $\mathcal{G}$ from v'. This implies that he also has a winning strategy from v: Play according to τ until a vertex v' with $r(v') = 0$ is reached, if such a vertex is reached at all. From there, mimic $\tau_{v'}$ when starting from v'. Every resulting disturbance-free play has a suffix that violates Win. Thus, by prefix-independence of Win, the whole play violates Win as well, i.e., it is winning for Player 1. Thus, $v \in W_1(\mathcal{G})$, which yields the desired contradiction, as winning regions are always disjoint. □

Having formally defined the disturbance update and the risk update, we now describe how to interleave them and show that this interleaving indeed results in a characterization of vertices of finite resilience. Let r_0 be the unique sound ranking with domain $W_1(\mathcal{G})$. Starting with r_0, we inductively define a sequence of rankings $(r_j)_{j \in \omega}$ such that

Def. r_j

- for all odd $j > 0$, r_j is the disturbance update of r_{j-1}, and
- for all even $j > 0$, r_j is the risk update of r_{j-1},

i.e., we construct $(r_j)_{j \in \omega}$ by alternating disturbance updates and risk updates.

Due to refinement, the r_j eventually stabilize, i.e., there is some j_0 such that $r_j = r_{j_0}$ for all $j \geq j_0$. Define $r^* = r_{j_0}$. Due to r_0 being sound and by Lemma 5.13 and

Def. r^*

Lemma 5.14, each r_j, and, in particular, r^* is sound. If $v \in \mathrm{dom}(r^*)$, let j_v be the minimal j with $v \in \mathrm{dom}(r_j)$, otherwise, j_v is undefined. We show that, for all $v \in \mathrm{dom}(r^*)$, once a rank is assigned to v during the iterative application of the two updates, it is never updated.

Lemma 5.15. *If $v \in \mathrm{dom}(r^*)$, then $r_{j_v}(v) = r_j(v)$ for all $j \geq j_v$.*

Proof. We show the following stronger result for every $v \in \mathrm{dom}(r^*)$:

- If j_v is odd, then $r_j(v) = \frac{j_v+1}{2}$ for every $j \geq j_v$.
- If j_v is even, then $r_j(v) = \frac{j_v}{2}$ for every $j \geq j_v$.

Intuitively, this property not only states that once a vertex is assigned a rank, it is not updated anymore in subsequent rounds, but also that in each round, only a single rank is assigned, namely rank $\frac{j+1}{2}$ in odd rounds j, and rank $\frac{j}{2}$ in even rounds j.

We first observe that the disturbance update increases the maximal rank by at most one and that the risk update does not increase the maximal rank at all. Furthermore, due to refinement, the rank of v is set and then only decreases. Hence, we obtain $r_j(v) \leq \frac{j_v+1}{2}$ and $r_j(v) \leq \frac{j_v}{2}$ for odd and even j_v, respectively. It remains to show a matching lower bound, which we do in the remainder of this proof.

We say that a vertex v is updated to $k \in \omega$ in r_j if $r_j(v) = k$ and either $v \notin \mathrm{dom}(r_{j-1})$ or both $v \in \mathrm{dom}(r_{j-1})$ and $r_{j-1}(v) \neq k$ (here, r_{-1} is the unique ranking with empty domain). Now, we show the following by induction over j, which implies the matching lower bound.

- If j is odd, then no v is updated in r_j to some $k < \frac{j+1}{2}$.
- If j is even, then no v is updated in r_j to some $k < \frac{j}{2}$.

For $j = 0$, we have $\frac{j}{2} = 0$, and clearly, no vertex is assigned a negative rank by r_0. For $j = 1$ and $j' = 2$, we obtain $\frac{j+1}{2} = \frac{j'}{2} = 1$. As r_0, r_1, and r_2 are sound, neither r_1 nor r_2 update any v to zero, which concludes the induction start.

Now let $j > 2$ and first consider the case where j is odd. Towards a contradiction, assume that $v \in V$ is updated in r_j to some value strictly less than $\frac{j+1}{2}$. Since j is odd, r_j is the disturbance update of r_{j-1}. Further, as v is updated in r_j, there exists some disturbance edge $(v, v') \in D$ such that $r_j(v) = r_{j-1}(v') + 1$. Thus, $r_{j-1}(v') < r_j(v) < \frac{j+1}{2}$, i.e., $r_{j-1}(v') \leq \frac{j+1}{2} - 2 = \frac{j-3}{2}$. First, we show $r_{j-3}(v') = r_{j-2}(v') = r_{j-1}(v')$, i.e., that the rank of v' is stable during the last two updates.

First assume towards a contradiction $r_{j-2}(v') \neq r_{j-1}(v')$. Then, v' is updated in r_{j-1} to some rank of at most $\frac{j-3}{2}$, which is in turn smaller than $\frac{j-1}{2}$, violating the induction hypothesis for $j-1$. Hence, $r_{j-2}(v') = r_{j-1}(v')$. The same reasoning yields a contradiction to the assumption $r_{j-3}(v') \neq r_{j-2}(v')$. Thus, we indeed obtain $r_{j-3}(v') = r_{j-2}(v') = r_{j-1}(v')$.

Since r_{j-2} is the disturbance update of r_{j-3}, we obtain $r_{j-2}(v) \leq r_{j-3}(v') + 1 = r_{j-1}(v') + 1 = r_j(v)$. Due to refinement, we obtain $r_{j-2}(v) \geq r_j(v)$, i.e., altogether $r_{j-2}(v) = r_{j-1}(v) = r_j(v)$. The latter equality contradicts our initial assumption, namely v being updated in r_j to $r_j(v)$.

Now consider the case where j is even. Again, assume towards a contradiction that $v \in V$ is updated in r_j to some value less than $\frac{j}{2}$. Since j is even, r_j is the risk update of r_{j-1}. Further, as v is updated in r_j, Player 1 wins the game $(\mathcal{A}, \mathrm{Win} \cap \textsc{Safety}(U))$ from v, where $U = \{v' \in \mathrm{dom}(r_{j-1}) \mid r_{j-1}(v') \leq r_j(v)\}$. Hence, he has a strategy τ such that every play starting in v and consistent with τ either violates Win or eventually visits some vertex v' with $r_{j-1}(v') \leq r_j(v)$. We claim $r_{j-2}(v') = r_{j-1}(v')$ for all $v' \in U$.

Towards a contradiction, assume $r_{j-2}(v') \neq r_{j-1}(v')$ for some $v' \in U$. Note that we have $r_{j-1}(v') \leq r_j(v) < \frac{j}{2}$ due to the definition of U. Thus, v' is updated in r_{j-1} to some value strictly less than $\frac{j}{2}$, which contradicts the induction hypothesis for $j-1$. Hence, we indeed obtain $r_{j-2}(v') = r_{j-1}(v')$ for all $v' \in U$.

Thus, there are two types of vertices v' in U:

1. Those for which $r_{j-3}(v')$ is defined, which implies $r_{j-3}(v') = r_{j-1}(v')$ due to the induction hypothesis and refinement, and
2. those where $r_{j-3}(v')$ is undefined, which implies $r_{j-2}(v') = r_{j-1}(v')$ due to the claim above.

We claim that Player 1 wins $(\mathcal{A}, \mathrm{Win} \cap \textsc{Safety}(\{v'' \in \mathrm{dom}(r_{j-3}) \mid r_{j-3}(v'') \leq r_j(v)\}))$ from v, which implies $r_{j-2}(v) = r_j(v)$. This then contradicts v being updated in r_j, our initial assumption.

To this end, we construct a strategy τ' for Player 1 from v that either violates Win or reaches a vertex v'' with $r_{j-3}(v'') \leq r_j(v)$ as follows: From v, τ' mimics τ until a vertex v' in U is reached, if such a vertex is reached at all. If v' is of the first type, then we have $r_{j-3}(v') = r_{j-1}(v') \leq r_j(v)$. If v' is of the second type, then v' is updated in r_{j-2} to some rank $r_{j-2}(v') = r_{j-1}(v') \leq r_j(v)$. As r_{j-2} is the risk update of r_{j-3}, Player 1 has a strategy $\tau_{v'}$ from v' such that all plays starting in v' and consistent with $\tau_{v'}$ either violate Win or encounter a vertex v'' with $r_{j-3}(v'') \leq r_{j-2}(v') \leq r_j(v)$. Thus, restarting in v', τ' mimics $\tau_{v'}$ from v' until such a vertex is reached (if it is reached at all). Thus, every play that starts in v and is consistent with τ' either violates Win (due to prefix-independence of Win) or reaches a vertex v'' with $r_{j-3}(v'') \leq r_j(v)$, which proves our claim. □

From the proof of Lemma 5.15, we obtain that the maximal rank assigned by r^* is bounded from above by the number of vertices of $\mathcal{G}$. This in turn implies that the r_j stabilize quickly, as $r_j = r_{j+1} = r_{j+2}$ implies $r_j = r^*$.

Corollary 5.16. *We have* $\mathrm{im}(r^*) = \{0, 1, \ldots, n\}$ *for some* $n < |V|$ *and* $r^* = r_{2|V|}$.

The main result of this section shows that each vertex ranked by r^* indeed has the resilience indicated by r^*.

Lemma 5.17. *Let* r^* *be defined for* $\mathcal{G}$ *as above, and let* $v \in V$. *If* $v \in \mathrm{dom}(r^*)$, *then* $r_{\mathcal{G}}(v) = r^*(v)$.

Proof. Let $v \in \mathrm{dom}(r^*)$. We prove $r_{\mathcal{G}}(v) \leq r^*(v)$ and $r_{\mathcal{G}}(v) \geq r^*(v)$ separately.

First, we show $r_{\mathcal{G}}(v) \leq r^*(v)$: Recall that the resilience-property is downwards-closed due to Remark 5.8, i.e., that an α-resilient strategy from v is also α'-resilient from v for every $\alpha' \leq \alpha$. Thus, to prove

$$r_{\mathcal{G}}(v) = \sup\{\alpha \in \omega + 2 \mid \text{Player 0 has an } \alpha\text{-resilient strategy for } \mathcal{G} \text{ from } v\} \leq r^*(v)$$

we just have to show that Player 0 has no $(r^*(v)+1)$-resilient strategy from v. By definition of resilience, for every strategy σ for Player 0, we have to show that there is a play ρ in $\mathcal{G}$ starting in v and consistent with σ that has at most $r^*(v)$ disturbances and is winning for Player 1. So, fix an arbitrary strategy σ for Player 0.

We define a play with the desired properties by constructing longer and longer finite prefixes before finally appending an infinite suffix. During the construction, we ensure that each such prefix ends in a vertex from $\mathrm{dom}(r^*)$ in order to be able to proceed with our construction.

The first prefix just contains $(v, 0)$, i.e., the prefix does indeed end in $\mathrm{dom}(r^*)$. Now, assume we have produced a prefix $\pi(v', b')$ ending in some vertex $v' \in \mathrm{dom}(r^*)$, which implies that $j_{v'}$ is defined. We consider three cases:

- If $j_{v'} = 0$, then $v' \in W_1(\mathcal{G})$ by soundness of r^*, i.e., Player 1 has a winning strategy τ from v'. Thus, we extend $\pi(v', b')$ by the unique disturbance-free play that starts in v' and is consistent with σ and τ, without its first vertex. In that case, the construction of the infinite play is complete.
- If $j_{v'} > 0$ is odd, then v' received its rank $r^*(v')$ during a disturbance update. Hence, there is some v'' such that $(v', v'') \in D$ with $r^*(v') - 1 = r^*(v'')$. In this case, we extend $\pi(v', b')$ by such a vertex v'' to obtain the new prefix $\pi(v', b')(v'', 1)$, which satisfies the invariant, as v'' is in $\mathrm{dom}(r^*)$. Further, we have $j_{v''} < j_{v'}$ as the rank of v'' had to be defined in order to be considered during the disturbance update assigning a rank to v'.
- If $j_{v'} > 0$ is even, then v' received its rank $r^*(v')$ during a risk update. We claim that Player 1 has a strategy $\tau_{v'}$ that guarantees one of the following outcomes from v': Either the resulting play violates Win or it encounters a vertex $v'' \neq v'$ that satisfies $r^*(v'') \leq r^*(v')$ and $j_{v''} < j_{v'}$.
  First assume that this claim indeed does hold true and consider the unique disturbance-free play ρ' that starts in v' and is consistent with σ and the strategy $\tau_{v'}$ as above. If ρ' violates Win, then we extend $\pi(v', b')$ by ρ' without its first vertex. In that case, the construction of the infinite play is complete.
  If ρ', however, does not violate Win, then we extend $\pi(v', b')$ by the prefix of ρ' without its first vertex and up to and including the first occurrence of a vertex v'' in ρ' satisfying the properties described above. This again satisfies the invariant.
  It remains to argue our claim: v' was assigned its rank $r^*(v') = r_{j_{v'}}(v')$ because it is in Player 1's winning region in the game with winning condition $\mathrm{Win} \cap \textsc{Safety}(U)$, for

  $$U = \left\{v'' \in \mathrm{dom}(r_{j_{v'}-1}) \mid r_{j_{v'}-1}(v'') \leq r_{j_{v'}}(v')\right\} \quad .$$

  Hence, from v', Player 1 has a strategy to either violate the winning condition or to reach U. Thus, $r_{j_{v'}-1}(v'') = r^*(v'')$ for every $v'' \in \mathrm{dom}(r_{j_{v'}-1})$ due to

Lemma 5.15 yields $r^*(v'') \leq r^*(v')$. Finally, we have $j_{v''} < j_{v'}$, as the rank of v' was assigned due to the vertices in U already having ranks.

Note that only in two cases, we extend the prefix to an infinite play. In the other two cases, we just extend the prefix to a longer finite one. Thus, we first show that this construction always results in an infinite play. To this end, let $\pi_0(v_0, b_0)$ and $\pi_1(v_1, b_1)$ be two prefixes constructed above such that $\pi_0(v_0, b_0)$ is a prefix of $\pi_1(v_1, b_1)$. A straightforward induction proves $j_{v_0} > j_{v_1}$. Hence, as the value can only decrease finitely often, at some point an infinite suffix is added. Thus, we at some point encounter the first case in the construction of the play and hence, we indeed construct an infinite play.

Finally, we have to show that the resulting play ρ has the desired properties, i.e., that it starts in v, is consistent with σ, has at most $r^*(v)$ disturbances and is winning for Player 1 The first two properties are clear by construction of ρ. Furthermore, by construction, it has a disturbance-free suffix that violates Win. Thus, due to prefix-independence of Win, the whole play also violates Win. It remains to show that it has at most $r^*(v)$ disturbances. To this end, let $\pi_0(v_0, b_0)$ and $\pi_1(v_1, b_1)$ be two prefixes of ρ such that $\pi_1(v_1, b_1)$ is obtained by extending $\pi_0(v_0, b_0)$ once. If the extension consists of taking the disturbance edge $(v_0, v_1) \in D$, then we have $r^*(v_1) = r^*(v_0) + 1$, again, due to construction of ρ. The only other possibility is the extension consisting of a finite disturbance-free play prefix that is consistent with the strategy τ_{v_0}. Then, by construction, we obtain $r^*(v_1) \leq r^*(v_0)$.

Thus, there are at most $r^*(v)$ many disturbances in ρ. Moreover, the current rank decreases with every disturbance edge and does not increase with the other type of extension. Furthermore, the current rank is always nonnegative. Hence, ρ witnesses the resilience of σ from v being at most $r^*(v)$. Since we picked σ arbitrarily, we obtain $r_{\mathcal{G}}(v) \leq r^*(v)$.

It remains to show $r_{\mathcal{G}}(v) \geq r^*(v)$. To this end, we construct a uniform strategy σ_f for Player 0 that is $r^*(v)$-resilient from every $v \in \mathrm{dom}(r^*)$. In other words, from every $v \in \mathrm{dom}(r^*)$, the strategy σ_f has to be winning even under $r^*(v) - 1$ disturbances. As every strategy is 0-resilient, we only have to consider those v with $r^*(v) > 0$.

The proof is based on the fact that r^* is both stable under the disturbance and under the risk update, i.e., the disturbance update and the risk update of r^* yield again r^*. Due to this, we obtain the following properties:

- Firstly, let $(v, v') \in D$ be a disturbance edge such that $r^*(v) > 0$. Then, we have $r^*(v') \geq r^*(v) - 1$.
- Secondly, for every vertex v with $v \in \mathrm{dom}(r^*)$ that satisfies $r^*(v) > 0$, Player 0 has a strategy σ_v that is winning for her from v in the game $\mathcal{G}_v = (\mathcal{A}, \mathrm{Win} \cap \textsc{Safety}(\{v' \in \mathrm{dom}(r^*) \mid r^*(v') < r^*(v)\}))$.

This latter property follows via determinacy of the game $\mathcal{G}_v$, as the risk update is formulated in terms of the winning region of Player 1. In particular, note that the set of unsafe vertices in the latter property is defined using a strict inequality.

Now, we define σ_f as follows: It always mimics a strategy σ_{v^*} for some $v^* \in \mathrm{dom}(r^*)$, which is initialized by the starting vertex. The strategy σ_{v^*} is mimicked until a consequential (w.r.t. σ_{v^*}) disturbance edge is taken, say by reaching the vertex v'. In that

case, the strategy σ_f discards the history of the play constructed so far, updates v^* to v', and begins mimicking $\sigma_{v'}$. This is repeated ad infinitum.

Now, consider a play ρ that starts in $\mathrm{dom}(r^*)$, is consistent with σ_f, and has less than $r^*(v)$ disturbances. The part up to the first consequential disturbance edge (if it exists at all) is consistent with σ_v. Now, let (v_0, v_0') be that disturbance edge. Then, we have $r^*(v_0) \geq r^*(v)$, since σ_v never visits vertices with a rank smaller than $r^*(v)$, due to it being a winning strategy for the safety condition. Thus, we conclude $r^*(v_0') \geq r^*(v_0) - 1 \geq r^*(v) - 1$. Similarly, the part between the first and the second consequential disturbance edge (if it exists at all) is consistent with $\sigma_{v_0'}$. Again, if (v_1, v_1') is the traversed disturbance edge, then we have $r^*(v_1') \geq r^*(v_1) - 1 \geq r^*(v) - 2$.

Continuing this reasoning shows that ρ eventually encounters a vertex v' such that $r^*(v') > 0$, and such that the suffix starting in this vertex is disturbance-free and consistent with $\sigma_{v'}$. The former property holds true, since ρ contains strictly less than $r^*(v)$ disturbances and as the rank is decreased by at most one for every disturbance edge. The latter two properties hold true by assumption on ρ.

Hence, the suffix satisfies Win, i.e., by prefix-independence, the complete play satisfies Win as well. As we picked ρ starting in some vertex from $\mathrm{dom}(r^*)$ and consistent with σ_f arbitrarily, σ_f is indeed $r^*(v)$-resilient from every $v \in \mathrm{dom}(r^*)$. □

We furthermore obtain that r^* indeed only assigns a rank to vertices of finite resilience.

Lemma 5.18. *Let r^* be defined for $\mathcal{G}$ as above, and let $v \in V$. If $v \notin \mathrm{dom}(r^*)$, then $r_{\mathcal{G}}(v) \in \{\omega, \omega + 1\}$.*

Proof. Let $X = V \setminus \mathrm{dom}(r^*)$. The disturbance update of r^* being r^* implies that every disturbance edge starting in X leads back to X. Similarly, the risk update of r^* being r^* implies $X = W_0(\mathcal{G}_X)$ for $\mathcal{G}_X = (\mathcal{A}, \mathrm{Win} \cap \textsc{Safety}(V \setminus X))$. Thus, from every $v \in X$, Player 0 has a strategy σ_v such that every disturbance-free play that starts in v and is consistent with σ_v satisfies the winning condition Win and never leaves X. Using these properties, we construct a strategy σ_ω that is ω-resilient from every $v \in X$, which implies $r_{\mathcal{G}}(v) \in \{\omega, \omega + 1\}$.

The definition of the strategy σ_ω here is similar to the one constructed in the proof of Lemma 5.17 yielding the lower bound on the resilience. Again, σ_ω always mimics a strategy σ_{v^*} for some $v^* \in X$, which is initialized by the starting vertex. The strategy σ_{v^*} is mimicked until a consequential (w.r.t. σ_{v^*}) disturbance edge is taken, say by reaching the vertex v'. In that case, the strategy σ_ω discards the history of the play constructed so far, updates v^* to v', and begins mimicking $\sigma_{v'}$. This is repeated ad infinitum.

Due to the properties of the disturbance edges and the strategies σ_v, such a play never leaves X, even if disturbances occur. Furthermore, if only finitely many disturbances occur, then the resulting play has a disturbance-free suffix that starts in some $v^* \in X$ and is consistent with σ_{v^*}. As σ_{v^*} is winning from v^* in $\mathcal{G}_X$, this suffix satisfies Win. Hence, by prefix-independence of Win, the complete play also satisfies Win. Thus, σ_ω is indeed an ω-resilient strategy from every $v \in X$. □

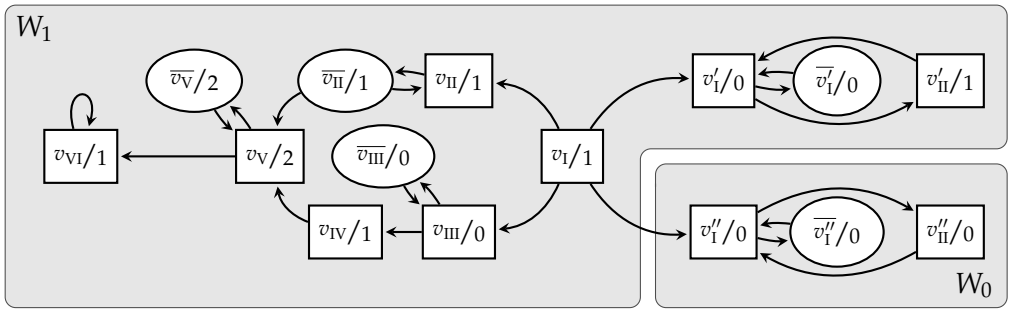


Figure 5.3: The rigged game obtained for the game shown in Figure 5.2.

Having thus shown that r^* indeed ranks all vertices of finite resilience with their resilience, we obtain the following upper bound on the resilience of vertices via Corollary 5.16, Lemma 5.17, and Lemma 5.18.

Corollary 5.19. *We have $r_{\mathcal{G}}(V) \cap \omega = \{0, 1, \ldots, n\}$ for some $n < |V|$.*

We have thus shown how to determine the vertices of finite resilience together with their resilience. In the following section we furthermore show how to determine the vertices with resilience $\omega + 1$.

5.2.2 Characterizing Resilience ω and $\omega + 1$

Our goal in this subsection is to determine the vertices of resilience ω and $\omega + 1$, i.e., those from which Player 0 can win even under an unbounded, but finite number of disturbances, and under an infinite number of disturbances, respectively. We first show how to determine vertices of resilience $\omega + 1$.

Recall that, by definition, the resilience of a vertex is in $\omega + 2$. Moreover recall that we have determined the vertices with resilience less than ω in the previous section. Hence, by determining those vertices with resilience $\omega + 1$, we obtain that those vertices not determined to have finite resilience in the previous section, nor determined to have resilience $\omega + 1$ in this section, have resilience ω. Thus, we have then determined the resilience of all vertices.

Intuitively, recall that a vertex has resilience $\omega + 1$ if and only if Player 0 can win the game from that vertex even if infinitely many disturbances occur. Thus, we give Player 1 control over the disturbance edges, as there cannot be more than infinitely many disturbances during a play.

→Sec. 5.1, Page 152

Example 5.20. Consider again the parity game with disturbances from →Figure 5.2. Intuitively, in order to give Player 1 control over the occurrence of disturbances, for each vertex v of Player 0, we add a vertex $\overline{v}$, which belongs to Player 0, and we give the original vertex v to Player 1. Upon entering vertex v, Player 1 may then choose to traverse an outgoing disturbance edge of v, or he may decide to give control to Player 0 by moving to vertex $\overline{v}$. From $\overline{v}$, Player 0 can then pick a non-disturbance edge of v to continue along.

We show the game without disturbances resulting from giving control over the disturbances to Player 1 in Figure 5.3. We observe that the winning region of Player 0 corresponds to the vertices of resilience $\omega + 1$ in the game of Figure 5.2. Dually, the winning region of Player 1 corresponds to the vertices of resilience at most ω, i.e., the vertices v_{I} through v_{VI} with finite resilience identified in ➔Example 5.12 and the vertices v'_{I} and v'_{II} with resilience ω. ➔Sec. 5.2, Page 156

From v''_{I}, Player 0 wins even if Player 1 controls whether the disturbance edge is traversed from v''_{I}, as both v''_{I} and v''_{II} have color zero. On the other hand, giving Player 1 control over the disturbance edges lets him win from v'_{I}, as he can use the disturbance edge incident to v'_{I} infinitely often to move to v'_{II}, which has color one. △

In the following, we prove this intuition of handing control over the disturbance to Player 1 to be correct for determining the vertices of resilience $\omega + 1$. To this end, we transform the arena of the game so that at a vertex of Player 0, first Player 1 gets to chose whether he wants to take one of the disturbance edges and, if not, gives control to Player 0, who is then able to use a standard edge. The resulting game is a game without disturbances: Since all disturbances are now under the control of Player 1, the corresponding disturbance edges are normal edges in the resulting game.

Again, since we want to obtain a uniform approach to characterizing vertices of resilience $\omega + 1$ that works for all winning conditions discussed in this work, we consider an arbitrary game with disturbances $(\mathcal{A}, \mathrm{Win})$.

Given a game with disturbances $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ with $\mathcal{A} = (V, V_0, V_1, E, D)$, we define the RIGGED GAME $\mathcal{G}_{\mathrm{rig}} = (\mathcal{A}', \mathrm{Win}')$ with $\mathcal{A}' = (V', V'_0, V'_1, E')$ such that $V'_0 = \{\overline{v} \mid v \in V_0\}$, $V'_1 = V$, and $V' = V'_0 \cup V'_1$. The set E' of edges is the union of the following sets: **Def.** rigged game

- D: Player 1 uses a disturbance edge.
- $\{(v, \overline{v}) \mid v \in V_0\}$: Player 1 does not use a disturbance edge and yields control to Player 0.
- $\{(\overline{v}, v') \mid (v, v') \in E \text{ and } v \in V_0\}$: Player 0 has control and picks a standard edge.
- $\{(v, v') \mid (v, v') \in E \text{ and } v \in V_1\}$: Player 1 takes a standard edge.

Further, $\mathrm{Win}' = \{\rho \in (V')^\omega \mid h(\rho) \in \mathrm{Win}\}$ where $h\colon (V')^\omega \to V^\omega$ is the homomorphism induced by $h(v) = v$ and $h(\overline{v}) = \epsilon$ for every $v \in V$. In particular, we construct $\mathcal{G}_{\mathrm{rig}}$ to be a game without disturbances. Hence, plays in $\mathcal{G}_{\mathrm{rig}}$ do not contain additional vertices denoting whether or not a disturbance edge has been taken.

We illustrate the construction of the rigged game from the parity game with disturbances shown in ➔Figure 5.2 in Figure 5.3. The following lemma generalizes and formalizes the observation of Example 5.20 that $W_0(\mathcal{G}_{\mathrm{rig}})$ characterizes the vertices of resilience $\omega + 1$ in $\mathcal{G}$. ➔Sec. 5.1, Page 152

Lemma 5.21. *Let v be a vertex of a game $\mathcal{G}$. We have $v \in W_0(\mathcal{G}_{rig})$ if and only if $r_{\mathcal{G}}(v) = \omega + 1$.*

Proof. We first show the implication from left to right. Let Player 0 win $\mathcal{G}_{\mathrm{rig}}$ from v, say with winning strategy σ'. In order to construct a strategy σ witnessing $r_{\mathcal{G}}(v) = \omega + 1$, we inductively translate play prefixes w in $\mathcal{G}$ into play prefixes $t'(w)$ in $\mathcal{G}_{\mathrm{rig}}$ satisfying the following invariant:

$t'((v_0, b_0) \cdots (v_j, b_j))$ starts in v_0 and ends in v_j

We begin by defining $t'(v_0, b_0) = v_0$. To define $t'((v_0, b_0) \cdots (v_j, b_j) \cdot (v_{j+1}, b_{j+1}))$, we consider several cases:

- If $b_{j+1} = 1$, then the play traverses the disturbance edge (v_j, v_{j+1}), i.e., we have $(v_j, v_{j+1}) \in D$. We mimick this move by defining

$$t'((v_0, b_0) \cdots (v_j, b_j) \cdot (v_{j+1}, b_{j+1})) = t'((v_0, b_0) \cdots (v_j, b_j)) \cdot v_{j+1} \ .$$

- If $b_{j+1} = 0$ and $v_j \in V_0$, then we have $(v_j, v_{j+1}) \in E$, i.e., the play did not traverse a disturbance edge and instead allowed Player 0 to pick the standard edge (v_j, v_{j+1}) to traverse. We mimick this move by defining

$$t'((v_0, b_0) \cdots (v_j, b_j) \cdot (v_{j+1}, b_{j+1})) = t'((v_0, b_0) \cdots (v_j, b_j)) \cdot \overline{v_j} \cdot v_{j+1} \ .$$

- If $b_{j+1} = 0$ and $v_j \in V_1$, then the play traversed the standard edge $(v_j, v_{j+1}) \in E$. We mimick this move by defining

$$t'((v_0, b_0) \cdots (v_j, b_j) \cdot (v_{j+1}, b_{j+1})) = t'((v_0, b_0) \cdots (v_j, b_j)) \cdot v_{j+1} \ .$$

It is easy to see that the invariant is satisfied in any case. Also, we lift t' to infinite plays by taking limits as usual.

Using this translation, we define a strategy σ for Player 0 in $\mathcal{G}$ via

$$\sigma(v_0 \cdots v_j) = \sigma'(t'((v_0, b_0) \cdots (v_j, b_j)) \cdot \overline{v_j}) \ ,$$

where $b_0 = 0$ and where for every $j' > 0$, $b_{j'} = 1$ if and only if $v_{j'} \neq \sigma(v_0 \cdots v_{j'-1})$, i.e., we reconstruct the consequential disturbances as described in ➔Section 5.1.1. A straightforward induction shows that for every play $\rho = (v_0, b_0)(v_1, b_1)(v_2, b_2) \cdots$ in $\mathcal{G}$ that is consistent with σ, the play $t'(\rho)$ is consistent with σ'. Hence, $t'(\rho) \in \mathrm{Win}'$ for every ρ starting in v. Furthermore, we have $h(t'(\rho)) = v_0 v_1 v_2 \cdots \in \mathrm{Win}$, as $t'(\rho) \in \mathrm{Win}'$. Thus, $\rho = (v_0, b_0)(v_1, b_1)(v_2, b_2) \cdots$ is winning for Player 0. As we have no restriction on the number of disturbances in ρ, the strategy σ is $(\omega + 1)$-resilient from v. Hence, $r_{\mathcal{G}}(v) = \omega + 1$, which concludes the proof of the implication from left to right.

➔Page 151

It remains to show the implication from right to left. To this end, let $r_{\mathcal{G}}(v) = \omega + 1$, i.e., Player 0 has an $(\omega + 1)$-resilient strategy σ from v in $\mathcal{G}$. We define a winning strategy σ' for Player 0 from v in $\mathcal{G}_{\mathrm{rig}}$. This time, we inductively define a translation t of play prefixes in $\mathcal{G}_{\mathrm{rig}}$ into play prefixes in $\mathcal{G}$. Since all vertices in $\mathcal{G}$ correspond to vertices of Player 1 in $\mathcal{G}_{\mathrm{rig}}$, it suffices to consider those prefixes that start and end in V_1'. For these, we construct t to satisfy the following invariant:

> If π starts in v_0 and ends in v_j, then $t(\pi)$ starts in v_0 and ends in v_j as well.

Recall that $\mathcal{G}_{\mathrm{rig}}$ is a game without disturbances. Thus, plays in $\mathcal{G}_{\mathrm{rig}}$ do not contain bits indicating whether a disturbance edge has been traversed and we have to reconstruct them from the traversal of the vertices v and $\overline{v}$. We define $t(v_0) = (v_0, 0)$ and consider several cases for the inductive step:

- First, assume we have a prefix of the form $v_0 \cdots v_j \cdot v_{j+1}$ for some $v_j \in V_0 \subseteq V_1'$, i.e., the move of Player 1 simulates traversing the disturbance edge $(v_j, v_{j+1}) \in D$. Then, we define

$$t(v_0 \cdots v_j \cdot v_{j+1}) = t(v_0 \cdots v_j) \cdot (v_{j+1}, 1) \ .$$

- Next, assume we have a prefix of the form $v_0 \cdots v_j \cdot v_{j+1}$ for some $v_j \in V_1 \subseteq V_1'$, i.e., the move of Player 1 simulates traversing the standard edge $(v_j, v_{j+1}) \in E$. Then, we define

$$t(v_0 \cdots v_j \cdot v_{j+1}) = t(v_0 \cdots v_j) \cdot (v_{j+1}, 0) \ .$$

- Finally, the last case is a prefix of the form $v_0 \cdots v_j \overline{v_j} \cdot v_{j+1}$ for some $v_j \in V_0'$, i.e., the move of Player 0 simulates traversing the standard edge $(v_j, v_{j+1}) \in E$. Then, we define

$$t(v_0 \cdots v_j \overline{v_j} \cdot v_{j+1}) = t(v_0 \cdots v_j) \cdot (v_{j+1}, 0) \ .$$

The invariant is satisfied in any case. Also, we again lift t to infinite plays via limits.

Now, we define a strategy σ' for Player 0 in $\mathcal{G}_{\text{rig}}$ via

$$\sigma'(v_0 \cdots v_j \overline{v_j}) = \sigma(t(v_0 \cdots v_j)) \ ,$$

where, for the sake of notational convenience, we assume that σ ignores the bits indicating whether or not a disturbance edge was taken present in $t(v_0 \cdots v_j)$.

The restriction to play prefixes ending in a vertex of the form $\overline{v}$ suffices, as these are the only vertices of Player 0 in $\mathcal{G}_{\text{rig}}$. A straightforward induction shows that for every play ρ that is consistent with σ', the play $t(\rho)$ is consistent with σ. Hence, $t(\rho)$ satisfies the winning condition if ρ starts in v, as σ is $(\omega+1)$-resilient from v. Let $t(\rho) = (v_0, b_0)(v_1, b_1)(v_2, b_2) \cdots$. Then, $v_0 v_1 v_2 \cdots \in \text{Win}$. Now, $h(\rho) = v_0 v_1 v_2 \cdots$ implies $\rho \in \text{Win}'$. Thus, σ' is a winning strategy for Player 0 from v, which concludes the implication from right to left. □

In the proof of Lemma 5.21 we construct an $\omega+1$-resilient strategy for Player 0 in $\mathcal{G}$ from a winning strategy for her in $\mathcal{G}_{\text{rig}}$. It is easy to see that this construction preserves positionality, which gives rise to the following corollary.

Corollary 5.22. *If Player 0 has a positional winning strategy that is winning for her from all vertices in $W_0(\mathcal{G}_{rig})$, then she has a positional strategy that is $\omega+1$-resilient from all vertices v in $\mathcal{G}$ with $r_{\mathcal{G}}(v) = \omega+1$.*

At this point, we are able to compute the resilience of those vertices v with $r_{\mathcal{G}}(v) \in \omega$ due to Lemma 5.17 and Lemma 5.18. Furthermore, we can identify the vertices v with $r_{\mathcal{G}}(v) = \omega+1$ due to Lemma 5.21. Since, for each vertex v, we have $r_{\mathcal{G}}(v) \in \omega \cup \{\omega, \omega+1\}$, all vertices not characterized by the statements above have resilience ω.

Thus, we have shown how to determine the resilience of all vertices in $\mathcal{G}$ and how to construct strategies witnessing the respective resilience out of winning strategies for variants of the underlying game without disturbances. In the following section, we show how to combine these results to effectively compute an optimally resilient strategy for Player 0 in $\mathcal{G}$.

5.2.3 Computing Optimally Resilient Strategies

To conclude our investigation of games with disturbances, we now show how to effectively compute the resilience of vertices and optimally resilient strategies for Player 0. Here, we focus on positional and finite-state strategies, which are sufficient for the winning conditions discussed in this work. It is, however, easy to see that the results from this section can be lifted to games with infinite-state winning strategies for Player 0 as well. Thus, we also provide approaches for effectively computing (finitely representable) optimally resilient infinite-state strategies.

→Sec. 5.2, Page 160
→Sec. 5.2, Page 163
→Sec. 5.2, Page 165

In the proof of →Lemma 5.17, we construct a strategy σ_f that is $r_{\mathcal{G}}(v)$-resilient from every v with $r_{\mathcal{G}}(v) \in \omega$. Furthermore, in the proof of →Lemma 5.18, we construct a strategy σ_ω that is ω-resilient from every v with $r_{\mathcal{G}}(v) \geq \omega$. Finally, in →Lemma 5.21 we construct a strategy $\sigma_{\omega+1}$ that is $\omega+1$ resilient from every v with $r_{\mathcal{G}}(v) = \omega+1$.

We obtain the former two strategies by combining winning strategies for the games $\mathcal{G}_U = (\mathcal{A}, \mathrm{Win} \cap \textsc{Safety}(U))$ with vertex set V where $U \subseteq V$, and the latter strategy by solving the game $\mathcal{G}_{\text{rig}}$. However, even if these winning strategies are positional, the strategies σ_f and σ_ω are in general not positional. Nonetheless, we show in the proof of Theorem Theorem 5.23 that such positional winning strategies and a positional one for $\mathcal{G}_{\text{rig}}$ can be combined into a single positional optimally resilient strategy.

In order to effectively compute this optimally resilient strategy, we need to assume that we are able to effectively compute the underlying strategies. More precisely, we have to assume that the games $\mathcal{G}_U$ as well as the rigged game $\mathcal{G}_{\text{rig}}$ are effectively solvable. This is indeed the case for all winning conditions discussed in this work.

Theorem 5.23. *Let $\mathcal{G}$ be a prefix-independent game with disturbances such that each $\mathcal{G}_U$ and $\mathcal{G}_{rig}$ is determined and can be effectively solved. Moreover, for each $\mathcal{G}_U$ and $\mathcal{G}_{rig}$, let Player 0 have a positional strategy that is winning from all vertices in her respective winning region.*

Then, the resilience of the vertices of $\mathcal{G}$ and a positional optimally resilient strategy can be effectively computed.

Proof. The effective computability of the resilience follows directly from the effectiveness requirements on $\mathcal{G}$: To compute the ranking r^*, it suffices to compute the disturbance and risk updates in alternation as discussed in →Section 5.2.1. The former are trivially effective while the effectiveness of the latter ones follows from our assumption. Lemma 5.17 shows that r^* correctly determines the resilience of all vertices with finite resilience. Finally, by solving the rigged game, we also correctly determine the resilience of the remaining vertices, as shown in Lemma 5.21. Again, this game can be effectively solved due to our assumption.

→Page 156

Thus, it remains to show how to compute a positional optimally resilient strategy. To this end we first construct for every vertex v of $\mathcal{G}$ a positional strategy σ_v satisfying the following properties.

- For every $v \in V$ with $r_{\mathcal{G}}(v) \in \omega \setminus \{0\}$, the strategy σ_v is winning for Player 0 from v for the game $(\mathcal{A}, \mathrm{Win} \cap \textsc{Safety}(\{v' \in V \mid r_{\mathcal{G}}(v') < r_{\mathcal{G}}(v)\}))$. The existence

of such a strategy has been shown in the proof of Lemma 5.17. It is effectively computable by assumption.

- For every $v \in V$ with $r_{\mathcal{G}}(v) = \omega$, the strategy σ_v is winning for Player 0 from v for the game $(\mathcal{A}, \mathrm{Win} \cap \textsc{Safety}(\{v' \in V \mid r_{\mathcal{G}}(v') \in \omega\}))$. The existence of such a strategy has been shown in the proof of Lemma 5.18. It is effectively computable by assumption.
- For every $v \in V$ with $r_{\mathcal{G}}(v) = \omega + 1$, the strategy σ_v is $(\omega+1)$-resilient from v. The existence of such a strategy follows from Corollary 5.22, as we assume Player 0 to win $\mathcal{G}_{\mathrm{rig}}$ with positional strategies. It is effectively computable by assumption.
- Finally, for every $v \in V$ with $r_{\mathcal{G}}(v) = 0$, we fix an arbitrary positional strategy σ_v for Player 0.

Furthermore, we fix a strict linear order $\prec$ on the vertices of $\mathcal{G}$ such that $v \prec v'$ implies $r_{\mathcal{G}}(v) \leq r_{\mathcal{G}}(v')$, i.e., we order the vertices by ascending resilience. We denote the non-strict variant of $\prec$ by $\preceq$. For a vertex v with $r_{\mathcal{G}}(v) \neq \omega + 1$, let R_v be the set of vertices reachable via disturbance-free plays that start in v and are consistent with σ_v. On the other hand, for a vertex v with with $r_{\mathcal{G}}(v) = \omega + 1$, let R_v be the set of vertices reachable via plays with arbitrarily many disturbances that start in v and are consistent with σ_v.

We claim $R_v \subseteq \{v' \in V \mid r_{\mathcal{G}}(v') \geq r_{\mathcal{G}}(v)\}$ for every $v \in V$ $(*)$. For v with $r_{\mathcal{G}}(v) \neq \omega + 1$ this follows immediately from the choice of σ_v. Thus, let us argue the claim for v with $r_{\mathcal{G}}(v) = \omega + 1$. Assume towards a contradiction that there exists a play starting in v and consistent with σ_v that reaches a vertex v' of resilience $r_{\mathcal{G}}(v') \neq \omega + 1$. Then, there exists a play ρ' starting in v' that is consistent with σ_v, has $r_{\mathcal{G}}(v') \in \omega$ many disturbances and is losing for Player 0. Thus the play obtained by first taking the play prefix to v' and then appending ρ' without its first vertex yields a play starting in v, consistent with σ_v, but losing for Player 0. The existence of this play implies that σ_v is not $(\omega+1)$-resilient from v, which yields the desired contradiction.

Let $m \colon V \to V$ be given as $m(v) = \min_{\prec} \{v' \in V \mid v \in R_{v'}\}$ and define the positional strategy σ as $\sigma(v) = \sigma_{m(v)}(v)$. By our assumptions, σ can be effectively computed. It remains to show that it is indeed optimally resilient.

To this end, we apply the following two properties of edges (v, v') that may be traversed during a play that is consistent with σ, i.e., we either have $(v, v') \in E$, or $(v, v') \in D$, which implies $v \in V_0$:

1. If $(v, v') \in E$, then we have $r_{\mathcal{G}}(v) \leq r_{\mathcal{G}}(v')$ and $m(v) \succeq m(v')$. The former property follows from minimality of $m(v)$ and $(*)$ while the latter one follows from the definition of R_v.
2. If $(v, v') \in D$, then we have to distinguish several subcases, which all follow immediately from the definition of resilience:
   - If $r_{\mathcal{G}}(v) \in \omega$, then $r_{\mathcal{G}}(v') \geq r_{\mathcal{G}}(v) - 1$.
   - If $r_{\mathcal{G}}(v) = \omega$, then $r_{\mathcal{G}}(v') = \omega$, and
   - If $r_{\mathcal{G}}(v) = \omega + 1$, then $r_{\mathcal{G}}(v') = \omega + 1$ and $m(v) \geq m(v')$ (here, the second property follows from the definition of R_v for v with $r_{\mathcal{G}}(v) = \omega + 1$, which allows for the occurrence of disturbance edges).

Now, consider a play $\rho = (v_0, b_0)(v_1, b_1)(v_2, b_2)\cdots$ that is consistent with σ. First, assume $r_{\mathcal{G}}(v_0) \in \omega$. We have to show that if ρ has less than $r_{\mathcal{G}}(v_0)$ disturbances, then it is winning for Player 0. If $r_{\mathcal{G}}(v_0) = 0$, then this claim vacuously holds true. Thus, assume $r_{\mathcal{G}}(v_0) > 0$. An inductive application of the above properties shows that in that case the last disturbance edge leads to a vertex of non-zero resilience. Furthermore, as the values $m(v_j)$ are only decreasing afterwards, they have to stabilize at some later point. Hence, there is some disturbance-free suffix of ρ that starts in some v' with non-zero resilience and that is consistent with the strategy $\sigma_{v'}$. Thus, the suffix is winning for Player 0 by the choice of $\sigma_{v'}$ and prefix-independence implies that ρ is winning for her as well.

Next, assume $r_{\mathcal{G}}(v_0) = \omega$. We have to show that if ρ has a finite number of disturbances, then it is winning for Player 0. Again, an inductive application of the above properties shows that in that case the last disturbance edge leads to a vertex of resilience ω or $\omega + 1$. Afterwards, the values $m(v_j)$ stabilize again. Hence, there is some suffix of ρ that starts in some v' with non-zero resilience and that is consistent with the strategy $\sigma_{v'}$. Thus, the suffix is winning for Player 0 by the choice of $\sigma_{v'}$ and prefix-independence implies that ρ is winning for her as well.

Finally, assume $r_{\mathcal{G}}(v_0) = \omega + 1$. Then, the above properties imply that ρ only visits vertices with resilience $\omega + 1$ and that the values $m(v_j)$ eventually stabilize. Hence, there is a suffix of ρ starting in some v' that is consistent with the $(\omega + 1)$-resilient strategy $\sigma_{v'}$. Thus, the suffix is winning for Player 0, no matter how many disturbances occurred. This again implies that ρ is winning for her as well. □

Next, we analyze the complexity of the algorithm sketched above in some more detail. The inductive definition of the r_j can be turned into an algorithm computing r^*, using the results of Lemma 5.15 to optimize the naive implementation, which has to solve $\mathcal{O}(|V|)$ many games (and compute winning strategies for some of them) with winning condition $\mathrm{Win} \cap \textsc{Safety}(U)$. Furthermore, the rigged game $\mathcal{G}_{\mathrm{rig}}$, which is of size $\mathcal{O}(|\mathcal{G}|)$, has to be solved and winning strategies have to be determined. Thus, the overall complexity is in general dominated by the complexity of solving these tasks.

Using similar arguments, one can also analyze games where positional strategies do not suffice. As above, assume $\mathcal{G}$ satisfies the same assumptions on determinacy and effectiveness, but only require that Player 0 has finite-state winning strategies for each game with winning condition $(\mathcal{A}, \mathrm{Win} \cap \textsc{Safety}(U))$ and for the rigged game $\mathcal{G}_{\mathrm{rig}}$. Then, one can show that she has a finite-state optimally resilient strategy. In fact, by reusing memory states, one can construct an optimally resilient strategy that it is not larger than any constituent strategy.

Corollary 5.24. *Let $\mathcal{G}$ be a prefix-independent game with disturbances such that each $\mathcal{G}_U$ and $\mathcal{G}_{rig}$ is determined and can be effectively solved. Moreover, let Player 0 have a winning strategy σ_U from her winning region in each $\mathcal{G}_U$ and σ_{rig} in $\mathcal{G}_{rig}$.*

If the σ_U are finite-state strategies of size s_U and if σ_{rig} is a finite-state strategy of size s_{rig} then an optimally resilient strategy of size at most $\max(\{s_U \mid U \subseteq V\} \cup \{s_{rig}\})$ can be effectively computed.

In the following section, we summarize the results obtained in this chapter and explicitly instantiate the results for the winning conditions discussed in this work.

5.3 Summary of Results

In this chapter we have first recalled the definition of games with intermittent unmodeled disturbances first introduced by Dallal, Neider, and Tabuada [DNT16] and defined the notion of optimally-resilient strategies, i.e., strategies that are still winning for Player 0 even if the maximal number of disturbances occur that allow her to win. Furthermore, we have shown how to compute optimally-resilient strategies for all games considered in this work by developing a general approach that relies only on very weak assumptions on the winning conditions.

We explicitly state the instantiation of our results from this chapter for parity games and their variants discussed in this work in the following corollary. The first, second, and third item of that corollary follow directly from Theorem 5.23, ➔Proposition 2.20, ➔Sec. 2.3, Page 20 ➔Proposition 2.27, and ➔Proposition 2.33. Recall that these propositions state that ➔Sec. 2.4, Page 24 parity games with n vertices and d odd colors can be solved in time $\mathcal{O}(n^{\log d+6})$, that ➔Sec. 2.4, Page 26 finitary parity games with n vertices can be solved in time $\mathcal{O}(n^4)$, and that parity games with costs with n vertices and d odd colors can be solved in time $\mathcal{O}(dn^2 2^{\log(d)})$, respectively. Similarly, the fourth item of that corollary follows from Theorem 5.23 and ➔Theorem 3.18. Recall that as a direct consequence of the proof of Theorem 3.18 ➔Sec. 3.2, Page 52 we furthermore obtain that parity games with weights with n vertices, d odd colors, and largest absolute weight W can be solved in time $\mathcal{O}(n^2(d(n')^{\log(d/\log n')+4.45}(W+1/n')))$, where $n' \in \mathcal{O}(n^2)$ due to Daviaud, Jurdziński, and Lazić [DJL18].

Corollary 5.25. *Let $\mathcal{G}$ be a game with disturbances.*

1. *If $\mathcal{G}$ is a parity game with n vertices and d odd colors, then Player 0 has a positional optimally-resilient strategy in $\mathcal{G}$ that can be computed in time $\mathcal{O}(n^{\log d+7})$.*
2. *If $\mathcal{G}$ is a finitary parity game with n vertices, then Player 0 has a positional optimally-resilient strategy in $\mathcal{G}$ that can be computed in time $\mathcal{O}(n^5)$.*
3. *If $\mathcal{G}$ is a parity game with costs with n vertices and d odd colors, then Player 0 has a positional optimally-resilient strategy in $\mathcal{G}$ that can be computed in time $\mathcal{O}(n(n^4 d)^{\log d+6})$.*
4. *If $\mathcal{G}$ is a parity game with weights with n vertices, d odd colors, and largest weight W, then Player 0 has a finite-state optimally-resilient strategy in $\mathcal{G}$ that can be computed in time $\mathcal{O}(n(n^2(d(n')^{\log(d/\log n')+4.45}(W+1/n'))))$, where $n' \in \mathcal{O}(n^2)$.*

The notion of resilience introduces a new metric for strategies: Strategies can be ordered by their resilience, i.e., a larger resilience against intermittent disturbances is better than a smaller one. This ordering is independent of the ordering induced by the cost of strategies in parity games with costs and parity games with weights, which we discussed in ➔Chapter 4. In the following section, we investigate the interplay ➔Page 83 between these two metrics, i.e., we investigate the relationship between the resilience of a strategy and the cost it realizes in quantitative parity games. Furthermore, we

discuss the memory requirements of strategies that are both resilient and enforce a certain cost of the resulting play.

CHAPTER 6

Tradeoffs

At this point, we have identified three metrics of strategies of Player 0 in finitary parity games and, by extension, in parity games with weights. Firstly, each strategy for her has a size, i.e., the amount of memory used to implement that strategy. Here, in general, smaller strategies are more desirable. Secondly, each strategy σ in a finitary parity game has an associated cost as discussed →Chapter 3 in →Chapter 4, defined as the upper bound on the cost of plays that are consistent with σ. Here, again, a smaller cost is more desirable. Thirdly, when playing on an arena with disturbances, each strategy for Player 0 is resilient against a number of disturbances, i.e., there exists a maximal number of disturbances which still allow her to win the resulting play. We discussed this setting in →Chapter 5. In contrast to the previous two characteristics of strategies, a larger resilience is more desirable.

→Page 29
→Page 83
→Page 149

In this section, we study the interplay between the three characteristics of size, cost, and resilience of strategies. To this end, we consider all possible pairings of these three characteristics and investigate the tradeoff between these metrics.

First, recall that we already stated in →Corollary 5.24 that optimally resilient strategies have the same size as arbitrary winning strategies for Player 0 in the underlying game. Hence, there exists no tradeoff between these two metrics, as resilience comes, in a sense, for free in terms of memory.

→Sec. 5.2, Page 170

Second, we consider the tradeoff between the cost of a strategy and the memory required to implement it. Recall that in a finitary parity game $\mathcal{G}$ with n vertices and d odd colors Player 0 has a positional winning strategy from her winning region due to →Proposition 2.25.1. If she, however, aims to bound the cost of the resulting plays from above by b, then she requires a strategy of size in $\mathcal{O}(b^d)$ due to →Lemma 4.39. In Section 6.1 we investigate whether this tradeoff is abrupt, i.e., whether there exists some bound b_0 such that Player 0 has a positional strategy that guarantees some bound b with $b_0 < b < \infty$ and such that she requires memory in $\mathcal{O}(b^d)$ in order to

→Sec. 2.4, Page 23
→Sec. 4.4, Page 131

enforce a bound $b \leq b_0$. We show that this is not the case and that this tradeoff may, in general, be gradual, i.e., Player 0 may require more and more memory in order to decrease the bound.

Third, we discuss the tradeoff between the cost of a strategy and its resilience in Section 6.2. Here, we show how to compute, for a fixed b, the maximal number of disturbances under which Player 0 can still enforce the cost of a play of at most b. Vice versa, we also show how to compute for a given number of disturbances r the minimal bound on the cost of the resulting play that Player 0 can ensure under the assumption that there are at most r disturbances. Furthermore, we argue that strategies witnessing these respective optimal values are effectively constructible and that computing them does not have a greater complexity in terms of runtime than solving the underlying parity game with weights optimally, as discussed in Chapter 4.

The results shown in Section 6.1 are based on work published in LMCS [WZ17]. The results shown in Section 6.2 constitute novel work.

6.1 Trading Memory for Optimality

➔Page 83

In ➔Chapter 4 we have investigated the problem of deciding the winner in parity games with weights and their special cases with respect to some given bound. There, we have shown that, in order to ensure an upper bound of b on the cost of the resulting play in a finitary parity game with d odd colors, $(b+2)^d$ memory states suffice (see

➔Sec. 4.4, Page 131

➔Lemma 4.39).

This upper bound, however, increases together with b, which is counterintuitive: Intuitively, increasing the bound b that Player 0 has to satisfy should make it easier for her to satisfy that bound, since more and more plays become winning for her. Thus, the size of strategies should decrease with increasing b. We illustrate the space of strategies for Player 0 in finitary parity games in Figure 6.1: For each finitary parity game $\mathcal{G}$ and each vertex v of $\mathcal{G}$, if Player 0 wins $\mathcal{G}$ from v, then there exists a minimal $b_0 \in \mathbb{N}$ such that Player 0 has a strategy σ_0 with $\text{Cost}(\sigma_0) \leq b_0$. Due to Lemma 4.39 we obtain $|\sigma_0| \leq (b+2)^d$. We indicate this upper bound by the dashed line in Figure 6.1. Since $\text{Cost}(\sigma_0) \leq b'$ for all $b' \geq b_0$, the size of σ_0 is an upper bound on the size of strategies that realize some cost greater than b_0, as indicated by the solid line in Figure 6.1. Furthermore, for $b' \geq b_1 = n$ we obtain that Player 0 has a positional strategy realizing cost at most b' if she has a winning strategy at all, due to ➔Proposition 2.25.1. This is indicated by the steep dropoff of the solid line at $b_1 = n$ in Figure 6.1. Finally, we indicate the set of pairs b and s such that Player 0 is known to have a strategy of cost at most b and size s by shading it in gray in Figure 6.1.

➔Sec. 2.4, Page 23

Before showing that this intuition in fact holds true, i.e., that there exist games in which it becomes "easier" to win for Player 0 with increasing bound b, we first argue that such a steep dropoff as illustrated in Figure 6.1 may actually occur. To this end, recall the construction of a finitary parity game from quantified Boolean formulas used to show PSPACE-hardness of the threshold problem for finitary parity games in ➔Section 4.3.1. In that section, we constructed, given a quantified Boolean

➔Page 117

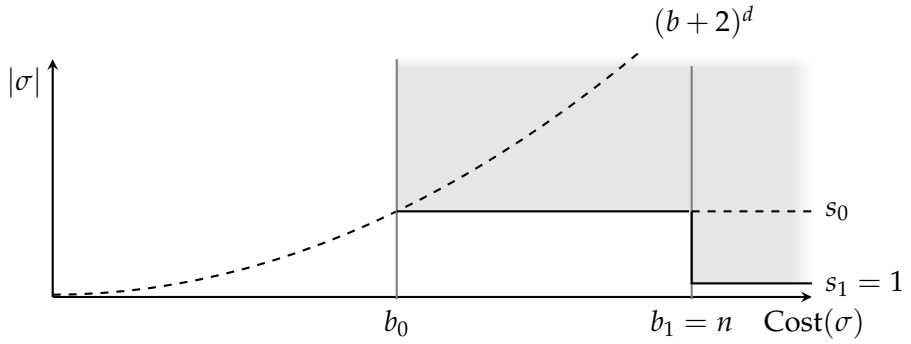


Figure 6.1: The structure of the space of strategies in finitary parity games for Player 0.

formula φ, a finitary parity game $\mathcal{G}_\varphi$ together with a designated vertex v of $\mathcal{G}_\varphi$ and a bound b_φ such that Player 0 has a strategy σ with $\mathrm{Cost}_v(\sigma) \leq b_\varphi$ in $\mathcal{G}_\varphi$ if and only if φ evaluates to true. Intuitively, the strategy σ prescribes truth values for the existentially quantified variables of φ depending on the values assigned to earlier universally quantified variables.

For some fixed $n \in \mathbb{N}$, consider the quantified Boolean formula

$$\varphi = \forall x_1 \cdots \forall x_n \exists y_1 \cdots \exists y_n. \bigwedge_{j \in \{1,\ldots,n\}} (x_j \leftrightarrow y_j) \ ,$$

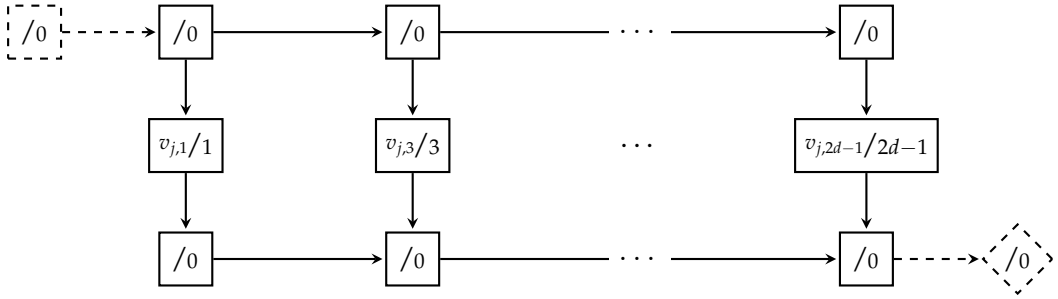
where we use the shorthand $x \leftrightarrow y$ to denote the formula $(\neg x \vee y) \wedge (\neg y \vee x)$, i.e., $x \leftrightarrow y$ denotes that the two variables x and y carry the same truth value. Although this formula is not given in the normal form assumed in Section 4.3.1 it is straightforward to rewrite it in that form by introducing additional, unused variables.

Clearly, φ evaluates to true, as witnessed by assigning each variable y_i the (previously chosen) value of x_i. A strategy σ for Player 0 in $\mathcal{G}_\varphi$ with $\mathrm{Cost}_v(\sigma) \leq b_\varphi$, however, clearly requires exponential memory, as it is required to store the choices made by Player 1 when choosing truth values for the variables x_1 through x_n. In contrast, we have argued in Section 4.3.1 that every strategy σ' for Player 0 in $\mathcal{G}_\varphi$ has $\mathrm{Cost}_v(\sigma') \leq b_\varphi + 2$. In particular, this holds true for all positional strategies for her. This observation gives rise to the following corollary.

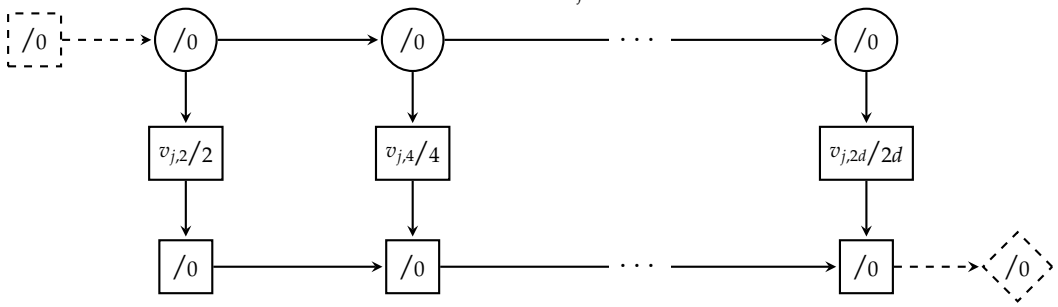
Corollary 6.1. *For each $n \in \mathbb{N}$ there exists a finitary parity game $\mathcal{G}_n$ with $\mathcal{O}(n)$ vertices, a bound $b_n \in \mathcal{O}(n)$, and a designated vertex v^* of $\mathcal{G}_n$ such that*

1. *there exists a strategy σ of size $\mathcal{O}(2^n)$ for Player 0 such that $\mathrm{Cost}_{v^*}(\sigma) \leq b_n$, such that*
2. *for all strategies σ' for Player 0 we have that $\mathrm{Cost}_{v^*}(\sigma') \leq b_n$ implies $|\sigma'| \geq |\sigma|$, and such that*
3. *there exists a positional strategy σ'' for Player 0 such that $\mathrm{Cost}_{v^*}(\sigma'') \leq b_n + 2$.*

The above corollary witnesses that there indeed exist games for which the space of strategies for Player 0 is structured as in Figure 6.1, i.e., in which that space features a sharp dropoff in terms of memory required in order to satisfy a given bound on the



(a) The j-th gadget G_j^1 for Player 1.



(b) The j-th gadget G_j^0 for Player 0.

Figure 6.2: The gadgets constituting the arena of $\mathcal{G}_d$.

incurred costs. In these games, there exists some bound b such that Player 0 requires exponential memory in order to enforce cost at most b, but she has a positional strategy that enforces bound $b + 2$.

We now show that there also exist games witnessing a contrasting situation: There exist finitary parity games in which, the larger the bound b is (in the interval $b_0 \leq b < b_1$), the smaller the minimal strategies enforcing cost of at most b are.

Theorem 6.2. *For each $d \geq 1$ there exists a finitary parity game $\mathcal{G}_d$ with $\mathcal{O}(d^2)$ vertices that contains a designated vertex v_I such that for every j with $1 \leq j \leq d$ there exists a strategy σ_j for Player 0 in $\mathcal{G}_d$ such that*

- $d^2 + 3d - 1 = Cost_{v_\mathrm{I}}(\sigma_1) > Cost_{v_\mathrm{I}}(\sigma_2) > \cdots > Cost_{v_\mathrm{I}}(\sigma_d) = d^2 + 2d$, *and*
- $1 = |\sigma_1| < |\sigma_2| < \cdots < |\sigma_d| = 2^{d-1}$.

Also, for every strategy σ' for Player 0 in $\mathcal{G}_d$ with $Cost(\sigma') \leq Cost(\sigma_j)$ we have $|\sigma'| \geq |\sigma_j|$.

Proof. We define $\mathcal{G}_d$ as the game from the proof of necessity of exponential memory for optimal strategies for Player 0 in finitary parity games, i.e., from the proof in
→ Page 133
→ Section 4.4.2. Recall that this game consists of d gadgets for each player, each of which contains $3d$ vertices. We reprint the gadgets G_j^1 and G_j^0 for Player 1 and Player 0, respectively, in Figure 6.2 and the overall construction of the arena of $\mathcal{G}_d$ in Figure 6.3 for the sake of completeness. Moreover, we again define v_I as the top-left vertex of the first gadget G_1^1 for Player 1.

Furthermore recall that, in Section 4.4.2, we defined the set of strictly increasing

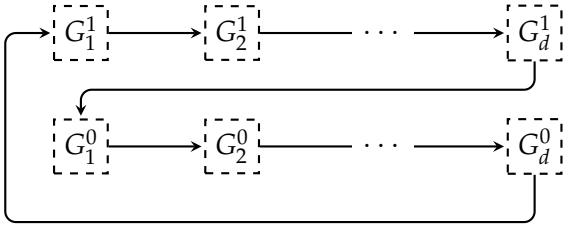


Figure 6.3: The finitary parity game $\mathcal{G}_d$ witnessing the existence of gradual tradeoffs for Player 0.

sequences of odd integers IncSeq_d and showed that we can implement an optimal strategy σ with cost at most $d^2 + 2d$ when starting from v_I using the set IncSeq_d as memory states.

Intuitively, the strategy σ stores up to $d-1$ requests made by Player 1 in his part of each round, as the final element of each increasing sequence is fixed to be $2d-1$. The idea behind the construction of the strategies σ_j for $1 \leq j \leq d$ is to restrict the memory of Player 0 such that she can only store up to $j-1$ requests. In the extremal cases of $j = 1$ and $j = d$ this implements a positional strategy that always moves to the vertex of color $2d$, and the strategy σ as above, respectively.

We implement σ_j via a memory structure $\mathcal{M}_d^j = (M_d^j, init_d^j, upd_d^j)$, which we define in the following. We again use strictly increasing odd sequences as the memory states of $\mathcal{M}_d^j$, but now restrict the maximal number of entries that differ from the maximal value of $2d-1$: Thus, we implement the above intuition that Player 0 stores at most $j-1$ requests.

To this end, we define the length-restricted set of strictly increasing odd sequences

$$\text{IncSeq}_d^j = \text{IncSeq}_d \cap \left\{ s = (c_1, \ldots, c_{j-1}, 2d-1, \ldots, 2d-1) \mid s \in \mathbb{N}^d \right\} \ ,$$

where $\mathbb{N}^d$ denotes the tuples of length d over the natural numbers, and pick $M_d^j = \text{IncSeq}_d^j$. Note that $M_d^d = M_d$ as defined in Section 4.4.2 and that M_d^1 is a singleton set containing only the sequence $(2d-1, \ldots, 2d-1)$ of length d. Clearly, the second claim of the theorem holds true, since $\text{IncSeq}_d^{j-1} \subset \text{IncSeq}_d^j$ for each $d \geq 1$ and each j with $1 \leq j \leq d$ and since we have already argued $|M_d| = 2^{d-1}$ in Section 4.4.2. In fact, we have $\left|\text{IncSeq}_d^j\right| = \Sigma_{0 \leq j' < j} \binom{d-1}{j'}$, since each sequence in IncSeq_d^j is isomorphic to a subset of the odd numbers up to $2d-3$ containing at most $j-1$ elements.

We define the initialization function

$$init_d^j(v) = (1, 3, \ldots, 2j-3, 2d-1, \ldots, 2d-1)$$

for all vertices v of $\mathcal{G}_d$, since we are only interested in plays starting in v_I in the following. Furthermore, we adapt the update function from Section 4.4.2 such that it only stores the first j relevant requests, obtaining the update function upd_d^j, which concludes the definition of $\mathcal{M}_d^j$.

Finally, we define the next-move function nxt_d^j to be identical to the next-move function nxt_d from Section 4.4.2 and define σ_d^j as the strategy implemented by $\mathcal{M}_d^j$ and nxt_d^j.

It remains to show $\mathrm{Cost}_{v_\mathrm{I}}(\sigma_j) = d^2 + 3d - j$ and that σ_j is a minimal strategy realizing that cost. To this end, we fix some j with $1 \leq j \leq d$ for the remainder of this proof.

First, we show that, starting in v_I, Player 1 can enforce a cost of $d^2 + 3d - j$ if Player 0 plays consistently with σ_j. Intuitively, Player 1 fills the memory of Player 0 as quickly as possible, and requests the minimal color that has not yet been requested afterwards. Thus, he maximizes the gap between the smallest request that Player 0 was not able to store using $\mathcal{M}_d^j$ and the "default" answer of $2d$.

More precisely, in each turn Player 1 requests the colors

$$1, 3, \ldots, 2j-3, 2j-1, 2j-1, \ldots, 2j-1 \ .$$

Playing consistently with σ_j, Player 0 then answers these requests with the sequence of colors

$$2, 4, \ldots, 2j-2, 2d, 2d, \ldots, 2d \ .$$

Hence, the cost of the resulting play is that incurred by answering a request for $2j-1$ in the j-th gadget of Player 1 with $2d$ in the j-th gadget of Player 0. Using arguments analogous to those from Section 4.4.2, the cost incurred by such a request-response-pair amounts to

$$d - \frac{2j-1+1}{2} + 2 + (d-j+j-1)(d+2) + \frac{2d}{2} = \\ d - j + 2 + (d-1)(d+2) + d = d^2 + 3d - j \ .$$

Since playing consistently with σ_d^j, Player 0 answers each request in every turn, the game restarts after the turn of Player 0, and hence Player 1 can enforce this cost infinitely often. Consequently, we obtain $\mathrm{Cost}_{v_\mathrm{I}}(\sigma_j) \geq d^2 + 3d - j$.

Note that the requests posed by Player 1 after visiting his j-th gadget in the above sequence are irrelevant for the cost of the play. Up to and including the request for color $2j-1$, however, this sequence of requests is indeed optimal for Player 1, i.e., he cannot enforce a higher cost, as we show in the following. Assume that Player 1 does not pose requests as specified above, but instead poses requests for the colors $c_1, \ldots, c_j$ in his gadgets. Then, either Player 1 reorders the requests for the colors specified above, or he requests larger colors than specified above. Formally, either

1. there exist some k and k' with $k < k' \leq j$, such that $c_k \geq c_{k'}$, or
2. there exists a $k \leq j$ with $2j - 1 < c_k \leq 2d - 1$.

In the first case, let k be minimal such that such a k' exists. Player 0 answers the first $k-1$ requests optimally before answering all remaining requests with costs at most $(d-1)(d+2)$, as she "ignores" the request for $c_{k'}$. In the second case, Player 0 again answers all requests up to the first request as described above optimally. Afterwards, she answers all succeeding requests with cost at most $d^2 + 2d + (2d - 1 -$

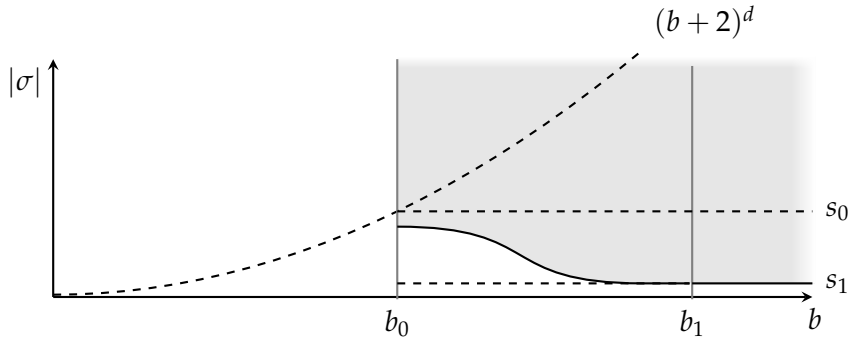


Figure 6.4: The structure of the space of strategies for Player 0 in the game $\mathcal{G}_d$ from Theorem 6.2.

$c_k)/2 \leq d^2 + 3d - j$. Hence, there exists no play ρ starting in v_I and consistent with σ_j that satisfies $\mathrm{Cost}(\rho) > d^2 + 3d - j$.

To conclude the proof, we observe that there exists no strategy σ' with $|\sigma'| < |\sigma_j|$ and $\mathrm{Cost}(\sigma') \leq \mathrm{Cost}(\sigma_j)$. The argument is nearly identical to the argument of minimality of the strategy constructed in the proof of ➔Lemma 4.45 and can in fact be obtained by replacing all occurrences of 2^{d-1} and $d^2 + 2d$ by $|\sigma_j|$ and $d^2 + 3d - j$, respectively. Hence, the strategies σ_j are minimal for their respective cost. □

➔Sec. 4.4, Page 137

The above theorem shows that, in general, Player 0 may be able to trade an improvement in the realized bound for a larger memory requirement of the witnessing strategy in finitary parity games. The strategies σ_j witnessing the increased bounds on the incurred cost differ from those constructed in ➔Chapter 4 in that they do not store the costs incurred by the requests, but only the order in which the requests are made. Due to the structure of the arena, the costs these requests have incurred can then be reconstructed. We show the structure of the space of strategies for Player 0 in $\mathcal{G}_d$ in Figure 6.4. The black solid line denotes the size of the strategies σ_j constructed in the proof of Theorem 6.2.

➔Page 83

Thus, this theorem shows that the strategies obtained in Chapter 4 are not minimal for the bound they realize. For a given finitary parity game $\mathcal{G}$ with d odd colors, a given vertex v^* of $\mathcal{G}$, and a given bound $b \in \mathbb{N}$, a minimal strategy σ with $\mathrm{Cost}_{v^*}(\sigma) \leq b$ may be found by enumerating all strategies for Player 0 of size at most $(b+2)^d$ and checking whether they have cost at most b. However, we conjecture that this approach is far from optimal. While clearly decidable, it remains open to determine an algorithm with optimal complexity of solving the following problem:

> "Given a finitary parity game $\mathcal{G}$, a vertex v of $\mathcal{G}$, and a bound $b \in \mathbb{N}$, compute a minimal strategy σ with $\mathrm{Cost}(\sigma) \leq b$."

We furthermore conjecture that a more efficient algorithm solving the above problem may be found by using the approach of bounded synthesis [FS13]. Using this approach, the problem of finding a strategy of size at most s is reduced to solving a

satisfiability problem, for which there exist efficient solvers. Completeness of this approach follows from the existence of an upper bound on the size of winning strategies. Since we have such an upper bound of $(b+2)^d$, we conjecture that minimal strategies realizing a given cost can be constructed efficiently using bounded synthesis.

Up to this point, we have only considered the structure of the space of strategies for Player 0. We now turn our attention to Player 1. Intuitively, the smaller the bound becomes, the "easier" (with respect to the memory required) it should be for Player 1 to win with respect to that bound. In fact, the following theorem shows that the games constructed for showing necessity of exponential memory for Player 1 in → Section 4.4.3 → Page 138 already witness that this intuition does indeed hold true.

Theorem 6.3. *For each $d \geq 1$ there exists a finitary parity game $\mathcal{G}_d$ with $\mathcal{O}(d^2)$ vertices that contains a designated vertex v_I, such that for every j with $1 \leq j \leq d$, there exists a strategy τ_j for Player 1 in $\mathcal{G}_d$ such that*

- $7 = Cost_{v_I}(\tau_1) < Cost_{v_I}(\tau_2) < \cdots < Cost_{v_I}(\tau_d) = 5(d-1)+7$, *and*
- $2 = |\tau_1| < |\tau_2| < \cdots < |\tau_d| = 2^d$.

Also, for every strategy τ' for Player 1 with $Cost_{v_I}(\tau') \geq Cost_{v_I}(\tau_j)$, we have $|\tau'| \geq |\tau_j|$.

Proof. We construct the game $\mathcal{G}_d$ out of the games $\mathcal{G}_d$ constructed in the proof of necessity of exponential memory for Player 1 in finitary parity games, i.e., from those games constructed in Section 4.4.3. For each j with $1 \leq j \leq d$ let $\mathcal{G}'_j$ be the game $\mathcal{G}_d$ constructed in Section 4.4.3.

We construct $\mathcal{G}_d$ such that it contains a designated initial vertex v_I, from which Player 1 may choose to move to the initial vertex of any of the $\mathcal{G}'_j$. Once the play ρ has moved into some $\mathcal{G}'_j$, it never leaves that part of the arena, i.e., the suffix starting at the second position of ρ is a play of $\mathcal{G}'_j$ starting in the initial vertex of that game.

For each j, the strategy τ_j defined in Section 4.4.3, augmented by a single move from v_I to the sub-game $\mathcal{G}'_j$, satisfies the properties above. Moreover, as the subgames $\mathcal{G}'_j$ are isolated from each other, each strategy τ' for Player 1 in $\mathcal{G}_d$ can be trivially transformed into a strategy for him in the subgame $\mathcal{G}'_j$ that τ' chooses at the beginning of $\mathcal{G}_d$. Hence, every strategy τ_j is of minimal size for the cost that it realizes. □

We illustrate the space of strategies for Player 1 in Figure 6.5. Again, the solid black line indicates the size of the strategies τ_j for Player 1 as constructed in the proof of Theorem 6.3, while the shaded area indicates the set of pairs of b and s such that Player 1 has a strategy of size at most s with cost exceeding b. Since Player 0 can enforce a cost of the play of at most $b_0 = 5(d-1)+8$, Player 1 has no strategy τ with $\mathrm{Cost}_{v_I}(\tau) > b_0$.

We conclude this section by noting that the above results also hold true for parity games with costs, since they subsume finitary parity games and Player 0 still has positional winning strategies (cf. → Proposition 2.34.1). → Sec. 2.4, Page 26 Since parity games with weights further generalize parity games with costs, the games constructed in the proofs of Theorem 6.2 and Theorem 6.3 also witness the existence of gradual tradeoffs in parity games with weights. Here, however, the dropoff between the size of strategies that

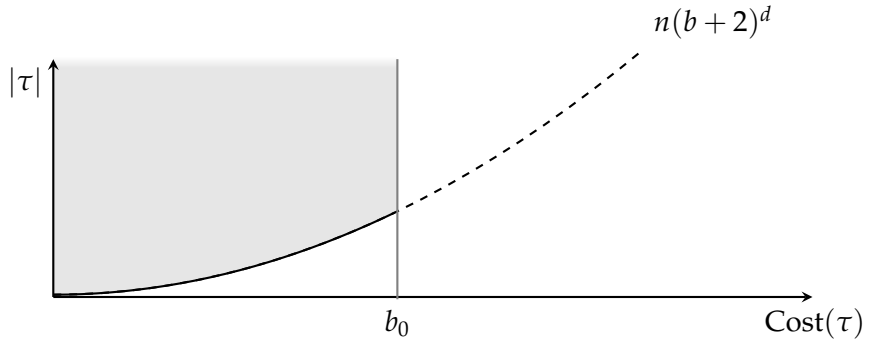


Figure 6.5: The structure of the space of strategies for Player 1 in the game $\mathcal{G}_d$ from Theorem 6.3.

are winning with respect to some bound and general winning strategies for Player 0 is not necessarily as steep as in the special cases considered previously, since Player 0 already requires exponential memory to win in such games (cf. ➔Theorem 3.34). ➔Sec. 3.4, Page 71

This concludes our discussion of tradeoffs between the quality of strategies and their memory requirements. In the following section we investigate the tradeoff between the resilience of a strategy and the cost of consistent plays that it guarantees.

6.2 Trading Resilience for Optimality

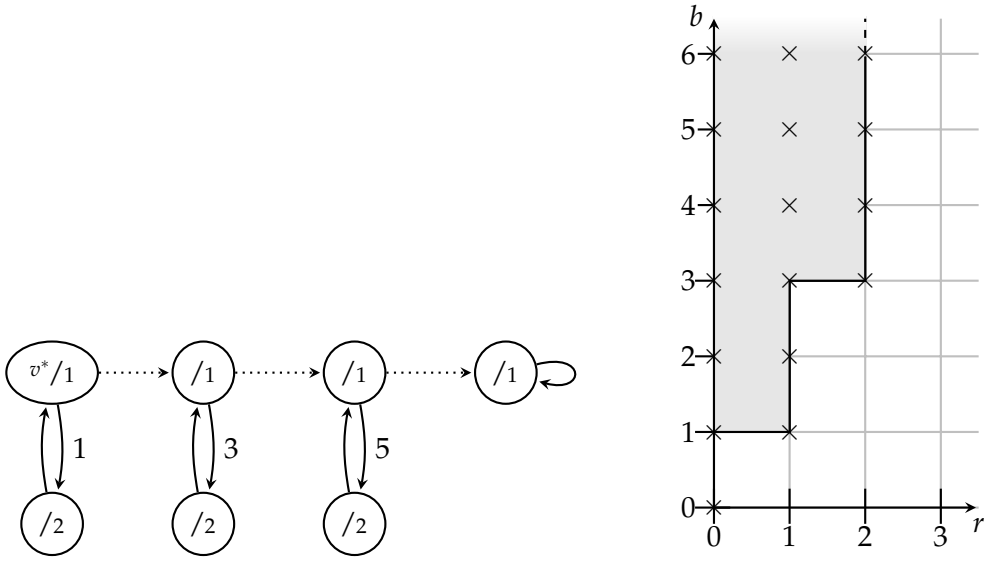
We now investigate the relationship between resilience and optimality, i.e., we consider the problem of playing optimally in parity games with weights in arenas with disturbances as defined in ➔Chapter 5. For the sake of readability, we assume that all arenas used in this chapter are arenas with disturbances, i.e., all games are games with disturbances. Otherwise, i.e., if the underlying arenas do not contain disturbance edges, the problems considered in this section trivially reduce to solving parity games with weights optimally as discussed in ➔Chapter 4. ➔Page 149 ➔Page 83

Our aim in this section is to answer the following question:

> "Given a parity game with weights $\mathcal{G}$, some vertex v^* of $\mathcal{G}$, some $r \in \omega + 2$ and some $b \in \mathbb{N}$, does Player 0 have a strategy σ such that for all plays ρ starting in v^* and consistent with σ we have that $\#_D(\rho) < r$ implies $\text{Cost}(\rho) \leq b$?"

This question thus combines the quantitative properties investigated in Chapter 4 and Chapter 5.

Recall that, intuitively, disturbances serve to describe real-world scenarios in which the action prescribed by a strategy for Player 0 differs from the action that is actually taken. Formally, an arena with disturbances is equipped with a set of disturbance edges, each of which is an outgoing edge of a vertex of Player 0. Whenever the token



(a) The game $\mathcal{G}$ discussed in Example 6.5.

(b) The marked positions denote the pairs $(r, b) \in \text{TrOff}_{v^*}(\mathcal{G})$, where $\mathcal{G}$ is the game shown in Figure 6.6a.

Figure 6.6: A parity game with weights and disturbances $\mathcal{G}$ and the set $\text{TrOff}_{v^*}(\mathcal{G})$.

is at a vertex v of Player 0, it may then occur that instead of moving along her chosen edge, the token moves along some disturbance edge attached to v.

In order to answer this question, we first introduce some notation. For the remainder of this section, fix some parity game with weights $\mathcal{G} = (\mathcal{A}, \textsc{WeightParity}(\Omega, \text{Weight}))$ with vertex set V containing n vertices, d odd colors, and largest absolute weight W. Furthermore, fix some vertex $v^* \in V$.

Def. $\text{TrOff}_{v^*}(\sigma)$

Now, let σ be a strategy for Player 0 in $\mathcal{G}$ and let v^* be a vertex of $\mathcal{G}$. We define the tradeoffs of σ (with respect to v^*) $\textsc{TrOff}_{v^*}(\sigma)$ as a subset of $(\omega + 2) \times \mathbb{N}$ as follows: $(r, b) \in \text{TrOff}_{v^*}(\sigma)$ if and only if for all plays ρ starting in v^* and consistent with σ we have that $\#_D(\rho) < r$ implies $\text{Cost}(\rho) \leq b$. It follows directly from the definition of $\text{TrOff}_{v^*}(\sigma)$ that it is downwards-closed in the first component and upwards-closed in the second one.

Remark 6.4. *Let σ be a strategy for Player 0 in $\mathcal{G}$, let $r \in \omega + 2$, and let $b \in \mathbb{N}$ such that $(r, b) \in \mathit{TrOff}_{v^*}(\sigma)$. Then, for all $r' \in \omega + 2$ with $r' \leq r$ and for all $b' \in \mathbb{N}$ with $b' \geq b$ we have $(r', b') \in \mathit{TrOff}_{v^*}(\sigma)$.*

Def. $\text{TrOff}_{v^*}(\mathcal{G})$

So far, the notion of tradeoffs only considers the tradeoffs of a single strategy. We lift this notion to games by defining $\text{TrOff}_{v^*}(\mathcal{G}) = \cup_\sigma \text{TrOff}_{v^*}(\sigma)$, where σ ranges over all strategies for Player 0.

Example 6.5. Let $\mathcal{G}$ be the parity game with weights and disturbances shown in Figure 6.6a. In $\mathcal{G}$, there exists only a single strategy for Player 0.

Each disturbance-free play starting in v^* has cost one, while each play with one disturbance has cost three. Moreover, each play with two disturbances has cost five. Furthermore, each play with three disturbances is losing for Player 0. We illustrate the set $\mathrm{TrOff}_{v^*}(\mathcal{G})$ in Figure 6.6b. △

We now answer the above question by computing the set $\mathrm{TrOff}_{v^*}(\mathcal{G})$. Even though this set is, in general, infinite, we show that it can be finitely represented. For the remainder of this section, we use the notion of computing $\mathrm{TrOff}_{v^*}(\mathcal{G})$ interchangeably with computing its finite representation.

Towards computing $\mathrm{TrOff}_{v^*}(\mathcal{G})$, we first remark that due to monotonicity, for each bound $b \in \mathbb{N}$ there exists a maximal number of disturbances $r_b \in \omega + 2$ such that Player 0 can enforce cost b from v^* if and only if less than r_b disturbances occur. This follows directly from downwards-closure of $\mathrm{TrOff}_{v^*}(\mathcal{G})$ in the first component, i.e., from Remark 6.4.

Def. r_b

Remark 6.6. *For each $b \in \mathbb{N}$ there exists a $r_b \in \omega + 2$ such that*
1. *for all $r \in \omega + 2$ with $r \leq r_b$ we have $(r, b) \in \mathit{TrOff}_{v^*}(\mathcal{G})$, and such that*
2. *for all $r \in \omega + 2$ with $r > r_b$ we have $(r, b) \notin \mathit{TrOff}_{v^*}(\mathcal{G})$.*

Furthermore, again due to Remark 6.4, it suffices to compute r_b for all $b \in \mathbb{N}$ to determine $\mathrm{TrOff}_{v^*}(\mathcal{G})$. It is, however, not clear a priori that this sequence eventually converges. Hence, at this point we have to show not only that each r_b is effectively computable, but also that we only need to compute a finite number of the r_b in order to determine $\mathrm{TrOff}_{v^*}(\mathcal{G})$.

We first show that each r_b is effectively computable. To this end, we define the PARITY CONDITION WITH WEIGHTS WITH RESPECT TO THE BOUND b as

Def. parity condition with weights with respect to the bound b

$$\textsc{WeightParity}_b(\Omega, \mathrm{Weight}) = \{\rho \in V^\omega \mid \mathrm{Cost}(\rho) \leq b\} \quad .$$

Recall that we fixed the game $\mathcal{G} = (\mathcal{A}, \textsc{WeightParity}(\Omega, \mathrm{Weight}))$. For each $b \in \mathbb{N}$ we then define the game

Def. $\mathcal{G}_b$

$$\mathcal{G}_b = (\mathcal{A}, \textsc{WeightParity}_b(\Omega, \mathrm{Weight})) \quad .$$

Via unraveling the definitions of r_b and $\mathrm{TrOff}_{v^*}(\mathcal{G})$, we obtain a strong connection between the elements of $\mathrm{TrOff}_{v^*}(\mathcal{G})$ and the resilience of v^* in the $\mathcal{G}_b$.

Remark 6.7. *Let $r \in \omega + 2$, and let $b \in \mathbb{N}$. We have $(r, b) \in \mathit{TrOff}_{v^*}(\mathcal{G})$ if and only if $r_{\mathcal{G}_b}(v^*) \geq r$.*

This observation allows us to compute r_b via computing the resilience of v^* in $\mathcal{G}_b$.

Corollary 6.8. *We have $r_b = r_{\mathcal{G}_b}(v^*)$.*

It remains to argue that we can effectively compute $r_{\mathcal{G}_b}(v^*)$. This follows from our results of Chapter 5 together with the following simple properties of the winning conditions of the $\mathcal{G}_b$.

Remark 6.9. *For each $b \in \mathbb{N}$ we have that*

1. $\textsc{WeightParity}_b(\Omega, \mathit{Weight})$ *is prefix-independent, and that*
2. $\textsc{WeightParity}_b(\Omega, \mathit{Weight}) \cap \textsc{Safety}(V')$ *is determined for all $V' \subseteq V$.*

Due to these properties, each game $\mathcal{G}_b$ satisfies the preconditions leveraged in the proofs from Chapter 5, which yields the following result.

Lemma 6.10. *For each $b \in \mathbb{N}$ the value r_b can be computed in exponential time.*

Proof. Recall that we defined V as the vertex set of $\mathcal{G}$ with cardinality n. Due to Remark 6.9 and the arguments from ➔Section 5.2 we can compute $r_{\mathcal{G}_b}(v^*)$ by solving at most n games of the form $(\mathcal{A}, \textsc{WeightParity}_b(\Omega, \mathrm{Weight}) \cap \textsc{Safety}(V'))$, where $V' \subseteq V$, and by solving the rigged game of $\mathcal{G}_b$. Since solving each of these games can easily be reduced to solving a parity game with weights with respect to some b, and as such games can be solved in exponential time due to ➔Theorem 4.12, we can compute $r_{\mathcal{G}_b}(v^*)$ in exponential time. This suffices to compute r_b due to Corollary 6.8. □

➔Page 155

➔Sec. 4.1, Page 96

As usual throughout this work exponential time in the statement of Lemma 6.10 refers to the description length of the input: Since the input here only consists of a single number, which we assume to be given in binary encoding, the value r_b can even be computed in polynomial time when measured in the value of b instead of the size of its encoding.

Thus, we have shown that the values r_b are effectively computable. It remains to argue that we only need to compute a finite number of these values in order to obtain $\mathrm{TrOff}_{v^*}(\mathcal{G})$. To this end, we show that the sequence $r_0, r_1, r_2, \ldots$ stabilizes after a bounded number of steps.

Lemma 6.11. *Let $b_0 = d(6n^2)(d+1)W(W+1)$. For all $b \geq b_0$ we have $r_b = r_{b_0}$.*

Proof. We show that for all $b \geq b_0$ we have $(r, b) \in \mathrm{TrOff}_{v^*}(\mathcal{G})$ if and only if $(r, b_0) \in \mathrm{TrOff}_{v^*}(\mathcal{G})$. This directly yields the desired result. The direction from right to left follows directly due to Remark 6.4. Hence, it remains to show the implication from left to right.

To this end, let $b \geq b_0$ and assume $(r, b) \in \mathrm{TrOff}_{v^*}(\mathcal{G})$. We show $(r, b_0) \in \mathrm{TrOff}_{v^*}(\mathcal{G})$. Let σ be a strategy for Player 0 such that $(r, b) \in \mathrm{TrOff}_{v^*}(\sigma)$. Such a strategy exists due to definition of $\mathrm{TrOff}_{v^*}(\mathcal{G})$.

Since every play ρ starting in v^* that is consistent with σ and that satisfies $\#_D(\rho) < r$ furthermore satisfies $\mathrm{Cost}(\rho) \leq b_0$, the strategy σ is winning from v^* for Player 0 even if less than r disturbances occur. Thus, we have $r_{\mathcal{G}}(v^*) \geq r$.

Since $\mathcal{G}$ clearly satisfies the preconditions of ➔Theorem 5.23, we can effectively compute an optimally resilient strategy σ' for Player 0 in $\mathcal{G}$. Recall that optimally resilient strategies are not larger than winning strategies for Player 0 in the underlying game due to ➔Corollary 5.24. Hence, we can assume that σ' is a finite-state strategy of size at most $d(6n)(d+1)(W+1) = s$ due to ➔Lemma 3.36.

➔Sec. 5.2, Page 168

➔Sec. 5.2, Page 170

➔Sec. 3.4, Page 72

Now, let ρ be a play starting in v^* that is consistent with σ' with $\#_D(\rho) < r$. Since σ' is optimally resilient and since we have $r_{\mathcal{G}}(v^*) \geq r$, we obtain that ρ is winning for Player 0. Furthermore, due to the same arguments underlying the proof of ➔Lemma 3.39, we obtain $\mathrm{Cost}(\rho) \leq nWs = b_0$. □ ➔Sec. 3.5, Page 75

A lower bound for the stabilization of the sequence $r_0, r_1, r_2, \ldots$ follows directly from the lower bound on the cost that Player 0 can ensure given in ➔Lemma 3.40. ➔Sec. 3.5, Page 77

Corollary 6.12. *For each $n > 0$ and each $W \geq 0$ there exists a parity game with weights $\mathcal{G}_{n,W}$ with $|\mathcal{G}_{n,W}| \in \mathcal{O}(n \log W)$ containing a vertex v such that for each $b \in \mathbb{N}$ and each $r \in \omega$ we have $(b, r) \in \mathit{TrOff}_{v^*}(\mathcal{G})$ if and only if either $r = 0$ or if $b \geq (n-1)W$.*

Due to the Lemma 6.11 and as argued above, it suffices to compute r_b for exponentially many values b in order to determine $\mathrm{TrOff}_{v^*}(\mathcal{G})$. Each of these computations requires exponential time due to Lemma 6.10, which yields the following result.

Theorem 6.13. *The following problem can be solved in exponential time:*

"Given a parity game with weights $\mathcal{G}$ and a vertex v^* of $\mathcal{G}$, compute $\mathit{TrOff}_{v^*}(\mathcal{G})$."

Proof. We have argued above that, in order to obtain $\mathrm{TrOff}_{v^*}(\mathcal{G})$, it suffices to compute the values r_b for $b \in \{0, \ldots, d(6n^2)(d+1)W(W+1)\}$. Each such r_b can be computed in exponential time due to Lemma 6.10. Thus, we compute $\mathrm{TrOff}_{v^*}(\mathcal{G})$ by computing exponentially many r_b, each of which can be computed in exponential time. □

We now show that in the special case of parity games with costs, we are even able to compute the set $\mathrm{TrOff}_{v^*}(\mathcal{G})$ in polynomial space. Hence, for the remainder of this section, assume that $\mathcal{G}$ is a parity game with costs.

To compute $\mathrm{TrOff}_{v^*}(\mathcal{G})$ in polynomial space, we first observe that the exponential runtime of computing r_b stated in Lemma 6.10 results from the ExpTime-membership of the threshold problem for parity games with weights due to Theorem 4.12. As we are, however, able to solve the threshold problem for parity games with costs in polynomial space due to ➔Theorem 4.26, we obtain the following corollary of Lemma 6.10. ➔Sec. 4.2, Page 115

Corollary 6.14. *If $\mathcal{G}$ is a parity game with costs, then for each $b \in \mathbb{N}$ the value r_b can be computed in polynomial space.*

Secondly, we note that we can decrease the number of values r_b computed in the proof of Theorem 6.13. To this end, we observe that in that proof we compute r_b for each $b \in \{0, \ldots, d(6n^2)(d+1)W(W+1)\}$, which suffices to compute $\mathrm{TrOff}_{v^*}(\mathcal{G})$ due to Lemma 6.11. In the proof of that lemma, however, we only leverage an upper bound of $s = d(6n)(d+1)(W+1)$ on the size of winning strategies for Player 0 in parity games with weights due to Lemma 3.36. In the special case of parity games with costs, in contrast, positional strategies suffice for Player 0 to win due to ➔Proposition 2.34.1. ➔Sec. 2.4, Page 26 Hence, analogous reasoning to the proof of Lemma 6.11 yields the following result.

Corollary 6.15. *Let $b_0 = nW$. If $\mathcal{G}$ is a parity game with costs, then for all $b \geq b_0$ we have $r_b = r_{b_0}$.*

Recall that we directly obtain a lower bound matching the upper bound given in Corollary 6.15 from Corollary 6.12

Plugging Corollary 6.14 and Corollary 6.15 into the proof of Theorem 6.13, however, does not yield the desired algorithm computing $\mathrm{TrOff}_{v^*}(\mathcal{G})$ in polynomial space: We still compute the value r_b for each $b \in \{0, \dots, nW\}$, i.e., we compute an exponential number of such r_b.

We now show that it indeed suffices to compute polynomially many such values in order to obtain $\mathrm{TrOff}_{v^*}(\mathcal{G})$. To this end, we first remark that the domain of the r_b is bounded due to the boundedness of the resilience as shown in →Corollary 5.19, Page 164, and due to Corollary 6.8.

Remark 6.16. *For each $b \in \mathbb{N}$ we have $r_b \in \{0, \dots, n-1, \omega, \omega+1\}$.*

Moreover, for each $r \in \{0, \dots, n-1, \omega, \omega+1\}$, we can determine the minimal b such that $r_b = r$ via a binary search over the range $b \in \{0, \dots, nW\}$ due to monotonicity of $\mathrm{TrOff}_{v^*}(\mathcal{G})$ in the second component as stated in Remark 6.4. This binary search determines the minimal such b via computing at most $\log(nW)$ many values r_b, each of which can be computed in polynomial time due to Corollary 6.14.

Furthermore, again due to Remark 6.4, computing the minimal b for each $r \in \{0, \dots, n-1, \omega, \omega+1\}$ suffices to obtain $\mathrm{TrOff}_{v^*}(\mathcal{G})$ due to similar reasoning as for the proof of Theorem 6.13. Hence, we obtain $\mathrm{TrOff}_{v^*}(\mathcal{G})$ by performing polynomially many binary searches, each of which requires polynomial space. This yields the following result.

Theorem 6.17. *The following problem can be solved in polynomial space:*

"Given a parity game with costs $\mathcal{G}$ and a vertex v^* of $\mathcal{G}$, compute $\mathrm{TrOff}_{v^*}(\mathcal{G})$."

The problem of computing $\mathrm{TrOff}_{v^*}(\mathcal{G})$ subsumes solving the threshold game for any given bound b for $\mathcal{G}$. As we have already shown the threshold game for parity games with costs to be PSpace-hard in →Theorem 4.31, Page 123, Theorem 6.17 settles the complexity of computing $\mathrm{TrOff}_{v^*}(\mathcal{G})$: The decision problem associated with computing that optimal bound is ExpTime-hard for parity games with weights due to →Theorem 4.38, Page 130, while it is PSpace-hard for the special case of parity games with costs due to →Theorem 4.31, Page 123.

To conclude this section, we turn our attention to computing strategies witnessing membership of a given $(b, r) \in \mathrm{TrOff}_{v^*}(\mathcal{G})$ in that set of tradeoffs. To this end, recall that, for each $b \in \mathbb{N}$, we defined $\mathcal{G}_b = (\mathcal{A}, \textsc{WeightParity}_b(\Omega, \mathrm{Weight}))$. Furthermore, due to Corollary 6.8 we obtain that for an optimally resilient strategy σ for Player 0 in $\mathcal{G}_b$ we have

$$\mathrm{TrOff}_{v^*}(\sigma) \cap ((\omega+2) \times \{b\}) = \mathrm{TrOff}_{v^*}(\mathcal{G}) \cap ((\omega+2) \times \{b\}) \ ,$$

i.e., the strategy σ_b witnesses $(r, b) \in \mathrm{TrOff}_{v^*}(\mathcal{G})$. Since each $\mathcal{G}_b$ satisfies the preconditions required for effectively computing optimally resilient strategies in →Section 5.2.3, Page 168, we are thus able to effectively compute a strategy witnessing $(r, b) \in \mathrm{TrOff}_{v^*}(\mathcal{G})$.

Remark 6.18. *Let $(r, b) \in TrOff_{v^*}(\mathcal{G})$. A strategy σ for Player 0 such that $(r, b) \in TrOff_{v^*}(\sigma)$ is effectively computable*

1. *in polynomial space, if $\mathcal{G}$ is a parity game with costs, and*
2. *in exponential time, otherwise.*

Furthermore, the strategy σ is at most of exponential size in $|\mathcal{G}|$.

This concludes our investigation of the tradeoff between optimality and resilience in parity games with weights and their special cases. We have defined the notion of a tradeoff between optimality and resilience in a parity game with weights and we have shown how to determine the tradeoffs of such a given game. Moreover, we have shown how to construct strategies witnessing the existence of a given tradeoff. In the following section, we summarize the results obtained in this chapter.

6.3 Summary of Results

In this section we have investigated tradeoffs between the quantitative metrics of strategies for Player 0 in parity games with weights, namely the size of the strategy, the cost it guarantees as discussed in Chapter 4, and its resilience against unmodeled intermittent disturbances as discussed in Chapter 5. To this end, we have investigated the tradeoff between the memory required to implement a strategy and the bound on the cost that strategy guarantees in a parity game with weights in Section 6.1 as well as the tradeoff between that bound on the cost and the resilience of a strategy against disturbances in Section 6.2. We have not investigated the tradeoff between the the amount of memory required in order to implement a strategy and the resilience of that strategy against intermittent disturbances, since we have already shown in ➔Corollary 5.24 that constructing optimally resilient strategies comes "for free" in terms of memory requirements.

➔Sec. 5.2, Page 170

First, we have shown in ➔Theorem 6.2 that Player 0 may, in general, relax the bound she guarantees on the cost of the play in exchange for smaller strategies realizing this increased bound on the cost. Dually, we have shown in ➔Theorem 6.3 that Player 1 has, in general, a similar choice: By gradually decreasing the bound on the cost that he has to violate, the memory required to implement such strategies decreases as well. We have, however, only shown that such tradeoffs are possible in general by providing and analyzing a sequence of parity games with weights exhibiting such tradeoffs. It remains open for future work how to determine the existence of such tradeoffs for a given parity game with weights.

➔Sec. 6.1, Page 176

➔Sec. 6.1, Page 180

Second, we have shown in Section 6.2 how to determine the possible tradeoffs between the cost of a play and the resilience of the strategy guaranteeing that cost in a parity game with weights. We have shown that computing these possible tradeoffs is as hard as solving the underlying game optimally, i.e., it is possible in exponential time for parity games with weights due to ➔Theorem 6.13, and it is possible in polynomial space for the special case of parity games with costs due to ➔Theorem 6.17. Furthermore, we have argued that strategies witnessing the above tradeoffs can be effectively

➔Sec. 6.2, Page 185

➔Sec. 6.2, Page 186

computed in exponential time for parity games with weights and in polynomial space for parity games with costs in ➔Remark 6.18.

➔Sec. 6.2, Page 187

In the following chapter, we summarize the results obtained in this thesis and discuss the problems left open for future work as well as possible extensions of this work.

CHAPTER 7

Conclusion and Outlook

In this chapter, we summarize the results obtained in this thesis, discuss their contribution, and give an overview over open problems and possible future work.

We have begun this work in ➔Chapter 3 by generalizing the model of parity games with costs to parity games with weights in ➔Section 3.1. While the existing model was quite restrictive with respect to its cost model, in that it only allowed nonnegative weights, parity games with weights allow arbitrary integer weights. Thus, they allow to model, e.g., charging and draining some resource attached to the modeled system. Hence, this novel model allows for modeling more complex systems than previous ones: The model of parity games with weights features greatly increased expressiveness when compared with the previously existing model of parity games with costs.

➔Page 29
➔Page 31

Following these basic definitions we have shown that the problem of solving parity games with weights is in NP ∩ coNP in ➔Theorem 3.18 and that it is polynomial-time equivalent to the problem of solving energy parity games in ➔Theorem 3.31. We have moreover shown that exponential memory suffices for Player 0 to implement a winning strategy in parity games with weights in ➔Theorem 3.34.1, but also that exponential memory is, in general, necessary for her to do so. For Player 1, the requirement of infinite memory for winning strategies for him follows from parity games with weights generalizing finitary parity games. Finally, we proved that if Player 0 wins a parity game with weights from some vertex v, then she has a strategy with at most exponential cost from v in ➔Theorem 3.38. That theorem moreover shows that there exists an asymptotically matching lower bound on the cost that Player 0 can ensure. We have summarized the results on parity games with weights and compare them with the special cases of parity games, finitary parity games, and parity games with costs in ➔Table 3.17 which we partially reprint as Table 7.1 for the sake of completeness.

➔Sec. 3.2, Page 52
➔Sec. 3.3, Page 69
➔Sec. 3.4, Page 71
➔Sec. 3.5, Page 75
➔Sec. 3.6, Page 79

In addition to greatly increasing the expressiveness of parity games and their vari-

	Complexity	Mem. Pl. 0/Pl. 1	Bounds
Parity Games	UP ∩ coUP quasi-poly.	pos./pos.	–
Finitary Parity Games	PTime	pos./inf.	$\mathcal{O}(nW)$
Parity Games with Costs	UP ∩ coUP quasi-poly.	pos./inf.	$\mathcal{O}(nW)$
Parity Games with Weights	NP ∩ coNP pseudo-quasi-poly.	$\mathcal{O}(nd^2W)$/inf.	$\mathcal{O}((ndW)^2)$

Table 7.1: Properties of the boundedness problem for variants of parity games.

ants, the parity condition with weights also allows ordering plays by the cost that they incur. So far, we have only considered the boundedness problem, i.e., the question whether, given some parity game with weights $\mathcal{G}$ and a vertex v of $\mathcal{G}$ whether there exists a strategy with finite cost from v. Such a strategy, however, does not necessarily suffice if one wants to determine, say, the minimal amount of some costly resource required to satisfy a specification. Thus, we have investigated the threshold problem for parity games with weights, i.e., the problem whether, given some parity game with weights $\mathcal{G}$, a vertex v of $\mathcal{G}$, and a bound $b \in \mathbb{N}$, Player 0 has a strategy of cost at most b from v.

→ Page 83
→ Sec. 4.1, Page 96
→ Sec. 4.3, Page 130
→ Sec. 4.2, Page 115
→ Sec. 4.3, Page 123
→ Sec. 4.4, Page 131
→ Sec. 4.4, Page 132
→ Sec. 4.6, Page 148

In → Chapter 4 we have shown the threshold problem to be ExpTime-complete for parity games with weights in → Theorem 4.12 and → Theorem 4.38, and we have shown the problem to be PSpace-complete for the special cases of parity games with costs and finitary parity games in → Theorem 4.26 and → Theorem 4.31. From the proof of ExpTime-membership, it moreover follows that Player 0, if she has a strategy of cost at most b from some vertex in a parity game with weights $\mathcal{G}$ with n vertices and with d odd colors, then she also has one of size in $\mathcal{O}((b^2+b)^d)$ (see → Lemma 4.39). Dually, if Player 1 has a strategy of cost greater than b in $\mathcal{G}$, then he has a strategy of size in $\mathcal{O}(n(b^2+b)^d)$ (see → Corollary 4.40). We have summarized our results on the threshold problem for parity games with weights and its special cases in → Table 4.24 which we reprint as Table 7.2 for the sake of completeness.

Thus, our results show that solving the threshold problem is harder than solving the boundedness problem. Moreover, for parity games with costs and finitary parity games, Player 0 requires exponential memory in order to play optimally, while positional strategies suffice for her to win. Thus, playing optimally comes at a price even in finitary parity games. As a consequence, when modeling a system as a parity game with weights, one should consider the implications of asking for optimal strategies: Depending on the scenario at hand, the benefits of a smaller strategy that can be computed faster may outweigh the benefits of a strategy that minimizes the cost of consistent plays.

After having thus enabled modeling of resources in the model of parity games, we

	Complexity	Mem. Pl. 0/Pl. 1
Finitary Parity Games	PSpace-c.	exp./exp.
Parity Games with Costs	PSpace-c.	exp./exp.
Parity Games with Weights	ExpTime-c.	exp./exp.

Table 7.2: Properties of the threshold problem for variants of parity games.

subsequently turned our attention to modeling disturbances that may occur when the modeled system is deployed in the real world in → Chapter 5. Here, we follow the approach of Dallal, Neider, and Tabuada [DNT16], and model disturbances as additional edges in infinite games, obtaining infinite games with disturbances. These disturbance edges are controlled by neither player, but are instead assumed to be rare events that occur nondeterministically. Their inclusion in the model of infinite games alleviates the burden on the designer of a game in that they allow modeling uncertainty about the result of actions, e.g., due to malfunctioning actuators, as discussed in Chapter 5. → Page 149

In games with disturbances, the question of solving a game is generalized: Instead of determining whether Player 0 has a winning strategy from some vertex, one instead asks for the maximal number of disturbances r that may occur while still allowing Player 0 to win the resulting play. We have shown that computing r is as hard as solving the underlying game without disturbances in → Theorem 5.23. Moreover, we have argued in the proof of that theorem that an optimally resilient strategy witnessing that Player 0 can win even under r disturbances can be computed effectively together with the computation of r and that such a strategy is not larger than a winning strategy in the underlying game without disturbances. Thus, resilience against intermittent disturbances can be obtained for free. Hence, if there exists a reasonable model of disturbances that may occur during the deployment of the system, there is no downside to computing strategies that are aware of such disturbances and that are resilient against the maximal number of them. → Sec. 5.2, Page 168

The models introduced in this work induce different metrics of quality for strategies, namely, its size, the cost of consistent plays that it guarantees, and its resilience. In → Chapter 6 we have considered the tradeoffs between these different measures of quality. We have shown that, in general, Player 0 can trade memory for optimality in → Theorem 6.2. That theorem shows that there exists a gradual tradeoff for her between the achieved upper bound b on the cost of a play and the amount of memory required to implement a strategy witnessing b. Moreover, we have shown that Player 0 can trade resilience for optimality and that computing the possible tradeoffs for her is as hard as solving the threshold problem for the underlying game in → Theorem 6.17. → Page 173 → Sec. 6.1, Page 176 → Sec. 6.2, Page 186

Open Problems and Future Work

In this work, we have extended the standard model of parity games with quantitative features that allow, e.g., to model charging and draining some resource, and with

capabilities for dealing with malfunctioning actuators and other disturbances during deployment. While we have studied the decision and optimization problems associated with these models exhaustively, there remain some technical open problems. We detail and discuss these problems in the following.

Definition of Cost in Parity Games with Weights After extending the model of parity games with costs to parity games with weights by allowing negative weights, we lifted the definition of the cost-of-response of a request for some color to the extended setting in ➔Section 3.1. To this end, we defined the cost-of-response as the amplitude of the shortest infix of the play starting at the given request and ending at an answer to that request. While, in our opinion, this definition naturally extends the cost-of-response to the setting of integer weights, other definitions of this notion are feasible. Such variations include, e.g., only considering the accumulated cost at the end of the shortest infix ending at an answer to the request, or by disallowing Player 0 to accumulate negative weight. We have detailed these alternatives in ➔Section 3.6. It remains open how to solve the associated decision problems for these alternative definitions.

➔Page 31

➔Page 78

Lower Bound on Solving Parity Games with Weights We have shown the boundedness problem for parity games with weights to be a member of NP ∩ coNP in ➔Theorem 3.18 and we have shown that it is as hard as the problem of solving energy parity games in ➔Theorem 3.31. There have been a number of decision problems that were only known to be in NP ∩ coNP for a long time, before being shown to in fact be in PTime. Thus, we conjecture the boundedness problem for parity games with weights to be in PTime as well.

➔Sec. 3.2, Page 52

➔Sec. 3.3, Page 69

Multidimensional Parity Games with Weights In this work we have only considered games with a single coloring and a single weight function. In contrast, Bruyère, Hautem, and Randour [BHR16] have shown the boundedness problem for finitary parity games with multiple colorings to be ExpTime-complete. It remains for future work to extend our results on the boundedness problem and for the threshold problem for parity games with costs to the extended setting with multiple coloring and multiple weight functions.

Threshold Problem for Parity Games with Weights with Unary Encoding Unless indicated otherwise, we have assumed the weight functions to be given in binary encoding in this work. Describing these functions in unary encoding induces an exponential blowup in the size of the game and could thus potentially decrease the complexity of the associated decision problems. We have argued in ➔Section 3.6 that unary encoding does not change our results on the complexity of the boundedness problem. Furthermore, we have shown in ➔Section 4.5 that the complexity of the threshold problem remains unchanged for parity games with costs whether the games are given in unary or binary encoding. For parity games with weights, however, there exists a gap: If the weights are given in unary encoding, ➔Theorem 4.12 yields ExpTime-membership of the threshold

➔Page 78

➔Page 143

➔Sec. 4.1, Page 96

problem, while our strongest lower bound is PSpace-hardness of the problem due to ➔Theorem 4.31. It remains open whether this problem is ExpTime-hard or in PSpace. ➔Sec. 4.3, Page 123

Minimal Strategies for Threshold Problem We have shown in ➔Theorem 6.2 that, in general, there exists a gradual tradeoff between cost and memory for Player 0 even in finitary parity games, i.e., Player 0 may decrease the amount of memory required for her to implement a strategy by allowing for costlier plays consistent with that strategy. Thus far, however, the only approach to finding a minimal strategy with a given cost from a given vertex consists of brute force, which has a prohibitive complexity that we conjecture to be far from optimal. It remains open to find an optimal approach to the problem of computing such a strategy. A starting point may be the approach of bounded synthesis [FS13], which reduces the problem to that of satisfiability for some underlying logic, and has shown to be applicable and feasible in practice for a wide range of benchmarks. ➔Sec. 6.1, Page 176

Infinite Arenas We have exclusively discussed the setting of finite arenas in this work. There does, however, also exist work of games of infinite duration played on infinite arenas [CF13], e.g., on arenas that are induced by a pushdown system [Wal01, LMS04, FZ12, Fri13, CH18]. It remains a problem for future work to investigate the problems of solving parity games with weights or games with disturbances on infinite arenas.

In this work we have exhaustively studied quantitative extensions of parity games. These extensions allow modeling resources that may be charged and drained and they allow taking disturbances, such as malfunctions of actuators, into account when modeling reactive systems. We have moreover investigated and discussed the tradeoffs occurring between the different metrics of quality of strategies in such games.

Classical qualitative parity games form the technical core for a multitude of techniques that have been successfully applied in the fields of program analysis and program synthesis. Thus, we hope that our work paves the way for practically applicable methods in those fields that allow not only to express functional requirements, but that also support modeling resources and other quantitative aspects of a system, and that are aware of malfunctions and other disturbances that may occur during deployment of the system.

Bibliography

[AAE04] Paul C. Attie, Anish Arora, and E. Allen Emerson. Synthesis of fault-tolerant concurrent programs. *ACM Transactions on Programming Languages and Systems*, 26(1):125–185, 2004.

[BBF+18] Giovanni Bacci, Patricia Bouyer, Uli Fahrenberg, Kim Guldstrand Larsen, Nicolas Markey, and Pierre-Alain Reynier. Optimal and robust controller synthesis - using energy timed automata with uncertainty. In Klaus Havelund, Jan Peleska, Bill Roscoe, and Erik P. de Vink, editors, *FM 2018*, volume 10951 of *LNCS*, pages 203–221. Springer, 2018.

[BCD+11] Lubos Brim, Jakub Chaloupka, Laurent Doyen, Raffaella Gentilini, and Jean-François Raskin. Faster algorithms for mean-payoff games. *Formal Methods in System Design*, 38(2):97–118, 2011.

[BCG+14] Roderick Bloem, Krishnendu Chatterjee, Karin Greimel, Thomas A. Henzinger, Georg Hofferek, Barbara Jobstmann, Bettina Könighofer, and Robert Könighofer. Synthesizing robust systems. *Acta Informatica*, 51(3-4):193–220, 2014.

[BCHJ09] Roderick Bloem, Krishnendu Chatterjee, Thomas A. Henzinger, and Barbara Jobstmann. Better quality in synthesis through quantitative objectives. In Ahmed Bouajjani and Oded Maler, editors, *CAV 2009*, volume 5643 of *LNCS*, pages 140–156. Springer, 2009.

[BEJK14] Roderick Bloem, Rüdiger Ehlers, Swen Jacobs, and Robert Könighofer. How to handle assumptions in synthesis. In Krishnendu Chatterjee, Rüdiger Ehlers, and Susmit Jha, editors, *SYNT 2014*, volume 157 of *EPTCS*, pages 34–50, 2014.

[BFL+08] Patricia Bouyer, Ulrich Fahrenberg, Kim Guldstrand Larsen, Nicolas Markey, and Jirí Srba. Infinite runs in weighted timed automata with

energy constraints. In Franck Cassez and Claude Jard, editors, *FORMATS 2008*, volume 5215 of *LNCS*, pages 33–47. Springer, 2008.

[BFRR17] Véronique Bruyère, Emmanuel Filiot, Mickael Randour, and Jean-François Raskin. Meet your expectations with guarantees: Beyond worst-case synthesis in quantitative games. *Information and Computation*, 254:259–295, 2017.

[BGH+15] Thomas Brihaye, Gilles Geeraerts, Axel Haddad, Benjamin Monmege, Guillermo A. Pérez, and Gabriel Renault. Quantitative games under failures. In *FSTTCS 2015*, volume 45 of *LIPIcs*, pages 293–306. Schloss Dagstuhl - LZI, 2015.

[BHR16] Véronique Bruyère, Quentin Hautem, and Mickael Randour. Window parity games: an alternative approach toward parity games with time bounds. In Domenico Cantone and Giorgio Delzanno, editors, *GandALF 2016*, volume 226 of *EPTCS*, pages 135–148, 2016.

[BJP+12] Roderick Bloem, Barbara Jobstmann, Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. Synthesis of Reactive(1) designs. *Journal of Computer and System Sciences*, 78(3):911–938, 2012.

[BL69] J. Richard Büchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969.

[BMR+18] Patricia Bouyer, Nicolas Markey, Mickael Randour, Kim G. Larsen, and Simon Laursen. Average-energy games. *Acta Informatica*, 55(2):91–127, 2018.

[BRS17] Romain Brenguier, Jean-François Raskin, and Ocan Sankur. Assume-admissible synthesis. *Acta Informatica*, 54(1):41–83, 2017.

[CD12] Krishnendu Chatterjee and Laurent Doyen. Energy parity games. *Theoretical Computer Science*, 458:49–60, 2012.

[CdAHS03] Arindam Chakrabarti, Luca de Alfaro, Thomas A. Henzinger, and Mariëlle Stoelinga. Resource interfaces. In Rajeev Alur and Insup Lee, editors, *EMSOFT 2003*, volume 2855 of *LNCS*, pages 117–133. Springer, 2003.

[CDF+18] Wojciech Czerwiński, Laure Daviaud, Nathanaël Fijalkow, Marcin Jurdziński, Ranko Lazić, and Paweł Parys. Universal trees grow inside separating automata: Quasi-polynomial lower bounds for parity games. *arXiv*, abs/1807.10546, 2018. Available at `http://arxiv.org/abs/1807.10546`.

[CF13] Krishnendu Chatterjee and Nathanaël Fijalkow. Infinite-state games with finitary conditions. In Simona Ronchi Della Rocca, editor, *CSL 2013*, volume 23 of *LIPIcs*, pages 181–196. Schloss Dagstuhl - LZI, 2013.

[CH06] Krishnendu Chatterjee and Thomas A. Henzinger. Finitary winning in omega-regular games. In Holger Hermanns and Jens Palsberg, editors, *TACAS 2006*, volume 3920 of *LNCS*, pages 257–271. Springer, 2006.

[CH18] Arnaud Carayol and Matthew Hague. Optimal strategies in pushdown reachability games. In Igor Potapov, Paul G. Spirakis, and James Worrell, editors, *MFCS 2018*, volume 117 of *LIPIcs*, pages 42:1–42:14. Schloss Dagstuhl - LZI, 2018.

[CHH09] Krishnendu Chatterjee, Thomas A. Henzinger, and Florian Horn. Finitary winning in omega-regular games. *ACM Transactions on Computional Logic*, 11(1):1:1–1:27, 2009.

[CHJ05] Krishnendu Chatterjee, Thomas A. Henzinger, and Marcin Jurdziński. Mean-payoff parity games. In *LICS 2005*, pages 178–187. IEEE Computer Society, 2005.

[Chu57] Alonzo Church. Applications of recursive arithmetic to the problem of circuit synthesis–summaries of talks. *Institute for Symbolic Logic, Cornell University*, 1957.

[CJK+17] Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasipolynomial time. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *STOC 2017*, pages 252–263. ACM, 2017.

[CKS81] Ashok K. Chandra, Dexter Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

[DJL18] Laure Daviaud, Marcin Jurdziński, and Ranko Lazić. A pseudo-quasi-polynomial algorithm for mean-payoff parity games. In Anuj Dawar and Erich Grädel, editors, *LICS 2018*, pages 325–334. ACM, 2018.

[DNT16] Eric Dallal, Daniel Neider, and Paulo Tabuada. Synthesis of safety controllers robust to unmodeled intermittent disturbances. In *CDC 2016*, pages 7425–7430. IEEE, 2016.

[EJ91] Ernest Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *FOCS 1991*, pages 368–377. IEEE Computer Society, 1991.

[EJS93] E. Allen Emerson, Charanjit S. Jutla, and A. Prasad Sistla. On model-checking for fragments of μ-calculus. In Costas Courcoubetis, editor, *CAV 1993*, volume 697 of *LNCS*, pages 385–396. Springer, 1993.

[EKA08] Ali Ebnenasir, Sandeep S. Kulkarni, and Anish Arora. FTSyn: a framework for automatic synthesis of fault-tolerance. *International Journal on Software Tools for Technology Transfer*, 10(5):455–471, 2008.

[EM79] Andrzej Ehrenfeucht and Jan Mycielski. Positional strategies for mean payoff games. *International Journal of Game Theory*, 8:109–113, 1979.

[ET14] Rüdiger Ehlers and Ufuk Topcu. Resilience to intermittent assumption violations in reactive synthesis. In Martin Fränzle and John Lygeros, editors, *HSCC 2014*, pages 203–212. ACM, 2014.

[FJS+17] John Fearnley, Sanjay Jain, Sven Schewe, Frank Stephan, and Dominik Wojtczak. An ordered approach to solving parity games in quasi polynomial time and quasi linear space. In Hakan Erdogmus and Klaus Havelund, editors, *SPIN 2017*, pages 112–121. ACM, 2017.

[Fri09] Oliver Friedmann. An exponential lower bound for the parity game strategy improvement algorithm as we know it. In *LICS 2009*, pages 145–156. IEEE Computer Society, 2009.

[Fri13] Wladimir Fridman. *A study of pushdown games*. PhD thesis, RWTH Aachen University, 2013.

[FS13] Bernd Finkbeiner and Sven Schewe. Bounded synthesis. *International Journal on Software Tools for Technology Transfer*, 15(5-6):519–539, 2013.

[FZ12] Wladimir Fridman and Martin Zimmermann. Playing pushdown parity games in a hurry. In Marco Faella and Aniello Murano, editors, *GandALF 2012*, volume 96 of *EPTCS*, pages 183–196, 2012.

[FZ14] Nathanaël Fijalkow and Martin Zimmermann. Parity and streett games with costs. *Logical Methods in Computer Science*, 10(2), 2014.

[GR09] Alain Girault and Éric Rutten. Automating the addition of fault tolerance with discrete controller synthesis. *Formal Methods in System Design*, 35(2):190–225, 2009.

[GTW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *LNCS*. Springer, 2002.

[HMU01] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 2001.

[HPSW16] Chung-Hao Huang, Doron A. Peled, Sven Schewe, and Farn Wang. A game-theoretic foundation for the maximum software resilience against dense errors. *IEEE Transactions on Software Engineering*, 42(7):605–622, 2016.

[JL17] Marcin Jurdziński and Ranko Lazić. Succinct progress measures for solving parity games. In *LICS 2017*, pages 1–9. IEEE Computer Society, 2017.

[JLS08] Marcin Jurdziński, François Laroussinie, and Jeremy Sproston. Model checking probabilistic timed automata with one or two clocks. *Logical Methods in Computer Science*, 4(3), 2008.

[Jur98] Marcin Jurdziński. Deciding the winner in parity games is in UP∩co-UP. *Information Processing Letters*, 68(3):119–124, 1998.

[Leh18] Karoliina Lehtinen. A modal μ perspective on solving parity games in quasi-polynomial time. In Anuj Dawar and Erich Grädel, editors, *LICS 2018*, pages 639–648. ACM, 2018.

[LMS04] Christof Löding, P. Madhusudan, and Olivier Serre. Visibly pushdown games. In Kamal Lodaya and Meena Mahajan, editors, *FSTTCS 2004*, volume 3328 of *LNCS*, pages 408–420. Springer, 2004.

[Mar75] Donald A. Martin. Borel determinacy. *Annals of Mathematics*, 102(2):363–371, 1975.

[MMS15] Fabio Mogavero, Aniello Murano, and Loredana Sorrentino. On promptness in parity games. *Fundamenta Informaticae*, 139(3):277–305, 2015.

[Mos91] Andrzej Wlodzimierz Mostowski. Games with forbidden positions. Technical report, Instytut Matematyki, Uniwersytet Gdański, 1991.

[MRT13] Rupak Majumdar, Elaine Render, and Paulo Tabuada. A theory of robust omega-regular software synthesis. *ACM Transactions on Embedded Computing Systems*, 13(3):48:1–48:27, 2013.

[NRY96] Anil Nerode, Jeffrey B. Remmel, and Alexander Yakhnis. Mcnaughton games and extracting strategies for concurrent programs. *Annals of Pure and Applied Logic*, 78(1-3):203–242, 1996.

[NWZ18a] Daniel Neider, Alexander Weinert, and Martin Zimmermann. Robust, expressive, and quantitative linear temporal logics. *arXiv*, 2018. Available at `http://arxiv.org/abs/1808.09028`.

[NWZ18b] Daniel Neider, Alexander Weinert, and Martin Zimmermann. Synthesizing optimally resilient controllers. In Dan Ghika and Achim Jung, editors, *CSL 2018*. Schloss Dagstuhl - LZI, 2018.

[Pap94] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

[Pnu77] Amir Pnueli. The temporal logic of programs. In *FOCS 1977*, pages 46–57. IEEE Computer Society, 1977.

[SM73] Larry J. Stockmeyer and Albert R. Meyer. Word problems requiring exponential time: Preliminary report. In Alfred V. Aho, Allan Borodin,

Robert L. Constable, Robert W. Floyd, Michael A. Harrison, Richard M. Karp, and H. Raymond Strong, editors, *STOC 1973*, pages 1–9. ACM, 1973.

[SWZ18] Sven Schewe, Alexander Weinert, and Martin Zimmermann. Parity games with weights. In Dan Ghika and Achim Jung, editors, *CSL 2018*. Schloss Dagstuhl - LZI, 2018.

[TCRM14] Paulo Tabuada, Sina Yamac Caliskan, Matthias Rungger, and Rupak Majumdar. Towards robustness for cyber-physical systems. *IEEE Transactions on Automatic Control*, 59(12):3151–3163, 2014.

[TN16] Paulo Tabuada and Daniel Neider. Robust linear temporal logic. In *CSL 2016*, volume 62 of *LIPIcs*, pages 10:1–10:21. Schloss Dagstuhl - LZI, 2016.

[TOLM12] Ufuk Topcu, Necmiye Ozay, Jun Liu, and Richard M. Murray. On synthesizing robust discrete controllers under modeling uncertainty. In Thao Dang and Ian M. Mitchell, editors, *HSCC 2012*, pages 85–94. ACM, 2012.

[Tur37] Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London mathematical society*, 2(1):230–265, 1937.

[Wal01] Igor Walukiewicz. Pushdown processes: Games and model-checking. *Information and Computation*, 164(2):234–263, 2001.

[Wil01] Thomas Wilke. Alternating tree automata, parity games, and modal mcalculus. *Bulletin of the Belgian Mathematical Society*, 8(2):359, 2001.

[WZ17] Alexander Weinert and Martin Zimmermann. Easy to win, hard to master: Optimal strategies in parity games with costs. *Logical Methods in Computer Science*, 13(3), 2017.

[Zer13] Ernst Zermelo. Über eine Anwendung der Mengenlehre auf die Theorie des Schachspiels. In *Proc. Fifth Congress of Mathematicians, Vol. 2*, pages 501–504. Cambridge Press, 1913.

[ZP95] Uri Zwick and Mike Paterson. The complexity of mean payoff games. In Ding-Zhu Du and Ming Li, editors, *COCOON 1995*, volume 959 of *LNCS*, pages 1–10. Springer, 1995.