Weak Cost Automata over Infinite Trees

Michael T. Vanden Boom

Balliol College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy in Computer Science

Trinity Term 2012

Abstract

Cost automata are traditional finite state automata enriched with a finite set of counters that can be manipulated on each transition. Based on the evolution of counter values, a cost automaton defines a function from the set of structures under consideration to $\mathbb{N} \cup \{\infty\}$, modulo an equivalence relation $\approx$ that ignores exact values but preserves boundedness properties.

Historically, variants of cost automata have been used to solve problems in language theory such as the star height problem. They also have a rich theory in their own right as part of the theory of regular cost functions, which was introduced by Colcombet as an extension to the theory of regular languages. It subsumes the classical theory since a language can be associated with the function that maps every structure in the language to 0 and everything else to ∞; it is a strict extension since cost functions can count some behaviour within the input.

Regular cost functions have been previously studied over finite words and trees. This thesis extends the theory to infinite trees, where classical parity automata are enriched with a finite set of counters. Weak cost automata, which have priorities $\{0, 1\}$ or $\{1, 2\}$ and an additional restriction on the structure of the transition function, are shown to be equivalent to a weak cost monadic logic. A new notion of quasi-weak cost automata is also studied and shown to arise naturally in this cost setting. Moreover, a decision procedure is given to determine whether or not functions definable using weak or quasi-weak cost automata are equivalent up to $\approx$, which also proves the decidability of the weak cost monadic logic over infinite trees.

The semantics of these cost automata over infinite trees are defined in terms of cost-parity games which are two-player infinite games where one player seeks to minimize the counter values and satisfy the parity condition, and the other player seeks to maximize the counter values or sabotage the parity condition. The main contributions and key technical results involve proving that certain cost-parity games admit positional or finite-memory strategies.

These results also help settle the decidability of some special cases of long-standing open problems in the classical theory. In particular, it is shown that it is decidable whether a regular language of infinite trees is recognizable using a nondeterministic co-Büchi automaton. Likewise, given a Büchi or co-Büchi automaton as input, it is decidable whether or not there is a weak automaton recognizing the same language.

Acknowledgements

I would like to thank my supervisor Luke Ong for his guidance during my studies in Oxford, and for encouraging me to explore this interesting area of automata theory.

I would also like to thank Thomas Colcombet for inviting me to LIAFA in Paris, and for generously sharing his time and knowledge with me. These research visits enabled a fruitful collaboration and friendship with Denis Kuperberg, which I hope will continue in the future. My examiners, Michael Benedikt and Mikołaj Bojańczyk, made this last stage of the DPhil an enjoyable and constructive process, and I am thankful for their feedback and support of my work as well.

This thesis would not have been possible without financial support from Oxford University Press and Balliol College. Additional funding was provided by ANR 2007 JCJC 0051 JADE, ANR 2010 BLAN 0202 02 FREC, and the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement 259454.

As always, I am also thankful for my family and friends who gave me the confidence to pursue this DPhil in Oxford, and have supported me in my meandering academic journey over many years.

Contents

List of Figures and Tables

Chapter 1

Introduction

Cost automata are an extension of finite state machines with a finite set of counters. These counters are initialized to value 0 and then take on values from the natural numbers $\mathbb{N}$ based on counter operations (such as increment or reset) on each transition. The counters cannot be used to affect control flow, so there is no test that can be used to determine the next state based on the counter value.[1] Instead, the counters are used as a way to assign values to runs and input structures. Just as a traditional automaton defines a language based on the structures that are accepted by it, a cost automaton defines a function from the set of input structures to $\mathbb{N} \cup \{\infty\}$, based on the evolution of the counter values during accepting runs.

Historically, variants of cost automata have been used to answer challenging questions from language theory such as the star height problem. They also have a rich theory in their own right as part of the theory of regular cost functions introduced by Colcombet [Col09c].

In this chapter, we briefly review some motivating examples from language theory, and describe how cost automata fit into the theory of regular cost functions. We then outline the main contributions of this thesis.

1.1 Motivation from language theory

Consider the following question from language theory, known as the *finite power property*:

> Given a regular language L of finite words, is there $n \in \mathbb{N}$ such that $L^* = (L + \epsilon)^n$?

[1] It is important to note that cost automata are not like counter automata in the sense of Minsky [Min67], which allow a zero-test and in the one counter case are really a form of pushdown automaton.

The question is whether the concatenation of any finite sequence of words in L can actually be represented by some bounded iteration of words in L. This problem was raised by Brzozowski in 1966 and solved independently by Simon [Sim78] and Hashiguchi [Has79].

Let us consider a technique that uses automata with counters to decide whether or not L satisfies the finite power property. Since L is regular, we can start by looking at the finite state automaton $\mathcal{A}$ with $L = L(\mathcal{A})$. Recall that the language $L(\mathcal{A})$ accepted by $\mathcal{A}$ is defined as the set of words u for which there is a run of $\mathcal{A}$ on u starting in the initial state and ending in a final state. By adding new ϵ transitions from any final state to the initial state, we get an automaton $\mathcal{A}'$ that recognizes L^*. In fact, we can describe the finite power property in terms of runs of $\mathcal{A}'$: the finite power property is satisfied if and only if there is some bound n such that for all $u \in L^*$, there is a run of the automaton that takes the new ϵ edges at most n times.

We want to capture the fact that these new edges are costly, whereas the original edges can be taken for free. One way to do this is to endow the automaton with a counter that is initialized to value 0 and incremented each time these new costly edges are taken and left unchanged when the original transitions from $\mathcal{A}$ are used. A cost automaton with one counter that can be incremented or left unchanged, as in this example, is known as a *distance automaton*, a model introduced by Hashiguchi [Has82]. The cost of a run of this distance automaton is the highest value the counter achieves, and the cost of a word is the minimum value over all accepting runs. This means that $\mathcal{A}'$ not only recognizes the language L^*, but also defines a function that maps every word $u \in L^*$ to the minimum number of times the new edges are taken in an accepting run on u. We can rephrase the problem in terms of this function: the language L satisfies the finite power property if and only if there exists $n \in \mathbb{N}$ such that for all words $u \in L(\mathcal{A}')$ the function defined by $\mathcal{A}'$ has value at most n on u. A function like this which is bounded over all accepted words is said to be *limited*.

Formally, the *limitedness problem* for a class $\mathscr{C}$ of automata with counters, such as the distance automata described above, asks:

> Given an automaton $\mathcal{A}$ with counting features that is in the class $\mathscr{C}$, is there $n \in \mathbb{N}$ such that every word accepted by the automaton has an accepting run of value at most n? In other words, is the function defined by the automaton bounded over its domain (of accepted words)?

Returning to the example, we see that the construction described above is a reduction of the decidability of the finite power property to the decidability of the limitedness

problem for distance automata. Hashiguchi proved that the limitedness problem is decidable for the class of distance automata [Has82], so this implies the decidability of the finite power property.

A number of questions emerged that could be reduced to the limitedness of functions defined by automata with counters in a similar way as the finite power property. Historically, the most famous problem like this is the *star height problem* which was introduced by Eggan [Egg63]:

> Given a regular language L of finite words and $n \in \mathbb{N}$, is there a regular expression for L using concatenation, union, and Kleene star ($*$) operators with at most n nestings of Kleene stars?

For instance, the star height of the regular expression $(b + aa^*b)^*aa^*$ is 2. However, it turns out that the star height of this language is 1, since it can be defined by a simpler expression, namely $(a + b)^*a$.

Hashiguchi [Has88] gave a reduction of the star height problem to the limitedness of distance automata, providing the first decidability proof for this problem. Unfortunately, this reduction was complicated and the algorithm was of non-elementary complexity.

More recently, Kirsten [Kir05] provided an alternative proof by reducing the star height problem to a more powerful automaton model called *nested distance desert automata*. Kirsten proved that the limitedness problem for this class of automata is also decidable (in fact, PSPACE-complete). This proof showed that the star height problem is in $2^{2^{\mathcal{O}(m)}}$ space, where m is the number of states in the given nondeterministic automaton recognizing L.

The nested distance desert automata that Kirsten introduced can be viewed as cost automata with a set of ordered counters that can be incremented, reset, or left unchanged on each transition, but with an additional nesting condition based on the counter ordering: if some counter is incremented or reset then all lower counters in the ordering must be reset and all higher counters must be left unchanged. We now describe informally the construction for star height 0 and 1, and how these multiple counters and resets are used.

Let L be a regular language given by a finite state automaton. The construction starts by converting this automaton into a special canonical form called the universal automaton for L. This transformation is effective but may result in an increase in the size of the automaton. Let $\mathcal{A}$ be this universal automaton for L.

The language L is of star height 0 if and only if it is finite. This means that any accepting path through the graph of $\mathcal{A}$ has to be bounded by the number of states (i.e. there are no loops), so we simply add a single counter to $\mathcal{A}$, and increment this counter on each transition. This new automaton $\mathcal{A}_0$ is limited if and only if L has star height 0.

A language L of star height 1 can always be written as a finite union of languages of the form $a_1K_1^* \cdots a_jK_j^*a_{j+1}$, where each a_i is an individual letter and each K_i is a language of star height 0 (see [Kir05]). This means that an automaton $\mathcal{A}$ for L may have loops corresponding to the iteration of some K_i. The new automaton $\mathcal{A}_1$ uses multiple copies of $\mathcal{A}$. The idea is that $\mathcal{A}_1$ simulates $\mathcal{A}$, but uses copies of the automaton during loops. Within each copy, every transition increments counter 1, in order to ensure that this copy (if it is limited) can only define a language of star height 0. Entering and exiting these copies must happen on the same state, and results in a reset of counter 1. This represents the fact that iterating languages of star height 0 is allowed in a language of star height 1. Edges in the master copy increment counter 2 and can never be reset. This represents the fact that each $a_1K_1^* \cdots a_jK_j^*a_{j+1}$ can contain only a finite concatenation of star height 0 languages. Based on this construction, it turns out that $\mathcal{A}_1$ is limited if and only if L has star height at most 1. Testing for larger star heights requires additional nested counters, but works in a similar way.

We have already seen two problems (the finite power property and the star height problem) that can be reduced to questions of limitedness for cost automata. Problems like this are not restricted to language theory: there are applications of automata with counters in areas such as speech recognition [Moh97], database theory [BPR11], and verification [AKY08], but we do not focus on these applications in this thesis.

These problems are also not restricted to languages of finite words. Colcombet and Löding showed the star height problem is decidable for regular languages of finite trees by reduction to a question of boundedness for functions defined by tree automata with counters [CL08a]. One of the motivating examples for this thesis is a question about regular languages of infinite trees. Regular languages of infinite trees are recognizable by *nondeterministic parity automata*. The *parity acceptance condition* is commonly given by a mapping from states to a set of priorities $\{i, i+1, \ldots, j\}$, and a run is accepting if the maximum priority occurring infinitely often on each branch in the run is even. The *parity index* is the range of priorities used by the automaton, and provides a measure of how complicated the language is (in a similar way as the star

height provides a measure of complexity for languages of finite structures). The *nondeterministic parity index problem* for regular languages of infinite trees asks:

> Given a regular language L of infinite trees and $i \leq j$, is there a nondeterministic parity automaton using only priorities $\{i, i+1, \ldots, j\}$ that recognizes L?

In [CL08b], Colcombet and Löding provided a reduction of the parity index problem to a question of boundedness for cost-parity automata on infinite trees (tree automata that combine the parity acceptance condition with the counting features of cost automata). However, they were unable to show that limitedness was decidable for this entire class of cost automata over infinite trees, so the decidability of the parity index problem remained open. This open problem provides one motivation for the study of cost automata over infinite trees pursued in this thesis.

1.2 Theory of regular cost functions

The problems mentioned in the previous section inspired many lines of research over the past few decades, particularly on algebraic and topological techniques to reason about automata with counters (or more general weighted automata, see Section 2.4). Building on this work, Colcombet [Col09c] showed that a theory could be developed around the functions defined by cost automata that subsumes the theory of regular languages.

This classical theory of regular languages and finite state automata without counters dates back to early work by Kleene [Kle56] and Rabin and Scott [RS59]. Regular languages are fundamental in computer science because they are a robust class of languages that enjoys strong closure properties and multiple representations using finite state automata, regular expressions, logic, and algebra. Regular languages also benefit from good decidability properties, namely the decidability of language inclusion and language emptiness.

Instead of recognizing a language like a traditional automaton, a cost automaton defines a function based on the evolution of the counters during accepting runs. What is a natural (and useful) decision procedure for these functions, in analogy to deciding language inclusion or language emptiness for traditional automata? One possibility would be to assert that a function f defined by a cost automaton evaluates to a particular value $n \in \mathbb{N}$ for all inputs u in some domain $\mathscr{D}$ of structures ($f(u) = n$ for all $u \in \mathscr{D}$). This is immediately decidable for a large class of structures because

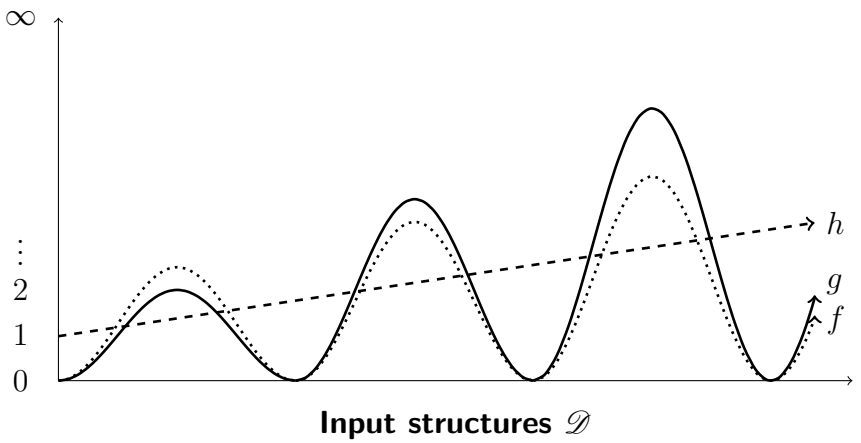

Figure 1.1. Functions such that $f \approx g$ but $f, g \not\approx h$.

we could encode this fixed bound in the state of a traditional automaton without counters, but it is not very interesting. Given two functions f and g, it would be more interesting to assert $f(u) = g(u)$, or $f(u) \leq g(u)$, across all inputs $u \in \mathscr{D}$. A result due to Krob [Kro94], however, implies that this is undecidable.

In order to recover decidability, Colcombet introduced the *domination preorder* (written $\preccurlyeq$) and the *boundedness relation* (written $\approx$) [Col09c]. These relations are weaker than $\leq$ and $=$, but retain information about boundedness which is useful for solving the sort of problems discussed in the previous section. These relations are defined over some domain $\mathscr{D}$ of input structures (usually labelled words or trees). We say a function $f : \mathscr{D} \to \mathbb{N} \cup \{\infty\}$ is *bounded* on some set $U \subseteq \mathscr{D}$ if there is some $n \in \mathbb{N}$ such that $f(u) \leq n$ for all $u \in U$. Given $f, g : \mathscr{D} \to \mathbb{N} \cup \{\infty\}$,

$$f \preccurlyeq g \text{ if for all } U \subseteq \mathscr{D}, \text{ if } g \text{ is bounded on } U \text{ then } f \text{ is bounded on } U.$$

Likewise, $f \approx g$ if $f \preccurlyeq g$ and $g \preccurlyeq f$. In other words,

$$f \approx g \text{ if for all } U \subseteq \mathscr{D}, f \text{ is bounded on } U \text{ if and only if } g \text{ is bounded on } U.$$

This means that f and g satisfying $f \approx g$ may not agree on exact values but do agree on boundedness properties across all subsets of the domain of input structures. This makes these relations perfectly suited for attacking problems of boundedness like those described in the previous section.

Consider the functions in Figure 1.1 (we assume the domain $\mathscr{D}$ of input structures is infinite and the functions continue in the expected way). On any subset $U \subseteq \mathscr{D}$ for which h is bounded (i.e. any finite set of input structures), f and g are also bounded. This means that $f \preccurlyeq h$ and $g \preccurlyeq h$. However, the infinite set of structures that has

output of value 0 on both f and g, has unbounded output value via h, so $h \not\preccurlyeq f$ and $h \not\preccurlyeq g$. On the other hand, $f \approx g$. This is true even though they do not agree on the exact values across all inputs and there are some inputs u with $f(u) < g(u)$ and some with $g(u) < f(u)$. From the perspective of boundedness properties, f and g are indistinguishable.

A *cost function* is defined as an equivalence class of functions under $\approx$, but we blur the distinction between a particular function and its equivalence class. By only considering functions up to $\approx$ (i.e. always viewing functions as cost functions), Colcombet was able to develop a rich theory that parallels the theory of regular languages. In particular, Colcombet [Col09c] showed that a class of *regular cost functions* over finite words can be equivalently defined (up to $\approx$) in terms of cost automata, a logic (known as *cost monadic second-order logic*), an extension of regular expressions (known as *BS-regular expressions*), and an algebraic notion (called *stabilization monoids*). Moreover, given two regular cost functions f, g over finite words, it is decidable whether or not $f \preccurlyeq g$ [Col09c] (although this result actually follows from earlier work in [BC06], see Section 2.4). Since functions rather than languages are the central objects, it is called the *theory of regular cost functions* [Col09c].

This theory of regular cost functions subsumes the classical theory of regular languages because a language L can be associated with its characteristic function χ_L that maps every word in the language to 0 and every word outside of the language to ∞. The classical problem of testing language inclusion $K \subseteq L$ is equivalent to testing $\chi_L \preccurlyeq \chi_K$. It is a strict extension of the classical theory since regular cost functions can count some behaviour within the input structures. Indeed, it subsumes the results of Hashiguchi, Kirsten, and others regarding the decidability of limitedness for distance automata and nested distance desert automata since these automata models are special cases of cost automata, and deciding limitedness is a special case of deciding $\preccurlyeq$ (see Remark 2.2).

This rich theory has been the subject of a number of papers [Col09c, Col09a, CKL10, Kup11] and is explained in more detail in the next chapter. The theory has also been extended to finite trees [CL10]. This thesis can be viewed as an extension of this theory to infinite trees.

In terms of automata, one of the key ideas in the theory is the use of two dual forms of cost automata called *B-automata* and *S-automata*. These dual B and S forms were already present in a slightly different form in earlier work due to Bojańczyk and Colcombet [BC06] (see Section 2.4). Roughly speaking, B-automata are designed to witness boundedness whereas S-automata are designed to witness unboundedness.

Switching between these two dual forms in the cost setting corresponds to finding an automaton for the complement language in the classical setting (see Example 2.3). A crucial and non-trivial result in the theory over finite words is that regular cost functions can be recognized by both B-automata and S-automata [Col09a, BC06]. Proving a similar result over infinite trees is one of the major problems tackled by this thesis, and one to which we are only able to give a partial solution.

1.3 Contributions and document structure

This thesis extends the theory of regular cost functions to infinite trees, particularly in relation to various weak forms of cost automata. Although the results have analogies in well-understood classical results, many of the extensions to the cost setting are non-trivial. We use the results to prove the decidability of special cases of challenging problems in language theory. The thesis is structured as follows.

Preliminaries

Chapter 2 provides additional background material on the theory of regular cost functions over finite words and finite trees. The goal is to provide definitions and examples. In particular, we provide examples of the two types of cost automata (the B and S form) and state the important result that every regular cost function over finite words or trees can be recognized by both a nondeterministic B-automaton and a nondeterministic S-automaton.

We then adapt this framework to infinite words and trees in Chapter 3. In place of classical acceptance and winning conditions, we use *objectives* that describe how to assign values based on both the counters and a classical condition (such as a Büchi condition or parity condition). We write, for instance, B-parity automata, for cost automata that combine the parity acceptance condition with the counting features of B-automata. We then describe the semantics of *alternating cost-parity automata* over infinite trees in terms of an infinite duration *cost game* where one player seeks to minimize the value and the other player seeks to maximize the value according to the objective. We also prove some basic results about cost-parity automata over infinite trees. For instance, we show that it is straightforward to convert alternating B-parity to alternating S-parity automata, and vice versa (just as it is easy to complement alternating automata in the classical setting).

Most of the definitions and results in this chapter are straightforward extensions of [CL10] and were presented in [VB11].

Strategies in cost-parity games

Chapter 4 is the technical core of the thesis. By adapting a result from [CL10], we show that for cost functions f and g over infinite trees, $f \preccurlyeq g$ is decidable when f is given by a nondeterministic S-parity automaton and g is given by a nondeterministic B-parity automaton. This decidability result establishes the goal of the remaining work: given any alternating cost-parity automaton, construct a nondeterministic B-parity automaton and nondeterministic S-parity automaton recognizing the same function (up to $\approx$). Being able to do this for all cost-parity automata would mean that $\preccurlyeq$ is decidable for all regular cost functions over infinite trees. In this thesis, however, we are able to accomplish this goal for a restricted class of cost-parity automata.

We start by looking more closely at the strategies required in cost-parity games. We show that simple strategies that use only *finite memory* are sufficient in certain cost-parity games with only two priorities. Even in these cases, finite memory strategies cannot always guarantee the optimal value in the game. Instead, we prove that we can bound the value when the player is restricted to finite memory strategies as a function of the number of counters and the value when arbitrary strategies are allowed. The proofs utilize a number of techniques including transformation between objectives (which parallels the transformation between winning conditions known from the literature, e.g. [GH82]) and a slicing technique (related to the breakpoint construction in [MH84]).

These memory results are used to show that alternating B-Büchi (respectively, S-Büchi) automata can be simulated by nondeterministic B-Büchi (respectively, S-Büchi) automata, generalizing the classical result due to Muller and Schupp [MS95] that alternating Büchi automata can be simulated by nondeterministic Büchi automata. The idea is that the nondeterministic automaton guesses a finite memory strategy based on the alternating automaton and then computes its value.

This chapter is a more complete presentation of work first described in [VB11].

Weak and quasi-weak cost automata

Chapters 5 and 6 describe weak and quasi-weak cost automata, two classes of cost automata over infinite trees for which $\preccurlyeq$ is decidable. These cost-parity automata over infinite trees use only two priorities and have certain restrictions on the structure of the transition function that make these automata weaker than arbitrary alternating cost-parity automata.

Weak cost automata are similar to classical weak automata: there is a fixed bound across all runs on the number of alternations between accepting and rejecting states/priorities. On the other hand, quasi-weak cost automata (introduced by Kuperberg and the author in [KVB11]) exhibit a variant of weakness that is new to the cost setting: the number of alternations is bounded based on the counter values, which can vary across runs. In both cases, the simulation result from Chapter 4 can be used to show that weak and quasi-weak cost automata can be converted to both nondeterministic B-Büchi and S-Büchi automata, so $\preccurlyeq$ is decidable.

We also show links to other formalisms. Going back to the work of Büchi [Büc60], automata were used to show the decidability of logics. We study an extension of weak monadic second-order logic (WMSO) with a predicate $|X| \leq N$ that can assert that the cardinality of X (where X is some second-order variable) is at most N. Sentences in this logic, called *cost WMSO*, define cost functions. We show that cost WMSO and weak cost automata define the same class of cost functions, and this equivalence is effective. This implies a decision procedure for this cost WMSO logic. Chapter 5 focuses on this result, which was first published in [VB11].

Rabin [Rab70] famously showed that weakly definable languages (languages definable in WMSO or recognizable by weak automata) could be characterized by definability of the language and its complement with Büchi automata. We show that it is the larger class of quasi-weak cost automata that admits a Rabin-style characterization: a cost function is recognizable by a quasi-weak cost automaton if and only if the function is recognizable by both a B-Büchi and S-Büchi automaton. This marks an interesting divergence from the classical theory. Constructing a quasi-weak automaton from nondeterministic B-Büchi and S-Büchi automata requires significant technical work, inspired by the construction due to Kupferman and Vardi [KV99]. This construction is given in Chapter 6 and is based on joint work with Kuperberg published in [KVB11].

Application to the parity index problem

In Chapter 7, we use the results about cost functions over infinite trees developed in this thesis in order to solve special cases of two challenging problems about regular languages of infinite trees.

First, we show that the results from Chapter 4 and [VB11] (combined with results due to Colcombet and Löding [CL08b]) prove that the *nondeterministic parity index problem* is decidable for the $[0, 1]$ or co-Büchi level: given an arbitrary parity

automaton, we can decide whether there is a nondeterministic co-Büchi automaton defining the same language.

We also show the decidability of a special case of the *weak definability problem*: given an (alternating) Büchi or co-Büchi automaton, we can decide whether or not there is a weak automaton recognizing the same language. We show that this result follows from the Rabin-style characterization in Chapter 6, but we also give an alternative construction that gives more intuition into why quasi-weak automata are needed. This application to the weak definability problem is based on joint work with Kuperberg [KVB11].

In Chapter 8, we conclude with open questions and further research directions based on the work in this thesis.

Chapter 2

Theory of Regular Cost Functions

The theory of regular cost functions is a robust quantitative extension to the theory of regular languages. In Sections 2.1 to 2.3, we review the technical background necessary for understanding cost automata and the larger theory of regular cost functions over finite words and finite trees. We primarily follow the presentation from Colcombet and Löding [Col09a, CL10], which is the starting point for our work over infinite trees later in this thesis. We also describe in Section 2.4 some related work that influenced the theory of regular cost functions.

In order to understand this new theory, it is helpful to remember connections with classical results. For instance, determinization and complementation are fundamental in the theory of regular languages. Unlike the classical setting, cost automata cannot always be determinized. Instead, cost automata over words can be made history deterministic, a form of nondeterminism with some deterministic-like properties. In place of closure under complementation, there is a duality theorem that allows a switch between dual B and S forms of a cost automaton. We provide a series of examples of these dual forms and history deterministic cost automata in Section 2.2. We also summarize other closure and decidability properties of these automata over finite words in Section 2.2 and finite trees in Section 2.3.

Notation and Conventions

Let $\mathbb{N}$ be the set of non-negative integers. For $i \leq j$, we write $[i, j]$ to denote the finite interval $\{i, i+1, \ldots, j\}$ of $\mathbb{N}$. We write $\mathbb{N}_\infty$ to denote $\mathbb{N} \cup \{\infty\}$, ordered such that $0 < 1 < \cdots < \infty$. We let $\inf \emptyset = \infty$ and $\sup \emptyset = 0$ (the usual convention).

An *alphabet* $\mathbb{A}$ is a finite set of symbols (each symbol is called a *letter*). The set of finite (respectively, infinite) words over $\mathbb{A}$ is denoted $\mathbb{A}^*$ (respectively, $\mathbb{A}^\omega$). The *empty word* is denoted ϵ, and $\mathbb{A}^+$ is $\mathbb{A}^* \setminus \{\epsilon\}$. If $u \in \mathbb{A}^* \cup \mathbb{A}^\omega$, then $u(i)$ denotes the

ith letter of u. We write $|u|$ for the *length* of u (∞ if $u \in \mathbb{A}^\omega$). If $a \in \mathbb{A}$, $|u|_a \in \mathbb{N}_\infty$ denotes the cardinality of $\{i : u(i) = a\}$, i.e. the number of a-labelled positions in u. We write $|\cdot|$ and $|\cdot|_a$ to denote the functions that map finite or infinite words u to $|u|$ and $|u|_a$, respectively.

Given a set S, we write $\mathscr{P}(S)$ for the *powerset* of S, the set of all subsets of S. Likewise, we write $\mathscr{B}^+(S)$ for the set of formulas that are *positive boolean combinations* of elements from S (formulas constructed by combining elements from S using conjunction and disjunction, but not negation). Given a formula φ, we write $\varphi[s'/s]$ for the result of replacing every occurrence of s with s'.

2.1 Boundedness relation and domination preorder

Functions rather than languages are the central components in the theory of regular cost functions. Given a domain $\mathscr{D}$ of structures (e.g. the set of words or trees over a fixed finite alphabet $\mathbb{A}$) we seek to analyse functions f from $\mathscr{D}$ to $\mathbb{N}_\infty$.

Let $f, g : \mathscr{D} \to \mathbb{N}_\infty$. Given a set $U \subseteq \mathscr{D}$, we write $f(U) = \{f(u) : u \in U\}$. We say $f(U)$ is *bounded* if there exists $n \in \mathbb{N}$ such that $\sup f(U) \leq n$. We say g *dominates* f (written $f \preccurlyeq g$) if and only if for all $U \subseteq \mathscr{D}$, if $g(U)$ is bounded then $f(U)$ is bounded. We write $f \approx g$ if and only if $f \preccurlyeq g$ and $g \preccurlyeq f$.

If we want to be more precise about the relationship between f and g, we can annotate $\preccurlyeq$ and $\approx$ with a *correction function* $\alpha : \mathbb{N} \to \mathbb{N}$, a non-decreasing function that satisfies $\alpha(n) \geq n$ for all n. We write $f \preccurlyeq_\alpha g$ if $f(u) \leq \alpha(g(u))$ for all $u \in \mathscr{D}$ (with the convention that $\alpha(\infty) = \infty$). Thus, α describes how much we may need to "stretch" g such that it dominates f. Note that $f \preccurlyeq g$ if and only if there is some correction function α such that $f \preccurlyeq_\alpha g$. To compare single values $m, n \in \mathbb{N}$, we also write $m \preccurlyeq_\alpha n$ if $m \leq \alpha(n)$. Finally, we write $f \approx_\alpha g$ if $f \preccurlyeq_\alpha g$ and $g \preccurlyeq_\alpha f$.

For the identity correction function $\mathrm{id}(n) = n$ for all $n \in \mathbb{N}$, $\preccurlyeq_{\mathrm{id}}$ and $\approx_{\mathrm{id}}$ coincide with the relations $\leq$ and $=$, respectively. In general, $\preccurlyeq_\alpha$ and $\approx_\alpha$ are weakened versions of $\leq$ and $=$ that ignore exact values of f and g, but preserve boundedness properties.

Example 2.1. Let $\mathscr{D} = \{a, b\}^*$ be the set of $\{a, b\}$-labelled words. Then $|\cdot|_a \approx_\alpha 2|\cdot|_a$ for $\alpha(n) = 2n$, but $|\cdot|_a \not\approx |\cdot|_b$ and $|\cdot|_a \not\approx |\cdot|$ (consider the set $U = \{b^n : n \in \mathbb{N}\}$ that is bounded on $|\cdot|_a$ but unbounded on $|\cdot|_b$ and $|\cdot|$).

However, $|\cdot| \approx_\alpha \max(|\cdot|_b, \text{max-seg}_a)$ for $\alpha(n) = (n+1)^2$ where $\text{max-seg}_a(u)$ is the maximum length of a segment of consecutive a's in u. One direction is immediate: $\max(|\cdot|_b, \text{max-seg}_a) \leq |\cdot|$. We also have $|\cdot| \preccurlyeq_\alpha \max(|\cdot|_b, \text{max-seg}_a)$: assuming that $\max(|\cdot|_b, \text{max-seg}_a)$ is bounded by some n on some set U of words, the longest word

that could be in U is of the form $(a^n b)^n a^n$, which has length $n(n+1)+n < (n+1)^2$, so $|\cdot|$ is bounded by $\alpha(n)$ on U.

We refer the reader back to Figure 1.1 for additional examples.

With these relations, we can formally define a *cost function* F to be an equivalence class of $\approx$, but we will blur the distinction between a particular function $f : \mathscr{D} \to \mathbb{N}_\infty$ and its equivalence class F. Unless otherwise stated, equivalence of functions in this thesis is always up to $\approx$.

Remark 2.2. The classical language inclusion problem and the limitedness problem are special cases of deciding the domination preorder.

- We can identify a language $L \subseteq \mathscr{D}$ with its *characteristic function* χ_L that maps $u \in L$ to 0 and $u \notin L$ to ∞. Then $K \subseteq L$ if and only if $\chi_L \preccurlyeq \chi_K$.
- Given a cost function f, the *limitedness problem* for f is equivalent to $f \preccurlyeq f'$ where $f'(u)$ is 0 if $f(u) < \infty$ and ∞ otherwise. A related problem called *boundedness* or *uniform universality* asks whether f is bounded across all inputs; this is equivalent to $f \preccurlyeq 0$ where 0 is a constant function mapping all inputs to 0.

2.2 Cost automata on finite words

We use cost automata to recognize cost functions. A *nondeterministic cost automaton $\mathcal{A}$ over finite words* is a tuple

$$\langle Q, \mathbb{A}, q_0, \Gamma, F, \Delta \rangle$$

where Q is a finite set of states, $\mathbb{A}$ is a finite alphabet, $q_0 \in Q$ is the initial state, Γ is a finite set of counters, $F \subseteq Q$ is a set of accepting states, and $\Delta : Q \times \mathbb{A} \times (\mathbb{C}^*)^\Gamma \times Q$ is the transition function where $\mathbb{C} := \{\mathtt{i}, \mathtt{r}, \mathtt{c}\}$ is the alphabet of atomic counter actions.

Each counter is initially assigned value 0, but can take on any value from $\mathbb{N}$ based on the sequence of *counter actions* from $\mathbb{C}$ on each transition. The counters can be *incremented* `i`, *reset* `r` to 0, *checked* `c`, or *left unchanged* ε.[1] We care about the value of the counter at the moment(s) when it is checked. Given a word w over the alphabet $\mathbb{C}$ describing the actions on some counter $\gamma \in \Gamma$, we define a set $C(w) \subseteq \mathbb{N}$ that collects all of the checked values of γ. For instance, $C(\mathtt{iriiicriic}) = \{2, 3\}$

[1] Although ϵ is usually used to denote the empty word, we use ε to denote the special empty word of counter actions.

since the first time the counter is checked it has value 3 and the second time it has value 2. In the case of a finite set of counters Γ and a word w over the alphabet $(\mathbb{C}^*)^\Gamma$, $C(w) := \bigcup_{\gamma\in\Gamma} C(\mathrm{pr}_\gamma(w))$ (where $\mathrm{pr}_\gamma(w)$ is the γ-projection of w). Notice that we do not care about when these checked values appear, how many times they appear, or which counter or counters they come from.

A *run* ρ of a cost automaton on an *input* word $u = a_1 \cdots a_n \in \mathbb{A}^*$ is a sequence of transitions $\rho = (q_0, a_1, c_1, q_1) \cdots (q_{n-1}, a_n, c_n, q_n) \in \Delta^*$. The run is *accepting* if $q_n \in F$. The *output* $\mathrm{out}(\rho)$ is the sequence of counter actions $c_1 \cdots c_n$.

Like a classical automaton, we say that $\mathcal{A}$ recognizes the *language* $L(\mathcal{A})$ where

$$L(\mathcal{A}) := \{u \in \mathbb{A}^* : \text{there is an accepting run of } \mathcal{A} \text{ on } u\}.$$

However, there are also two dual semantics, B and S, that are used to assign values.[2] Given a sequence of counter actions $w \in (\mathbb{C}^*)^\Gamma$, the *$B$-value* is the maximum checked value of any counter, whereas the *S-value* is the minimum checked value:

$$\mathrm{value}_B(w) := \sup C(w) \quad \text{and} \quad \mathrm{value}_S(w) := \inf C(w).$$

The value of a run ρ is the value of the output, so we have $\mathrm{value}_B(\rho) := \mathrm{value}_B(\mathrm{out}(\rho))$ and $\mathrm{value}_S(\rho) := \mathrm{value}_S(\mathrm{out}(\rho))$. Finally, a value is assigned to each word $u \in \mathbb{A}^*$ using one of these semantics:

$$[\![\mathcal{A}]\!]_B(u) := \inf\{\mathrm{value}_B(\rho) : \rho \text{ is an accepting run of } \mathcal{A} \text{ over } u\}$$
$$[\![\mathcal{A}]\!]_S(u) := \sup\{\mathrm{value}_S(\rho) : \rho \text{ is an accepting run of } \mathcal{A} \text{ over } u\}$$

The convention is that $\inf \emptyset = \infty$ and $\sup \emptyset = 0$, so in the case that there are no accepting runs, the B-semantics assign value ∞ and the S-semantics assign value 0. If there are accepting runs, then the B-semantics (respectively, S-semantics) minimize (respectively, maximize) the checked counter values over the accepting runs.

We refer to an automaton as a *B-automaton* or an *S-automaton* depending on the desired semantics. When the intended semantics are clear, we will just write $[\![\mathcal{A}]\!]$ for the function from $\mathbb{A}^*$ to $\mathbb{N}_\infty$ defined by $\mathcal{A}$. We say that $\mathcal{A}$ *recognizes* a function $f : \mathbb{A}^* \to \mathbb{N}_\infty$ if $f \approx [\![\mathcal{A}]\!]$. That is, $\mathcal{A}$ recognizes the cost function $[\![\mathcal{A}]\!]$.

[2] This B and S notation was originally used in [BC06] to stand for sequences of counter values that were bounded and strongly unbounded (converged to infinity); see Section 2.4 for more information.

2.2.1 Examples

We give a few examples (mostly from [Col09a]). In each case, $\mathbb{A} := \{a, b\}$.

Example 2.3. If $\mathcal{A}$ is a traditional automaton without counters recognizing a language L, then $[\![\mathcal{A}]\!]_B = \chi_L$ (the characteristic function for L that maps $u \in L$ to 0 and $u \notin L$ to ∞) and $[\![\mathcal{A}]\!]_S = \chi_{\overline{L}}$, where $\overline{L}$ is the complement of L, in this case $\mathbb{A}^* \setminus L$.

This example supports the idea that B and S are dual semantics. It shows that switching between B and S in the cost setting takes the place of complementing languages in the classical setting.

Of course, cost automata are designed to count some behaviour within the input structure. Perhaps the simplest example is a cost automaton counting the occurrences of some letter $a \in \mathbb{A}$ in a word $u \in \mathbb{A}^*$.

Example 2.4. Consider the function $|\cdot|_a$ that maps a word u to the number of a-labelled positions in u.

The B-automaton $\mathcal{B}$ that recognizes this function has a single state (which is initial and accepting) and a single counter that is incremented and checked (`ic`) each time an a is seen, and left unchanged (ε) otherwise. Because this automaton is deterministic, for each word u there is only a single run ρ, and this run is accepting. Since the counter is checked each time an a is seen, $C(\mathrm{out}(\rho)) = \{1, 2, \ldots, |u|_a\}$ and $\mathrm{value}_B(\rho) = \sup C(\mathrm{out}(\rho)) = |u|_a$, so $[\![\mathcal{B}]\!]_B = |u|_a$.

The corresponding S-automaton $\mathcal{S}$ increments (`i`) the counter when reading an a, and then guesses when the end of the word is reached and performs a check-reset (`cr`).[3] For a given word u, there is only one accepting run ρ, which moves from the initial state to the accepting state on the last letter. If the last letter is a, then $\mathrm{value}_S(\rho) = \inf\{|u|_a - 1\} = |u|_a - 1$; if the last letter is b, then $\mathrm{value}_S(\rho) = \inf\{|u|_a\} = |u|_a$. Notice that it does not define exactly the function $|\cdot|_a$ but it does recognize the cost function corresponding to $|\cdot|_a$. Indeed, since the value may be off by one when the word ends in a, the S-automaton recognizes $|u|_a$ with a correction function of $\alpha(n) = n + 1$, written $[\![\mathcal{S}]\!]_S \approx_\alpha |u|_a$.

These automata are shown in Figure 2.4. The initial state (respectively, final state(s)) are indicated by unlabelled ingoing (respectively, outgoing) edges, as usual. An edge labelled $e : c$ means that when reading input $e \in \mathbb{A}$ the output action is c.

[3] The automaton could also use just a check (`c`) instead of a check-reset (`cr`). The counter actions in this example (and in the remaining examples) have been chosen to be "simple", which will be explained on page 20.

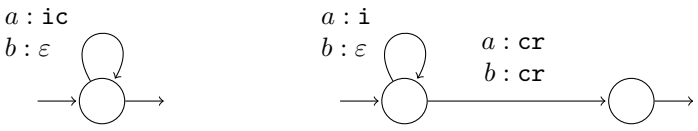

Figure 2.1. B-automaton and S-automaton recognizing $|u|_a$ (see Example 2.4).

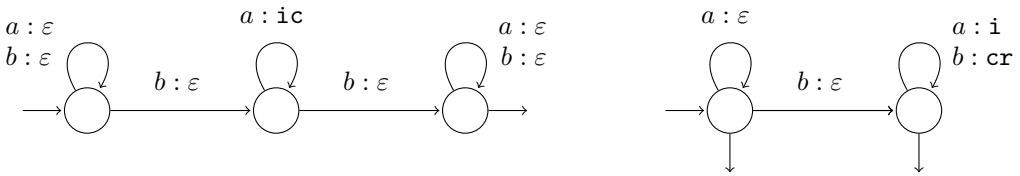

Figure 2.2. B-automaton and S-automaton recognizing min-block$_a$ (see Example 2.5).

Example 2.5. Now consider the function min-block$_a$ that maps a word u to the minimum size of a block of a's surrounded by b's in u (and ∞ if there is no such block). This function is recognized by a nondeterministic B-automaton with one counter and a deterministic S-automaton with one counter, shown in Figure 2.2.

The checked value from each accepting run ρ of the B-automaton corresponds to the size of a particular block of a's surrounded by b's. Hence, the infimum over the values of the accepting runs is exactly the minimum length of a block of a's surrounded by b's. Notice that if there are no such blocks (i.e. there are less than two b's in the word), then there will be no accepting run of the automaton and the value will be $\inf \emptyset = \infty$ as desired.

The S-automaton is deterministic and all runs are accepting. Let ρ be the unique run for some word u. If u has no block of a's surrounded by b's, then the counter is never checked so $\text{value}_S(\rho) = \inf \emptyset = \infty$ as desired. Otherwise, the automaton counts every block of a's surrounded by b's so $\text{value}_S(\rho) = \inf C(\text{out}(\rho)) = \inf \{n : ba^n b \text{ is a factor of } u\} = \text{min-block}_a(u)$.

Since nondeterminism in a B-automaton (respectively, S-automaton) is resolved into taking an infimum (respectively, supremum), one should think of a B-automaton as trying to find the accepting run that will minimize (respectively, maximize) the counter value. This nondeterminism is essential. Indeed, it is straightforward to prove that there is no deterministic B-automaton recognizing min-block$_a$, even up to $\approx$.

The automata in the previous example can be adapted to show that max-block$_a$ (the maximum length of a block of a's surrounded by b's, or 0 if there is no such block) is recognizable by both a B- and S-automaton. In that case, the B-automaton is deterministic and the S-automaton uses nondeterminism to guess the largest block to count.

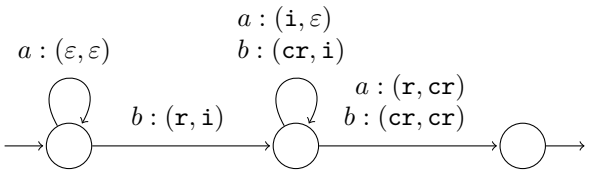

Figure 2.3. S-automaton recognizing $g(u) = \min\{\text{min-block}_a(u), |u|_b\}$ (see Example 2.6).

We now consider a simple example that makes use of two counters.

Example 2.6. Let $g(u) = \min\{\text{min-block}_a(u), |u|_b\}$. The B-automaton recognizing g guesses at the outset whether to run the B-automaton from Example 2.5 or Example 2.4 (adjusted to count b's instead of a's).

The S-automaton recognizing g has two counters γ_1 and γ_2, where γ_1 is responsible for computing $\text{min-block}_a(u)$ and γ_2 is responsible for computing $|u|_b$. This automaton is shown in Figure 2.3; we write $e : (c_1, c_2)$ where c_i is the action for γ_i when reading $e \in \mathbb{A}$. Consider some word u. If u has no b's, then $g(u) = 0$ and there is no accepting run of the automaton so the value is 0 as desired. If u has one b, then $g(u) = 1$ and there is either no accepting run (if u ends with b) or a single accepting run with checked value 1 (if u ends with a). Otherwise, if u has at least two b's, then there is a single accepting run ρ and $C(\text{out}(\rho))$ contains the sizes of each block of a's surrounded by b's (taken from the checked values of γ_1), and $|u|_b$ if the word ends in a or $|u|_b - 1$ if the word ends in b (taken from the checked value of γ_2). Overall, this S-automaton recognizes g with a correction function of $\alpha(n) = n + 1$.

In general, increasing the number of counters increases the expressive power of the cost automata. That is, for all k, there is a cost function definable using a cost automaton with k counters that is not definable using less than k counters [Col12b].

2.2.2 Duality

The examples above hide some of the difficulties inherent in finding both a B- and S-automaton recognizing a cost function. This is due to the fact that in most of the examples so far, either the B or S form of the automaton is deterministic. When starting from a deterministic cost automaton, it is straightforward to construct a (nondeterministic) cost automaton for its dual form. For instance, a deterministic S-automaton can be simulated by a nondeterministic B-automaton by using exactly the same counter actions except for checks: the B-automaton guesses a single check to duplicate and ignores all other checks from the output of the original S-automaton [Col09a, Proposition 5]. Recall that Example 2.3 showed that switching between

B and S forms in the cost setting is like complementation in the classical setting; since complementation is easy for deterministic automata, it makes sense that dualizing is easy in this case as well. Unfortunately, unlike the classical setting, not all cost automata are determinizable [Col09c].

Consider the following example: take $f(u) = \min\{\max\{|v|_a, |w|_b\} : u = vw\}$. There is a very natural B-automaton recognizing f that guesses the factorization of u into v and w and then counts the appropriate letters in each segment. What is the corresponding S-automaton? Since nondeterminism in an S-automaton corresponds to taking a maximum, we can no longer use the nondeterminism to guess the factorization, so it is not clear how to construct an S-automaton recognizing the correct function.

To aid in this transformation between B and S forms, Colcombet shows that these cost functions can also be defined using an algebraic structure. Reasoning about automata using algebra is nothing new. It is well-known that a finite automaton can be represented by a *monoid* (a set with an associative binary operation $\cdot$ and an identity) that defines the same language of finite words (we refer the interested reader to [PP04]). Unfortunately, the classical translation of a finite automaton to a finite monoid carries no quantitative information about the automaton. This is due to the properties of the product in a monoid: if e is *idempotent* (i.e. $e^2 = e$), then $e^n = e$ for all $n \geq 1$. This means we cannot count the number of times we take the product of such an element e in a standard monoid.

In order to get around this limitation, Colcombet introduces algebraic structures called *stabilization monoids* $\langle M, \cdot, \leq, \sharp \rangle$ in [Col09a] that enhance a monoid with an ordering $\leq$ and a *stabilization operator* $\sharp$. The stabilization operation is defined only on idempotent elements. The idea is that e^n should be e when n is "small" and $e^\sharp$ when n is "large". Of course, this vague notion about what is "small" and "large" would not be useful if we were interested in particular values of these functions, but it is exactly what is needed for cost functions that care only about boundedness.

It is this rich algebraic structure where much of the technical work for the theory of regular cost functions over finite words is performed. In general, in order to convert between the B and S forms, the original automaton is converted first into a stabilization monoid. From this stabilization monoid both a B- or S-automaton can be constructed that is equivalent to the original, up to $\approx$.

In fact, we get more from this conversion through the algebra. The cost automata that come out of this transformation have the following features:

- *Simple counter actions*: atomic counter actions for B-automata (respectively, S-automata) are restricted to $\mathbb{B} := \{\varepsilon, \mathtt{ic}, \mathtt{r}\}$ (respectively, $\mathbb{S} := \{\varepsilon, \mathtt{i}, \mathtt{r}, \mathtt{cr}\}$).

- *Hierarchical counter actions*: the set of counters Γ is some set $[1, k]$ and if the action for counter j is not equal to ε, then all counters $j' < j$ are reset ($\mathtt{r}$) and counters $j' > j$ are left unchanged (ε).[4] We write, e.g. hB-automaton for a B-automaton with hierarchical counter actions.

- *History deterministic transition relation*: a nondeterministic transition relation with some deterministic-like properties; this is defined formally in the next section and is crucial for the extension of this theory to trees.

We point out that Examples 2.4–2.6 have simple counter actions. The one-counter automata in Examples 2.4–2.5 are trivially hierarchical, but the two-counter automaton in Example 2.6 does not have hierarchical counter actions (an S-automaton with hierarchical counters can be constructed for it, but we leave this as an exercise for the reader). The deterministic automata in the examples so far are trivially history deterministic, but the nondeterministic cost automata are not (history deterministic versions will be given in the next section).

Remark 2.7. Special types of cost automata have been previously studied under different names.

- *Distance automata* ([Has82]) are B-automata with one counter and restricted actions $\{\mathtt{ic}, \varepsilon\}$.

- *Desert automata* ([Kir04, Bal04]) are B-automata with one counter and restricted actions $\{\mathtt{ic}, \mathtt{r}\}$.

- *Nested distance desert automata* ([Kir05]) and R-automata ([AKY08]) are B-automata with hierarchical simple counter actions.

- ωB- and ωS-automata ([BC06]) are discussed in Section 2.4.

[4] This definition is slightly different than in [Col09a] where if the action for counter j is not equal to ε, then lower counters $j' < j$ are allowed action $\mathtt{cr}$ or $\mathtt{r}$. This is not a significant difference because these two forms are equivalent, up to $\approx$, using the transducer described in Lemma 2.17 which reads arbitrary S-actions and converts into hierarchical actions in line with the definition above. The definition we give here is better suited for later applications in this thesis, namely for the construction in Chapter 6.

By passing through the algebra, we get the following important result that Colcombet refers to as the *duality theorem*. Recalling Example 2.3, this theorem can be viewed as the cost version of a combined determinization and complementation procedure. Indeed, Colcombet has given an alternative proof using ideas from Safra's determinization construction [Col11].

Theorem 2.8 ([Col09c, Theorem 1]). *It is effectively equivalent for a cost function f over finite words to be recognized by the following types of cost automata:*

- *nondeterministic B-automaton;*
- *nondeterministic S-automaton;*
- *history deterministic hB-automaton with simple, hierarchical counter actions;*
- *history deterministic hS-automaton with simple, hierarchical counter actions.*

Such a function f is called a regular cost function over finite words.

From now on, we will assume cost automata have simple counter actions unless otherwise stated. Indeed, assuming simple and hierarchical counter actions is often useful in proofs because it gives more structure to the automata.

A nice corollary of this result concerns the closure properties of cost automata. The natural closure properties of these automata differ from the classical setting since they are on functions rather than languages. For instance, instead of closure under union and intersection, cost automata are closed under taking min and max (of functions); the constructions (using a disjoint union and product) are the same as in the classical setting, however.

The most interesting cases (indeed, the cases where Theorem 2.8 is needed) correspond to projection in the classical setting. In the cost setting, these operations are inf-*projection* and sup-*projection*. Let $h : \mathbb{A}' \longrightarrow \mathbb{A}$ be a map between alphabets $\mathbb{A}'$ and $\mathbb{A}$ such that $\mathbb{A}' \supseteq \mathbb{A}$ and $h(a) = a$ for all $a \in \mathbb{A}$. We write $h(u') = u$ for the extension of h to words that relabels $u' \in (\mathbb{A}')^*$ according to h. The op-*projection* of some cost function $g : (\mathbb{A}')^* \longrightarrow \mathbb{N}_\infty$ over $h : \mathbb{A}' \longrightarrow \mathbb{A}$ is the function $g_{\mathrm{op},h} : \mathbb{A}^* \longrightarrow \mathbb{N}_\infty$ such that

$$g_{\mathrm{op},h}(u) := \mathrm{op}\,\{g(u') : h(u') = u\}$$

where op is inf or sup. The idea is that on input u, the op-projection of g over h combines (using the operation op) all of the values of g on words u' that could be projected to u via h.

B-automata are naturally closed under inf-projection since the nondeterminism resolves into taking an infimum: on input u, the B-automaton guesses some u' such that $h(u') = u$ and then simulates the B-automaton for g on u'. Likewise, S-automata are naturally closed under sup-projection. Because of Theorem 2.8 we can switch between these dual forms and get closure for all cost automata (and hence all regular cost functions) under these operations.

Proposition 2.9 ([Col09c]). *Regular cost functions over finite words are closed under* min*,* max*,* inf*-projection, and* sup*-projection.*

These closure properties can be useful when proving that some cost function is regular. For instance, consider again $f(u) = \min\{\max\{|v|_a, |w|_b\} : u = vw\}$. Let $\mathbb{A}' := \{a, b, \$\}$ and $\mathbb{A} := \{a, b\}$. It is easy to construct a B-automaton that reads words u' over $\mathbb{A}'$, and counts the number of a's before \$ and the number of b's after \$ (and rejects if there is not exactly one \$ in the word). Let $h : \mathbb{A}' \to \mathbb{A}$ such that $h(\$) = \epsilon$ and $h(e) = e$ for $e \in \{a, b\}$ (so h "erases" the \$). Then $f = [\![\mathcal{A}]\!]_{\inf,h}$, so f is regular by Proposition 2.9.

2.2.3 Decidability

As mentioned in Chapter 1, these cost automata were designed as a tool for decision procedures about questions of boundedness. Hence, the crucial result in the theory of regular cost functions over finite words is that the domination preorder and the boundedness relation are decidable.

Theorem 2.10 ([Col09c, Theorem 4] following from [BC06]). *Given regular cost functions f and g over finite words, it is decidable whether or not $f \preccurlyeq g$.*

Like dualization, the proof of this result goes through algebra: it uses a saturation procedure based on the algebraic representation of a regular cost function as a stabilization monoid. We refer interested readers to Section 2.4 and [Col09c].

By Remark 2.2, this subsumes the decidability of the language inclusion problem (for classical automata), and the limitedness for distance and nested distance desert automata. As mentioned in Chapter 1, regular cost functions over finite words can also be defined using a cost monadic second-order logic, so Theorem 2.10 implies the decidability of this cost logic. We delay further discussion of the logic until Chapter 5.

2.2.4 History determinism

Crucial for the extension to the tree case is the fact that cost automata over words can be made history deterministic. This property was mentioned in Theorem 2.8, and we explain this concept now. History determinism was introduced by Colcombet [Col09c] as a weakening of the standard notion of determinism. Informally, a history deterministic cost automaton is a nondeterministic automaton where all of the "guesses" depend only on the history. That is, there should be an oracle which given the history of the word, and the desired value for the run, is able to choose deterministically the next transition. Formally, we define this oracle in terms of a family $(\vartheta_n)_{n\in\mathbb{N}}$ of translation strategies, one for each possible value n, described below.

Fix some nondeterministic cost automaton $\mathcal{A} = \langle Q, \mathbb{A}, q_0, \Gamma, F, \Delta\rangle$. A *translation strategy* $\vartheta : \mathbb{A}^* \times \mathbb{A} \to \Delta$ for $\mathcal{A}$ describes how to deterministically construct the next move in a run of $\mathcal{A}$ over a word $u \in \mathbb{A}^*$ given the prefix of the word already read (i.e. the "history"). We can extend this into a mapping $\tilde{\vartheta} : \mathbb{A}^* \to \Delta^*$ defined such that

$$\tilde{\vartheta}(u) = \vartheta(\epsilon, a_1)\vartheta(a_1, a_2)\vartheta(a_1a_2, a_3)\cdots\vartheta(a_1a_2\cdots a_{k-1}, a_k)$$

for $u = a_1a_2\cdots a_k \in \mathbb{A}^*$. We say ϑ *drives the run* $\tilde{\vartheta}(u) \in \Delta^*$ if $\tilde{\vartheta}(u)$ is a valid run of $\mathcal{A}$ on u. From now on, we restrict to translation strategies ϑ such that $\tilde{\vartheta}(u)$ is a valid run of $\mathcal{A}$ over u for all $u \in \mathbb{A}^*$.

Given a family of translation strategies $(\vartheta_n)_{n\in\mathbb{N}}$, we can define the *B-semantics relative to* $(\vartheta_n)_{n\in\mathbb{N}}$ and *S-semantics relative to* $(\vartheta_n)_{n\in\mathbb{N}}$ as follows:

$$[\![\mathcal{A}]\!]^{\vartheta}_B(u) := \inf\left\{n : \tilde{\vartheta}_n(u) \text{ is an accepting run of value at most } n\right\},$$
$$[\![\mathcal{A}]\!]^{\vartheta}_S(u) := \sup\left\{n : \tilde{\vartheta}_n(u) \text{ is an accepting run of value at least } n\right\}.$$

A cost automaton $\mathcal{A}$ is *history deterministic* if and only if there is some family of translation strategies $(\vartheta_n)_{n\in\mathbb{N}}$ and some correction function α such that $[\![\mathcal{A}]\!]^{\vartheta} \approx_\alpha [\![\mathcal{A}]\!]$. If we want to make the correction function explicit, we say that $\mathcal{A}$ is α-history-deterministic. In other words, for all $u \in \mathbb{A}^*$, the value obtained when $\mathcal{A}$ is restricted to accepting runs driven by $(\vartheta_n)_{n\in\mathbb{N}}$ on u is $\approx_\alpha$-equivalent to the value obtained when considering any accepting run of $\mathcal{A}$ on u.

We can assume that $(\vartheta_n)_{n\in\mathbb{N}}$ is *monotonic* in the following sense (which depends on the semantics being used): if $\mathcal{A}$ is a B-automaton and $\tilde{\vartheta}_n(u)$ is an accepting run of value at most n, then for all $n' \geq n$, $\tilde{\vartheta}_{n'}(u)$ is an accepting run of value at most n'; likewise, if $\mathcal{A}$ is an S-automaton and $\tilde{\vartheta}_n(u)$ is an accepting run of value at least n, then for all $n' \leq n$, $\tilde{\vartheta}_{n'}(u)$ is an accepting run of value at least n'.

Unpacking the definition above, a B-automaton $\mathcal{A}$ is α-history-deterministic if there is a family of translation strategies $(\vartheta_n)_{n\in\mathbb{N}}$ such that for all $u \in \mathbb{A}^*$,

$$\text{if } [\![\mathcal{A}]\!]_B(u) \leq n \text{ then } \tilde{\vartheta}_{\alpha(n)}(u) \text{ is accepting and } \text{value}_B(\tilde{\vartheta}_{\alpha(n)}(u)) \leq \alpha(n).$$

Likewise, an S-automaton $\mathcal{A}$ is α-history-deterministic if for all $u \in \mathbb{A}^*$,

$$\text{if } [\![\mathcal{A}]\!]_S(u) \geq \alpha(n) \text{ then } \tilde{\vartheta}_n(u) \text{ is accepting and } \text{value}_S(\tilde{\vartheta}_n(u)) \geq n.$$

The important point is that history determinism is weaker than the standard notion of determinism. In a deterministic automaton, the next state is determined entirely by the current state and input letter. In a history deterministic automaton, the desired value n, the history of the play, and the input letter uniquely determine the next transition. Moreover, an automaton may not be able to actually implement this deterministic choice since it may need unbounded memory to store, for instance, the counter values on a partial run.

Remark 2.11. Usually, we seek to simultaneously prove that a cost automaton $\mathcal{A}$ is history deterministic and recognizes a cost function f. If $\mathcal{A}$ is a B-automaton (respectively, S-automaton) it suffices to prove that $[\![\mathcal{A}]\!]_B^\vartheta \preccurlyeq f \preccurlyeq [\![\mathcal{A}]\!]_B$ (respectively, $[\![\mathcal{A}]\!]_S \preccurlyeq f \preccurlyeq [\![\mathcal{A}]\!]_S^\vartheta$) for some family ϑ of translation strategies. This follows from the fact that $[\![\mathcal{A}]\!]_B \leq [\![\mathcal{A}]\!]_B^\vartheta$ (respectively, $[\![\mathcal{A}]\!]_S^\vartheta \leq [\![\mathcal{A}]\!]_S$).

We now consider a few examples of history deterministic cost automata. We will not prove in a formal way that these automata are history deterministic, but we will describe informally the translation strategies that witness the history determinism. We will return to this notion in Section 3.2, where some more formal proofs about the properties of history deterministic automata (in the context of games) will be given.

Example 2.12. Consider the automata recognizing $|u|_a$ given in Example 2.4. The B-automaton $\mathcal{B}$ is deterministic so it is trivially history deterministic: there is a unique run ρ of $\mathcal{B}$ on any input word, so all of the strategies ϑ_n should simply select ρ.

The S-automaton $\mathcal{S}$ for $|u|_a$ is not history deterministic, however, since it requires a guess about the end of the word, and this is not dependent on the history. We can make it history deterministic by adding a loop on the final state that on any input leaves the counter unchanged (shown in Figure 2.4). We call this automaton $\mathcal{S}'$. There is no longer a single accepting run of the automaton on a given input. Instead, $\mathcal{S}'$ is allowed to move to the accepting state at any point during the run, but must check-reset the counter at that time. The idea is that $\mathcal{S}'$ now guesses when the

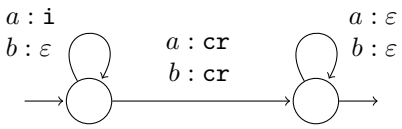

Figure 2.4. History deterministic S-automaton recognizing $|u|_a$ (see Examples 2.4 and 2.12).

counter reaches a high enough value. This sort of guessing is acceptable in a history deterministic automaton since the choice depends only on the history. Formally, the translation strategy ϑ_n stays in the initial state until the counter value reaches n, and then proceeds to the accepting state.

Note that the optimal value for $\mathcal{S}'$ on some input always corresponds to the value from a run driven by one of the translation strategies ϑ_n. For this reason, we say that $\mathcal{S}'$ is id-history-deterministic, where id is the identity correction function $\mathrm{id}(n) = n$. Although $\mathcal{S}'$ is id-history-deterministic, it only recognizes $|u|_a$ up to the correction function $\alpha(n) = n + 1$ since the value is off by one if the last letter is an a. Overall, we have $[\![\mathcal{S}']\!]^{\vartheta} = [\![\mathcal{S}']\!] \approx_\alpha |\cdot|_a$ for $\alpha(n) = n + 1$.

Example 2.13. Now consider another function that counts the number of a's in an input word u if u ends in a, but otherwise assigns value 0. The most natural B-automaton for this function would simply guess at the beginning the last letter, only counting the number of a's if the last letter was guessed to be a, and then checking at the end of the word that the initial guess was correct. As in the previous example, this is not history deterministic because there is no way to drive the run given just the history (the guess depends on the future).

The history deterministic version is shown in Figure 2.5. The translation strategies $(\vartheta_n)_{n\in\mathbb{N}}$ witnessing the history determinism are defined as follows. The strategy ϑ_n stays in state q as long as the next letter will not result in the counter value exceeding n, otherwise it moves to state r. Fix some u. If the last letter in u is b, then for all n, ϑ_n will drive an accepting run of value at most n so $[\![\mathcal{A}]\!]^{\vartheta}(u) = 0$. If the last letter is a, then only ϑ_n for $n \geq |u|_a$ will drive accepting runs, so $[\![\mathcal{A}]\!]^{\vartheta}(u) = |u|_a$. The idea is that the automaton tries to stay in the accepting state q for as long as possible. While there, it is forced to count the number of a's. It delays the guess about the last letter for as long as possible, only guessing that the last letter is b if the counter value gets too high (where the threshold depends on the translation strategy being used). Again, this is id-history-deterministic.

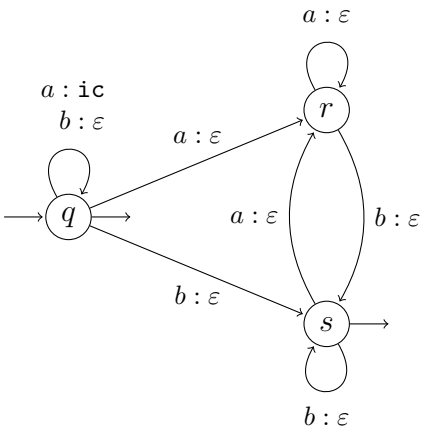

Figure 2.5. History deterministic B-automaton (see Example 2.13).

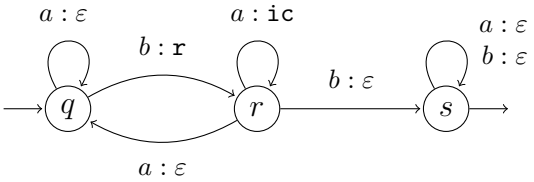

Figure 2.6. History deterministic B-automaton recognizing min-block$_a$ (see Examples 2.5 and 2.14).

Example 2.14. The last example is a history deterministic B-automaton shown in Figure 2.6 that recognizes min-block$_a$ as defined in Example 2.5. The S-automaton for min-block$_a$ is deterministic and therefore trivially history deterministic.

Recall that the original B-automaton recognizing min-block$_a$ guessed the smallest block of a's surrounded by b's, which is not dependent solely on the past and cannot be implemented by a translation strategy. The automaton below instead starts counting every block of a's (with the exception of an initial prefix of a's). The only nondeterminism is in state r. The translation strategies ϑ_n are defined such that ϑ_n stays in state r for as long as possible, and only moves to state q if the counter would otherwise exceed value n. Thus, ϑ_n can be seen as the strategy that is trying to prove that there is a block of size n, and therefore min-block$_a(u) \leq n$. If there is such a block, then the automaton will stay in state r long enough to be able to take the transition from r to the accepting state s. If not, then there will be no accepting run driven by ϑ_n.

As a side note, we remark that an automaton without counters is history deterministic if and only if it contains a deterministic sub-automaton for the same language,

i.e. the automaton can be made deterministic simply by removing edges [Col12a]. Restricted to automata without counters, history deterministic automata correspond to the *good for games automata* that were introduced independently by Henzinger and Piterman [HP06]. This name was chosen because these automata compose well with games. Indeed, it is these good properties with games that make history deterministic automata important in this work. We will make this more precise in Section 3.2.

Transducers

We can use cost automata over words as transducers that read a word of counter actions and output different actions that preserve the value, at least up to $\approx$.

As a simple example of this idea, we start by describing a B-automaton transducer that converts a word over the alphabet $\{\mathtt{ic}, \varepsilon\}^\Gamma$ to actions over $\{\mathtt{ic}, \varepsilon\}$. Although increasing the number of counters usually increases expressivity, when restricting to actions $\{\mathtt{ic}, \varepsilon\}$, multiple counters do not add to the expressivity, and therefore can be converted to a sequence of actions using a single counter.

Lemma 2.15 ([CL12]). *For all Γ, there is a deterministic B-automaton $\mathcal{D}^\Gamma_{\{\mathtt{ic},\varepsilon\}}$ with one counter and actions $\{\mathtt{ic}, \varepsilon\}$ such that $[\![\mathcal{D}^\Gamma_{\{\mathtt{ic},\varepsilon\}}]\!](u) \approx_\alpha \mathrm{value}_B(u)$ for all words $u \in (\{\mathtt{ic}, \varepsilon\}^\Gamma)^*$ where $\alpha(n) = n \cdot |\Gamma|$.*

Proof. The automaton $\mathcal{D}^\Gamma_{\{\mathtt{ic},\varepsilon\}}$ has one state (which is initial and accepting) and one counter. When reading some $c \in \{\mathtt{ic}, \varepsilon\}^\Gamma$, if there is some $\gamma \in \Gamma$ such that $\mathrm{pr}_\gamma(c) = \mathtt{ic}$, then $\mathcal{D}^\Gamma_{\{\mathtt{ic},\varepsilon\}}$ outputs $\mathtt{ic}$. Otherwise, $\mathcal{D}^\Gamma_{\{\mathtt{ic},\varepsilon\}}$ outputs ε.

We aim to show that $[\![\mathcal{D}^\Gamma_{\{\mathtt{ic},\varepsilon\}}]\!](u) \approx_\alpha \mathrm{value}_B(u)$ for all words $u \in (\{\mathtt{ic}, \varepsilon\}^\Gamma)^*$. We actually break the proof into two directions, first showing $\mathrm{value}_B(u) \leq [\![\mathcal{D}^\Gamma_{\{\mathtt{ic},\varepsilon\}}]\!](u)$ and then showing $[\![\mathcal{D}^\Gamma_{\{\mathtt{ic},\varepsilon\}}]\!](u) \preccurlyeq_\alpha \mathrm{value}_B(u)$ for $\alpha(n) = n \cdot |\Gamma|$.

Fix some $u \in (\{\mathtt{ic}, \varepsilon\}^\Gamma)^*$ and assume that $[\![\mathcal{D}^\Gamma_{\{\mathtt{ic},\varepsilon\}}]\!](u)$ is bounded by some $n \in \mathbb{N}$. Since $\mathcal{D}^\Gamma_{\{\mathtt{ic},\varepsilon\}}$ always increments the counter when there is an underlying increment for some counter $\gamma \in \Gamma$, each of the original counters can achieve value at most n. Hence $\mathrm{value}_B(u) \leq [\![\mathcal{D}^\Gamma_{\{\mathtt{ic},\varepsilon\}}]\!](u)$.

Now assume $\mathrm{value}_B(u) \leq n \in \mathbb{N}$. Then each counter $\gamma \in \Gamma$ is incremented at most n times in u. Each increment induces an output $\mathtt{ic}$ in the run of $\mathcal{D}^\Gamma_{\{\mathtt{ic},\varepsilon\}}$, and since these increments could could occur at different positions in u, the value according to $\mathcal{D}^\Gamma_{\{\mathtt{ic},\varepsilon\}}$ is at most $n \cdot |\Gamma|$. This means that $[\![\mathcal{D}^\Gamma_{\{\mathtt{ic},\varepsilon\}}]\!](u) \preccurlyeq_\alpha \mathrm{value}_B(u)$ for $\alpha(n) = n \cdot |\Gamma|$. □

A similar result holds when the actions are restricted to $\{\mathtt{ic}, \mathtt{r}\}$. This was observed in [CKL10] where B-automata using only actions $\mathtt{ic}$ and $\mathtt{r}$ are called temporal B-automata.

Lemma 2.16 ([CKL10]). *For all Γ, there is a deterministic B-automaton $\mathcal{D}^{\Gamma}_{\{\mathtt{ic},\mathtt{r}\}}$ with one counter and actions $\{\mathtt{ic}, \mathtt{r}\}$ such that $[\![\mathcal{D}^{\Gamma}_{\{\mathtt{ic},\mathtt{r}\}}]\!](u) \approx_\alpha \mathrm{value}_B(u)$ for all words $u \in (\{\mathtt{ic}, \mathtt{r}\}^\Gamma)^*$ where $\alpha(n) = 2n + 1$.*

Proof. The B-automaton $\mathcal{D}^{\Gamma}_{\{\mathtt{ic},\mathtt{r}\}}$ has states $Q := \mathscr{P}(\Gamma)$. All states are accepting and the initial state is $\emptyset$. When in state $q \in \mathscr{P}(\Gamma)$ and reading action $c \in \{\mathtt{ic}, \mathtt{r}\}^\Gamma$, we let $q' = q \cup \left\{\gamma : \mathrm{pr}_\gamma(c) = \mathtt{r}\right\}$. If $q' \neq \Gamma$, then the automaton outputs $\mathtt{ic}$ and moves to state q'. Otherwise, if $q' = \Gamma$, then the automaton outputs $\mathtt{r}$ and moves to state $\emptyset$, and continues operating as before.

Fix some $u \in (\{\mathtt{ic}, \mathtt{r}\}^\Gamma)^*$ such that $\mathrm{value}_B(u) \leq n \in \mathbb{N}$. The run of $\mathcal{D}^{\Gamma}_{\{\mathtt{ic},\mathtt{r}\}}$ starts from state $\emptyset$. Since $\mathrm{value}_B(u) \leq n$, after at most $n + 1$ positions every counter must be reset at least once. Hence, during this part of the run, $\mathcal{D}^{\Gamma}_{\{\mathtt{ic},\mathtt{r}\}}$ can output $\mathtt{ic}$ at most n times before it outputs $\mathtt{r}$ and moves back to state $\emptyset$. Continuing to reason in this way, we can build up the run of $\mathcal{D}^{\Gamma}_{\{\mathtt{ic},\mathtt{r}\}}$ witnessing $[\![\mathcal{D}^{\Gamma}_{\{\mathtt{ic},\mathtt{r}\}}]\!](u) \leq n$.

Now assume for the sake of contradiction that there is some $u \in (\{\mathtt{ic}, \mathtt{r}\}^\Gamma)^*$ such that $[\![\mathcal{D}^{\Gamma}_{\{\mathtt{ic},\mathtt{r}\}}]\!](u) \leq n \in \mathbb{N}$ but $\mathrm{value}_B(u) > 2n+1$. Then there is some some counter γ and some subword v of length $2n + 2$ in u that has $2n + 2$ increments for γ but no resets for γ. Let q be the state of $\mathcal{D}^{\Gamma}_{\{\mathtt{ic},\mathtt{r}\}}$ when it starts reading v. If $\gamma \notin q$, then $\mathcal{D}^{\Gamma}_{\{\mathtt{ic},\mathtt{r}\}}$ will output only $\mathtt{ic}$ on v, contradicting $[\![\mathcal{D}^{\Gamma}_{\{\mathtt{ic},\mathtt{r}\}}]\!](u) \leq n$. If $\gamma \in q$, then $\mathcal{D}^{\Gamma}_{\{\mathtt{ic},\mathtt{r}\}}$ can reach state $\emptyset$ and output $\mathtt{r}$ at most once while reading v. Due to the length of v, there must be a subword of v of length at least $n + 1$ where $\mathcal{D}^{\Gamma}_{\{\mathtt{ic},\mathtt{r}\}}$ outputs only $\mathtt{ic}$, contradicting $[\![\mathcal{D}^{\Gamma}_{\{\mathtt{ic},\mathtt{r}\}}]\!](u) \leq n$. □

We now describe automata that read arbitrary counter actions and output hierarchical counter actions. We include these results for completeness and also to give an example of a more involved cost automaton construction. These results are not new. The first part of the proof is given in [CL10] and the second part was known but unpublished [CL12]. Readers familiar with the latest appearance record construction [GH82] will recognize similarities here; this is not surprising, since the latest appearance record construction is used to derive a more structured acceptance condition (going from a Muller condition condition to a parity condition, see Table 3.1), and here we are trying to derive more structured counter actions.

Lemma 2.17 ([CL10, CL12]). *For all Γ, there is a deterministic hB-automaton $\mathcal{H}_B^\Gamma$ such that $[\![\mathcal{H}_B^\Gamma]\!] \approx_\alpha \mathrm{value}_B$ and there is a history deterministic hS-automaton $\mathcal{H}_S^\Gamma$ such that $[\![\mathcal{H}_S^\Gamma]\!] \approx_\alpha \mathrm{value}_S$ with $\alpha(n) = k \cdot (n+1)^k$ where $k = |\Gamma|$.*

Proof. Fix some set $\Gamma = [1, k]$ of counters. Our first goal is to build a deterministic hB-automaton $\mathcal{H}_B^\Gamma$ with the same counters Γ such that $[\![\mathcal{H}_B^\Gamma]\!] \approx_\alpha \mathrm{value}_B^\Gamma$ but using hierarchical counter actions. We indicate the new hierarchical actions by writing $\mathtt{op}_i$ for the hierarchical action that performs action $\mathtt{op}$ on counter i, resets counter $i' \leq i$, and leaves $i' > i$ unchanged. We write $\mathtt{r}_0$ for the operation that leaves all counters unchanged.

The set of states in $\mathcal{H}_B^\Gamma$ are of the form $\langle \gamma_1, \ldots, \gamma_k \rangle$ where $\langle \gamma_1, \ldots, \gamma_k \rangle$ is a permutation of the counters in Γ (i.e. a permutation of $[1, k]$). Every state is accepting.

The idea is that $\mathcal{H}_B^\Gamma$ mimics the original action, but only on the highest counter (the counter that is the furthest to the right in the permutation). If the counter is reset, then it is moved to the beginning of the permutation.

Consider a state $q = \langle \gamma_1, \ldots, \gamma_k \rangle$ and an input letter $a \in \mathbb{B}^\Gamma$. We describe the edges in the transition relation starting from q. Let $i \in [1, k]$ be the largest index such that $\mathrm{pr}_{\gamma_i}(a) \neq \varepsilon$. Then

- $q \xrightarrow{a : \mathtt{r}_0} \langle \gamma_1, \ldots, \gamma_k \rangle$ if no such i exists;
- $q \xrightarrow{a : \mathtt{ic}_i} \langle \gamma_1, \ldots, \gamma_k \rangle$ if $\mathrm{pr}_{\gamma_i}(a) = \mathtt{ic}$;
- $q \xrightarrow{a : \mathtt{r}_i} \langle \gamma_i, \gamma_1, \ldots, \gamma_{i-1}, \gamma_{i+1}, \ldots, \gamma_k \rangle$ if $\mathrm{pr}_{\gamma_i}(a) = \mathtt{r}$.

We prove that $[\![\mathcal{H}_B^\Gamma]\!] \approx_\alpha \mathrm{value}_B$.

We start by showing that $[\![\mathcal{H}_B^\Gamma]\!] \leq \mathrm{value}_B$. Assume that there is some $u \in (\mathbb{B}^\Gamma)^*$ such that $[\![\mathcal{H}_B^\Gamma]\!](u) = n \in \mathbb{N}$. Then the run of $\mathcal{H}_B^\Gamma$ on u that outputs hierarchical actions w has a subsequence w' that contains n occurrences of $\mathtt{ic}_i$ and no higher counter operation. Since no higher counter operation appears in w', the counter γ indexed by position i in the permutation remains stable while $\mathcal{H}_B^\Gamma$ is reading the corresponding subsequence u' of u. Moreover, the actions $\mathtt{ic}_i$ are induced in w' by reading $\mathtt{ic}$ for γ in u', and there can be no $\mathtt{r}$ for γ in this subsequence, otherwise γ would not be stable in the permutation. This means that γ achieves value at least n in u, so $n = [\![\mathcal{H}_B^\Gamma]\!](u) \leq \mathrm{value}_B(u)$ for all u.

Next, we show $\mathrm{value}_B \preccurlyeq_\alpha [\![\mathcal{H}_B^\Gamma]\!]$. Assume for the sake of contradiction that there is $u \in (\mathbb{B}^\Gamma)^*$ such that $[\![\mathcal{H}_B^\Gamma]\!](u) < n$ but $\mathrm{value}_B(u) \geq \alpha(n)$. Let u' be a subsequence that contains $\alpha(n)$ increments of some counter γ with no reset for γ. Each $\mathtt{ic}$ for γ

induces $\mathtt{ic}_i$ where $\gamma = \gamma_i$ in the permutation. Because there are no resets for γ in u', γ can only move right in the permutation and this can only occur at most k times.

Since $\alpha(n) = k \cdot (n+1)^k$, there must be a subsequence u'' of u' with at least $(n+1)^k$ increments for γ during which the index i of γ in the permutation remains stable. While the position of γ remains stable at some position i, there are no resets of counters indexed by some $i' \geq i$. Hence, each increment of γ results in $\mathtt{ic}_i$. However, if some counter $\gamma_{i'}$ for $i' > i$ is incremented, then this induces a reset of counter i. We can show by induction on k that while the counters in positions $i' \geq i$ remain stable like this and $[\![\mathcal{H}_B^\Gamma]\!](u) < n$, there must be strictly less than $(n+1)^k$ increments for γ, which implies the desired contradiction.[5]

Our next goal is to build a history deterministic hS-automaton $\mathcal{H}_S^\Gamma$ using the same counters Γ such that $[\![\mathcal{H}_S^\Gamma]\!] \approx \mathrm{value}_S$ but with hierarchical counter actions. The set of states in $\mathcal{H}_S^\Gamma$ includes a sink state $\bot$ and all tuples of the form $(X, \langle \gamma_j, \ldots, \gamma_k \rangle)$ where $X \subseteq \Gamma$, $j = |X| + 1$, and $\langle \gamma_j, \ldots, \gamma_k \rangle$ is a permutation of $\Gamma \setminus X$. We write $\langle Y, \gamma_j, \ldots, \gamma_k \rangle$ to denote the permutation that lists the elements of Y in ascending order followed by $\gamma_j, \ldots, \gamma_k$. The initial state is $(\emptyset, \langle \Gamma \rangle)$. Every state except $\bot$ is accepting.

The *active counters* are the counters in the permutation $\gamma_j, \ldots, \gamma_k$. The *safe counters* are the counters in X. The idea is that the automaton mimics the original action on the highest active counter (the counter furthest to the right in the permutation). Once a counter reaches a high value, it gets check reset and moved to the safe set X where further increments are ignored. If $\mathtt{cr}$ or $\mathtt{r}$ is seen for a counter in the safe set, then this counter gets moved back to the beginning of the active permutation. If $\mathtt{r}$ is seen for an active counter, then it is moved to the beginning of the permutation. However, if $\mathtt{cr}$ is ever seen for an active counter, then the automaton moves immediately to the rejecting sink state. Thus, in order to maximize the value of the run, $\mathcal{H}_S^\Gamma$ must ensure that any counter that is actually check-reset in w is moved to the safe set before this occurs.

Consider a state $q = (X, \langle \gamma_j, \ldots, \gamma_k \rangle)$ and an input letter $a \in \mathbb{S}^\Gamma$. We describe the edges in the transition relation starting from q.

[5] The base case for $k = 1$ is trivial. For $k > 1$, the inductive hypothesis is that there are less than $(n+1)^{k-1}$ increments for γ when utilizing counters up to $k-1$ and keeping $[\![\mathcal{H}_B^\Gamma]\!](u) < n$. After such a block of increments, however, there could be an increment of $\gamma' = \gamma_k$ that induces a reset for lower counters and allows another application of the inductive hypothesis and another block of increments for γ. Since $[\![\mathcal{H}_B^\Gamma]\!](u) < n$ and γ' is never reset (since the positions $i' \geq i$ are stable in the permutation), there can be at most $n-1$ increments of γ'. This means we can account for at most $(n-1)(n+1)^{k-1} < (n+1)^k$ increments for γ.

If there is $i \in [j,k]$ such that $\mathrm{pr}_{\gamma_i}(a) = \mathtt{cr}$ then $q \xrightarrow{a:\mathtt{r}_0} \bot$ is the only possible transition.

Otherwise, let $Y = \Big\{\gamma \in X : \mathrm{pr}_\gamma(a) \in \{\mathtt{r}, \mathtt{cr}\}\Big\}$.

If $\mathrm{pr}_{\gamma_i}(a) = \varepsilon$ for every $i \in [j,k]$, then $q \xrightarrow{a:\mathtt{r}_0} (X \setminus Y, \langle Y, \gamma_j, \ldots, \gamma_k\rangle)$.

If $i \in [j,k]$ is the largest index in the active set of counters such that $\mathrm{pr}_{\gamma_i}(a) \neq \varepsilon$, then $q \xrightarrow{a:\mathtt{cr}_i} ((X \cup \{\gamma_i\}) \setminus Y, \langle Y, \gamma_j, \ldots, \gamma_{i-1}, \gamma_{i+1}, \ldots, \gamma_k\rangle)$ is a possible transition, as well as one of the following transitions depending on $\mathrm{pr}_{\gamma_i}(a)$:

- $q \xrightarrow{a:\mathtt{i}_i} (X \setminus Y, \langle Y, \gamma_j, \ldots, \gamma_k\rangle)$ if $\mathrm{pr}_{\gamma_i}(a) = \mathtt{i}$;
- $q \xrightarrow{a:\mathtt{r}_i} (X \setminus Y, \langle Y, \gamma_i, \gamma_j, \ldots, \gamma_{i-1}, \gamma_{i+1}, \ldots, \gamma_k\rangle)$ if $\mathrm{pr}_{\gamma_i}(a) = \mathtt{r}$.

Consider the family $(\vartheta_n)_{n\in\mathbb{N}}$ of translation strategies that wait to take the $\mathtt{cr}_i$ transition until counter i reaches value n. By Remark 2.11, it suffices to show that $[\![\mathcal{H}_S^\Gamma]\!] \preccurlyeq \mathrm{value}_S \preccurlyeq_\alpha [\![\mathcal{H}_S^\Gamma]\!]^\vartheta$.

We start by showing $[\![\mathcal{H}_S^\Gamma]\!] \leq \mathrm{value}_S$. Let $u \in (\mathbb{S}^\Gamma)^*$ and let w be some output sequence from a run of $\mathcal{H}_S^\Gamma$ on u. Assume that $\mathrm{value}_S(u) = n \in \mathbb{N}$. Then there is some counter γ and some subword u' of u beginning with a reset for γ (or the initial position in u) and ending with a check-reset for γ such that there are n increments and no intermediate resets or check-resets for γ in u'. Because u' starts with the initial position or a reset for γ, the state $(X, \langle\gamma_j, \ldots, \gamma_k\rangle)$ when $\mathcal{H}_S^\Gamma$ starts to read u' must have $\gamma = \gamma_i$ for some $i \in [j,k]$. If the automaton $\mathcal{H}_S^\Gamma$ does not move γ to the safe set while reading u', then it will move to state $\bot$ at the end of u' so the run has value 0. Otherwise, γ is moved to the safe set while $\mathcal{H}_S^\Gamma$ is reading u'. Before it is moved to the safe set, it may move right from position i to position $i+1$ in the permutation, but doing so resets the value of counter i and counter $i+1$. Let i' be the highest index of γ before it is moved to the safe set. In this position, there can be at most n increments for $\mathtt{i}_{i'}$ (induced by the original increments for γ in u'). Hence, counter i' is check-reset with value at most n when γ is moved to the safe set. Since the value of w is the minimum checked counter value, this means that $[\![\mathcal{H}_S^\Gamma]\!](u) \leq n$ as desired.

Next, we show $\mathrm{value}_S \preccurlyeq_\alpha [\![\mathcal{H}_S^\Gamma]\!]^\vartheta$. Assume for the sake of contradiction that there is some u with $\mathrm{value}_S(u) \geq \alpha(n)$ but $[\![\mathcal{H}_S^\Gamma]\!]^\vartheta(u) < n$.

If $\mathrm{value}_S(u) = \infty$, then there is no $\mathtt{cr}$ for any counter in u. This means that for all n, the run driven by ϑ_n is accepting and has value n, so $[\![\mathcal{H}_S^\Gamma]\!]^\vartheta(u) = \infty$ as well.

If $\mathrm{value}_S(u) \in \mathbb{N}$ and at least $\alpha(n)$, then consider any subsequence u' of u that begins after a reset for some counter γ (or is the initial position in u) and ends with a check-reset for γ. There must be at least $\alpha(n)$ increments and no intermediate resets

or check-resets for γ in u' since $\text{value}_S(u) \geq \alpha(n)$. Because u' starts with the initial position or a reset for γ, the state $(X, \langle \gamma_j, \ldots, \gamma_k \rangle)$ when $\mathcal{H}_S^\Gamma$ starts to read u' must have $\gamma = \gamma_i$ for some $i \in [j, k]$.

Since there are no intermediate resets or check-resets for γ on u', γ can only be moved right in the active permutation during any run of $\mathcal{H}_S^\Gamma$ on u' (unless it is check-reset, in which case it is moved to the safe set, and remains there until the end of u'). Recall that the translation strategy ϑ_n moves an active counter into the safe set once it reaches value n. After at most $(n+1)^k$ increments for γ, some counter must have reached value n and have been moved into the safe set (this could be shown formally by induction on k). This means that after at most $(n+1)^k$ increments, γ must have reached value n and have been moved to the safe set, or γ must have been moved right in the permutation because another counter (higher in the permutation) was moved to the safe set. Since there are k counters, $k(n+1)^k$ increments for γ ensure that γ has been moved into the safe set after being check-reset with a value of n. Because this is true for any such subsequence u', the run driven by ϑ_n has value at least n, a contradiction. □

2.3 Extension to finite trees

Building on the results described earlier, Colcombet and Löding extended the theory of regular cost functions to finite trees [CL10], so versions of Theorem 2.8, Proposition 2.9, and Theorem 2.10 also hold for regular cost functions over finite trees.

For the purposes of this thesis, we do not need to define cost automata over finite trees formally, but the idea is that a traditional top-down tree automaton is enriched with a finite set of counters. In a *nondeterministic cost automaton over finite trees*, one copy of the automaton is sent in each direction of the tree with the transition relation describing the state and counter action associated with each copy. The B-value (respectively, S-value) of the run is the maximum (respectively, minimum) checked value across all branches in the run, and the value of the input tree is the minimum (respectively, maximum) value over all accepting runs.

As part of the development of this theory over finite trees, *alternating cost automata* over finite trees are also utilized. An alternating automaton generalizes the nondeterministic model by allowing the automaton to send more than one copy (or none at all) in each direction. Colcombet and Löding show that alternation does not increase the power of the automaton model, and that any alternating cost automa-

ton can be *simulated* by a nondeterministic cost automaton using either the B- or S-semantics.

Theorem 2.18 ([CL10, Theorem 12]). *It is effectively equivalent for a cost function f over finite trees to be recognized by alternating B-, hB-, S-, and hS-automata, as well as their nondeterministic versions. We call such a cost function a* regular cost function over finite trees.

Using this simulation and duality result, closure properties and decidability of the domination preorder can also be shown.

Proposition 2.19 ([CL10, Lemma 11]). *Regular cost functions over finite trees are closed under* min, max, inf*-projection, and* sup*-projection.*

Theorem 2.20 ([CL10, Theorem 13]). *Given regular cost functions f and g over finite trees, it is decidable whether or not $f \preccurlyeq g$.*

Instead of the algebraic approach used to prove these results over words, these results over trees exploit a connection between automata and two-player turn-based games. This approach will be explained in the next chapter since it is essential for the development of the theory of regular cost functions over infinite trees as well.

2.4 Discussion

In this chapter, we have summarized the main results in the theory of regular cost functions over finite words and finite trees, including the duality theorem (Theorems 2.8 and 2.18) and the decidability of $\preccurlyeq$ (Theorems 2.10 and 2.20). We conclude this chapter by discussing some related work that influenced (and continues to influence) the theory of regular cost functions.

Variants of cost automata have been studied under many different names including distance automata [Has82], nested distance desert automata [Kir05], BS-automata [BC06], and R-automata [AKY08]. This theory of regular cost functions, however, grew out of two main lines of work: research by Hashiguchi [Has82], Kirsten [Kir05], and others who were studying problems that could be reduced to limitedness of functions (the most famous problem like this being the star height problem discussed in Chapter 1); and research by Bojańczyk and Colcombet [Boj04, BC06] on extensions of monadic second-order logic that can assert properties related to boundedness.

Many of the algebraic methods that are central to Colcombet's theory of regular cost functions over finite words were developed earlier in the context of *weighted automata*, a model that allows each transition to be assigned a weight from some set S in a semiring $(S, \oplus, \otimes, \mathrm{id}_\oplus, \mathrm{id}_\otimes)$, rather than the counter operations seen in cost automata (see, e.g. [DKV09]). A *weighted automaton over* $(S, \oplus, \otimes, \mathrm{id}_\oplus, \mathrm{id}_\otimes)$ is a nondeterministic automaton such that each transition is assigned some weight in S, and the weight of a path is found by $\otimes$-multiplying the weights of the transitions on that path. The weight of a word is the $\oplus$-sum of the weights of all accepting paths (with weight $\mathrm{id}_\oplus$ assigned to any word that is not accepted).

Classical nondeterministic finite state automata can be viewed as weighted automata over the Boolean semiring $(\{0,1\}, \vee, \wedge, 0, 1)$ where each transition has weight 1 (this makes sense since in the traditional model of finite automata a word is either accepted or rejected). One of the most common semirings in the weighted automata literature, however, is the *min-plus* or *tropical* semiring $(\mathbb{R} \cup \{\infty\}, \min, +, \infty, 0)$ introduced by Simon [Sim78] in his study of the finite power property. Indeed, the distance automata described earlier can be viewed as a special case of weighted automata over the tropical semiring in which the weights are restricted to 0 or 1, since the weight of an accepted word is obtained by taking the minimum (over all accepting runs) of the sum of the weights on the run. Following Hashiguchi's proof of the decidability of limitedness for distance automata [Has82], Leung [Leu88] gave an alternative proof using an extension of the tropical semiring and the new idea of a stabilization operator $\sharp$. These ideas continued to develop through the work of Simon, Leung, Kirsten, and others, and their influence can be seen in Colcombet's theory of regular cost functions.

The notion of the dual B and S forms of automata with counters also came from earlier work, this time due to Bojańczyk and Colcombet [BC06]. In this work, they introduced ωBS-automata (which are actually defined on infinite rather than finite words). An *ωBS-automaton* is a nondeterministic finite state automaton with a finite set of counters Γ (partitioned into a set of B-counters Γ_B and a set of S-counters Γ_S). Similar to cost automata, the transitions are labelled with actions `i`, `r`, or ε. Like classical automata, however, these automata recognize languages rather than functions.

The acceptance condition describes desired asymptotic behaviour for the counter values. Given a run ρ of an ωBS-automaton and a counter $\gamma \in \Gamma$, let $\mathrm{value}(\rho, \gamma)_i$ be the value of counter γ immediately before the ith reset. This induces a sequence of values in $\mathbb{N}_\infty$ (denoted $\mathrm{value}(\rho, \gamma)$) that is used to define the acceptance condition. A

run ρ is accepting if for every counter $\gamma \in \Gamma$, the sequence value(ρ, γ) is infinite (i.e. γ is reset infinitely many times) and

$$\text{if } \gamma \in \Gamma_B \text{ then } \gamma \text{ is } \underline{\text{b}}\text{ounded:} \quad \limsup_i \text{value}(\rho, \gamma)_i < \infty,$$
$$\text{if } \gamma \in \Gamma_S \text{ then } \gamma \text{ is } \underline{\text{s}}\text{trongly unbounded:} \quad \liminf_i \text{value}(\rho, \gamma)_i = \infty.$$

This is where the B and S notation comes from. If the automaton has counters of only one type, then it is called an ωB-automaton or ωS-automaton as appropriate. For instance, an ωB-automaton could define the language consisting of all words $a^{n_1}ba^{n_2}b\cdots$ such that $(n_i)_{i\in\mathbb{N}}$ is bounded. The word $(a^{100}b)^\omega$ would be in this language, but $a^1ba^2ba^3b\cdots$ would not.

The class of ωBS-automata is closed under union, intersection, and projection, but fails to be closed under complement [BC06]. However, the complement of a language definable using an ωB-automaton is an ωS-automaton, and vice versa. This is a complicated proof and results in a nonelementary blowup of the state space. However, a proof of Theorem 2.8 and Theorem 2.10 can be extracted from this result. For example, say we are trying to show that some B-automaton $\mathcal{A}$ which reads finite words over $\mathbb{A}$ satisfies $[\![\mathcal{A}]\!] \preccurlyeq 0$ (the simplest case of deciding the domination preorder). Consider the ωB-automaton $\mathcal{A}'$ based on $\mathcal{A}$ defined as follows: to every final state of $\mathcal{A}$, add a new edge that when reading a new symbol \$, resets all counters and moves the automaton back to its initial state. Then $[\![\mathcal{A}]\!] \preccurlyeq 0$ if and only if $L(\mathcal{A}') = (\mathbb{A}\cup\{\$\})^\omega$ if and only if $L(\mathcal{A}'') = \emptyset$ where $\mathcal{A}''$ is the complement of $\mathcal{A}'$. By the results in [BC06], the ωS-automaton $\mathcal{A}''$ can be found effectively and the emptiness of $L(\mathcal{A}'')$ is decidable. Hence, $[\![\mathcal{A}]\!] \preccurlyeq 0$ is decidable.

The motivation for studying these automata came from an extension of monadic second-order logic with an *unbounding quantifier* $\mathbb{U}$ (originally introduced in its negated form in [Boj04]), where $\mathbb{U}X.\varphi(X)$ expresses "there is no bound on the size of sets X satisfying $\varphi(X)$". Fragments of this logic (with some restrictions on negation) correspond to definability via ωB-automata and ωS-automata, and hence these fragments are decidable. However, no automaton model is known for the full logic (and in fact, it has been shown that no nondeterministic automaton model with a Borel acceptance condition can suffice [HST10]), so the decidability of the full logic remains open.

This prompted Toruńczyk [Tor11] to introduce a related algebraic structure (which he also calls a stabilization semigroup) to reason about these boundedness problems. His approach is inspired by the more topological viewpoint of Leung, and uses the idea of profinite words. It provides yet another proof of Theorem 2.10, but the

decidability of the full monadic second-order logic with $\mathbb{U}$ over infinite words remains open. Despite these connections to the theory of regular cost functions, the work on ωBS-automata and the unbounding quantifier appears to be largely orthogonal to Colcombet's work. Indeed, when moving from words to trees, the decidability results in these two lines of research no longer appear to be reducible to one another. It would be interesting to study more closely the relationship between these lines of work in the future.

Chapter 3

Cost Automata and Games

Cost automata over infinite structures combine a classical acceptance condition with the counting features described in the previous chapter. We capture these conditions in the form of an objective that describes how to assign values to plays based on both the classical condition and the counter values. We begin by describing this automaton model over infinite words in Section 3.1 and then over infinite trees in Section 3.3.

In Section 3.2, we also define two-player cost games. These games are infinite-duration two-player games on labelled graphs where one player seeks to minimize the value of the play and the other player seeks to maximize the value based on the objective. There is a strong connection between automata over infinite trees and games in the classical theory, and this connection extends to the cost setting. Indeed, the semantics of cost automata over infinite trees are defined in terms of these cost games. Moreover, we show that we can convert between cost games with different objectives, in analogy to classical conversions between different types of winning conditions, and this allows us to prove a duality theorem (Theorem 3.15) for alternating cost-parity automata over infinite trees.

Many of the proofs in this chapter are adaptations of the corresponding results for cost automata over finite trees in [CL10], and were first presented in [VB11].

3.1 Automata on infinite words

For automata over finite words, the notion of an accepting run is very natural: a run is accepting if the final state of the automaton after reading the entire word belongs to some special set of accepting states F. When moving to infinite structures, however, this condition no longer makes sense since there is no final state in the run.

Name	Description	Condition
Büchi	$F \subseteq Q$	$\mathrm{Inf}(\rho) \cap F \neq \emptyset$
co-Büchi	$F \subseteq Q$	$\mathrm{Inf}(\rho) \cap F = \emptyset$
Muller	$\mathcal{F} \subseteq \mathscr{P}(Q)$	$\exists F \in \mathcal{F}. F = \mathrm{Inf}(\rho)$
Rabin	$\mathcal{F} \subseteq \mathscr{P}(Q) \times \mathscr{P}(Q)$	$\exists (E,F) \in \mathcal{F}. \mathrm{Inf}(\rho) \cap E = \emptyset \wedge \mathrm{Inf}(\rho) \cap F \neq \emptyset$
Streett	$\mathcal{F} \subseteq \mathscr{P}(Q) \times \mathscr{P}(Q)$	$\forall (E,F) \in \mathcal{F}. \mathrm{Inf}(\rho) \cap F \neq \emptyset \Rightarrow \mathrm{Inf}(\rho) \cap E \neq \emptyset$
parity	$P \subseteq \mathbb{N}$, $\Omega : Q \longrightarrow P$	$\max(\Omega(\mathrm{Inf}(\rho)))$ is even

Table 3.1. Common acceptance conditions for automata over infinite words.

Instead, the acceptance condition for an automaton $\mathcal{A} = \langle Q, \mathbb{A}, q_0, \mathrm{Acc}, \Delta \rangle$ over infinite words is usually based on the set of states that occur infinitely many times in a run ρ of $\mathcal{A}$, which we denote by $\mathrm{Inf}(\rho)$. Table 3.1 summarizes some of the most common acceptance conditions, and the finite description Acc of this condition. We assume some familiarity with these acceptance conditions and the theory of regular languages over infinite words (see [Tho97] for a survey).

In this thesis, we concentrate on the *parity condition* which can be defined by a set of *priorities* $P \subseteq \mathbb{N}$, together with a mapping $\Omega : Q \longrightarrow P$ from states to priorities. A run ρ satisfies the parity condition if the maximum priority that occurs infinitely often in the run (after mapping states to priorities using Ω) is even.[1]

We remark that the parity condition over priorities $P = [0, 2j+1]$ can be viewed as a special case of a Rabin condition, where $\mathcal{F} = \{(E_0, F_0), \ldots, (E_j, F_j)\}$ and $E_i = \{q : \Omega(q) > 2i\}$ and $F_i = \{q : \Omega(q) = 2i\}$ for $i \in [0, j]$. It is straightforward to see that a run satisfies the parity condition if and only if there exists some $(E, F) \in \mathcal{F}$ such that $\mathrm{Inf}(\rho) \cap E = \emptyset$ and $\mathrm{Inf}(\rho) \cap F \neq \emptyset$. Likewise, the parity condition can be viewed as a special case of the Streett condition (the dual of the Rabin condition) where $E_i = \{q : \Omega(q) > 2i+1\}$ and $F_i = \{q : \Omega(q) = 2i+1\}$ for $i \in [0, j]$.

A *Büchi* (respectively, *co-Büchi*) condition described by accepting states F can be seen as a special case of a parity condition where $\Omega : Q \longrightarrow [1, 2]$ (respectively, $\Omega : Q \longrightarrow [0, 1]$) and

$$\Omega(q) = \begin{cases} 1 \text{ (respectively, } 0) & \text{if } q \notin F \\ 2 \text{ (respectively, } 1) & \text{if } q \in F \end{cases}.$$

We will usually view a Büchi and co-Büchi condition as a parity condition like this.

[1] This condition is also known as the *max-parity* condition to distinguish it from the min-parity condition which requires the minimum priority occurring infinitely often to be even. In this thesis, parity condition always means max-parity condition.

Moreover, it turns out that any language definable using nondeterministic automata over infinite words with one of the conditions in Table 3.1 can be expressed using a deterministic parity automaton (see, e.g. [GTW02]).

Theorem 3.1. *It is effectively equivalent for a language L of infinite words to be recognizable by the following types of automata:*

- *nondeterministic Büchi automata;*
- *deterministic or nondeterministic Rabin or Streett automata;*
- *deterministic or nondeterministic Muller automata;*
- *deterministic or nondeterministic parity automata.*

We say such a language L is regular (*or* ω-regular).

The fact that deterministic parity automata capture all regular languages of infinite words is one motivation for using automata with the parity acceptance condition. Other motivations arise in the context of games, which will be examined in Chapter 4.

3.1.1 Objectives

In this thesis, we will combine the classical parity condition with the cost features described in Chapter 2. Following [CL10], we define a general notion of objective that will take the place of acceptance conditions for automata and winning conditions for games.

An *objective* O is a tuple $\langle \mathbb{C}, f, \mathit{goal} \rangle$ where $\mathbb{C}$ is a finite alphabet of actions, $f : \mathbb{C}^\omega \longrightarrow \mathbb{N}_\infty$ is a *valuation* that maps sequences of actions to a value in $\mathbb{N}_\infty$, and $\mathit{goal} \in \{\min, \max\}$ describes how f should be optimized.

For example, a parity condition with priorities P is described by the *parity objective* $\mathrm{Cost}^P_{\mathrm{parity}} := \langle P, \mathrm{cost}^P_{\mathrm{parity}}, \min \rangle$ where $P \subseteq \mathbb{N}$ is a finite set of priorities and $\mathrm{cost}^P_{\mathrm{parity}} : P^\omega \longrightarrow \{0, \infty\}$ maps a word $u \in P^\omega$ to 0 if the maximum infinitely-occurring priority in u is even and ∞ otherwise. For instance, $\mathrm{cost}^{[1,2]}_{\mathrm{parity}}((12)^\omega) = 0$ but $\mathrm{cost}^{[1,2]}_{\mathrm{parity}}(2^{100}1^\omega) = \infty$. The goal is min since satisfying the parity condition corresponds to minimizing the valuation $\mathrm{cost}^P_{\mathrm{parity}}$.

We want to enrich this parity objective with conditions involving counters so that values come from $\mathbb{N}_\infty$ (instead of only $\{0, \infty\}$). We do this by defining objectives that combine a classical parity condition with particular atomic counter actions and valuations.

- The *B-parity objective* (over counters Γ and priorities P) is defined as $\mathrm{Cost}_B^{\Gamma,P} := \langle \mathbb{B}^\Gamma \times P, \mathrm{cost}_B^{\Gamma,P}, \min \rangle$ such that

$$\mathrm{cost}_B^{\Gamma,P}(u) := \sup\left(C(u) \cup \left\{\mathrm{cost}^P_{\mathrm{parity}}(u)\right\}\right),$$

  where $C(u)$ is the set of checked counter values in u (see Section 2.2) and $\mathrm{cost}^P_{\mathrm{parity}}(u)$ is the function described above that interprets the parity condition on the projection of u to its last component. The atomic actions for each counter are simple B-actions from $\mathbb{B} = \{\varepsilon, \mathtt{ic}, \mathtt{r}\}$. If the parity condition is satisfied, then the value is the supremum of the checked counter values; otherwise, the counters are ignored and the value is ∞. For example, if $u = \Big((\mathtt{ic},2)(\mathtt{ic},2)(\varepsilon,1)(\mathtt{r},2)(\mathtt{ic},1)\Big)^\omega$, then $\mathrm{cost}_B^{\{1\},[1,2]}(u) = \sup(\{1,2,3\} \cup \{0\}) = 3$.

- The *hB-parity objective* is a variant of the B-parity objective with *hierarchical counters* $\Gamma = [1,k]$ such that whenever $\gamma \in \Gamma$ is incremented or reset, all $\gamma' < \gamma$ are reset. Formally, $\mathrm{Cost}_{hB}^{\Gamma,P} := \langle H_B^\Gamma \times P, \mathrm{cost}_B^{\Gamma,P}, \min \rangle$ where

$$H_B^\Gamma := \left\{c \in \mathbb{B}^\Gamma : \mathrm{pr}_\gamma(c) \neq \varepsilon \text{ implies } \mathrm{pr}_{\gamma'}(c) = \mathtt{r} \text{ for all } \gamma' < \gamma\right\}.$$

- The *S-parity objective* (over counters Γ and priorities P) is defined as $\mathrm{Cost}_S^{\Gamma,P} := \langle \mathbb{S}^\Gamma \times P, \mathrm{cost}_S^{\Gamma,P}, \max \rangle$ where

$$\mathrm{cost}_S^{\Gamma,P}(u) := \inf\left(C(u) \cup \left\{\overline{\mathrm{cost}^P_{\mathrm{parity}}}(u)\right\}\right)$$

  and $\overline{\mathrm{cost}^P_{\mathrm{parity}}}(u)$ is 0 (respectively, ∞) if $\mathrm{cost}^P_{\mathrm{parity}}(u)$ is ∞ (respectively, 0). The atomic actions for each counter are simple S-actions from $\mathbb{S} = \{\varepsilon, \mathtt{i}, \mathtt{r}, \mathtt{cr}\}$. If the parity condition is not satisfied then the counters are ignored and the value assigned is 0; otherwise, the minimum checked value is used (∞ if no counter is checked). For example, if $u = (\mathtt{i},0)(\mathtt{r},1)(\mathtt{i},0)(\varepsilon,1)(\mathtt{i},1)(\mathtt{cr},0)((\mathtt{i},0))^\omega$ then $\mathrm{cost}_S^{\{1\},[0,1]}(u) = \inf(\{2\} \cup \{\infty\}) = 2$.

- The *hS-parity objective* is a variant of the S-parity objective with hierarchical counters $\Gamma = [1,k]$. Formally, $\mathrm{Cost}_{hS}^{\Gamma,P} := \langle H_S^\Gamma \times P, \mathrm{cost}_S^{\Gamma,P}, \max \rangle$ where

$$H_S^\Gamma := \left\{c \in \mathbb{S}^\Gamma : \mathrm{pr}_\gamma(c) \neq \varepsilon \text{ implies } \mathrm{pr}_{\gamma'}(c) = \mathtt{r} \text{ for all } \gamma' < \gamma\right\}.$$

Given an objective $O = \langle \mathbb{C}, f, \mathit{goal} \rangle$, the *dual objective* $\overline{O}$ is obtained from O by switching min to max and vice versa.

The objectives above (together with their dual versions) are the most common objectives, but additional objectives will be introduced as needed (see, e.g. Section 4.2).

3.1.2 Cost automata on infinite words

A *nondeterministic cost automaton* $\mathcal{A}$ on infinite words is a tuple

$$\langle Q, \mathbb{A}, q_0, O, \Delta \rangle$$

where Q is a finite set of states, $\mathbb{A}$ is a finite alphabet, $q_0 \in Q$ is the initial state, $O = \langle \mathbb{C}, f, \mathit{goal} \rangle$ is an objective, and $\Delta \subseteq Q \times \mathbb{A} \times \mathbb{C} \times Q$ is the transition relation. If for all $(q, a) \in Q \times \mathbb{A}$ there is a unique element $(c, r) \in \mathbb{C} \times Q$ such that $(q, a, c, r) \in \Delta$, then we say that $\mathcal{A}$ is *deterministic*.

We will often refer to a cost automaton by its objective, i.e. writing O automaton for a cost automaton with objective O. For notational simplicity, we will write, e.g. hB-$[1, 2]$ or hB-Büchi instead of $\mathrm{Cost}_{hB}^{\Gamma,[1,2]}$. We will also use the term *cost-parity automaton* to describe an automaton that has one of the four standard objectives described above.

A *run* ρ of $\mathcal{A}$ over *input* $u = a_1 a_2 \cdots \in \mathbb{A}^\omega$ is $(q_i, a_{i+1}, c_{i+1}, q_{i+1})_{i \in \mathbb{N}} \in \Delta^\omega$, an infinite word that describes a possible sequence of states and actions during the operation of $\mathcal{A}$ on u.[2] The *output* $\mathrm{out}(\rho)$ from the run ρ is $c_1 c_2 \cdots \in \mathbb{C}^\omega$.

The objective $O = \langle \mathbb{C}, f, \mathit{goal} \rangle$ is used to assign a value to runs and words. The *value of a run* ρ is $\mathrm{value}(\rho) := f(\mathrm{out}(\rho))$. The *value of a word* u is

$$[\![\mathcal{A}]\!](u) := \begin{cases} \inf \{\mathrm{value}(\rho) : \rho \text{ is a run of } \mathcal{A} \text{ over } u\} & \text{if } \mathit{goal} = \min \\ \sup \{\mathrm{value}(\rho) : \rho \text{ is a run of } \mathcal{A} \text{ over } u\} & \text{if } \mathit{goal} = \max \end{cases}.$$

Thus, $[\![\mathcal{A}]\!] : \mathbb{A}^\omega \longrightarrow \mathbb{N}_\infty$ is a function that maps words $u \in \mathbb{A}^\omega$ to a value in $\mathbb{N}_\infty$. If $[\![\mathcal{A}]\!] \approx g$ then we say that $\mathcal{A}$ *recognizes* g.

If we want to emphasize that the objective O is a B-parity objective or S-parity objective, we will write $[\![\mathcal{A}]\!]_B$ or $[\![\mathcal{A}]\!]_S$, respectively. Notice that the definition above coincides with the semantics of B- and S-automata described in the previous chapter. For instance, recall that a nondeterministic B-automaton maps a word to the minimum of the values over all accepting runs on that word. Since non-accepting runs are immediately assigned value ∞ by the B-objectives, $\inf \{\mathrm{value}(\rho) : \rho \text{ is a run of } \mathcal{A} \text{ over } u\}$ is exactly the minimum of the values over all accepting runs.

Remark 3.2. One peculiarity with this model is that counter and priority actions both occur on transitions, whereas priorities usually label states. We remark that it is straightforward to translate between transition-labelled automata and the more common state-labelled automata, at the price of increasing the number of states.

[2] We require that a run is infinite: if there is no infinite sequence $(q_i, a_{i+1}, c_{i+1}, q_{i+1})_{i \in \mathbb{N}} \in \Delta^\omega$ describing the operation of $\mathcal{A}$ on $u = a_1 a_2 \cdots \in \mathbb{A}^\omega$, then there is no run of $\mathcal{A}$ on u.

3.1.3 History determinism

The notion of history determinism in Section 2.2 extends to cost automata over infinite words in a natural way.

Given a cost automaton $\mathcal{A} = \langle Q, \mathbb{A}, q_0, \langle \mathbb{C}, f, \mathit{goal} \rangle, \Delta \rangle$ and a family of translation strategies $(\vartheta_n)_{n\in\mathbb{N}}$ with $\vartheta_n : \mathbb{A}^* \times \mathbb{A} \rightarrowtail \Delta$, we define

$$[\![\mathcal{A}]\!]^{\vartheta}(u) := \begin{cases} \inf\left\{n : \text{value}(\tilde{\vartheta}_n(u)) \leq n\right\} & \text{if } \mathit{goal} = \min \\ \sup\left\{n : \text{value}(\tilde{\vartheta}_n(u)) \geq n\right\} & \text{if } \mathit{goal} = \max \end{cases}$$

where $\tilde{\vartheta}_n(u) \in \Delta^\omega$ is the run driven by ϑ_n on a word $u \in \mathbb{A}^\omega$ (we restrict to translation strategies for which $\tilde{\vartheta}_n$ is a valid run for all $u \in \mathbb{A}^\omega$). Then $\mathcal{A}$ is *α-history-deterministic* if and only if $[\![\mathcal{A}]\!]^{\vartheta} \approx_\alpha [\![\mathcal{A}]\!]$.

Equivalently, $\mathcal{A}$ is α-history-deterministic if there is a family of translation strategies $(\vartheta_n)_{n\in\mathbb{N}}$ such that for all $u \in \mathbb{A}^\omega$,

$$\begin{aligned} &\text{if } \mathit{goal} = \min \text{ and } [\![\mathcal{A}]\!](u) \leq n \text{ then } \text{value}(\tilde{\vartheta}_{\alpha(n)}(u)) \leq \alpha(n), \\ &\text{if } \mathit{goal} = \max \text{ and } [\![\mathcal{A}]\!](u) \geq \alpha(n) \text{ then } \text{value}(\tilde{\vartheta}_n(u)) \geq n. \end{aligned}$$

Note that the runs driven by the translation strategies are now infinite and the definition allows arbitrary objectives, but otherwise the definition matches the finite word case. Remark 2.11 still applies and will be used in the lemmas below.

Many classical results about regular languages of infinite words can be lifted to the cost setting. For instance, history deterministic cost-Büchi automata are strictly less expressive than nondeterministic cost-Büchi automata (just as deterministic Büchi automata are strictly less expressive than nondeterministic Büchi automata over infinite words), but history deterministic cost-parity automata recognize the entire class of regular cost functions over infinite words.

Theorem 3.3 ([Col12b]). *It is effectively equivalent for a cost function over infinite words to be recognizable by the following types of automata:*

- *nondeterministic B-Büchi and S-Büchi automata;*
- *nondeterministic or history deterministic B-parity and S-parity automata.*

We call such a cost function a regular cost function over infinite words.

Another well known classical result states that a regular language L of infinite words is recognizable by a deterministic Büchi automaton if and only if $L = \lim U$ for some regular language U where

$$\lim U = \{u : u(0) \cdots u(i) \in U \text{ for infinitely many } i \in \mathbb{N}\}.$$

The following lemma can be viewed as a generalization of part of this result.

Lemma 3.4. *Let g be a regular cost function over finite words and let*

$$f(u) = \inf\{n : \exists \textit{ infinitely many prefixes } v \textit{ of } u \textit{ such that } g(v) \leq n\},$$
$$f'(u) = \sup\{n : \exists \textit{ infinitely many prefixes } v \textit{ of } u \textit{ such that } g(v) \geq n\}$$

be cost functions over infinite words. Then f (respectively, f') is recognizable by a history deterministic hB-Büchi (respectively, hS-Büchi) automaton.

Proof. By Theorem 2.8, we can assume that there is a α_{hd}-history-deterministic hB-automaton $\mathcal{A}_{\text{fin}}$ (over finite words) with a family ϑ of translation strategies such that for all u we have

$$f(u) \approx_\alpha \inf\{n : \exists \text{ infinitely many prefixes } v \text{ of } u \text{ such that } [\![\mathcal{A}_{\text{fin}}]\!](v) \leq n\}.$$

We can view this $\mathcal{A}_{\text{fin}}$ as a hB-$[1,2]$ automaton $\mathcal{A}$ over infinite words, with the same structure but where the accepting states in $\mathcal{A}_{\text{fin}}$ now output priority 2 and all other states output priority 1. We claim $\mathcal{A}$ is equivalent to f and is history deterministic (witnessed by the same family ϑ of translation strategies). By Remark 2.11, it suffices to show that $[\![\mathcal{A}]\!]^\vartheta \preccurlyeq f \preccurlyeq [\![\mathcal{A}]\!]$.

Let $\beta := \alpha_{\text{hd}} \circ \alpha$. We first prove $[\![\mathcal{A}]\!]^\vartheta \preccurlyeq_\beta f$. Assume f is bounded by N on some set U of input words. Then for every $u \in U$, there must be some infinite set of indices I such that $[\![\mathcal{A}_{\text{fin}}]\!]$ is bounded by $\alpha(N)$, and hence $[\![\mathcal{A}_{\text{fin}}]\!]^\vartheta$ is bounded by $\beta(N)$ on prefixes $u(0)\cdots u(i)$ for all $i \in I$. This means that using the translation strategy $\vartheta_{\beta(N)}$ to drive the automaton $\mathcal{A}_{\text{fin}}$ on $u(0)\cdots u(i)$ results in an accepting run bounded by $\beta(N)$. Moreover, because $\vartheta_{\beta(N)}$ deterministically specifies how to construct the run, the runs driven by $\vartheta_{\beta(N)}$ on prefixes $u(0)\cdots u(i)$ and $u(0)\cdots u(i')$ are identical on any shared prefix. Thus, this same translation strategy $\vartheta_{\beta(N)}$ can be used to drive an infinite run of $\mathcal{A}$ on u that witnesses infinitely many priority 2 (at each $i \in I$) and has value bounded by $\beta(N)$.

Now we show that $f \preccurlyeq_\alpha [\![\mathcal{A}]\!]$. If $[\![\mathcal{A}]\!]$ is bounded by N on some set U, then for any $u \in U$, there is a run of $\mathcal{A}$ on u with value bounded by N and with infinitely many

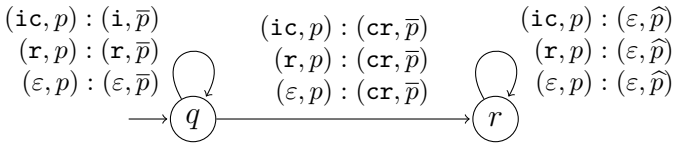

Figure 3.2. History deterministic S-$[i+1, j+1]$ automaton recognizing $\mathrm{cost}_B^{\{1\},[i,j]}$ (see Lemma 3.5).

priority 2 at positions indexed by some infinite set I. Thus, for all $i \in I$, there is a run of $\mathcal{A}_{\mathrm{fin}}$ on $u(0) \cdots u(i)$ that ends in an accepting state and has valued bounded by N. Hence, $\inf\{n : \exists \text{ infinitely many prefixes } v \text{ of } u \text{ such that } [\![\mathcal{A}_{\mathrm{fin}}]\!](v) \leq n\} \leq N$, so we have $f(u) \leq \alpha(N)$.

The proof for f' is similar, so we omit it. □

As discussed in Section 2.2.4, we often use history deterministic automata as transducers that convert between different types of counter actions while preserving the value. We begin with examples of cost-parity automata that approximate the value of a sequence of counter actions (say, from a B-parity automaton) by using different actions (say, from an S-parity automaton). We present these examples as part of lemmas that will be used later in this thesis.

Lemma 3.5. *For all $i < j$,*[3] *there is a* id*-history-deterministic S-$[i+1, j+1]$ automaton $\mathcal{A}_{\mathrm{cost}_B^{\Gamma,[i,j]}}$ (respectively, B-$[i+1, j+1]$ automaton $\mathcal{A}_{\mathrm{cost}_S^{\Gamma,[i,j]}}$) such that*

$$[\![\mathcal{A}_{\mathrm{cost}_B^{\Gamma,[i,j]}}]\!]_S = \mathrm{cost}_B^{\Gamma,[i,j]} \quad \text{and} \quad [\![\mathcal{A}_{\mathrm{cost}_S^{\Gamma,[i,j]}}]\!]_B = \mathrm{cost}_S^{\Gamma,[i,j]}\,.$$

Proof. We first define in Figure 3.2 an S-parity automaton $\mathcal{A}_{\mathrm{cost}_B^{\{1\},[i,j]}}$ recognizing $\mathrm{cost}_B^{\{1\},[i,j]}$. This automaton reads words over $\mathbb{B} \times [i, j]$ (the alphabet of actions for a B-parity automaton with one counter and priorities $[i, j]$). We label a transition with $(c, p) : (c', p')$ if on input (c, p), the counter action output is c' and the priority output is p'. We write $\overline{p}$ for $p + 1$, and $\widehat{p}$ for some fixed even priority in $[i + 1, j + 1]$.

The history deterministic translation strategies $(\vartheta_n)_{n\in\mathbb{N}}$ are defined such that $\vartheta_n(u, a)$ stays in state q if the counter value is less than n but moves to, and then remains in, state r if the counter value is at least n.

By Remark 2.11, it is sufficient to prove $[\![\mathcal{A}_{\mathrm{cost}_B^{\{1\},[i,j]}}]\!] \preccurlyeq \mathrm{cost}_B^{\{1\},[1,2]} \preccurlyeq [\![\mathcal{A}_{\mathrm{cost}_B^{\{1\},[i,j]}}]\!]^{\vartheta}$. In fact, we prove that $[\![\mathcal{A}_{\mathrm{cost}_B^{\{1\},[i,j]}}]\!] \leq \mathrm{cost}_B^{\{1\},[1,2]} \leq [\![\mathcal{A}_{\mathrm{cost}_B^{\{1\},[i,j]}}]\!]^{\vartheta}$.

[3] If $i = j$, then a similar result holds but the history deterministic cost automata may require priorities $[i + 1, j + 2]$.

It is easy to see that $[\![\mathcal{A}_{\mathrm{cost}_B^{\{1\},[i,j]}}]\!] \leq \mathrm{cost}_B^{\{1\},[i,j]}$ since if $\mathrm{cost}_B^{\{1\},[i,j]}(u) = N \in \mathbb{N}$, then the parity condition in u must be satisfied and the counter value must not exceed N. This means that any accepting run of $\mathcal{A}_{\mathrm{cost}_B^{\{1\},[i,j]}}$ on u must move from state q to r. While in q any `ic` is converted to `i` for the S-counter (and `r` and ε are preserved), and on the transition from q to r this S-counter value is checked. This means that the maximum value that the S-counter can achieve is N (when the transition to r is taken after some maximal sequence of `ic` without `r`), so $[\![\mathcal{A}_{\mathrm{cost}_B^{\{1\},[i,j]}}]\!](u) \leq \mathrm{cost}_B^{\{1\},[i,j]}$.

Now we seek to prove that $\mathrm{cost}_B^{\{1\},[i,j]} \leq [\![\mathcal{A}_{\mathrm{cost}_B^{\{1\},[i,j]}}]\!]^{\vartheta}$. Assume $\mathrm{cost}_B^{\{1\},[i,j]}(u) = N$. If $N < \infty$, then the parity condition is satisfied but the counter achieves value N. This means the run driven by ϑ_N will witness exactly value N, so $[\![\mathcal{A}_{\mathrm{cost}_B^{\{1\},[i,j]}}]\!]^{\vartheta}(u) \geq N$. If $N = \infty$, then either (i) the counter values are unbounded or (ii) the counter values are below $M \in \mathbb{N}$, but the parity condition is not satisfied. In the case of (i), $\vartheta_{N'}$ will drive an accepting run which witnesses value N' for all N'. In the case of (ii), the run driven by $\vartheta_{N'}$ for $N' > M$ will stay forever in state q and witness value ∞ since the counter is never checked and the priorities $\overline{p}$ satisfy the parity condition. In either case, this means $[\![\mathcal{A}_{\mathrm{cost}_B^{\{1\},[i,j]}}]\!]^{\vartheta}(u) \geq N$ as desired.

In order to define $\mathcal{A}_{\mathrm{cost}_B^{\Gamma,[i,j]}}$ for $|\Gamma| > 1$, we take the product of $|\Gamma|$-many copies of $\mathcal{A}_{\mathrm{cost}_B^{\{1\},[i,j]}}$ where each copy is responsible for processing counter actions from a particular counter $\gamma \in \Gamma$. That is, there is a transition from $(s_1, \ldots, s_{|\Gamma|})$ to $(s'_1, \ldots, s'_{|\Gamma|})$ labelled $(a, p) : (a', \overline{p})$ if for all $\gamma \in \Gamma$, there is a transition from s_γ to $s'_\gamma = q$ labelled $(\mathrm{pr}_\gamma(a), p) : (\mathrm{pr}_\gamma(a'), \overline{p})$. Likewise, there is a transition $(a, p) : (a', \widehat{p})$ if for all γ there is a transition from s_γ to s'_γ labelled $(\mathrm{pr}_\gamma(a), p) : (\mathrm{pr}_\gamma(a'), p')$, and there is some γ such that $s'_\gamma = r$. The idea is that the run should be accepting as soon as one copy is able to prove (by moving to state r) that the counter has reached a large value. We leave the proof of correctness and history determinism to the reader.

We now turn to the second part, defining a B-parity automaton reading S-parity actions. Figure 3.3 shows the B-$[i+1, j+1]$ automaton $\mathcal{A}_{\mathrm{cost}_S^{\{1\},[i,j]}}$.

This looks more complicated, but it actually uses a similar principle as Example 2.14. The idea is that on input $u \in (\mathbb{S} \times [i, j])^\omega$, the B-automaton is trying to prove that $\mathrm{cost}_S^{\{1\},[i,j]}(u)$ is low, either because the parity condition was not satisfied, or because the counter was checked with a low value. The only nondeterminism is when reading `i` in state r. The translation strategy ϑ_n moves to state q if the B-counter would exceed value n and otherwise stays in state r.

By Remark 2.11, it suffices to show that $[\![\mathcal{A}_{\mathrm{cost}_S^{\{1\},[i,j]}}]\!]^{\vartheta} \leq \mathrm{cost}_S^{\{1\},[i,j]} \leq [\![\mathcal{A}_{\mathrm{cost}_S^{\{1\},[i,j]}}]\!]$.

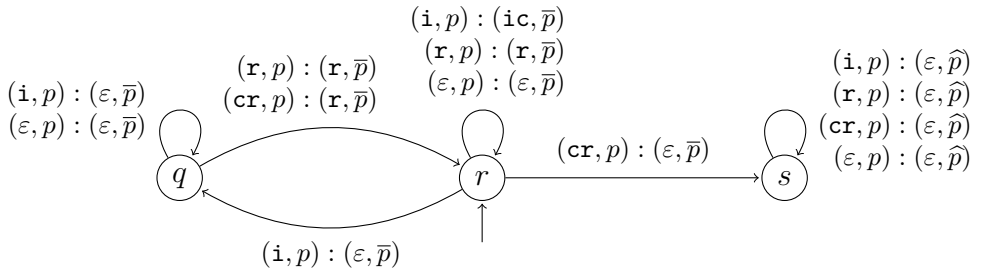

Figure 3.3. History deterministic B-$[i+1, j+1]$ automaton recognizing $\mathrm{cost}_S^{\{1\},[i,j]}$ (see Lemma 3.5).

Assume $[\![\mathcal{A}_{\mathrm{cost}_S^{\{1\},[i,j]}}]\!](u) \leq N \in \mathbb{N}$. We distinguish between two cases depending on whether the run ρ of $\mathcal{A}_{\mathrm{cost}_S^{\{1\},[i,j]}}$ on u with $\mathrm{value}(\rho) \leq N$ stays in states q and r or moves to state s.

If ρ remains in state q and r, then it must be the case that the priority sequence in u does not satisfy the parity condition (if not, the priorities $\overline{p}$ generated during the run of $\mathcal{A}_{\mathrm{cost}_S^{\{1\},[i,j]}}$ would not satisfy the parity condition, contradicting the fact that $\mathrm{value}(\rho) \leq N$). In this case, $\mathrm{cost}_S^{\{1\},[i,j]}(u) = 0 \leq N$ as desired.

Otherwise, assume that during the course of ρ the automaton moves to state s. In order to be able to take the transition to s, there must be some subword in u that starts with $\mathtt{r}$ (or the beginning of the word) and has a sequence of $\mathtt{i}$ without $\mathtt{r}$ ending in $\mathtt{cr}$. Because $\mathrm{value}(\rho) \leq N$, this must mean there were at most N occurrences of $\mathtt{i}$ in this subword. Hence, this subword witnesses the fact that the S-counter is checked with value at most N, so $\mathrm{cost}_S^{\{1\},[i,j]}(u) \leq N$.

Now we must show $[\![\mathcal{A}_{\mathrm{cost}_S^{\{1\},[i,j]}}]\!]^{\vartheta} \leq \mathrm{cost}_S^{\{1\},[i,j]}$. Assume $\mathrm{cost}_S^{\{1\},[i,j]}(u) = N \in \mathbb{N}$. This means that either (i) there is some point at which the counter is checked with value N or (ii) the counter is always checked with value greater than 0 but the priorities in u do not satisfy the parity condition so $N = 0$. It is clear that the translation strategy ϑ_N yields a run of value at most N. If the counter is checked with a value at most N in u, then the translation strategy ϑ_N will result in the same value (since it remains in state r and mimics each of the original increments preceding the $\mathtt{cr}$ before moving to state s when reading the $\mathtt{cr}$). In the case of (ii), then ϑ_0 will remain forever in state q and r, never incrementing the counter. The priorities output by ϑ_0 will satisfy the parity condition since the original sequence of priorities in u did not satisfy the parity condition, so this run will have value 0. In either case, $[\![\mathcal{A}_{\mathrm{cost}_S^{\{1\},[i,j]}}]\!]^{\vartheta}(u) \leq N$ as desired.

As in the previous case, in order to construct $\mathcal{A}_{\mathrm{cost}_S^{\Gamma,[i,j]}}$ for $|\Gamma| > 1$, we take $|\Gamma|$-many copies of the automaton. The counter actions output on transition come from

the individual copies. If the input priority is p, then output priority is $\widehat{p}$ if there is some copy that has moved to state s, and $\overline{p}$ otherwise. □

It is also possible to translate arbitrary actions into hierarchical actions.

Lemma 3.6. *For all Γ and $i < j$, there is a deterministic hB-$[i,j]$ automaton $\mathcal{H}_B^{\Gamma,[i,j]}$ and an* id*-history-deterministic hS-$[i,j]$ automaton $\mathcal{H}_S^{\Gamma,[i,j]}$ such that*

$$[\![\mathcal{H}_B^{\Gamma,[i,j]}]\!] \approx_\alpha \mathrm{cost}_B^{\Gamma,[i,j]} \quad \text{and} \quad [\![\mathcal{H}_S^{\Gamma,[i,j]}]\!] \approx_\alpha \mathrm{cost}_S^{\Gamma,[i,j]}$$

for $\alpha(n) = k \cdot (n+1)^k$ where $k = |\Gamma|$.

Proof. We use the automata $\mathcal{H}_B^\Gamma$ and $\mathcal{H}_S^\Gamma$ from Lemma 2.17. First, note that correctness of these automata over finite words from $(\mathbb{B}^\Gamma)^*$ or $(\mathbb{S}^\Gamma)^*$ implies correctness (of the same automata, now viewed as Büchi automata) run over infinite words w from $(\mathbb{B}^\Gamma)^\omega$ or $(\mathbb{S}^\Gamma)^\omega$. This follows from Lemma 3.4 and the fact that

$$\mathrm{value}_B(w) = \inf\{n : \exists \text{ infinitely many prefixes } v \text{ of } w \text{ such that } \mathrm{value}_B(v) \leq n\},$$
$$\mathrm{value}_S(w) = \sup\{n : \exists \text{ infinitely many prefixes } v \text{ of } w \text{ such that } \mathrm{value}_S(v) \geq n\}.$$

We seek automata that read both counter actions and priorities. For this, $\mathcal{H}_B^{\Gamma,[i,j]}$ simulates $\mathcal{H}_B^\Gamma$ to get the desired hierarchical actions but also outputs the priorities unchanged so the parity condition is preserved.

Likewise, $\mathcal{H}_S^{\Gamma,[i,j]}$ simulates $\mathcal{H}_S^\Gamma$ and outputs the priorities unchanged, unless the automaton enters state $\bot$ (the rejecting sink state), in which case the output is set to some odd priority in $[i,j]$ to indicate that this is not a valid run for $\mathcal{H}_S^\Gamma$. □

We can also show that restricted $\mathbb{B}$ actions ($\{\mathtt{ic}, \varepsilon\}$ or $\{\mathtt{ic}, \mathtt{r}\}$) over multiple counters can be converted to actions over a single counter. We omit the straightforward proof which simply adapts the transducers in Lemmas 2.15 and 2.16 to the setting of infinite words.

Lemma 3.7. *For all Γ, $i \leq j$, and restricted $\mathbb{B}$-actions $\mathbb{B}' \in \{\{\mathtt{ic}, \varepsilon\}, \{\mathtt{ic}, \mathtt{r}\}\}$, there is a deterministic hB-$[i,j]$ automaton $\mathcal{D}_{\mathbb{B}'}^{\Gamma,[i,j]}$ using only one counter and actions $\mathbb{B}'$ such that over the alphabet $(\mathbb{B}')^\Gamma$,*

$$[\![\mathcal{D}_{\mathbb{B}'}^{\Gamma,[i,j]}]\!] \approx_\alpha \mathrm{cost}_B^{\Gamma,[i,j]}$$

where $\alpha(n) = n \cdot |\Gamma|$ if $\mathbb{B}' = \{\mathtt{ic}, \varepsilon\}$ and $\alpha(n) = 2n+1$ if $\mathbb{B}' = \{\mathtt{ic}, \mathtt{r}\}$.

3.2 Cost games

Before proceeding to cost automata over infinite trees, we introduce two player games known as *cost games*. The semantics of cost automata over infinite trees will be defined in terms of these games.

Unlike the classical game setting where a player either wins or loses, a cost game is assigned a value in $\mathbb{N}_\infty$ based on some objective O. The objective O describes the aim of the first player, traditionally known as *Eve*. The dual objective $\overline{O}$ of some $O = \langle \mathbb{C}, f, goal \rangle$ is obtained by changing *goal* from min to max or max to min, and represents the aim of the opponent, known as *Adam*.

Formally, a *cost game* $\mathcal{G} = \langle V, v_0, O, \delta \rangle$ is played in an *arena* that is defined by a (possibly infinite) set of *positions* V, an *initial position* $v_0 \in V$, an objective $O = \langle \mathbb{C}, f, goal \rangle$ for Eve, and a *control function* $\delta : V \longrightarrow \mathscr{B}^+(\mathbb{C} \times V)$ where $\mathscr{B}^+(\mathbb{C} \times V)$ is the set of positive boolean combinations of elements from $\mathbb{C} \times V$. We assume each $\delta(v)$ is written in disjunctive normal form (as a disjunction of conjunctive clauses).

The *dual game* is $\overline{\mathcal{G}} = \langle V, v_0, \overline{O}, \overline{\delta} \rangle$ where $\overline{O}$ is the dual objective of O (switching min to max, or vice versa), and $\overline{\delta}$ is the result of exchanging conjunctions and disjunctions in δ (and then rewriting each $\overline{\delta}(v)$ in disjunctive normal form). This has the effect of switching the roles of the two players in the game.

The set of *moves* in $\mathcal{G}$ is $E_\mathcal{G} := \{(v, c, w) \in V \times \mathbb{C} \times V : (c, w) \text{ appears in } \delta(v)\}$; we write $E_\mathcal{G}(v)$ for the set of moves starting from a particular $v \in V$. This notation is chosen because $\mathcal{G}$ can be viewed as a graph where the nodes are the positions V and the edges are $E_\mathcal{G}$. A *play* π is an infinite sequence of moves $(v_i, c_{i+1}, v_{i+1})_{i \in \mathbb{N}} \in E_\mathcal{G}^\omega$ such that v_0 is the initial position and the tuple $(v_i, c_{i+1}, v_{i+1}) \in E_\mathcal{G}(v_i)$ for all $i \in \mathbb{N}$. The *output* $\text{out}(\pi)$ of a play π is $c_1 c_2 \cdots \in \mathbb{C}^\omega$.

Given a set σ of plays, let $\text{pref}(\sigma)$ denote the set of prefixes of plays in σ. We say $(v_0, c_1, v_1) \cdots (v_j, c_{j+1}, v_{j+1}) \in \text{pref}(\sigma)$ is a partial play *ending* in v_{j+1} (we say $\epsilon \in \text{pref}(\sigma)$ ends in v_0). At a position $v \in V$, the positive boolean combination given by $\delta(v)$ can be viewed as a subgame in which Eve selects a disjunct in $\delta(v)$ and Adam selects a conjunct within this disjunct. For instance, if there is some partial play π ending in $v \in V$ with $\delta(v) = (c, v) \vee ((c', v') \wedge (c'', v''))$, then $E_\mathcal{G}(v) = \{(v, c, v), (v, c', v'), (v, c'', v'')\}$. Eve can choose a disjunct, say, $(c', v') \wedge (c'', v'')$, and Adam can choose one of the conjuncts in this disjunct, say (c'', v''). The play is then extended to $\pi \cdot (v, c'', v'')$ and c'' describes the cost for making this move.

A *strategy* σ for Eve in $\mathcal{G}$ is a set of plays such that if a partial play $\pi \in \text{pref}(\sigma)$ ends in position v, there must be a single disjunct $(c'_1, v'_1) \wedge \cdots \wedge (c'_j, v'_j)$ in $\delta(v)$ such

that for every conjunct (c'_i, v'_i) for $i \in [1, j]$, $\pi \cdot (v, c'_i, v'_i) \in \text{pref}(\sigma)$. The idea is that a strategy for Eve describes deterministically how she should play for every possible history. Unless otherwise indicated, when we speak of a strategy σ in $\mathcal{G}$, we mean a strategy for Eve. A strategy $\overline{\sigma}$ for Adam in $\mathcal{G}$ is a set of plays such that $\overline{\sigma}$ is a strategy for Eve in $\overline{\mathcal{G}}$. Note that fixing a strategy σ for Eve in $\mathcal{G}$ and a strategy $\overline{\sigma}$ for Adam in $\mathcal{G}$ induces a single play π such that $\pi \in \sigma$ and $\pi \in \overline{\sigma}$. Strategies in cost games are the subject of Chapter 4.

The objective $O = \langle \mathbb{C}, f, \mathit{goal} \rangle$ describes how to assign values in the game. For a play $\pi = (v_i, c_{i+1}, v_{i+1})_{i \in \mathbb{N}}$, the value is $\text{value}(\pi) := f(\text{out}(\pi))$. If *goal* is min, then the value of a strategy σ for Eve is $\text{value}(\sigma) := \sup\{\text{value}(\pi) : \pi \in \sigma\}$ and the value of the game is $\text{value}(\mathcal{G}) := \inf\{\text{value}(\sigma) : \sigma \text{ is a strategy for Eve in } \mathcal{G}\}$. In other words, Eve seeks to minimize over all strategies the maximum value of all plays compatible with the strategy. Dually, if *goal* is max, then $\text{value}(\sigma) := \inf\{\text{value}(\pi) : \pi \in \sigma\}$ and $\text{value}(\mathcal{G}) := \sup\{\text{value}(\sigma) : \sigma \text{ is a strategy for Eve in } \mathcal{G}\}$.

Just like cost automata, we will refer to cost games by their objective (e.g. a B-Büchi or B-$[1, 2]$ game is a cost game with objective $\text{Cost}_B^{\Gamma,[1,2]}$).

Remark 3.8. This form of two-player game differs slightly from the usual presentation of games in, e.g. [GTW02].

First, the priorities (and counter actions) label edges in the game graph rather than positions. It is straightforward to convert to a game where the positions are labelled with priorities and counter actions instead of edges (at the price of increasing the size of the game graph).

Second, the control function δ is given by a positive boolean combination of elements from $\mathbb{C} \times V$ (written in disjunctive normal form) where Eve selects a disjunct and Adam selects a conjunct, rather than partitioning V based on which player controls the move from each position. This allows a more immediate link with the alternating automata that will be described in the next section. Because we assume that $\delta(v)$ is defined for all $v \in V$, it is also important to note that there are no finite plays in these games. It is possible to adapt this definition to allow finite duration cost games (see [CL10]) where every play is finite and the objective can utilize the terminal symbol of the play in order to determine the value, but for the purposes of this thesis, we will usually be working with infinite duration cost games.

We now provide two examples of cost games.

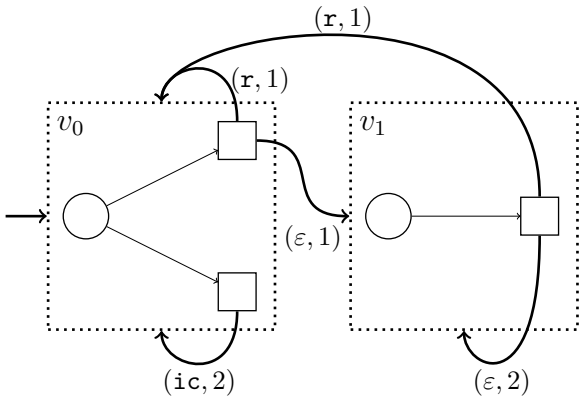

Figure 3.4. B-$[1,2]$ cost game $\mathcal{G}_B$ with value$(\mathcal{G}_B) = 1$ (see Example 3.9).

Example 3.9. Consider the B-$[1,2]$ game $\mathcal{G}_B := \langle \{v_0, v_1\}, v_0, \mathrm{Cost}_B^{\{1\},[1,2]}, \delta \rangle$ where

$$\delta(v_0) := \Big(((\mathtt{r}, 1), v_0) \wedge ((\varepsilon, 1), v_1)\Big) \vee ((\mathtt{ic}, 2), v_0),$$
$$\delta(v_1) := \ ((\mathtt{r}, 1), v_0) \wedge ((\varepsilon, 2), v_1).$$

We have given a (slightly unconventional) graphical representation of this game in Figure 3.4. The play starts at v_0. If the play has reached position $v \in \{v_0, v_1\}$, a subgame is played between Eve (circles) and Adam (boxes) where Eve selects a disjunct (taking one of the light edges), and then Adam selects a conjunct (taking one of the dark edges leading to position v' and labelled with action c). The play then moves to position v' with cost c, and the players continue as before.

In this example, Eve only has one choice to make, namely the disjunct to select when in position v_0 (in v_1, there is only one disjunct so she has no choice). We now consider some strategies for Eve, and the cost associated with them.

Consider the strategy σ where Eve always selects the second conjunct when in v_0. There is only one play consistent with this strategy, so $\sigma = \{\pi\}$ where $\pi = (v_0, (\mathtt{ic}, 2), v_0)^\omega$. The parity condition is satisfied on π, but value$(\pi) = \infty$ since there is no bound on the checked values of the counter. This means that value$(\sigma) = \infty$, so σ is not a good strategy for Eve, the minimizing player.

Next consider the strategy σ' where Eve always selects the first conjunct. There are infinitely many plays in σ'. However, $\pi' = (v_0, (\mathtt{r}, 1), v_0)^\omega \in \sigma'$ has value ∞ because the parity condition is not satisfied, so value$(\sigma') = \infty$.

However, there are strategies for Eve that witness value$(\mathcal{G}_B) = 1$. Let σ'' be the strategy for Eve that in v_0 selects the first disjunct if the counter currently has value 1 and selects the second disjunct if the counter currently has value 0. Assume the play

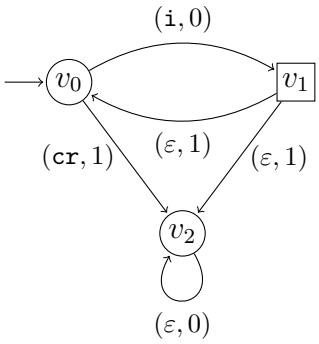

Figure 3.5. S-$[0,1]$ cost game $\mathcal{G}_S$ with value$(\mathcal{G}_S) = \infty$ (see Example 3.10).

is in position v_0 with value 0. Then Eve will select the second conjunct, and the play outputs $(\mathtt{ic}, 2)$ and stays in v_0. Since the counter now has value 1, Eve will select the first conjunct. If the play moves to v_1 and stabilizes there, then the remaining output is $(\varepsilon, 2)$, so the parity condition is satisfied and the play has value 1. If the play eventually returns to v_0, $(\mathtt{r}, 1)$ is output so the reasoning can proceed as above to show that the counter never exceeds value 1 and Eve can always guarantee to visit priority 2. Hence, value$(\sigma'') = 1$ and value$(\mathcal{G}_B) \leq 1$. There is no strategy of value 0, so value$(\mathcal{G}_B) = 1$.

We will often consider much simpler examples where it is not necessary to use the full power of an alternating control function. In particular, if for all positions v, $\delta(v)$ is purely disjunctive or purely conjunctive, then the positions can be partitioned into nodes controlled by Adam (positions v where $\delta(v)$ is purely conjunctive and has at least two conjuncts) and nodes controlled by Eve (positions v where $\delta(v)$ is purely disjunctive). This is graphically represented as a game graph where Adam's positions are boxes and Eve's positions are circles (and edges are labelled with the cost of the move, as usual). This more traditional graphical presentation of games will be used in the remaining examples of cost games in this thesis.

Example 3.10. Consider the S-$[0,1]$ game $\mathcal{G}_S = \langle \{v_0, v_1, v_2\}, v_0, \mathrm{Cost}_S^{\{1\},[0,1]}, \delta \rangle$ in Figure 3.5 where

$$\begin{aligned}
\delta(v_0) &:= ((\mathtt{i}, 0), v_1) \vee ((\mathtt{cr}, 1), v_2), \\
\delta(v_1) &:= ((\varepsilon, 1), v_0) \wedge ((\varepsilon, 1), v_2), \\
\delta(v_2) &:= ((\varepsilon, 0), v_2).
\end{aligned}$$

Consider the strategy σ_n for Eve that at v_0 selects the first disjunct (i.e. takes the transition to v_1) if the counter has value less than n, and otherwise, selects

the second disjunct (i.e. takes the transition to v_2). Let $\pi \in \sigma_n$. If Adam always selects the transition back to v_0, then Eve will be able to move to v_2 and check-reset the counter when it reaches value n, so $\text{value}(\pi) = n$. If not, then Adam must select the transition from v_1 to v_2 during π. But this means π satisfies the parity condition by stabilizing in priority 0 and the counter is never checked, so $\text{value}(\pi) = \infty$. Hence $\text{value}(\sigma_n) = n$. Note that there is no single strategy that witnesses value ∞. However, there is a strategy σ_n with $\text{value}(\sigma_n) = n$ for each n, so $\text{value}(\mathcal{G}_S) = \sup\{\text{value}(\sigma) : \sigma \text{ is a strategy for Eve in } \mathcal{G}_S\} = \infty$.

Changing objectives

It is often helpful to convert between cost games with different objectives in a way that preserves the value, at least up to some correction function. This is similar to the notion of converting between different types of winning conditions known from literature (e.g. converting a Muller game to an equivalent parity game).

One simple transformation of a game/objective that preserves the value is dualization. As mentioned earlier, the dual of a game $\mathcal{G}$ is obtained by switching disjunctions and conjunctions in the control function and using the dual objective (i.e. replacing min with max, and vice versa). This switches the roles of Adam and Eve in the game.

Proposition 3.11. *Let $\mathcal{G}$ be a cost-parity game. Then* $\text{value}(\mathcal{G}) = \text{value}(\overline{\mathcal{G}})$.

Proof. Although cost games are assigned a value, we can convert the game into a more standard setting with a winner or loser, by parameterizing the game based on the value. That is, given a cost game $\mathcal{G}$ with $goal = \min$ (respectively, $goal = \max$), we can consider a family of games $\mathcal{G}_n$ for each $n \in \mathbb{N}_\infty$ such that Eve wins game $\mathcal{G}_n$ if there is a strategy σ for Eve such that every $\pi \in \sigma$ satisfies $\text{value}(\pi) \leq n$ (respectively, $\text{value}(\pi) > n$), and Adam wins otherwise.

We fix now some cost game $\mathcal{G}$ with objective $\text{Cost}_B^{\Gamma,P}$ (the argument is similar for other cost-parity objectives). Let $\mathbb{C}$ be the output alphabet.

Proving that $\text{value}(\mathcal{G}) = \text{value}(\overline{\mathcal{G}})$ comes down to the fact that these parametrized cost-parity games are *determined*, that is, from every position, exactly one of the two players has a winning strategy. We assume determinacy for now and describe why this implies that $\text{value}(\mathcal{G}) = \text{value}(\overline{\mathcal{G}})$.

Let $\text{value}(\mathcal{G}) = n$. Then Eve can win $\mathcal{G}_n$. Since dualizing reverses the roles of the players, this means Adam wins $\overline{\mathcal{G}}_n$, so there is no winning strategy σ for Eve in $\overline{\mathcal{G}}_n$ such that every $\pi \in \sigma$ satisfies $\text{value}(\pi) > n$. This implies that there is no strategy σ in $\overline{\mathcal{G}}$ with $\text{value}(\sigma) > n$, so $\text{value}(\overline{\mathcal{G}}) \leq n$.

However, Eve cannot win $\mathcal{G}_{n-1}$ (if she could, it would contradict value$(\mathcal{G}) = n$). Hence, since we are assuming determinacy, there is a winning strategy for Adam in $\mathcal{G}_{n-1}$ and, equivalently, a winning strategy for Eve in $\overline{\mathcal{G}}_{n-1}$. This implies that there is a strategy σ for Eve in $\overline{\mathcal{G}}_{n-1}$ such that every $\pi \in \sigma$ satisfies value$(\pi) > n-1$. Hence, this strategy σ can also be viewed as a strategy in $\overline{\mathcal{G}}$ with value$(\sigma) > n-1$. Since value$(\overline{\mathcal{G}})$ is the maximum value over all strategies for Eve, this means that value$(\overline{\mathcal{G}}) \geq n$. Putting this together, we get value$(\overline{\mathcal{G}}) = n$ as desired.

It remains to show that $\mathrm{Cost}_B^{\Gamma,P}$ games are determined. For this, we rely on a famous result due to Martin [Mar75] that says that games with Borel winning conditions are determined. The Borel hierarchy is a topological classification of sets. We need only basic definitions for our purpose here, but refer the reader to [Kec95] for additional information. We view the space of output words $\mathbb{C}^\omega$ as a topological space where open sets are of the form $X \cdot \mathbb{C}^\omega$ where X is any set of finite words. These open sets are denoted $\mathbf{\Sigma}_1^0$. The complement of an open set is denoted $\mathbf{\Pi}_1^0$. Taking a countable union (respectively, countable intersection) of sets from $\mathbf{\Pi}_{\beta'}^0$ (respectively, $\mathbf{\Sigma}_{\beta'}^0$) for $\beta' < \beta$ results in a set in $\mathbf{\Sigma}_\beta^0$ (respectively, $\mathbf{\Pi}_\beta^0$).

The parity condition is in $\mathbf{\Sigma}_3^0 \cap \mathbf{\Pi}_3^0$. The idea is that we can define an open set $P_p^m = (\mathbb{C}^*(\mathbb{B}^\Gamma \times p))^m \mathbb{C}^\omega$ for all m that consists of words with at least m occurrences of some priority p. The set P_p of words where p occurs infinitely often is $P_p = \bigcap_{m \in \mathbb{N}} P_p^m$, so it is in $\mathbf{\Pi}_2^0$. The parity condition is then

$$\bigcap_{\text{odd } p} \bigcup_{\text{even } p' > p} \overline{P_p} \cup P_{p'}$$

where p and p' range over the finite set P of priorities. This is a finite boolean combination of sets in $\mathbf{\Sigma}_2^0$ and $\mathbf{\Pi}_2^0$, and hence is in $\mathbf{\Sigma}_3^0 \cap \mathbf{\Pi}_3^0$.

We can analyse counter actions in a similar way. The set of words C_γ^m in $\mathbb{C}^\omega$ that witness checked value greater than m for counter γ is an open set (since all such words can be described by a finite prefix that witnesses value greater than m). Hence the counter condition for $\mathcal{G}_n$ is

$$\overline{\bigcup_{\gamma \in \Gamma} C_\gamma^n}$$

which is in $\mathbf{\Pi}_1^0$ (the complement of an open set).

This means the overall winning condition for $\mathcal{G}_n$ is also in $\mathbf{\Sigma}_3^0 \cap \mathbf{\Pi}_3^0$, so the winning condition is Borel as required. □

Another way to change objective is by composing a cost game with certain cost automata over infinite words. Let $\widetilde{\mathcal{G}} = \langle V, v_0, \langle \mathbb{A}, f, \mathit{goal} \rangle, \delta \rangle$ be a cost game and

consider a cost automaton over infinite words $\mathcal{A} = \langle Q, \mathbb{A}, q_0, \langle \mathbb{C}, g, \mathit{goal}\rangle, \Delta\rangle$ such that $f \approx [\![\mathcal{A}]\!]$.

Because $\mathcal{A}$ recognizes f, it is easy to see that the game $\mathcal{G} = \langle V, v_0, \langle \mathbb{A}, [\![\mathcal{A}]\!], \mathit{goal}\rangle, \delta\rangle$ satisfies $\text{value}(\mathcal{G}) \approx_\alpha \text{value}(\widetilde{\mathcal{G}})$, but we can do more. Notice that the value of a play that outputs actions from an alphabet $\mathbb{A}$ can be computed by running the automaton $\mathcal{A}$ which uses a different objective and outputs actions from an alphabet $\mathbb{C}$. We can make this explicit by considering the *composition of* $\mathcal{A}$ *and* $\mathcal{G}$ which is the game $\mathcal{A} \circ \mathcal{G} := \langle Q \times V, (q_0, v_0), \langle \mathbb{C}, g, \mathit{goal}\rangle, \delta'\rangle$ where

$$\delta'((q, v)) := \delta(v)\left[\bigvee\{(c, (q', v')) : (q, a, c, q') \in \Delta\}/(a, v')\right].$$

In this new game $\mathcal{A} \circ \mathcal{G}$, the state of the valuation function described by $\mathcal{A}$ is made explicit in the positions in the game, and the objective of this new game has changed to the objective from $\mathcal{A}$. We can view a play in $\mathcal{A} \circ \mathcal{G}$ as a play in $\mathcal{G}$ together with a run of $\mathcal{A}$ used to compute its value.

If $\mathcal{A}$ is deterministic, then the output from this composition will clearly yield the same value. If $\mathcal{A}$ is nondeterministic, however, then this is not necessarily the case: in order to compute the value of two plays with some common prefix, $\mathcal{A}$ could disagree about moves on the shared prefix, and therefore be unable to correctly assign values to both plays.

It turns out that if $\mathcal{A}$ is history deterministic, this composition results in a game with a different objective but the same value (up to $\approx$). Thus, a nice way to summarize history deterministic cost automata is that they compose well with games. This was shown in [CL10, Lemma 7] for history deterministic cost automata on finite words composed with finite duration cost games, and can be easily adapted to the infinite setting in this thesis.

Lemma 3.12. *Let $\mathcal{A} = \langle Q, \mathbb{A}, q_0, \langle \mathbb{C}, g, \mathit{goal}\rangle, \Delta\rangle$ be an α_{hd}-history-deterministic cost automaton over infinite words and $\mathcal{G} = \langle V, v_0, \langle \mathbb{A}, [\![\mathcal{A}]\!], \mathit{goal}\rangle, \delta\rangle$ be a cost game. Then* $\text{value}(\mathcal{G}) \approx_{\alpha_{\text{hd}}} \text{value}(\mathcal{A} \circ \mathcal{G})$.

Proof. Assume *goal* is min (and notice that *goal* is the same in $\mathcal{A}$ and $\mathcal{G}$). Let $\mathcal{G}' := \mathcal{A} \circ \mathcal{G}$.

We first show that $\text{value}(\mathcal{G}) \leq \text{value}(\mathcal{G}')$. Assume $\text{value}(\mathcal{G}') \in \mathbb{N}$, and let σ' be a strategy in $\mathcal{G}'$ witnessing this bounded value. Then any play $\pi' \in \sigma'$ can be written in the form $((q_i, v_i), c_{i+1}, (q_{i+1}, v_{i+1}))_{i\in\mathbb{N}}$ and can be transformed into a play $\pi := (v_i, a_{i+1}, v_{i+1})_{i\in\mathbb{N}}$ in $\mathcal{G}$ where $(q_i, a_{i+1}, c_{i+1}, q_{i+1}) \in \Delta$ for all $i \in \mathbb{N}$. Notice that $[\![\mathcal{A}]\!](a_1 a_2 \cdots) \leq g(c_1 c_2 \cdots)$ since $g(c_1 c_2 \cdots)$ is the value of a particular run of $\mathcal{A}$ on

$a_1a_2\cdots$ and the $[\![\mathcal{A}]\!](u)$ takes the minimum value over all runs since *goal* = min. This means that $\text{value}(\pi) = [\![\mathcal{A}]\!](a_1a_2\cdots) \leq g(c_1c_2\cdots) = \text{value}(\pi')$.

Doing this transformation for all plays $\pi' \in \sigma'$ results in a set of plays σ that is a strategy in $\mathcal{G}$ satisfying $\text{value}(\sigma) \leq \text{value}(\sigma')$ as desired.

In the other direction, we rely on the history determinism of $\mathcal{A}$ and show that $\text{value}(\mathcal{G}') \preccurlyeq_{\alpha_{\text{hd}}} \text{value}(\mathcal{G})$. Let $(\vartheta_n)_{n\in\mathbb{N}}$ be the translation strategies for $\mathcal{A}$. Assume that $\text{value}(\mathcal{G}) \leq n \in \mathbb{N}$. Then there is a strategy σ in $\mathcal{G}$ such that $\text{value}(\sigma) \leq n$. Any play $\pi \in \sigma$ is of the form $(v_i, a_{i+1}, v_{i+1})_{i\in\mathbb{N}}$ with $[\![\mathcal{A}]\!](a_1a_2\cdots) \leq n$. By the definition of history determinism, this means that the unique run $\rho = (q_i, a_{i+1}, c_{i+1}, q_{i+1})_{i\in\mathbb{N}}$ driven by $\vartheta_{\alpha_{\text{hd}}(n)}$ satisfies $\text{value}(\rho) \leq \alpha_{\text{hd}}(n)$. We use ρ and π to define the play $\pi' := ((q_i, v_i), c_{i+1}, (q_{i+1}, v_{i+1}))$ in $\mathcal{G}'$ with $\text{value}(\pi') \leq \alpha_{\text{hd}}(n)$.

Let σ' be the set of all plays π' obtained by doing this transformation starting from some play $\pi \in \sigma$. Because the plays in σ' agree on all shared prefixes (since they were driven by the same translation strategy $\vartheta_{\alpha_{\text{hd}}(n)}$), we see that σ' is actually a strategy in $\mathcal{G}'$, with $\text{value}(\sigma') \leq \alpha_{\text{hd}}(n)$ as desired.

The proof when *goal* is max is similar. □

3.3 Cost automata on infinite trees

The *unlabelled infinite binary tree* is $\mathcal{T} = \{0,1\}^*$. Hence, the *positions* or *nodes* in a binary tree are words over $\{0,1\}$. The *root* of $\mathcal{T}$ is denoted ϵ and a *branch* is an infinite sequence $x_0x_1\cdots$ of positions such such that $x_0 = \epsilon$ and $x_{i+1} \in x_i \cdot \{0,1\}$. A *frontier* E is a set of positions such that for any branch π, $E \cap \pi$ is a singleton. For $x, y \in \mathcal{T}$, we write $x \leq y$ if x is a prefix of y, and $x < y$ if $x \leq y$ and $|x| < |y|$. For positions $x < y$, $[x, y] := \{x' : x \leq x' \leq y\}$ (respectively, $[x, y) := \{x' : x \leq x' < y\}$) is the set of positions from x up to and including y (respectively, x up to but not including y).

Given a finite alphabet $\mathbb{A}$, the set $\mathcal{T}_{\mathbb{A}}$ of *complete* $\mathbb{A}$*-labelled binary trees* is composed of mappings $t : \mathcal{T} \to \mathbb{A}$ that map a position $x \in \mathcal{T}$ to its label $t(x) \in \mathbb{A}$. A branch π induces an infinite word over $\mathbb{A}^\omega$ that describes the sequence of labels on that path; we often identify π with this infinite word. If $t \in \mathcal{T}_{\mathbb{A}}$ and $x \in \mathcal{T}$, then t_x denotes the *subtree* of t with root at x, so $t_x(y) = t(x \cdot y)$.

We could also define labelled trees over a ranked alphabet $\mathbb{A}$, where each symbol has some finite arity and the resulting trees in $\mathcal{T}_{\mathbb{A}}$ have finite branching determined by the rank of the symbol at a particular position (this is done in [CL10]). Occasionally, we will encounter graphs with a tree structure and finite branching like this (see,

e.g. the definition of a strategy tree in Section 4.1). However, for notational simplicity, we restrict the input for automata over infinite trees to complete labelled binary trees (i.e. trees where every symbol has arity 2), knowing that the definitions and results could be extended to trees over a finite ranked alphabet.

We are now ready to describe cost automata on infinite trees. An (*alternating*) *cost automaton $\mathcal{A}$ over infinite trees* is a tuple

$$\langle Q, \mathbb{A}, q_0, O, \delta \rangle$$

with a finite set of states Q, an alphabet $\mathbb{A}$, an initial state $q_0 \in Q$, an objective $O = \langle \mathbb{C}, f, goal \rangle$, and a transition function $\delta : Q \times \mathbb{A} \longrightarrow \mathscr{B}^+([0,1] \times \mathbb{C} \times Q)$.

Given $t \in \mathcal{T}_{\mathbb{A}}$, we represent $\mathcal{A}$ *acting on* t in terms of the cost game

$$\mathcal{A} \times t := \langle Q \times \mathcal{T}, (q_0, \epsilon), O, \delta' \rangle$$

where $\delta'((p,x)) = \delta(p, t(x))[(c, (q, xk))/(k, c, q)]$. That is, a position in the game corresponds to a state of the automaton and a location in the input tree; the control function δ' modifies the transition function δ of the automaton to map to the appropriate positions in the game. We set $[\![\mathcal{A}]\!](t) := \text{value}(\mathcal{A} \times t)$, so $\mathcal{A}$ defines a function $[\![\mathcal{A}]\!] : \mathcal{T}_{\mathbb{A}} \longrightarrow \mathbb{N}_\infty$. We say that $\mathcal{A}$ *recognizes* a cost function g if $[\![\mathcal{A}]\!] \approx g$.

If $\delta(q,a)$ is a disjunction of clauses $(0, c', q') \wedge (1, c'', q'')$ for all $(q,a) \in Q \times \mathbb{A}$, then we say the automaton is *nondeterministic*. In this case, we often use a transition relation $\Delta : Q \times \mathbb{A} \times (\mathbb{C} \times Q) \times (\mathbb{C} \times Q)$ in place of δ, where $(q, a, (c', q'), (c'', q'')) \in \Delta$ if and only if $(0, c', q') \wedge (1, c'', q'')$ is a disjunct in $\delta(q,a)$. Since the only choices of Adam are in the branching, a strategy σ in some game $\mathcal{A} \times t$ can be viewed as a labelling of the binary tree such that each branch in the tree corresponds to a play in σ, with positions labelled by the state of the automaton and edges labelled with the action. We call such a labelling R a *run* of $\mathcal{A}$ over t, and write $R(x)$ for the state of the automaton at position x in this run. For a nondeterministic B-parity (respectively, S-parity) automaton, the value of a run is the supremum (respectively, infimum) of the values across all branches in the run tree.

3.3.1 Examples

Let $\mathbb{A} := \{a, b\}$ and consider the following examples of cost automata over infinite trees.

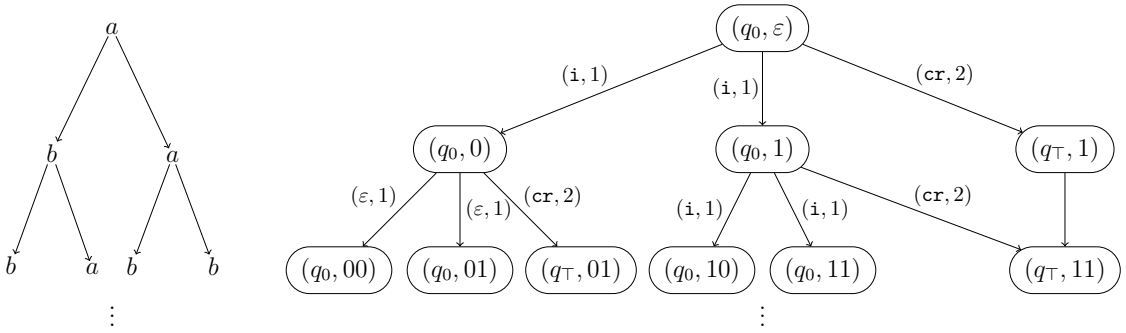

Figure 3.6. Input tree t and game $\mathcal{A} \times t$ (see Example 3.13).

Example 3.13. Let $g(t) = \sup\{|\pi|_a : \pi \text{ is a branch in } t\}$ be the function that maps a tree t to the maximum over all branches of the number of a-labelled positions on the branch. We describe a simple S-$[1,2]$ automaton $\mathcal{A}$ recognizing this function.

Let $\mathcal{A} := \langle \{q_0, q_\top\}, \mathbb{A}, q_0, \mathrm{Cost}_S^{\{1\},[1,2]}, \delta \rangle$ where

$$\delta(q_0, e) := (0, (c, 1), q_0) \vee (1, (c, 1), q_0) \vee (1, (\mathtt{cr}, 2), q_\top)$$

where c is $\mathtt{i}$ if $e = a$ and ε otherwise. The state $q_\top$ represents an accepting sink state, so $\delta(q_\top, e) := (1, (\varepsilon, 2), q_\top)$ for all $e \in \mathbb{A}$. Eve controls every move and is the maximizing player. The idea is that at each position, Eve chooses whether to check-reset the counter (if she has already seen a lot of a's), or select the direction in the tree with the most a's and continue counting. Figure 3.6 shows an initial finite subtree of some input tree t, and the game $\mathcal{A} \times t$ that would come out of it.

Example 3.14. Consider the function $f(t) = |t|_a$ that counts the number of a's in an $\mathbb{A}$-labelled binary tree. There is no cost-parity automaton computing exactly this function, but there is an automaton $\mathcal{A}$ such that $[\![\mathcal{A}]\!] \approx_\alpha f$ for $\alpha(n) = 2^n$.

The nondeterministic B-$[1,2]$ automaton recognizing this cost function acts as follows. Initially, all transitions have priority 2. Eve makes a guess about which subtrees have an a. If the current position is labelled with an a, or she guesses that more than one subtree from the current position has an a, then the counter is incremented. Adam chooses the direction. If it is in a direction that Eve guessed had no a's, then for the remainder of the play the counter is left unchanged, and priority 2 is output unless Adam witnesses an a, in which case the play stabilizes in priority 1. Otherwise, if it is in a direction that Eve guessed had an a, then priority 2 is output and play continues as described above.

Assume there is a tree that has $n \in \mathbb{N}$ a's. Then any play can increment the counter at most n times (when these n a's are on the same branch). Hence, $[\![\mathcal{A}]\!](t) \leq |t|_a$ for all t.

In the other direction, we prove that if $[\![\mathcal{A}]\!](t) \leq n \in \mathbb{N}$, then $|t|_a < 2^n$ where 2 is the maximum arity (rank) of any label in $\mathbb{A}$. We proceed by induction on n. If $n = 0$, then the result is obvious: if the counter is never incremented, then t must have no a's. For $n > 0$, consider the optimal strategy σ for Eve in $\mathcal{A} \times t$ that witnesses value n. Let d be the minimum depth at which a play in σ witnesses an increment. Take a play $\pi \in \sigma$ that witnesses an increment at this minimal depth d and corresponds to position x in t. The subtrees from x can have value at most $n-1$ via $\mathcal{A}$, so by the inductive hypothesis each of the subtrees can have at most $2^{n-1} - 1$ positions labelled a. So the tree rooted at x can have at most $2(2^{n-1} - 1) + 1 < 2^n$ positions labelled a.

We claim that there can be no a's outside of the subtree rooted at x. Assume by contradiction that there were another subtree t' of t that contained a's but was not a subtree of t_x. Then there must be a position $y < x$ which has both t_x and t' as subtrees. There must be an increment at this position (otherwise, Eve would have needed to guess at y that at most one subtree had an a, and Adam could prove otherwise), which is a contradiction based on the choice of depth d and position x. Hence, $|t|_a < 2^n$ as desired.

3.3.2 Duality

A natural expressivity question for cost automata is whether cost automata with different objectives define the same class of cost functions. For alternating cost automata over infinite trees, it is straightforward to show that the B-parity and S-parity objectives (as well as their hierarchical counterparts) are equivalent.

Theorem 3.15. *It is effectively equivalent for a cost function f over infinite trees to be recognizable by cost automata with the following objectives: B-parity, hB-parity, S-parity, hS-parity. Moreover, for $i < j$,*

- every B-$[i, j]$ automaton is effectively equivalent to an hB-$[i, j]$, S-$[i+1, j+1]$, and hS-$[i+1, j+1]$ automaton, and
- every S-$[i, j]$ automaton is effectively equivalent to an hS-$[i, j]$, B-$[i+1, j+1]$, and hB-$[i+1, j+1]$ automaton.

Proof. Assume we are starting with an alternating B-parity automaton $\mathcal{B}$ with objective $\mathrm{Cost}_B^{\Gamma,P}$. We want to construct an equivalent alternating S-parity automaton $\mathcal{S}$. Let $\mathcal{A}_{\mathrm{cost}_B^{\Gamma,P}}$ be the id-history-deterministic S-automaton (over infinite words)

described in Lemma 3.5 which recognizes $\text{cost}_B^{\Gamma,P}$. It would suffice to show that $[\![\mathcal{B}]\!] \approx [\![\mathcal{S}]\!]$, but in fact, we can show that $[\![\mathcal{B}]\!] = [\![\mathcal{S}]\!]$.

Fix some tree t. Then the cost game $\overline{\mathcal{B} \times t}$ is like the game $\mathcal{B} \times t$ but with the roles of the players reversed (conjunctions and disjunctions in the control function switched) and the goal of the objective changed to max. Notice that in this new game, the value for Eve is now the maximum over all strategies σ of the minimum value of all plays $\pi \in \sigma$. This is like an S-parity automaton except for the fact that the values of the plays π are computed using $\text{cost}_B^{\Gamma,P}$. Composing with the S-automaton $\mathcal{A}_{\text{cost}_B^{\Gamma,P}}$ results in a cost game with S-actions and priorities P' as desired. Note that if $P = [i, j]$ for $i < j$, then $P' = [i + 1, j + 1]$ (if $i = j$, then $P' = [i + 1, j + 2]$).

More formally, let $\mathcal{S}$ be an alternating S-parity automaton such that $\mathcal{S} \times t$ is isomorphic to $\mathcal{A}_{\text{cost}_B^{\Gamma,P}} \circ \overline{\mathcal{B} \times t}$ for all t. Then

$$[\![\mathcal{S}]\!](t) = \text{value}(\mathcal{S} \times t) = \text{value}(\mathcal{A}_{\text{cost}_B^{\Gamma,P}} \circ \overline{\mathcal{B} \times t}) = \text{value}(\overline{\mathcal{B} \times t})$$

by Lemma 3.5 and Lemma 3.12. Moreover, by Proposition 3.11, $\text{value}(\overline{\mathcal{B} \times t}) = \text{value}(\mathcal{B} \times t) = [\![\mathcal{B}]\!](t)$. Hence, $[\![\mathcal{S}]\!](t) = [\![\mathcal{B}]\!](t)$ for all t as desired.

The other transformations are similar, using the other history deterministic automata in Lemmas 3.5 and 3.6. Going from B to hB or S to hS is even simpler since dualization of the game is not necessary, but these conversions introduce the correction function from Lemma 3.6. □

3.4 Cost automata with both counter types

We usually work with cost automata with a single objective, and consequently only one type of counter, B or S. For some technical constructions later in this thesis, however, we must work with both counter types simultaneously in the form of a nondeterministic BS-Büchi automaton, described in joint work with Kuperberg [KVB11]. We define this type of automaton in this chapter with the other automaton models, but recommend omitting this section on the first reading and returning to it when needed in Chapter 6. For notational clarity, we use the traditional description of the Büchi condition via a set of accepting states, rather than a parity condition (to make it easier to distinguish between B-accepting and S-accepting states).

A *nondeterministic BS-Büchi automaton* over infinite words is a tuple

$$\mathcal{A} = \langle Q, \mathbb{A}, q_0, \Gamma_B, F_B, \Gamma_S, F_S, \Delta \rangle.$$

The set Γ_B (respectively, Γ_S) is the set of *B-counters* (respectively, *S-counters*); we assume Γ_B and Γ_S are disjoint. Likewise, F_B (respectively, F_S) is the set of *B-accepting states* (respectively, *S-accepting states*); these sets are not required to be disjoint. The transition function is $\Delta \subseteq Q \times \mathbb{A} \times \mathbb{C} \times Q$ where $\mathbb{C} = \mathbb{B}^{\Gamma_B} \times \mathbb{S}^{\Gamma_S}$ is the set of counter actions.

A run ρ of $\mathcal{A}$ over $u = a_1 a_2 \cdots \in \mathbb{A}^\omega$ is $(q_i, a_{i+1}, c_{i+1}, q_{i+1})_{i \in \mathbb{N}} \in \Delta^\omega$. We say ρ is *B-accepting* if $\mathrm{Inf}(\rho) \cap F_B \neq \emptyset$. The *$B$-value* is $\mathrm{value}_B(\rho) := \sup\left\{\bigcup_{\gamma \in \Gamma_B} C(\mathrm{pr}_\gamma(\mathrm{out}(\rho)))\right\}$, the supremum over the checked counter values coming from B-counters. The corresponding notions of *S-accepting* and *S-value* are defined as expected by replacing B with S and sup with inf in the definitions above.

We can define functions $[\![\mathcal{A}]\!]_B$ and $[\![\mathcal{A}]\!]_S$ as expected (by restricting to the B-part or S-part of the run):

$$[\![\mathcal{A}]\!]_B(u) := \inf\{\mathrm{value}_B(\rho) : \rho \text{ is a } B\text{-accepting run of } \mathcal{A} \text{ on } u\},$$
$$[\![\mathcal{A}]\!]_S(u) := \sup\{\mathrm{value}_S(\rho) : \rho \text{ is an } S\text{-accepting run of } \mathcal{A} \text{ on } u\}.$$

We can also define semantics that are related to both counter types. In particular, the *S-semantics relative to the B-value* is a function $[\![\mathcal{A}]\!]_S^B : \mathbb{N} \to \mathbb{A}^\omega \to \mathbb{N}_\infty$ that seeks to maximize the value over S-accepting runs that also have some bounded B-value:

$$[\![\mathcal{A}]\!]_S^B(m)(u) := \sup\left\{\mathrm{value}_S(\rho) : \begin{array}{l}\rho \text{ is an } S\text{-accepting run of } \mathcal{A} \text{ on } u \\ \text{and } \mathrm{value}_B(\rho) \leq m\end{array}\right\}.$$

There is a corresponding notion of B-semantics relative to the S-value, but this will not be needed.

We say $\mathcal{A}$ is *BS-hierarchical* (and call $\mathcal{A}$ a nondeterministic *hBS*-Büchi automaton) if the counters $\Gamma_B \cup \Gamma_S$ are globally numbered $[1, k]$ (for $k = |\Gamma_B| + |\Gamma_S|$) and for any action on $\mathbb{B}^{\Gamma_B} \times \mathbb{S}^{\Gamma_S}$ there is some $i \in [1, k]$ such that ε is performed on all counters $j > i$ and $\mathtt{r}$ on all counters $j < i$.

Example 3.16. Let $\mathcal{A}$ and $\mathcal{A}'$ be nondeterministic BS-automata on infinite words over $\mathbb{A} := \{a, b, c\}$ pictured in Figure 3.7, each with one B- and one S-counter. We write $a : (d, d')$ if on input a, the output is action d (respectively, d') for the S-counter (respectively, B-counter). We omit self-loops with $c : (\varepsilon, \varepsilon)$. All states are B-accepting, but only the states that are not initial are S-accepting.

These automata are very similar. For instance, $[\![\mathcal{A}]\!]_B = [\![\mathcal{A}']\!]_B = |\cdot|_b$. The key difference is $\mathcal{A}'$ is BS-hierarchical, with the B-counter above the S-counter.

Notice that we have $[\![\mathcal{A}]\!]_S \approx |\cdot|_a$ (if there are a finite number of a's, then the best run of $\mathcal{A}$ moves to the accepting state when reading the final a; otherwise, for

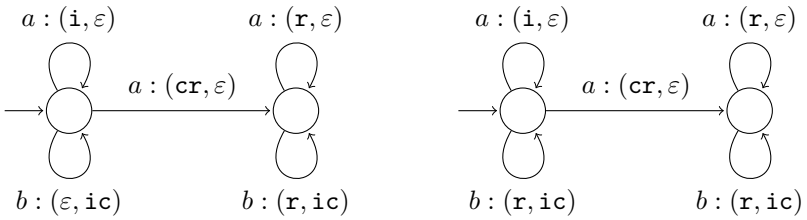

Figure 3.7. Nondeterministic BS-Büchi automata $\mathcal{A}$ and $\mathcal{A}'$ with $\mathcal{A} \cong \mathcal{A}'$ (see Example 3.16).

every n, there is an accepting run of $\mathcal{A}$ such that the S-counter has value n). In $\mathcal{A}'$, however, the B-counter is higher than the S-counter so $\mathcal{A}'$ forces a reset of the S-counter when a b is read in the initial state. Since there is no a priori bound on the number of b's in the input, this means $[\![\mathcal{A}']\!]_S \not\approx [\![\mathcal{A}]\!]_S$. However, for any fixed m and any u such that $[\![\mathcal{A}]\!]_B(u) \leq m$, the S-value of $\mathcal{A}$ on u is $\approx_{\beta_m}$-equivalent to $\mathcal{A}'$ on u with $\beta_m(n) = n(m+1)$. This means that $[\![\mathcal{A}]\!]_S^B(m) \approx_{\beta_m} [\![\mathcal{A}]\!]_S^B(m)$.

In order to capture the fact that these BS-Büchi automata are very similar, we introduce a new equivalence relation $\cong$ which we call *BS-equivalence*. We write $\mathcal{A} \cong_\alpha^\beta \mathcal{A}'$ for a correction function α and a family of correction functions $(\beta_m)_{m\in\mathbb{N}}$ if

- $[\![\mathcal{A}]\!]_B \approx_\alpha [\![\mathcal{A}']\!]_B$, and
- for all $m \in \mathbb{N}$, $[\![\mathcal{A}]\!]_S^B(m) \preccurlyeq_{\beta_m} [\![\mathcal{A}']\!]_S^B(\alpha(m))$ and $[\![\mathcal{A}']\!]_S^B(m) \preccurlyeq_{\beta_m} [\![\mathcal{A}]\!]_S^B(\alpha(m))$.

We say that $\mathcal{A}$ and $\mathcal{A}'$ are *BS-equivalent*, written $\mathcal{A} \cong \mathcal{A}'$, if there exists α and $(\beta_m)_{m\in\mathbb{N}}$ such that $\mathcal{A} \cong_\alpha^\beta \mathcal{A}'$. The idea is that the B-semantics are preserved between $\mathcal{A}$ and $\mathcal{A}'$ as usual, and the S-semantics relative to the B-value are also preserved (after adjusting for any differences in the bound on the B-value). Although it is technical, this definition captures the notion that two BS-Büchi automata behave in a similar fashion (as in Example 3.16).

It turns out that given any nondeterministic BS-Büchi automaton like $\mathcal{A}$, there is a nondeterministic hBS-Büchi automaton $\mathcal{A}'$ satisfying $\mathcal{A} \cong \mathcal{A}'$. There is a similar result in [BC06] for automata with both B- and S-counters but in a setting where only boolean properties about boundedness and unboundedness are considered, unlike the quantitative setting here.

This translation can be done effectively by a nondeterministic BS-Büchi transducer that reads an infinite word of non-hierarchical counter actions and outputs BS-hierarchical counter actions. It does this in a history deterministic way, satisfying both the property of a history deterministic S-automaton as well as the stronger

property that every run, including the runs driven by the history deterministic translation strategies, preserve the B-value (up to some correction α).

Let $\mathcal{I}(\Gamma_B, \Gamma_S)$ denote the BS-Büchi automaton that reads words over $\mathbb{B}^{\Gamma_B} \times \mathbb{S}^{\Gamma_S}$ and outputs these actions unchanged.

Theorem 3.17 ([KVB11]). *For all sets Γ_B, Γ_S of counters, there exists effectively an hBS-automaton $\mathcal{H}(\Gamma_B, \Gamma_S)$ such that $\mathcal{H}(\Gamma_B, \Gamma_S) \approx_\alpha^\beta \mathcal{I}(\Gamma_B, \Gamma_S)$. Moreover, there is a family of translation strategies $(\vartheta_n)_{n\in\mathbb{N}}$ such that for all $u \in \mathbb{B}^{\Gamma_B} \times \mathbb{S}^{\Gamma_S}$, if $\mathrm{value}_B(u) \leq m$ and $\mathrm{value}_S(u) \geq \beta_m(n)$ then*

- for all runs ρ of $\mathcal{H}(\Gamma_B, \Gamma_S)$ over u, $\mathrm{value}_B(\rho) \leq \alpha(m)$;
- the run ρ of $\mathcal{H}(\Gamma_B, \Gamma_S)$ driven by ϑ_n over u is S-accepting with $\mathrm{value}_S(\rho) \geq n$.

The transducer $\mathcal{H}(\Gamma_B, \Gamma_S)$ has the same set of B counters, but extra copies of the S-counters. The principle of the automaton is to split the input word into sequences of S-actions from $\{\mathtt{i}, \varepsilon\}^*$ that are between resets of the B-counters. It uses one copy of the S-counter to count the number of S-increments within each sequence, and another copy to count the sequences with at least one S-increment. If the S-value is high compared to the B-value, then the transducer will also have a high S-value, obtained from one of the copies. Every state is B-accepting and S-accepting except a single sink state which is not S-accepting but is still B-accepting. We omit the formal technical proof but refer the interested reader to [KVB11].

Note that all of these definitions for nondeterministic BS-Büchi automata can be extended to infinite trees in the expected way. This means that we can use the transducers to transform arbitrary nondeterministic BS-Büchi automata over words or trees into hierarchical BS-Büchi automata which are easier to work with.

In particular, we can package a nondeterministic B-Büchi and S-Büchi automaton into a single nondeterministic BS-Büchi automaton and then convert into an $\approx$-equivalent hBS-Büchi automaton. Let $\mathcal{U} = \langle Q_\mathcal{U}, \mathbb{A}, q_0^\mathcal{U}, \Gamma_B^\mathcal{U}, F_B^\mathcal{U}, \Delta_\mathcal{U}\rangle$ (respectively, $\mathcal{U}' = \langle Q_{\mathcal{U}'}, \mathbb{A}, q_0^{\mathcal{U}'}, \Gamma_S^{\mathcal{U}'}, F_S^{\mathcal{U}'}, \Delta_{\mathcal{U}'}\rangle$) be a nondeterministic B-Büchi (respectively, S-Büchi) automaton over trees. Then the nondeterministic BS-Büchi automaton $\mathcal{U}\times\mathcal{U}'$ defined by $\langle Q_\mathcal{U} \times Q_{\mathcal{U}'}, \mathbb{A}, (q_0^\mathcal{U}, q_0^{\mathcal{U}'}), \Gamma_B^\mathcal{U}, F_B^\mathcal{U} \times Q_{\mathcal{U}'}, \Gamma_S^{\mathcal{U}'}, Q_\mathcal{U} \times F_S^{\mathcal{U}'}, \Delta\rangle$ has transitions Δ which combine transitions from $\Delta_\mathcal{U}$ and $\Delta_{\mathcal{U}'}$ on the same input. Lemma 3.12 and Theorem 3.17 imply the following result.

Lemma 3.18. $\mathcal{H}(\Gamma_B^\mathcal{U}, \Gamma_S^{\mathcal{U}'}) \circ (\mathcal{U} \times \mathcal{U}') \approx \mathcal{U} \times \mathcal{U}'$.

3.5 Discussion

This chapter has established the framework for cost automata on infinite words and infinite trees. In particular, we have defined the semantics of alternating cost automata on infinite trees in terms of a two-player infinite-duration cost game in which one player seeks to minimize the value and the other player seeks to maximize the value according to some objective that takes the place of a classical winning condition.

In addition to definitions and examples, the most important results in this chapter are related to transforming objectives. We showed that history deterministic automata on words compose well with cost games (Lemma 3.12). Moreover, certain history deterministic automata can be used to convert cost games to use a different objective without changing the value. This allowed us to prove a duality result for alternating cost-parity automata in Theorem 3.15, showing that alternating B-parity automata over infinite trees and alternating S-parity automata over infinite trees recognize the same cost functions.

One of the advantages of using alternating automata in the classical setting is that complementation is easy: one only needs to switch conjunctions and disjunctions in the transition function, and dualize the acceptance condition in order to construct an automaton that accepts the complement language. Although the duality result here was more involved, it still required only a switch of conjunctions and disjunctions in the transition function, and then a composition with a history deterministic cost automaton which dualized the objective.

In the classical theory, it is more challenging to show that alternating parity automata can be simulated by nondeterministic parity automata. In the next chapter, we turn to a much finer analysis of the strategies required in these cost games, which will help us show that certain alternating cost-parity automata over infinite trees can be simulated by nondeterministic cost-parity automata.

Chapter 4

Strategies in Cost Games

Strategies in games played on finite or infinite graphs can be complicated, with the move at a given position depending on the entire history of the play leading to it. A well-known result in the theory, however, is that parity games are positionally determined [EJ91, Mos91]. Determinacy means that from each position in the game one of the two players has a winning strategy. Positional determinacy implies that if Eve, say, has a winning strategy, then Eve has a simple winning strategy where each move only depends on the current position and does not require any memory of the play leading to that position. Likewise, Muller games have finite memory determinacy [GH82], which implies that there is a strategy for the winning player where each move only depends on the current position and a state (taken from a finite set of memory states) that is a function of the history of the play.

Büchi had already realized a connection between determinacy and complementation [Büc77]. Later, Muller and Schupp [MS84, MS95] showed a connection between positional determinacy of parity games and simulation of alternating automata with nondeterministic automata. Positional determinacy of parity games also has algorithmic consequences when solving a parity game on a finite game graph (i.e. determining which player has a winning strategy from a given position) and can be used to show that this problem is in NP $\cap$ co-NP. Hence, the shape of strategies in games has proven to be quite important.

We prove in Section 4.2 that some cost-parity games admit finite memory (or even positional) strategies while others do not. Unlike the classical case, restricting to finite memory strategies cannot guarantee the exact value obtained using arbitrary strategies; instead, we guarantee an $\approx_\alpha$-equivalent value where α is a correction function that depends on the number of counters. We use these results to prove in Section 4.3

that certain alternating cost-Büchi automata can be simulated by nondeterministic cost-Büchi automata.

We also show in Section 4.4 that $f \preccurlyeq g$ is decidable when f is given as a nondeterministic S-parity automaton and g as a nondeterministic B-parity automaton, and describe how this is related to the classical decision procedure testing language inclusion for regular languages of infinite trees.

This chapter is an expanded presentation of results first mentioned in [VB11] and is one of the main contributions of this thesis.

4.1 Preliminaries

Let $\mathcal{G} = \langle V, v_0, O, \delta \rangle$ be a cost game with $O = \langle \mathbb{C}, f, \mathit{goal} \rangle$, where $\delta(v)$ is written in disjunctive normal form for all $v \in V$. Recall that a strategy σ for Eve in $\mathcal{G}$ is a set of plays such that if a partial play $\pi \in \mathrm{pref}(\sigma)$ ends in position v, then there is a single disjunct in $\delta(v)$, say $(c'_1, v'_1) \wedge \cdots \wedge (c'_j, v'_j)$, such that for every conjunct (c'_i, v'_i) for $i \in [1, j]$, $\pi \cdot (v, c'_i, v'_i) \in \mathrm{pref}(\sigma)$.

We use a memory structure in order to describe the amount of memory used by a strategy. A *memory structure* for $\mathcal{G}$ is a tuple $\mathcal{M} = \langle M, m_0, \mathrm{update} \rangle$ such that M is a set of memory states, m_0 is the initial state, and $\mathrm{update} : M \times E_{\mathcal{G}} \to M$ is the memory update function. The function $\mathrm{update}^* : M \times E_{\mathcal{G}}^* \to M$ is defined inductively such that $\mathrm{update}^*(m, \epsilon) = m$ and for some partial play $\pi = \pi(0) \cdots \pi(j) \in E_{\mathcal{G}}^+$,

$$\mathrm{update}^*(m, \pi) = \mathrm{update}^*(\mathrm{update}(m, \pi(0)), \pi(1) \cdots \pi(j)).$$

The *size of the memory structure* is the cardinality of M.

We say that a strategy σ for Eve is a *strategy using finite memory* l if there is a memory structure $\mathcal{M} = \langle M, m_0, \mathrm{update} \rangle$ of size $l \in \mathbb{N}$ and a *next-move function* $\mathrm{next} : M \times V \to \mathscr{B}^+(\mathbb{C} \times V)$ that satisfies the following conditions:

- for all $m \in M$ and $v \in V$, $\mathrm{next}(m, v)$ is a single disjunct in $\delta(v)$;
- $\pi = (v_0, c_1, v_1) \cdots (v_i, c_{i+1}, v_{i+1}) \in \mathrm{pref}(\sigma)$ if and only if (c_{i+1}, v_{i+1}) is a conjunct in $\mathrm{next}(\mathrm{update}^*(m_0, (v_0, c_1, v_1) \cdots (v_{i-1}, c_i, v_i)), v_i)$.

That is, next describes how Eve should choose her next move (a disjunct in $\delta(v)$) based on the current position v and the memory state m given by update^*, which is a function of the history of the play.

In the special case that $l = 1$, we say σ is a *positional* or *memoryless strategy*, and next can be written as a function $\mathrm{next} : V \to \mathscr{B}^+(\mathbb{C} \times V)$.

A nice way to picture a strategy σ is in the form of a tree where each branch represents a play $\pi \in \sigma$. We will call this the *strategy tree* T corresponding to σ (sometimes called a *run tree* or *computation tree* in the literature [MS84]). Formally, we view this as a graph where the initial position s_0 (the root) is labelled with v_0, and if there is a path π in T ending in a position s labelled with $v \in V$, then there is an edge labelled with action $c' \in \mathbb{C}$ from s to s' labelled with $v' \in V$ (written (s, c', s')) if and only if $\pi \cdot (v, c', v') \in \text{pref}(\sigma)$. We write S for the set of positions in the strategy tree T and $h : S \longrightarrow V$ for the homomorphism that maps a position $s \in S$ in the strategy tree to its label $v \in V$ which is the corresponding game position.

For an arbitrary strategy and corresponding strategy tree T, Eve's choice at a particular $v \in V$ may depend on the history of the play leading to v. Thus, there may be $s, s' \in h^{-1}(v)$ such that the moves in T from s and s' are different. In a positional strategy, however, the moves must be identical for any $s, s' \in h^{-1}(v)$. For a finite memory strategy, the moves must be identical for any positions $s, s' \in h^{-1}(v)$ with histories π and π' such that $\text{update}^*(m_0, \pi) = \text{update}^*(m_0, \pi')$.

Later in the chapter, we will take a strategy tree T, and construct a positional strategy from it by choosing for each $v \in V$ a single element of $h^{-1}(v)$ for Eve to play like, regardless of the history of the play. In those settings, we view a positional strategy as a mapping from V to S.

4.1.1 Classical results

A natural question to ask is how much memory is needed to describe the winning strategies in some family of games. A fundamental result in the theory of infinite games is that parity games are positionally determined, and therefore represent a class of games with particularly simple winning strategies.

Theorem 4.1 ([EJ91, Mos91]). *Parity games are positionally determined.*

As mentioned earlier, a game is determined if from each position in the game, one of the two players has a winning strategy. The determinacy of parity games follows from the fact that the games are Borel (see the proof of Proposition 3.11). Moreover, the winning player always has a winning positional strategy. The fact that positional strategies are sufficient has been proven in many different ways (see [GTW02] for a survey). Later in this chapter, we will adapt the signature-based approach found in, e.g. [Wal96, Jur00, GW06] (see Section 4.2).

Regardless of the proof method, the positional determinacy of parity games has important algorithmic consequences. Given a positional strategy in a parity game on

a finite graph, it can be checked in polynomial time whether this strategy is winning for a certain player from a given initial vertex (it amounts to checking certain graph properties related to loops). Hence, this leads to an NP algorithm for determining, for each node, whether Eve has a winning strategy from that position. Since it is symmetric in the players, this leads to an NP $\cap$ co-NP algorithm for *solving a parity game* (determining the winning regions for each player). In fact, this has been improved to a UP $\cap$ co-UP algorithm in [Jur98], where UP is the complexity class of problems solvable using an unambiguous nondeterministic polynomial time, i.e. a nondeterministic polynomial time Turing machine with at most one accepting computation for each input.

Theorem 4.2 ([Jur98]). *Solving a parity game on a finite game graph is in the complexity class* UP $\cap$ co-UP*.*

A major open question in this field is whether there is a polynomial time algorithm for solving parity games on finite graphs.

As mentioned in Chapter 3, another common winning condition is the Muller condition. The *latest appearance record construction* [GH82] allows a Muller game to be converted to a parity game such that Eve has a winning strategy from the initial position in the original Muller game if and only if she has a winning strategy from the initial position in the new parity game. Moreover, this reduction is effective and is obtained by composing the original Muller game with a deterministic finite state automaton, so the positional strategy in the parity game can be converted to a finite memory strategy in the original Muller game.

Theorem 4.3 ([GH82]). *Finite memory determinacy holds for Muller games. Solving Muller games on finite game graphs is decidable.*

Indeed, there are many other problems that can also be reduced to solving parity games. For instance, deciding the emptiness problem for tree automata with a parity acceptance condition (i.e. whether $L(\mathcal{A}) = \emptyset$ for a parity automaton $\mathcal{A}$) is polynomial time equivalent to solving a parity game. Likewise, in Section 4.4 we show how to reduce the decidability of the domination preorder $\preccurlyeq$ for some cost functions over infinite trees to solving a Muller game, and hence solving a parity game.

4.1.2 Limitations with cost games

In classical parity and Muller games, the previous results imply that finite memory strategies are sufficient: when looking for a winning strategy, we can always restrict

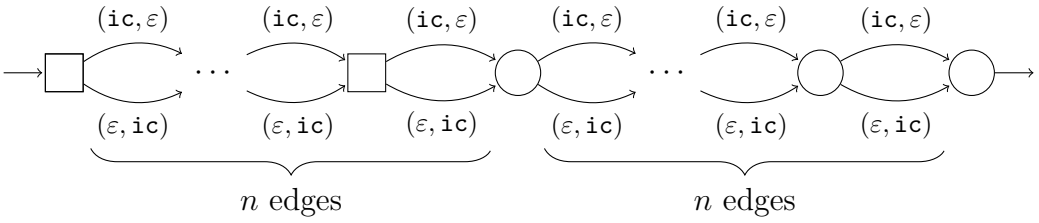

Figure 4.1. The B-game $\mathcal{G}_n$ with no finite memory strategy of size n achieving the optimal value (see Proposition 4.4).

the search to finite memory strategies. However, by restricting Eve to finite memory strategies in cost games, she may not be able to achieve the optimal value. This is the case even in very simple games on acyclic finite game graphs (called *finite duration cost games* in [CL10]). We write, e.g. Cost_B^Γ for the objective that is similar to the B-parity objective defined in Chapter 3, but with a condition involving a terminal symbol rather than priorities since the plays are finite.

Proposition 4.4. *Let $\mathscr{G}$ be the family of acyclic cost games with finite game graphs and objective Cost_B^Γ where $|\Gamma| = 2$. For all $n \in \mathbb{N}$, there is a game $\mathcal{G}_n \in \mathscr{G}$ such that any finite memory strategy σ of size n satisfies $\mathrm{value}(\sigma) > \mathrm{value}(\mathcal{G})$.*

Proof. Fix some n and consider the game $\mathcal{G}_n$ shown in Figure 4.1 with counters $\Gamma = \{\gamma_1, \gamma_2\}$ and objective Cost_B^Γ which is based on an example due to Löding [Löd09]. We write (c_1, c_2) for action c_1 on γ_1 and c_2 on γ_2. The idea is that Adam (respectively, Eve) controls which counter to increment and check in the first set of n positions (respectively, second set of n positions).

Assume for the sake of contradiction that there is a finite memory strategy σ in the game $\mathcal{G}_n$ using memory structure $\mathcal{M} = \langle M, m_0, \mathrm{update}\rangle$ with $|M| \leq n$ such that $\mathrm{value}(\mathcal{G}_n) = \mathrm{value}(\sigma) = n$.

Consider a set of plays $\{\pi_0, \ldots, \pi_n\}$ such that π_i is a partial play ending in the first position controlled by Eve that assigns value i to γ_1 (and hence value $n - i$ to γ_2). By the pigeon-hole principle there must be $i \neq j$ such that $\mathrm{update}^*(m_0, \pi_i) = \mathrm{update}^*(m_0, \pi_j)$.

There must be a unique partial play π_k starting from the first position controlled by Eve such that $\pi_i\pi_k \in \sigma$ and $\pi_j\pi_k \in \sigma$ (there can be only one such play because $\mathrm{update}^*(m_0, \pi_i) = \mathrm{update}^*(m_0, \pi_j)$). Assume γ_1 (respectively, γ_2) is incremented k (respectively, $n - k$) times on π_k.

Then $\mathrm{value}(\sigma) = \sup\{\mathrm{value}(\pi) : \pi \in \sigma\} \geq \max\{\mathrm{value}(\pi_i\pi_k), \mathrm{value}(\pi_j\pi_k)\}$.

The least possible value for value($\pi_i\pi_k$) is n (in the case that $k = n-i$). Indeed, if $k \neq n-i$, then value(σ) $\geq$ value($\pi_i\pi_k$) $> n$ which contradicts the initial assumption.

So suppose that $k = n - i$. Then

$$\begin{aligned}\text{value}(\pi_j\pi_k) &= \max\{j+k, (n-j)+(n-k)\} \\ &= \max\{n+(j-i), n+(i-j)\} > n\end{aligned}$$

since $i \neq j$ implies that $j - i > 0$ or $i - j > 0$. Hence, value(σ) $> n =$ value($\mathcal{G}_n$), contradicting the initial assumption.

Note that these games $\mathcal{G}_n$ could be defined in terms of an alternating cost automaton $\mathcal{A}$ acting on words of the form $a^n b^n$, where $\mathcal{A}$ has one state (which is initial and accepting) and two counters, and on input a (respectively, b), Adam (respectively, Eve) selects which of the two counters to increment and check. □

The previous result is not actually problematic for the desired applications in this thesis. Indeed, the theme in the theory of regular cost functions is that exact values do not matter, since we only care about functions up to the cost function equivalence. Hence, for a family of cost games we seek a correction function α such that finite memory strategies achieve values that are $\approx_\alpha$-equivalent to the values obtained using any strategies.

Definition 4.5. We say a cost game $\mathcal{G} = \langle V, v_0, \langle \mathbb{C}, f, \mathit{goal}\rangle, \delta\rangle$ admits *α-m strategies* for Eve if restricting Eve to finite memory strategies of size m results in an $\approx_\alpha$-equivalent value:

$$\text{value}(\mathcal{G}) \approx_\alpha \text{op}\{\text{value}(\sigma) : \sigma \text{ is a strategy using finite memory of size } m \text{ in } \mathcal{G}\}$$

where op is inf (respectively, sup) if $\mathit{goal} = \min$ (respectively, max). If $m = 1$, then we say $\mathcal{G}$ is *α-positional* for Eve.

Given a family $\mathscr{G}$ of cost games, if there exists α and $m \in \mathbb{N}$ such that all cost games $\mathcal{G} \in \mathscr{G}$ have α-m strategies for Eve, then we say that *finite memory* (α-m) *strategies suffice for Eve in $\mathscr{G}$*. If $m = 1$, then we say that α-positional strategies suffice for Eve.

The first positive result like this is due to Colcombet and Löding [CL08a] for cost games on acyclic finite game graphs using hB and $\overline{hB}$ objectives, written Cost^Γ_{hB} and $\overline{\text{Cost}^\Gamma_{hB}}$ (as mentioned before, these objectives are similar to the hB-parity and $\overline{hB\text{-parity}}$ objectives defined in Chapter 3, but with a condition involving a terminal symbol rather than priorities since the plays are finite).

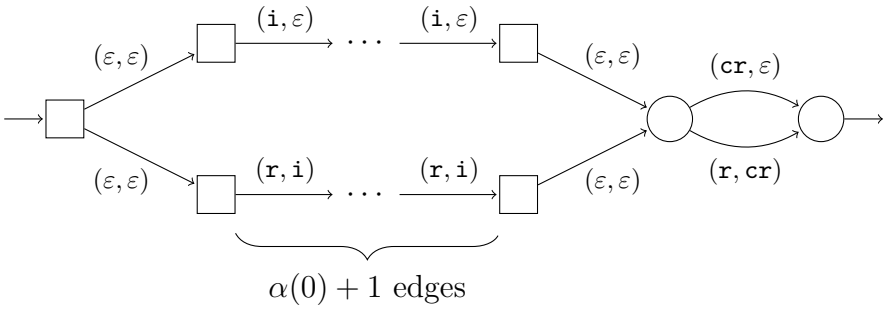

Figure 4.2. The hS-game $\mathcal{G}_\alpha$ with no α-positional strategy (see Proposition 4.7).

Proposition 4.6 ([CL08a, Lemma 7],[CL10, Theorem 8])**.** *Let $\mathscr{G}$ be the family of acyclic cost games with finite game graphs and objective $\mathrm{Cost}^\Gamma_{hB}$ or $\overline{\mathrm{Cost}^\Gamma_{hB}}$ for hierarchical counters Γ. Then α-positional strategies suffice for Eve in $\mathscr{G}$ for $\alpha(n) = n^{|\Gamma|}$.*

We will usually phrase the results in terms of strategies for Eve like this. Because the dual objective $\overline{\mathrm{Cost}^\Gamma_{hB}}$ is equivalent to switching the roles of the two players, another way to state this result is that both Adam and Eve admit α-positional strategies in $\mathrm{Cost}^\Gamma_{hB}$ games.

On the other hand, S-games do not necessarily admit positional strategies, even when played on a finite acyclic game graph.

Proposition 4.7. *Let $\mathscr{G}$ be the family of acyclic cost games with finite game graphs and objective $\mathrm{Cost}^\Gamma_{hS}$ or $\overline{\mathrm{Cost}^\Gamma_{hS}}$ for hierarchical counters Γ. For all α, there is a game $\mathcal{G}_\alpha \in \mathscr{G}$ that is not α-positional for Eve.*

Proof. Fix some α. Consider the game $\mathcal{G}_\alpha \in \mathscr{G}$ shown in Figure 4.2.

The idea in $\mathcal{G}_\alpha$ is that Adam chooses one of two counters to increment $\alpha(0) + 1$ times, and then Eve chooses one of the two counters to check-reset. Since Eve is the maximizing player in an hS-game, the optimal strategy for Eve is to always check-reset the counter that Adam increments. Hence, $\mathrm{value}(\mathcal{G}_\alpha) = \alpha(0) + 1$.

However, a positional strategy σ for Eve must select a single counter to check-reset regardless of how Adam plays. If Eve check-resets counter 1 (respectively, counter 2), then the play $\pi \in \sigma$ in which Adam increments counter 2 (respectively, counter 1), has value 0. Hence, $\mathrm{value}(\sigma) = 0$ for any positional strategy σ, contradicting the fact that $\mathcal{G}_\alpha$ is α-positional.

Likewise, consider the $\overline{hS}$-game played on the same game graph, but where Eve is now the minimizing player. The value of the game is 0 (since Eve can choose to check-reset the counter that was not incremented by Adam) but the value of any positional strategy is $\alpha(0) + 1$. This shows that this $\overline{hS}$-game is not α-positional.

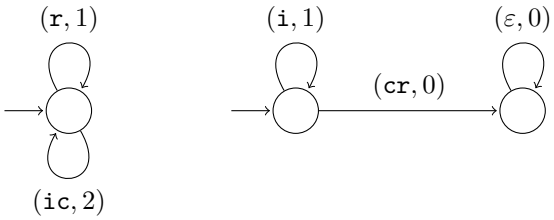

Figure 4.3. Cyclic hB-$[1, 2]$ and hS-$[0, 1]$ games with no α-positional strategies (see Proposition 4.8).

Similar to Proposition 4.4, we point out that these games could be defined as $\mathcal{A} \times u$ for a suitable alternating hS-automaton or alternating $\overline{hS}$-automaton on finite words over the alphabet $\{a, b\}$. □

The results are also negative when allowing cyclic game graphs.

Proposition 4.8. *There is a cyclic hB-$[1, 2]$ game $\mathcal{G}$ and cyclic hS-$[0, 1]$ game with one counter such that for all α, there is no α-positional strategy for Eve.*

Proof. Consider the following games in Figure 4.8. We claim that neither admits α-positional strategies for any α.

Consider the hB-$[1, 2]$ game on the left that has value 1 (the optimal strategy alternates between selecting the $(\mathtt{r}, 1)$ edge and the $(\mathtt{ic}, 2)$ edge which ensures that the parity condition is satisfied and the counter achieves value at most 1). However, a positional strategy must select only one of these edges. If it selects $(\mathtt{r}, 1)$ then the parity condition is not satisfied; if it selects $(\mathtt{ic}, 2)$, then the parity condition is satisfied but the counter value is unbounded. Hence, any positional strategy has value ∞, and there is no correction function α such that $\alpha(1) = \infty$.

Next consider the hS-$[0, 1]$ game on the right that has value ∞ (there is a family of strategies σ_n for Eve that stay in the initial position until the counter achieves value n; hence, the supremum over these strategies is ∞). However, a positional strategy must select for the initial position only a single edge to be taken. If the $(\mathtt{i}, 1)$ loop is selected, then the only play consistent with this strategy has value 0 because the parity condition is not satisfied; if the $(\mathtt{cr}, 0)$ edge is taken, then the only play consistent with this strategy has value 0 since the counter is checked without it ever being incremented. In either case, the value according to a positional strategy is 0, and there is no correction function α such that $\alpha(0) = \infty$. In fact, it is not hard to see that even finite memory strategies (of any size) do not suffice for Eve. □

Note that this is very different than the classical setting where parity games admit finite memory (in fact, positional) strategies even on cyclic game graphs.

This is not a major setback, since we are primarily interested in the games $\mathcal{A} \times t$ where $\mathcal{A}$ is an alternating cost-parity automaton and t is a tree. These games are *finite branching* (there are only finitely many outgoing edges from any position) and *acyclic* (there are no cycles in the game graph). The finite branching condition is often needed in order to apply *König's lemma* which, in one common form, states that a finitely branching tree with infinitely many nodes must contain an infinite branch.

Some proofs in the next section also require a chronological game graph. We say a game $\mathcal{G} = \langle V, v_0, O, \delta \rangle$ is *chronological* if there is a mapping $\text{depth} : V \longrightarrow \mathbb{N}$ from positions in the game to $\mathbb{N}$ such that $\text{depth}(v_0) = 0$ and if v' appears in $\delta(v)$, then $\text{depth}(v') = \text{depth}(v) + 1$. Notice that if a game is chronological then it is acyclic. For games $\mathcal{A} \times t$, $\text{depth}((q, x))$ is simply $|x|$, the depth of position x in the tree t.

4.2 Shape of strategies in cost-parity games

In this section, we describe the shape of strategies in certain cost-parity games. Theorem 4.9, Corollary 4.11, and Corollary 4.12 summarize the results for cost games that allow infinite plays (in contrast to the finite duration games mentioned in Proposition 4.6).

Theorem 4.9. *Let $\mathcal{G}$ be a cost game $\langle V, v_0, O, \delta \rangle$. For all sets P of priorities, $\mathcal{G}$ admits α-m strategies for Eve if*

(a) $O = \langle \{\mathtt{ic}, \varepsilon\} \times P, \text{cost}_B^{\{1\},P}, \min \rangle$, $\alpha = \text{id}$, $m = 1$;[1] or

(b) $O = \langle \{\mathtt{ic}, \mathtt{r}\} \times P, \text{cost}_B^{\{1\},P}, \min \rangle$, $\alpha(n) = 2n$, $m = 2$, and $\mathcal{G}$ is chronological.

For all sets $\Gamma = [1, k]$ of counters, $\mathcal{G}$ admits α-m strategies for Eve if $\mathcal{G}$ is chronological and finite branching and

(c) $O = \text{Cost}_{hB}^{\Gamma,[1,2]}$, $\alpha(n) = (n+1)^k$, $m = 2$; or

(d) $O = \overline{\text{Cost}_{hB}^{\Gamma,[0,1]}}$, $\alpha(n) = (n+1)^k$, $m = 2$.

[1] A similar result holds for $O = \langle \{\mathtt{ic}, \varepsilon\} \times P, \text{cost}_B^{\{1\},P}, \max \rangle$, $\alpha = \text{id}$, $m = 1$ [CL12].

Parts (a) and (b) are not new. Part (a) was previously known [CL12] but unpublished, and part (b) follows from the published work in [CKL10]. These cases allow any set of priorities P but restrict to a subset of actions from $\mathbb{B}$. The games in part (a) are called *distance-parity games* since the allowed counter actions correspond to the actions in the distance automata of Hashiguchi [Has82], and the games in part (b) are called *desert-parity games* since they correspond to the actions in the desert automata of Kirsten [Kir04]. We give proofs of these results in Sections 4.2.1 and 4.2.2 for completeness, and in order to introduce techniques that will be used in the new parts (c) and (d).

The proofs of parts (c) and (d) given in Sections 4.2.3 and 4.2.4 are contributions of this thesis. These proofs are an improved and expanded presentation of results first stated in [VB11][2], and are fundamental for the results in later chapters.

A key technique in the proofs of parts (b)–(d) is to use composition with history deterministic automata to convert between games with different objectives. We have already seen in Lemma 3.12 that composition with history deterministic automata preserves the value of the game. The memory required for strategies in the original game is also related to the memory required for strategies in the composed game and the number of states in the history deterministic automaton.

Lemma 4.10. *Let $\mathcal{A} = \langle Q, \mathbb{A}, q_0, \langle \mathbb{C}, g, goal\rangle, \Delta\rangle$ be an α-history-deterministic cost automaton over infinite words and $\mathcal{G} = \langle V, v_0, \langle \mathbb{A}, [\![\mathcal{A}]\!], goal\rangle, \delta\rangle$ be a cost game. If $\mathcal{A} \circ \mathcal{G}$ admits finite memory strategies of size N for Eve, then $\mathcal{G}$ admits finite memory strategies of size $|Q| \cdot N$ for Eve.*

Proof. Assume that $goal = \min$, $\text{value}(\mathcal{G}) \leq n$, and $\mathcal{G}' := \mathcal{A} \circ \mathcal{G}$ admits β-N strategies. By Lemma 3.12, $\text{value}(\mathcal{G}') \leq \alpha(n)$ and since we are assuming $\mathcal{G}'$ admits β-N strategies, there is a finite memory strategy σ' of size N in $\mathcal{G}'$ with $\text{value}(\sigma') \leq \beta(\alpha(n))$. It suffices to show that there is a finite memory strategy of size $|Q| \cdot N$ with value at most $\beta(\alpha(n))$ in $\mathcal{G}$.

There is a memory structure $\mathcal{M}' = \langle M, m_0, \text{update}'\rangle$ with $|M| = N$ and next-move function $\text{next}' : M \times (Q \times V) \to \mathscr{B}^+(\mathbb{C} \times (Q \times V))$ describing Eve's choices in σ'. We construct a new memory structure $\mathcal{M} := \langle Q \times M, (q_0, m_0), \text{update}\rangle$ for $\mathcal{G}$. We need to define next and update (which will then induce a finite memory strategy σ for $\mathcal{G}$).

Fix some $m \in M$, $q \in Q$, and $v \in V$ and assume that $\text{next}'(m, (q, v)) = (c_1, (q_1, v_1)) \wedge \cdots \wedge (c_j, (q_j, v_j))$ and let $m_i := \text{update}'(m, ((q, v), c_i, (q_i, v_i))$ for all

[2] In [VB11], the assumption about chronological game graphs was implied but not stated explicitly. Part (d) was also not stated clearly in that paper.

$i \in [1, j]$. Recall that there is an edge from (q, v) to (q_i, v_i) with action c_i in $\mathcal{G}'$ if and only if there is an edge from v to v_i labelled by some a_i in $\mathcal{G}$ and there is some $(q, a_i, c_i, q_i) \in \Delta$. This means that there is some $(q, a_i, c_i, q_i) \in \Delta$ for all $i \in [1, j]$ and we can define $\text{next}((q, m), v) := (a_1, v_1) \wedge \cdots \wedge (a_j, v_j)$ and $\text{update}((q, m), (v, a_i, v_i)) := (q_i, m_i)$. By doing this for all m, q, and v, next and update induce a finite memory strategy σ of size $|Q| \cdot N$ in $\mathcal{G}$.

For every $\pi' \in \sigma'$, there is a corresponding play $\pi \in \sigma$. Consider $\text{out}(\pi') = u' \in \mathbb{C}^\omega$ and $\text{out}(\pi) = u \in \mathbb{A}^\omega$. By the definition of next and update above, the sequence u' is the output from a run of $\mathcal{A}$ on u. But $\text{value}(\pi)$ in $\mathcal{G}$ is defined as the minimum value across all runs of $\mathcal{A}$ on u. This means that $\text{value}(\pi) \leq \text{value}(\pi') \leq \beta(\alpha(n))$. Since this is true for all $\pi \in \sigma$, σ is the desired finite memory strategy of size $|Q| \cdot N$ with value at most $\beta(\alpha(n))$.

The proof is similar for $goal = \max$. □

For instance, using the deterministic transducers in Lemma 3.7, we can reduce B-parity games with multiple counters but restricted actions $\{\mathtt{ic}, \varepsilon\}$ or $\{\mathtt{ic}, \mathtt{r}\}$ to the single counter distance-parity or desert-parity cases from part (a) or (b).

Corollary 4.11. *Fix $\mathbb{B}' \in \{\{\mathtt{ic}, \varepsilon\}, \{\mathtt{ic}, \mathtt{r}\}\}$. For all $i \leq j$, finite memory strategies suffice for Eve in B-$[i, j]$ games on chronological game graphs where the actions for each counter are restricted to $\mathbb{B}'$.*

Likewise, because there is a finite memory reduction from B-parity games to hB-parity games (via the deterministic automaton in Lemma 3.6), we get the following corollary of Theorem 4.9 parts (c) and (d).

Corollary 4.12. *Finite memory strategies suffice for Eve in B-$[1, 2]$ games and $\overline{B\text{-}[0, 1]}$ games on chronological game graphs with finite branching.*

In the remainder of this section, we prove each part of Theorem 4.9 in turn.

4.2.1 Positional strategies in distance-parity games

There have been a number of approaches for proving positional determinacy in parity games. We use *signature assignments* in the style of [EJ91, Wal96] and [Jur00] (where they are called *small progress measures*). As an introduction to this signature assignment method, we prove that id-positional strategies suffice for Eve in distance-parity games.[3]

[3]There are alternative approaches to proving this result, including a proof using an attractor construction and induction on the number of priorities [CL12].

We start with a strategy τ that witnesses a bounded cost n in some distance-parity game $\mathcal{G} = \langle V, v_0, O, \delta \rangle$. Without loss of generality, we assume that $P = [i, j]$ for $i \in [0, 1]$ (this can be accomplished by shifting the range of indices by multiples of 2). We consider the corresponding strategy tree T, and let V (respectively, S) denote the set of positions in $\mathcal{G}$ (respectively, T). Let $h : S \longrightarrow V$ denote the homomorphism that maps a position in the strategy tree to the corresponding game position.

Using τ, Eve's choice at a particular $v \in V$ may depend on the history of the play leading to v. Thus, there may be $s, s' \in h^{-1}(v)$ such that the moves possible from s and s' are different. We will define a positional strategy σ by selecting a single element of $h^{-1}(v)$ for Eve to play like (regardless of the history) when she reaches a position v. As a result, we think of σ as a mapping from V to S.

The proof consists of the following steps.

1. Define a signature for each node in T that captures relevant information about the game. Specify an ordering on signatures (in our case, this will always be a lexicographic ordering on vectors of ordinals or natural numbers).

2. Construct a positional strategy σ by mapping $v \in V$ to the element in $h^{-1}(v)$ with the least signature.

3. Prove that value$(\sigma) \approx_\alpha$ value(τ).

In this distance-parity case, the components of the signature need to include information about both the counters and the priorities. Since we want to minimize the number of increments and the number of high odd priorities, the components of the signature should measure these things.

- Let $\beta(s) \in [0, n]$ be the number of times a path from s in T can increment the counter. The strategy should try to minimize β in order to minimize the cost.

- For $p \in P$, let $\eta_p(s)$ be an ordinal that represents the maximum number of times a path from s in T visits some priority p before visiting a higher priority. Formally, let X_p^0 be the set of $s \in S$ such that every path from s visits a priority greater than p before visiting priority p, and for all ordinals $\beta > 0$, let X_p^β be the set of $s \in S$ such that for any $s' \in S$ with a path from s to s' visiting priority p, every successor of s' is in $\bigcup_{\beta' < \beta} X_p^{\beta'}$. Then $\eta_p(s) := \inf \left\{ \beta : s \in X_p^\beta \right\}$. This is well-defined when T is a strategy tree witnessing value n and is well-known from the classical setting (see, e.g. [GW06] for more information). The strategy should try to minimize η_p for odd p in order to satisfy the parity condition.

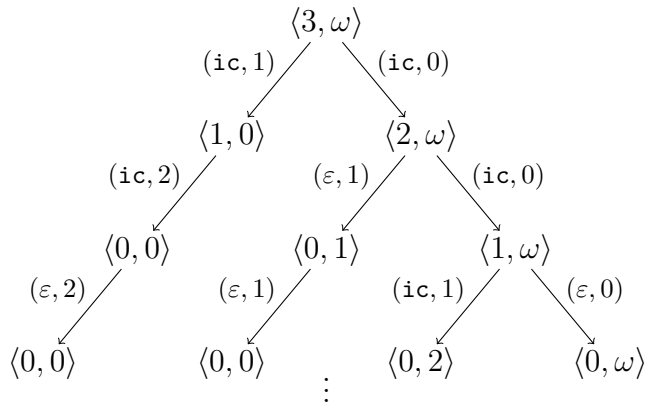

Figure 4.4. A distance-parity strategy tree annotated with the signature.

Based on these components, we define the *signature* at s in T to be

$$\mathrm{sig}(s) := \langle \beta(s), \eta_{p'}(s), \eta_{p'-2}(s), \ldots, \eta_3(s), \eta_1(s) \rangle$$

where p' is the maximum odd priority in P. We write $\mathrm{sig}(s) < \mathrm{sig}(s')$ if the vector from $\mathrm{sig}(s)$ is less than $\mathrm{sig}(s')$ under the standard lexicographical ordering.

We give an example in Figure 4.4 of a strategy tree that comes from a distance-parity game using priorities $[0, 2]$ and a bounded cost of 3 (at most 3 increments on any play). Each node s is labelled with the signature $\langle \beta(s), \eta_1(s) \rangle$. We assume there are no increments outside of the portion of the strategy tree shown in Figure 4.4. The spine 1^ω uses only priority 0, but starting from each node 1^i is a branch with $i + 1$ occurrences of priority 1 before the play stabilizes in priority 2. Note that $\beta(s) \leq 3$ for all $s \in S$ since the strategy tree has a bounded cost 3. However, despite the fact that each branch only has finitely many positions with priority 1 before priority 2 is visited, $\eta_1(s) = \omega$ for $s \in 1^*$ because there is no natural number bound on the number of priority 1 before priority 2 on the branches leading from the spine.

The following lemma describes the conditions under which the components of the signature decrease.

Lemma 4.13. *Let T be the strategy tree corresponding to a strategy τ witnessing some bounded value n for a distance-parity game $\mathcal{G}$. Then for any edge $(s, (c, p), s')$ in T,*

(a) $\beta(s') \leq \beta(s) \leq n$, and $\beta(s') < \beta(s)$ if $c = \mathtt{ic}$;

(b) for all odd $q > p$, $\eta_q(s') \leq \eta_q(s)$;

(c) if p is odd, then $\eta_p(s') < \eta_p(s)$.

The fact that T comes from a distance-parity game where the counter can only be incremented or left unchanged (never reset) means that β is non-increasing (and has a maximum value of n). The components related to the priorities can increase: if p is the current priority, η_q for $q < p$ can increase. However, η_q for $q > p$ must either decrease or stay the same since visiting priority p cannot increase the maximum number of times a priority $q > p$ is visited before a higher priority is seen. Also, if p is an odd priority, than the η_p component of the signature must strictly decrease. Note that this lemma would not hold if T did not represent a strategy where each branch satisfied the parity condition and had counter value at most n.

We can now prove that distance-parity games are id-positional. As mentioned earlier, this result was previously known but unpublished [CL12].

Proposition 4.14. *If $\mathcal{G}$ is a distance-parity game, then $\mathcal{G}$ is* id*-positional for Eve.*

Proof. Fix an optimal strategy τ witnessing value$(\mathcal{G}) = n$. For each node s in the corresponding strategy tree T, we define

$$\text{sig}(s) := \langle \beta(s), \eta_{p'}(s), \eta_{p'-2}(s), \ldots, \eta_3(s), \eta_1(s) \rangle$$

as described above. We use this to construct a positional strategy $\sigma : V \longrightarrow S$ where $\sigma(v)$ selects the node $s \in h^{-1}(v)$ with the lexicographically-least signature. For $v \in V$, let $\text{sig}(v) := \text{sig}(\sigma(v))$.

It is clear that value$(\tau) \leq$ value(σ) (otherwise τ would not be optimal).

We now show that σ satisfies value$(\sigma) \leq$ value(τ). Suppose for the sake of contradiction that value$(\sigma) >$ value(τ). Then there exists $\pi = (v_0, c_1, v_1)(v_1, c_2, v_2) \cdots$ in σ such that value$(\pi) > n$. By Lemma 4.13 and the definition of σ, β is non-increasing on π, and if the counter is incremented then β must strictly decrease. Hence, the counter is incremented at most n times, and there must be some position in π after which β is constant.

This means that the only way for π to exceed value n is if the parity condition is not satisfied on π. Let i be the position in π after which β is constant and any priority that occurs is in $\text{Inf}(\pi)$ (the set of infinitely occurring priorities in π). Let o be the maximum priority occurring infinitely often in this suffix; we know o must be odd since we are assuming that the parity condition is not satisfied.

Consider some $v = v_{i'}$ for $i' \geq i$ and the move $(v, (c, p), w)$ from $v = v_{i'}$ to $w = v_{i'+1}$ in $\mathcal{G}$ according to π. There is a corresponding move $(r, (c, p), s)$ from $r = \sigma(v)$ to some position s in T with $h(s) = w$ (but s is not necessarily $\sigma(w)$). Then $\text{sig}(v) = \text{sig}(r) \geq \text{sig}(s) \geq \text{sig}(w)$ by definition of σ and Lemma 4.13. Moreover, if

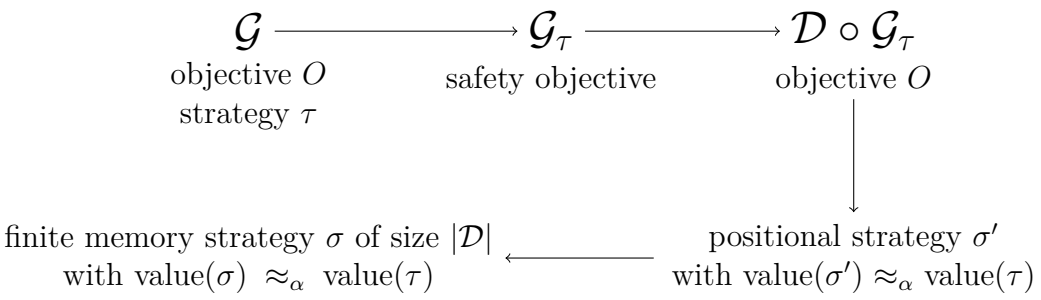

Figure 4.5. Proof structure for Theorem 4.9 parts (b)–(d).

$p = o$ (the maximum odd priority occurring infinitely many times) then $\mathrm{sig}(r) > \mathrm{sig}(s)$ by Lemma 4.13, which implies that $\mathrm{sig}(v) > \mathrm{sig}(w)$. Hence, we can build an infinite descending chain of signatures

$$\mathrm{sig}(v_{i_0}) > \mathrm{sig}(v_{i_1}) > \cdots$$

where $i \leq i_0 < i_1 < \cdots$ index the infinitely many moves in π visiting priority o. This contradicts the well-foundedness of vectors of ordinals. □

4.2.2 Finite memory strategies in desert-parity games

Now let $\mathcal{G}$ be a chronological desert-parity game with priorities $P = [i, j]$ for $i \in [0, 1]$. Recall that the allowed counter actions in $\mathcal{G}$ are $\{\mathtt{ic}, \mathtt{r}\}$. Minimizing the signature from Proposition 4.14 does not work because β is no longer non-increasing: after a reset, the number of increments from a given position may increase, which means the value of β may also increase. Therefore, minimizing the signature from the previous section would always keep the counter values bounded, but could result in a play that does not satisfy the parity condition because minimizing β would always take precedence over minimizing the number of visits to odd priorities.

Assume $\mathrm{value}(\mathcal{G}) = n \in \mathbb{N}$. Then there is some strategy τ that witnesses value n. We seek a finite memory strategy of size two that witnesses value at most $2n$.

The idea is that we will convert the desert-parity game $\mathcal{G}$ into a more structured desert-parity game $\mathcal{D} \circ \mathcal{G}_\tau$ (via $\mathcal{G}_\tau$, which has a different objective). In the game $\mathcal{D} \circ \mathcal{G}_\tau$ it is possible to prove that there is a positional strategy using a similar technique as Proposition 4.14. We then show that this positional strategy can be used to find a finite memory strategy in the original game, where the memory depends on $\mathcal{D}$ (see Figure 4.5).

The first step is to define a *desert-safety game* $\mathcal{G}_\tau = \langle V, v_0, O', \delta' \rangle$ based on $\mathcal{G}$ and τ. This new game is identical to $\mathcal{G}$ except *signals* \$ have been added to the

output for counter actions leading to depths that are multiples of $n+1$.[4] Formally,

$$\delta'(v) = \begin{cases} \delta(v)[(c\$, w)/(c, w)] & \text{if } \mathrm{depth}(v) = j(n+1) - 1 \text{ for some } j \geq 1 \\ \delta(v) & \text{otherwise} \end{cases}$$

For a particular $j \in \mathbb{N}$, the game positions between depths $j(n+1)$ and $(j+1)(n+1)-1$ (i.e. between occurrences of \$) are called a *slice*. The new objective in $\mathcal{G}_\tau$ is

$$O' := \langle (\{\mathtt{ic}, \mathtt{r}\} \times P) \cdot \{\epsilon, \$\}, f', \min \rangle$$

where f' assigns the usual cost to a play based on $\mathrm{cost}_B^{\{1\},P}$ but also ensures that there is at least one reset between the signals in the word (and assigns value ∞ if not). This new game can be viewed as a more structured version of the original game since the signals \$ provide information about depths. However, the value of the game is the same.

Lemma 4.15. $\mathrm{value}(\mathcal{G}) = \mathrm{value}(\mathcal{G}_\tau) = n$.

Proof. The strategy τ from $\mathcal{G}$ can be played in $\mathcal{G}_\tau$ simply by adding the \$ output at appropriate depths. It is not hard to see that τ still has value n in $\mathcal{G}_\tau$. Therefore, $\mathrm{value}(\mathcal{G}_\tau) \leq \mathrm{value}(\mathcal{G})$. Assume for the sake of contradiction that there were a strategy τ' in $\mathcal{G}_\tau$ with value strictly less than n. Then this strategy (with \$ removed) could be played in $\mathcal{G}$ to witness value less than n, contradicting the fact that $\mathrm{value}(\mathcal{G}) = n$. Together, this means $\mathrm{value}(\mathcal{G}) = \mathrm{value}(\mathcal{G}_\tau)$. □

Now define $\mathcal{D}$ to be the deterministic desert-P automaton over infinite words that recognizes the valuation f' in O' (see Figure 4.6). It reads words over the alphabet $(\{\mathtt{ic}, \mathtt{r}\} \times P) \cdot \{\epsilon, \$\}$, checking whether there is at least one reset between \$, and otherwise outputs counter actions and priorities according to the original operations (notice that only priority 1 is output in state q_{rej}, so this can be viewed as a rejecting sink state that could be omitted without changing the value).

Let $\mathcal{G}' := \mathcal{D} \circ \mathcal{G}_\tau$. Because $\mathcal{D}$ is deterministic, the composition game $\mathcal{D} \circ \mathcal{G}_\tau$ has the same value as $\mathcal{G}_\tau$ (by Lemma 3.12) but now uses actions from a traditional desert-P game. Despite the fact that it has the same objective as the original game, the new game carries additional information: namely, a position in the game graph is now of the form (q, v) and the state q of $\mathcal{D}$ stores whether or not a reset has been visited yet

[4] The notation $\mathcal{G}_n$ (rather than $\mathcal{G}_\tau$) would probably be more indicative of the relationship with $\mathcal{G}$ since this new desert-safety game depends on $\mathrm{value}(\tau) = n$, but does not require any other information about τ. However, for consistency in notation with the later sections, we use the notation $\mathcal{G}_\tau$ here.

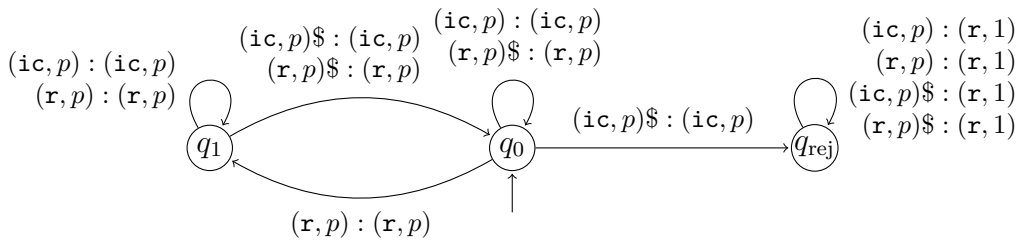

Figure 4.6. Deterministic desert-parity automaton $\mathcal{D}$ recognizing the valuation in a desert-safety game.

in the current slice. The additional structure in this game makes it easier to define a positional strategy with a low value. Indeed, it prevents the positional strategy defined below from jumping between parts of the strategy tree in such a way that it avoids doing a reset.

Based on this observation, we will use a signature that only captures information relevant to the parity condition. The components of the signature are the η_p defined on page 78. These components decrease as described previously when in a strategy tree witnessing a bounded value n.

Lemma 4.16. *Let T' be the strategy tree corresponding to a strategy τ' witnessing some bounded value n in a desert-parity game $\mathcal{G}'$. Then for any edge $(s, (c, p), s')$ in T',*

(a) for all odd $q > p$, $\eta_q(s') \leq \eta_q(s)$;

(b) if p is odd, then $\eta_p(s') < \eta_p(s)$.

Lemma 4.17. *There is a positional strategy σ' in $\mathcal{G}'$ such that* $\text{value}(\mathcal{G}') \approx_\alpha \text{value}(\sigma')$ *for $\alpha(n) = 2n$.*

Proof. Assume $\text{value}(\mathcal{G}') = n \in \mathbb{N}$. Fix some strategy τ' in $\mathcal{G}'$ and let T' be the corresponding strategy tree witnessing value n. We define

$$\text{sig}(s) := \langle \eta_{p'}(s), \eta_{p'-2}(s), \ldots, \eta_3(s), \eta_1(s) \rangle$$

where η_p is defined on page 78 and p' is the highest odd priority. Let $\sigma' : V \longrightarrow S$ be the positional strategy such that $\sigma'(v)$ selects the node $s \in h^{-1}(v)$ with the lexicographically-least signature.

It is clear that $\text{value}(\sigma') \geq \text{value}(\mathcal{G}')$ (since $\text{value}(\mathcal{G}')$ is defined as the minimum value over all strategies, which includes σ').

We now show that σ' satisfies $\text{value}(\sigma') \preccurlyeq_\alpha \text{value}(\mathcal{G}')$ for $\alpha(n) = 2n$. Suppose for contradiction that $\text{value}(\sigma') > 2n$. Then there is some $\pi = (v_0, c_1, v_1)(v_1, c_2, v_2) \cdots$ in σ' such that $\text{value}(\pi) > 2n$.

If π fails to satisfy the parity condition, then we can use an argument similar to the end of the proof of Proposition 4.14 to build an infinite descending chain of signatures, which is a contradiction.

Hence, in order for value$(\pi) > 2n$, there must be some segment of π witnessing $2n+1$ increments without a reset. Because of the length of this segment, there is a subsegment $(w_0, (d_1, p_1), w_1) \cdots (w_n, (d_{n+1}, p_{n+1}), w_{n+1})$ with $d_k = \texttt{ic}$ for $k \in [1, n+1]$ and $w_0 = v_{j(n+1)}$ for some $j \geq 0$. This means that this subsegment represents a portion of π spanning an entire slice (from depth $j(n+1)$ to $(j+1)(n+1)-1$) and having $n+1$ increments. Let $s_k := \sigma'(w_k)$ for $k \in [0, n+1]$.

Because w_0 is a position at the beginning of the slice, the game position must store the fact that no reset has been seen yet in the current slice (this is the part of the state coming from the memory of $\mathcal{D}$). Now σ' plays like s_0, moving to some game position w_1 and outputting $\texttt{ic}$. Since no reset occurred, the position w_1 must also record the fact that no reset has yet occurred in this slice. Continuing like this, we eventually reach position w_n that for the same reasoning must also record that no reset has yet been seen in the slice. But $d_{n+1} = \texttt{ic}$, which means that there is some branch from a position in $h^{-1}(w_0)$ to s_{n+1} in T' that witnesses no resets in this slice. This contradicts the fact that T' is a strategy tree of value n, since a branch in T' without a reset in some slice should yield value ∞.

Another way to think about this is to consider the worst scenario for σ'. We have just seen that any play consistent with σ' must respect the condition that a reset is visited in each slice. But the positional strategy σ' could jump between branches in T' in such a way that it builds a segment $\texttt{r}(\texttt{ic})^n(\texttt{ic})^n\texttt{r}$ that spans two slices and satisfies the condition that each slice has a reset, but yields value $2n$.

We have shown that every play in σ' must satisfy the parity condition and have counter value at most $2n$, which means that value$(\mathcal{G}') \approx_\alpha$ value(σ') for $\alpha(n) = 2n$. □

We can use this lemma to show that the original desert-parity game $\mathcal{G}$ admits a finite memory strategy.

Proposition 4.18. *If $\mathcal{G}$ is a chronological desert-parity game, then there exists an α-2 finite memory strategy σ for Eve such that* value$(\mathcal{G}) \approx_\alpha$ value(σ) *for $\alpha(n) = 2n$.*

Proof. Assume $\mathcal{G}$ is a desert-parity game with value$(\mathcal{G}) = n \in \mathbb{N}$. Then there is some optimal strategy τ witnessing value n. As described above, we construct $\mathcal{G}_\tau$ based on $\mathcal{G}$ and τ, and by Lemma 4.15, value$(\mathcal{G}_\tau)$ = value$(\mathcal{G})$. Moreover, by Lemma 3.12, value$(\mathcal{D} \circ \mathcal{G}_\tau)$ = value$(\mathcal{G}_\tau)$, since $\mathcal{D}$ is id-history-deterministic (indeed $\mathcal{D}$ is deterministic) and $[\![\mathcal{D}]\!] = f'$, the valuation in the objective for $\mathcal{G}_\tau$. Hence, there is a strategy τ'

in $\mathcal{G}' := \mathcal{D} \circ \mathcal{G}_\tau$ with value$(\tau') = n$. This means we can apply Lemma 4.17 to get a positional strategy σ' for Eve in $\mathcal{G}'$ with value at most $2n$.

As described in the Lemma 4.10, this implies that there is a finite memory strategy $\sigma_{\mathcal{G}_\tau}$ in $\mathcal{G}_\tau$ of value at most $2n$, and the size of the memory depends on the number of states in $\mathcal{D}$. Although the automaton in Figure 4.6 uses three states, we can actually omit the rejecting sink state q_{rej} since any strategy of low value will never need to use this state. Hence there is a finite memory strategy of size 2 in $\mathcal{G}_\tau$ of value at most $2n$ using some memory structure $\mathcal{M}_{\mathcal{G}_\tau} = \langle \{q_0, q_1\}, q_0, \text{update}_{\mathcal{G}_\tau} \rangle$ where $\text{update}_{\mathcal{G}_\tau}$ mimics the transition function in $\mathcal{D}$.

Finally, we can take this finite memory strategy $\sigma_{\mathcal{G}_\tau}$ and remove all \$ from the output to get a finite memory strategy σ of size 2 in $\mathcal{G}$ with value at most $2n$. The memory structure is based on $\mathcal{M}_{\mathcal{G}_\tau}$ but must account for the fact that \$ no longer appears explicitly in the output. This means that $\mathcal{M} := \langle \{q_0, q_1\}, q_0, \text{update} \rangle$ where $\text{update}(m, (v, (c, p), v')) := \text{update}_{\mathcal{G}_\tau}(m, (v, (c, p)b, v'))$ and b is \$ if $\text{depth}(v) = j(n+1) - 1$ for some $j \geq 1$, and ϵ otherwise. □

Thus, we have shown that finite memory strategies of size 2 suffice for Eve in desert-parity games.

This result also follows from [CKL10]. In that work, *temporal cost functions* are studied over finite words. This subclass of regular cost functions admits various equivalent presentations, including B-automata where the counter actions are restricted to `ic` and `r`, as well as the one counter version that corresponds to desert automata. Temporal cost functions also admit a "clock-based presentation" in which a regular language over words is labelled with ticks of a clock (where the ticks of the clock do not depend on the particular input). This approach can be extended to infinite trees [CKL10, Col12b] and the slices in the proof here roughly correspond to the ticks of the clock in the clock-based presentation.

4.2.3 Finite memory strategies in hB-[1,2] games

We now turn to Theorem 4.9, part (c). For notational convenience, we will use the same names (such as $\mathcal{G}$, $\mathcal{G}_\tau$, and $\mathcal{D}$), even though these define different objects in this case. In this way, the proof outline in Figure 4.5 still applies.

hB-safety games

We start by examining the properties of *hB-safety games*. Let $\mathbb{C} := H_B^{[1,k]} \times [1, 2]$ be the output alphabet for an hB-$[1, 2]$ game (see Section 3.1), and let $\mathbb{C}' := \mathbb{C} \cdot \{\epsilon, \$\}$.

The *hB-safety objective* over hierarchical counters Γ is

$$\text{Safety-Cost}_{hB}^{\Gamma,[1,2]} = \langle \mathbb{C}', \text{safety-cost}_{hB}^{\Gamma,[1,2]}, \min \rangle.$$

The valuation $\text{safety-cost}_{hB}^{\Gamma,[1,2]}$ is a function that reads words over $\mathbb{C}'$, the usual alphabet for hB-$[1,2]$ games extended with signals \$. For $u \in (\mathbb{C}')^\omega$, $\text{safety-cost}_{hB}^{\Gamma,[1,2]}(u)$ is defined as follows.

- If there are finitely many \$ in u, then $\text{safety-cost}_{hB}^{\Gamma,[1,2]}(u) := \infty$.
- If there are infinitely many \$ in u, then $u = u_0\$u_1\$\cdots$ such that each $u_i \in \mathbb{C}^*$ contains no \$. If there is some u_i in u such that there is no priority 2, then $\text{safety-cost}_{hB}^{\Gamma,[1,2]}(u) := \infty$. Otherwise, $\text{safety-cost}_{hB}^{\Gamma,[1,2]}(u) := \text{cost}_{hB}^{\Gamma,[1,2]}(h_\$(u))$ where $h_\$$ removes all \$ from u.

The idea is that this valuation not only ensures that priority 2 is visited infinitely often, but also that it is visited at least once in each slice between signals \$.

It is not hard to see that there is a two state deterministic hB-$[1,2]$ automaton $\mathcal{D}$ on words over the input alphabet $\mathbb{C}'$ and with objective $\langle \mathbb{C}, \text{cost}_{hB}^{\Gamma,[1,2]}, \min \rangle$ recognizing $\text{safety-cost}_{hB}^{\Gamma,[1,2]}$. This automaton has two states which are used to remember whether priority 2 has been seen in the current slice between signals \$, and otherwise outputs actions from $\mathbb{C}$ unchanged.

By Lemma 3.12, we can compose $\mathcal{D}$ with any hB-safety game $\mathcal{G}$ to get a game with an hB-$[1,2]$ objective but the same value. This composition $\mathcal{D} \circ \mathcal{G}$ admits positional strategies.

Lemma 4.19. *For every chronological hB-safety game $\mathcal{G}$ with finite branching, there is a positional strategy σ' in $\mathcal{G}' := \mathcal{D} \circ \mathcal{G}$ such that* $\text{value}(\mathcal{G}') \approx_\alpha \text{value}(\sigma')$ *for $\alpha(n) = (n+1)^k$ where k is the number of hierarchical counters.*

Proof. Fix a strategy τ' witnessing a bounded cost n in $\mathcal{G}' := \mathcal{D} \circ \mathcal{G}$. For each node s in the corresponding strategy tree T', we define

$$\text{sig}(s) := \langle \beta_k(s), \beta_{k-1}(s), \ldots, \beta_1(s) \rangle$$

where $\beta_j(s)$ is the number of times a path from s can increment counter j before a reset. Note that $\beta_j(s) \leq n$ for all $j \in [1,k]$ and all $s \in S$. Moreover, in a move (s, c, s') where counter j is the highest counter with action not equal to ε, $\beta_{j'}(s') \leq \beta_{j'}(s)$ for $j' \geq j$, and $\beta_j(s') < \beta_j(s)$ if the action on counter j is `ic`.

We use this to construct a positional strategy $\sigma' : V \longrightarrow S$ where $\sigma'(v)$ selects the node $s \in h^{-1}(v)$ with the lexicographically-least signature. We let $\mathrm{sig}(v) := \mathrm{sig}(\sigma'(s))$.

We must show that σ' satisfies $\mathrm{value}(\sigma') \preccurlyeq_\alpha \mathrm{value}(\tau')$ where $\alpha(n) := (n+1)^k$. Suppose for contradiction that $\mathrm{value}(\sigma') > \alpha(\mathrm{value}(\tau'))$. The first thing to notice is that each play consistent with σ' must satisfy the parity condition. Assume not. Then there is some position after which no transitions of priority 2 occur, so there is some i such that no transitions of priority 2 occur between d_i and d_{i+1}. Let v be the position in the play at d_i. We proceed by induction on the length l of the play between d_i and d_{i+1} that does not witness transitions of priority 2. If $l = 1$ (i.e. $d_{i+1} - d_i = 1$), then the fact that no transition of priority 2 occurred between d_i and d_{i+1} means that there is some node $s \in h^{-1}(v)$ at depth d_i in T' such that there is a move from s that is not labelled with priority 2. This contradicts the fact that T' is a strategy tree for τ' witnessing value n. Otherwise, if $l > 1$, let v be the position at depth d_i and let v' be the next position in the play using strategy σ'. We can assume the transition between v and v' is labelled with priority 1 (otherwise we immediately have a contradiction). The important point is that the positions in $\mathcal{G}'$ record the state of $\mathcal{D}$. This means that any $s' \in h^{-1}(v')$ with parent s in T' (note that $h(s)$ is not necessarily v) must also have an edge that is labelled with priority 1 and must be at depth $d_i + 1$ because the game graph is chronological. This means we can apply the inductive hypothesis from v' to get the desired contradiction.

The only other way $\mathrm{value}(\sigma') > \alpha(\mathrm{value}(\tau'))$ is if there is some counter j (for $1 \leq j \leq k$) that is incremented more than $(n+1)^k$ times on $(v_0, c_0, v_1)(v_1, c_1, v_2)\cdots$. Consider a segment of the play starting at some position v_{i_0} that witnesses more than $(n+1)^k$ increments for some counter j at positions $I = \{i_0, i_1, \ldots, i_{(n+1)^k}, \ldots\}$ without any intermediate resets for counter j or any increments or resets for counters $j' > j$. The maximum signature at v_{i_0} is $\langle n, \ldots, n\rangle$. If $i \in I$ then $\mathrm{sig}(v_{i+1}) < \mathrm{sig}(v_i)$ and one of the first $k - j + 1$ coordinates witnesses this strict decrease; otherwise the first $k - j + 1$ coordinates must remain the same or decrease between v_i and v_{i+1}. A counting argument (or an induction on j) shows that after at most $(n+1)^{k-j+1}$ increments, the first $k - j + 1$ components of the signature have decreased to 0, so no further increments from counter j are possible. This means there must be at most $(n+1)^{k-j+1}$ increments on this segment, contradicting the fact that this segment starting at v_{i_0} has more than $(n+1)^k$ increments. □

By Lemma 4.10, this implies that any hB-safety game admits finite memory strategies for Eve.

Corollary 4.20. *For all chronological hB-safety games $\mathcal{G}$ with finite branching, there exists an α-2 finite memory strategy σ for Eve in $\mathcal{G}$ such that* $\text{value}(\mathcal{G}) \approx_\alpha \text{value}(\sigma)$ *for $\alpha(n) = (n+1)^k$ where k is the number of hierarchical counters.*

hB-[1,2] games

Fix a game $\mathcal{G} = \langle V, v_0, O, \delta \rangle$ which is an hB-$[1,2]$ game with k hierarchical counters $\Gamma = [1,k]$ (so $O = \text{Cost}_{hB}^{[1,k],[1,2]}$). We assume that $\mathcal{G}$ has a chronological and finite branching game graph.

Assume $\text{value}(\mathcal{G}) = n \in \mathbb{N}$. Fix τ in $\mathcal{G}$ with $\text{value}(\tau) = n$, and let T be the corresponding strategy tree.

We say a hB-safety game $\mathcal{G}'$ is *based on* a hB-$[1,2]$ game $\mathcal{G}$ if the arenas are identical except for the fact that signals \$ have been added to some edges in the game graph. We now define a particular hB-safety game $\mathcal{G}_\tau$ based on $\mathcal{G}$.

We begin by defining inductively a strictly increasing sequence of depths $(d_i)_{i \in \mathbb{N}}$ where $d_0 := 0$ and

$$d_{i+1} := \inf \{ d : \forall \text{ branches } \pi \text{ in } T.\ \text{priority 2 occurs on } \pi \text{ between depth } d_i \text{ and } d \}.$$

In other words, d_{i+1} is chosen such that all paths in T have at least one transition labelled with priority 2 between depths d_i and d_{i+1}. This is well-defined since T is finite branching: if there were no bound on the depth at which a priority 2 transition is reached, then König's lemma would imply that there is an infinite branch in T with only finitely many priority 2, a contradiction. For each $i \in \mathbb{N}$, the game positions between depths d_i and $d_{i+1} - 1$ are said to be in *slice i*.[5]

These depths are used to transform $\mathcal{G}$ into a hB-safety game $\mathcal{G}_\tau$ based on $\mathcal{G}$. Formally $\mathcal{G}_\tau := \langle V, v_0, \text{Safety-Cost}_{hB}^{\Gamma,[1,2]}, \delta' \rangle$ so the game positions are unchanged and if $v \in V$ is not at a depth d_i for any $i \in \mathbb{N}$, then $\delta'(v) := \delta(v)$. However, for v at depth d_i for some $i \in \mathbb{N}$, we update the output so it produces a signal: $\delta'(v) := \delta(v)[(c\$, v')/(c, v')]$.

This new game has the same value as the original.

[5]For readers familiar with the breakpoint construction due to Miyano and Hayashi [MH84], these depths $(d_i)_{i \in \mathbb{N}}$ correspond to the breakpoints used in the proof that alternating Büchi automata can be simulated by nondeterministic Büchi automata. It is not surprising that a similar idea is used here, since this proof that finite memory strategies suffice in B-$[1,2]$ games will be a key ingredient in the proof that alternating B-$[1,2]$ can be simulated by nondeterministic B-$[1,2]$ automata (see Section 4.3).

Lemma 4.21. *If $\mathcal{G}$ is a hB-$[1,2]$ game with a chronological and finite branching game graph, then*

$$\text{value}(\mathcal{G}) = \text{value}(\mathcal{G}_\tau) = \inf\{\text{value}(\mathcal{G}') : \mathcal{G}' \text{ is a } hB\text{-safety game based on } \mathcal{G}\}.$$

Proof. Let $\mathcal{G}'$ be a hB-safety game based on $\mathcal{G}$. Given a strategy σ' in $\mathcal{G}'$, $h_\$(\sigma')$ is a strategy in $\mathcal{G}$ where $h_\$$ removes \$ from the plays in σ'. Moreover, the value of $h_\$(\sigma')$ in $\mathcal{G}$ is at most the value of σ' in $\mathcal{G}'$ since the safety-cost objective in $\mathcal{G}'$ is more restrictive than the hB-$[1,2]$ objective in $\mathcal{G}$ (not only requiring infinitely many priority 2, but requiring priority 2 at specific moments in the play based on \$). This shows the property that $\text{value}(\mathcal{G}) \leq \inf\{\text{value}(\mathcal{G}') : \mathcal{G}' \text{ is a } hB\text{-safety game based on } \mathcal{G}\}$ and, since $\mathcal{G}_\tau$ is a particular hB-safety game based on $\mathcal{G}$, $\text{value}(\mathcal{G}) \leq \text{value}(\mathcal{G}_\tau)$.

Recall that the strategy τ for $\mathcal{G}$ witnesses $\text{value}(\mathcal{G}) = \text{value}(\tau)$. Playing according to τ in $\mathcal{G}_\tau$ (adding \$ at appropriate depths) shows that $\text{value}(\mathcal{G}_\tau) \leq \text{value}(\tau) = \text{value}(\mathcal{G})$. Since $\mathcal{G}_\tau$ is a hB-safety game based on $\mathcal{G}$, this means that we also have $\inf\{\text{value}(\mathcal{G}') : \mathcal{G}' \text{ is a } hB\text{-safety game based on } \mathcal{G}\} \leq \text{value}(\mathcal{G})$. □

We can put these results together to get a finite memory strategy in $\mathcal{G}$.

Proposition 4.22. *If $\mathcal{G}$ is a hB-$[1,2]$ game with k hierarchical counters and a chronological and finite branching game graph, then α-2 finite memory strategies suffice for Eve for $\alpha(n) = (n+1)^k$.*

Proof. We proceed in a similar fashion as in the proof of Proposition 4.18, first constructing a finite memory strategy $\sigma_{\mathcal{G}_\tau}$ in $\mathcal{G}_\tau$ using Corollary 4.20. The desired strategy is obtained from $\sigma_{\mathcal{G}_\tau}$ by removing all occurrences of \$. As explained in the proof of Lemma 4.21, this is a strategy in $\mathcal{G}$ that has the same value as in $\mathcal{G}_\tau$. The memory structure is $\mathcal{M} := \langle\{q_0, q_1\}, q_0, \text{update}\rangle$ where $\text{update}(m, (v, c, v')) = \text{update}_{\mathcal{G}_\tau}(m, (v, cb, v'))$ where b is \$ if $\text{depth}(v) = d_i$ for some $i \in \mathbb{N}$ and is ϵ otherwise. □

No positional strategy in hB-[1,2] games

The strategy result for hB-$[1,2]$ games is optimal in the sense that there is, in general, no strategy using less memory.

Proposition 4.23. *There is a chronological hB-$[1,2]$ game $\mathcal{G}$ with one counter and finite branching such that for all α, there is no α-positional strategy σ in $\mathcal{G}$ such that $\text{value}(\mathcal{G}) \approx_\alpha \text{value}(\sigma)$.*

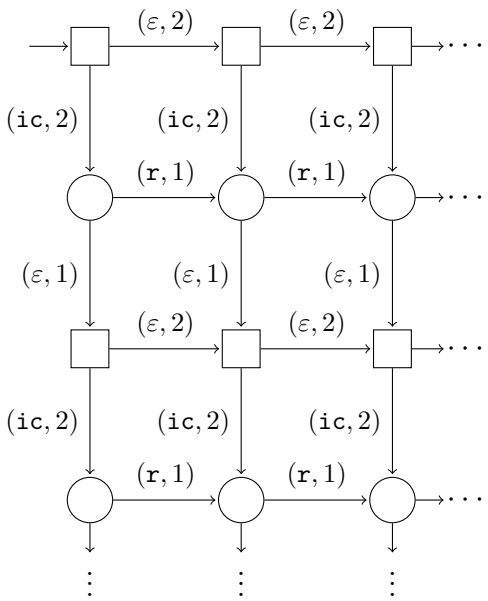

Figure 4.7. The hB-$[1, 2]$ game $\mathcal{G}$ with no α-positional strategy (see Proposition 4.23).

Proof. Consider the game shown in Figure 4.7. The game graph is an infinite grid, and is chronological with finite branching. This game has value 1, witnessed by the following optimal strategy for Eve: as soon as the play increments the counter and enters a position controlled by Eve, Eve moves right one position which resets the counter, and then moves down one position which passes the control back to Adam. Notice that neither player has an incentive to keep control of the play indefinitely by moving always right.

Assume there is some α-positional strategy σ witnessing value at most $\alpha(1) < \infty$ in $\mathcal{G}$. Consider the first row of positions controlled by Eve. If there are finitely many positions where σ moves down, then Adam need only pass control to Eve at a position which is to the right of this position, and the play will stabilize in priority 1 and yield value ∞. Otherwise, if there are infinitely many positions where σ moves down, regardless of Adam's starting position there must be some node he can reach which is controlled by Eve and for which σ then moves down immediately. Similar reasoning for the other rows means that Adam can either reach a position where Eve will move indefinitely right (and stabilize in priority 1) or he can forever pass control to Eve at positions where she moves immediately down (so the counter is incremented infinitely many times and never reset). In either case, Adam can witness a play of value ∞, contradicting the fact that σ is a positional strategy of finite value. □

4.2.4 Finite memory strategies in the dual of hB-[0,1] games

We now turn to Theorem 4.9 (d), which is more technical, but uses many of the same techniques as in the previous sections.

$\overline{\text{hB}}$-safety games

Recall that in a $\overline{hB\text{-}[0,1]}$ game, Eve is seeking to maximize the hB-$[0,1]$ valuation. Under this valuation, a play is assigned value ∞ if the counter values are unbounded, or there are infinitely many priority 1 (since this means the play is rejecting for the parity condition). As a result, it is helpful to know where these priority 1 transitions occur (just as it was helpful to know where priority 2 transitions occurred in the previous section on hB-$[1,2]$ games). In analogy to the previous section, we define a variant of a $\overline{hB\text{-}[0,1]}$ game which we call a *$\overline{hB}$-safety game*. A $\overline{hB}$-safety game has additional signals \$ added in the output related to the occurrence of priority 1 transitions.

Formally, a $\overline{hB}$-safety game has objective

$$\text{Safety-Cost}_{\overline{hB}}^{\Gamma,[0,1]} := \langle \mathbb{C}', \text{safety-cost}_{\overline{hB}}^{\Gamma,[0,1]}, \max \rangle$$

with $\mathbb{C} := H_B^{[1,k]} \times [0,1]$, $\mathbb{C}' := \mathbb{C} \cdot \{\epsilon, \$\}$, and $\text{safety-cost}_{\overline{hB}}^{\Gamma,[0,1]}(u)$ for $u \in (\mathbb{C}')^\omega$ defined as follows.

- If there are finitely many \$ in u, then $\text{safety-cost}_{\overline{hB}}^{\Gamma,[0,1]}(u) := 0$.
- If there are infinitely many \$ in u, then $u = u_0\$u_1\$\cdots$ such that each $u_i \in \mathbb{C}^*$ contains no \$. If there is some u_i in u such that there is no priority 1 in u_i (but there is priority 1 in each $u_{i'}$ for $i' < i$), then $\text{safety-cost}_{\overline{hB}}^{\Gamma,[0,1]}(u) := \text{value}_B(h(u_0u_1\cdots u_i))$, where h removes the priorities, keeping only counter actions. In that case, the value is the normal B-value on the prefix $u_0u_1\cdots u_i$. Otherwise, if there is priority 1 in every u_i, then $\text{safety-cost}_{\overline{hB}}^{\Gamma,[0,1]}(u) := \infty$.

The idea is that it is more difficult to obtain a high value in a $\overline{hB}$-safety game compared to a $\overline{hB\text{-}[0,1]}$ game because of the additional requirements enforced by \$ (e.g. a play with infinitely many priority 1 has value ∞ in a $\overline{hB\text{-}[0,1]}$ game, but a play with infinitely many priority 1 in a $\overline{hB}$-safety game might not be assigned value ∞ due to the requirements placed by \$).

Note that there is a deterministic $\overline{hB\text{-}[0,1]}$ automaton $\mathcal{D}$ over the alphabet $\mathbb{C}'$ that recognizes $\text{safety-cost}_{\overline{hB}}^{\Gamma,[0,1]}$ (it uses the state to remember whether priority 1 has been seen between signals \$, and whether the output is still being analysed). We now

consider the composition $\mathcal{D} \circ \mathcal{G}$, which is a $\overline{hB\text{-}[0,1]}$ game since $\mathcal{D}$ translates the game $\mathcal{G}_\tau$ into an $\overline{hB\text{-}[0,1]}$ objective.

Lemma 4.24. *Let $\mathcal{G}$ be a chronological $\overline{hB}$-safety game with finite branching. Then α-positional strategies suffice for Eve in $\mathcal{G}' := \mathcal{D} \circ \mathcal{G}$ for $\alpha(n) = (n+1)^k$ where k is the number of hierarchical counters.*

Proof. Fix a strategy τ' witnessing at least cost $(n+1)^k$ in $\mathcal{G}' := \mathcal{D} \circ \mathcal{G}$ and let T' be the corresponding strategy tree. Let $h : S \to V$ be the homomorphism between the set of positions S in T' and the set of positions V in $\mathcal{G}'$.

We define a signature for nodes $s \in S$ as follows: let $\mathrm{sig}(s) := \mathrm{pval}(\pi_s)$ where π_s is the path from the root to s in T' and pval is defined inductively as $\mathrm{pval}(\epsilon) := \langle 0, \ldots, 0 \rangle$ (a vector of k components initialized to 0), and $\mathrm{pval}(\pi \cdot (s', (c', p'), s'')) := \mathrm{pval}(\pi) \oplus c'$ for

$$\langle n_k, \ldots, n_1 \rangle \oplus c = \begin{cases} \langle n_k, \ldots, n_{j+1}, 0, \ldots, 0 \rangle & \text{if } c \text{ is } \texttt{r} \text{ for counter } j \\ \omega & \text{if } c \text{ is } \texttt{ic} \text{ for counter } j \text{ and } n_{j'} = n \\ & \quad \text{for all } j' \geq j \\ \langle n_k, \ldots, n_{j'} + 1, 0, \ldots, 0 \rangle & \text{if } c \text{ is } \texttt{ic} \text{ for counter } j \text{ and } j' \geq j \\ & \quad \text{is the least such that } n_{j'} < n \end{cases}$$

and $\omega \oplus c = \omega$.

The idea is that Eve is choosing her move based on the worst history leading to that position, and pval provides a summary of the value of the play (the idea for this valuation is taken from [CL08a]).

Let $\sigma' : V \to S$ where $\sigma'(v)$ selects $s \in h^{-1}(v)$ with the lexicographically-least signature. We extend sig to $v \in V$ by setting $\mathrm{sig}(v) := \mathrm{sig}(\sigma'(v))$. It is clear that σ' is a positional strategy. It remains to show that minimizing this signature results in a positional strategy that can guarantee value at least n.

Suppose for contradiction that $\mathrm{value}(\sigma') < n < \infty$, so there is some play $\pi \in \sigma'$ with $\mathrm{value}(\pi) = m < n$. Then π must not visit priority 1 infinitely often (otherwise the value of the play would be immediately ∞). Thus, there is some least i such that no transitions of priority 1 occur between d_i and d_{i+1}. Let v be the game position at depth d_{i+1} in π. For all $s \in h^{-1}(v)$, there must be no priority 1 between d_i and d_{i+1} (since this is true at v and the state of $\mathcal{D}$ is recorded in the game position). Hence, for all $s \in h^{-1}(v)$, the path from the root of T' to s must have value at least $(n+1)^k$. Moreover, this means that on this path there is a sequence of at least $(n+1)^k$ increment checks for some counter j (with no higher counter operations) which by a

straightforward counting argument means that pval reaches ω at some point on the path. Hence $\text{sig}(s) = \omega$, for all $s \in h^{-1}(v)$, which implies that $\text{sig}(v) = \omega$.

We prove by induction on the depth of v' in π that $\text{sig}(v') \leq \langle n_k, n_{k-1}, \ldots, n_1 \rangle$ where n_j is the current value of counter j in π on the partial play ending in v'. Since the maximum counter values are $m < n$ at all points along the play leading to v, this contradicts the fact that $\text{sig}(v) = \omega$.

The base case for depth 0 is trivial since the initial game position v'_0 is $\langle 0, 0, \ldots, 0 \rangle$ and the counters are initialized to value 0.

Assume the result holds up to v' with $\text{sig}(v') \leq \langle n_k, \ldots, n_1 \rangle$ at depth d. If the next move according to π is (v', c', v''), then there is some edge $(s', (c', p'), s'')$ in T' with $\sigma'(v') = s'$, $\text{sig}(s') = \text{sig}(v')$, and $s'' \in h^{-1}(v'')$.

- If c' resets counter j (or performs ε, which can be thought of as a reset of counter $j = 0$), then $\text{sig}(s'') \leq \langle n_k, n_{k-1}, \ldots, n_{j+1}, 0, \ldots, 0 \rangle$. Since counters $j' \leq j$ were reset and no other counters were touched, $\text{sig}(v'') \leq \text{sig}(s'')$ satisfies the desired condition.

- If c' increments and checks counter j, then the signature at s'' satisfies $\text{sig}(s'') \leq \langle n_k, n_{k-1}, \ldots n_{j+1}, n_j + 1, 0, \ldots, 0 \rangle$ since it must be the case that counter j has value less than m (otherwise it would contradict $\text{value}(\pi) = m < n$). But an increment for counter j resets all lower counters, so $\text{sig}(v'') \leq \text{sig}(s'')$ still satisfies the condition that the signature components are bounded by the counter values on π up to v''.

But this implies that $\text{sig}(v') \leq \langle m, \ldots, m \rangle$ at all depths up to and including d_{i+1}, which is a contradiction. □

By applying Lemma 4.10, we get the following result.

Corollary 4.25. *For all chronological finite branching $\overline{hB}$-safety games $\mathcal{G}$, α-2 finite memory strategies suffice for Eve in $\mathcal{G}$ for $\alpha(n) = (n+1)^k$ where k is the number of hierarchical counters.*

$\overline{\text{hB-[0,1]}}$ games

Let $\mathcal{G} = \langle V, v_0, O, \delta \rangle$ be a $\overline{hB\text{-}[0,1]}$ game with k hierarchical counters $\Gamma = [1, k]$ and a chronological and finite branching game graph.

Assume that $\text{value}(\mathcal{G}) \geq (n+1)^k$, and fix a strategy τ that witnesses at least cost $(n+1)^k$ for the $\overline{hB\text{-}[0,1]}$ game $\mathcal{G}$. Let T be the strategy tree corresponding to τ.

We define inductively a strictly increasing sequence of depths $(d_i)_{i\in\mathbb{N}}$ where $d_0 := 0$ and d_{i+1} is the least d such that all paths in T satisfy one of the following conditions:

- the counters witness value $(n+1)^k$ before depth d, or
- at least one transition is labelled with priority 1 between depths d_i and d.

This is well-defined: assume by contradiction that there is some position $s \in T$ from which there is no bound on the depth at which either of the conditions is satisfied. Because of the finite branching inherent in these games, König's lemma would imply that there is an infinite path from s on which neither of the conditions are satisfied, i.e. a path in T through s with only finitely many priority 1 transitions (and no priority 1 transitions after s) and counter values less than n, so this path has value less n. But this means value$(\tau) < n$, a contradiction.

We say a $\overline{hB}$-safety game $\mathcal{G}'$ is *based on* a $\overline{hB}$-$[0,1]$ game $\mathcal{G}$ with an chronological game graph if the arenas are identical except for the fact that signals \$ have been added to some edges in the game graph. We transform $\mathcal{G}$ into a $\overline{hB}$-safety game $\mathcal{G}_\tau = \langle V, v_0, \text{Safety-Cost}^{\Gamma,[0,1]}_{\overline{hB}}, \delta' \rangle$. If v is not at a depth d_i for any $i \in \mathbb{N}$, then $\delta'(v) := \delta(v)$. Otherwise, for v at depth d_i for some $i \in \mathbb{N}$, we update the output so $\delta'(v) := \delta(v)[(c\$, v')/(c, v')]$.

This new game must have value at least $(n+1)^k$ since by adding \$ at appropriate depths, τ can be transformed into a strategy in $\mathcal{G}_\tau$ that witnesses value at least $(n+1)^k$. There is also a nice relationship between $\overline{hB}$-$[0,1]$ games and the $\overline{hB}$-safety games based on it.

Lemma 4.26. value$(\mathcal{G}) = \sup\left\{\text{value}(\mathcal{G}') : \mathcal{G}' \textit{ is a } \overline{hB}\textit{-safety game based on } \mathcal{G}\right\}$ *and* value$(\mathcal{G}_\tau) \geq$ value(τ).

Proof. Playing according to τ in $\mathcal{G}_\tau$ (and adding \$ at the depths $(d_i)_{i\in\mathbb{N}}$) witnesses the fact that $\sup\left\{\text{value}(\mathcal{G}') : \mathcal{G}' \text{ is a } \overline{hB}\text{-safety game based on } \mathcal{G}\right\} \geq$ value$(\mathcal{G})$ and value$(\mathcal{G}_\tau) \geq$ value(τ).

Now assume by contradiction that there is a strategy σ in a $\overline{hB}$-safety game $\mathcal{G}'$ based on $\mathcal{G}$ such that value$(\sigma) >$ value$(\mathcal{G})$. This means that on each play $\pi \in \sigma$, either there are infinitely many priority 1 in π (with at least one such transition between each signal \$) or the counters witness a value exceeding n. In either case, $\text{cost}^{\Gamma,[0,1]}_B(h_\$(\pi)) > n$ where $h_\$$ removes the signal output. But this means that value$(\mathcal{G}) > n$, contradicting the initial assumption. □

We can now conclude that the original $\overline{hB}$-$[0,1]$ game $\mathcal{G}$ admits finite memory strategies.

Proposition 4.27. *If $\mathcal{G}$ is a $\overline{hB\text{-}[0,1]}$ game with k hierarchical counters and a chronological finite branching game graph, then α-2 strategies suffice for Eve for $\alpha(n) = (n+1)^k$.*

Proof. By Corollary 4.25, for any strategy τ of value at least $(n+1)^k$ in $\mathcal{G}$, there is a finite memory strategy of size 2 of value at least n in $\mathcal{G}_\tau$. This same strategy (with \$ removed) can be played in $\mathcal{G}$ and witnesses value at least n. Therefore, if $\text{value}(\mathcal{G}) \geq (n+1)^k$, then restricting to finite memory strategies of size 2 in $\mathcal{G}$ still ensures a value of at least n, which is enough to conclude that finite memory strategies suffice in $\mathcal{G}$. □

4.3 Simulation using nondeterministic cost-Büchi automata

The alternating automata on trees introduced by Muller and Schupp [MS84] and described in the cost setting in Section 3.3 are a powerful and useful automaton model. The structure of the transition function makes it straightforward to complement a classical alternating parity automaton and to convert between B- and S- versions of alternating cost-parity automata as shown in Theorem 3.15.

The increased power, however, comes at a price: the notion of a run is more complicated than in a nondeterministic automaton since an alternating automaton can launch several independent copies of itself, and these copies cannot communicate with one another. This is problematic in certain constructions (see, for example, the decidability result in Section 4.4). In these situations, it is useful to be able to *simulate* an alternating automaton with a nondeterministic automaton.

We can use the results about finite memory strategies from Theorem 4.9 in order to prove that alternating cost-Büchi automata can be simulated by nondeterministic cost-Büchi automata. This theorem generalizes the classical result that alternating Büchi automata are equivalent to nondeterministic Büchi automata ([MH84, MS95]). It incorporates ideas from the complementation of parity automata known from the literature (see, for instance, [Tho97]) and the simulation result for alternating cost automata over finite trees ([CL10]).

Theorem 4.28. *Let $\mathcal{A}$ be an alternating B-$[1,2]$ (respectively, S-$[1,2]$) automaton. Then there exists effectively a nondeterministic hB-$[1,2]$ (respectively, hS-$[1,2]$) automaton $\mathcal{A}_{\text{nd}}$ such that $[\![\mathcal{A}]\!] \approx [\![\mathcal{A}_{\text{nd}}]\!]$.*

The proof of this simulation theorem uses the idea of a *tree annotated with a strategy*. Given a cost automaton $\mathcal{A} = \langle Q, \mathbb{A}, q_0, \langle \mathbb{C}, f, goal \rangle, \delta \rangle$, input tree t, and finite memory strategy σ for Eve using memory structure $\mathcal{M} = \langle M, m_0, \text{update} \rangle$ in a game $\mathcal{A} \times t$, we consider the tree t_σ that is annotated with this strategy. This annotated tree uses an extended alphabet

$$\mathbb{A}' := \mathbb{A} \times \mathcal{P}((Q \times M) \times \mathbb{C} \times (Q \times M) \times [0,1]).$$

The tree t_σ is defined such that $t_\sigma(x) := (t(x), E_\sigma(x))$ where $E_\sigma(x)$ describes the set of moves possible from position x according to σ:

$$\left\{ ((q,m), c, (q',m'), d) : \begin{array}{l} ((q,x), c, (q', xd)) \text{ is a possible move from } (q,x) \text{ via } \sigma \\ \text{and update}(m, ((q,x), c, (q', xd))) = m' \end{array} \right\}.$$

Let $\tau = d_1 d_2 \cdots \in [0,1]^\omega$ be a branch in t. We write $\sigma|_\tau$ for σ restricted to plays that stay on τ. Given t_σ and τ, let $w_\sigma^\tau := (a_0, d_1)(a_1, d_2) \cdots$ such that $a_0 = t_\sigma(\epsilon)$ and $a_j = t_\sigma(d_1 \cdots d_j)$ for all $j > 0$, so w_σ^τ is the word that describes the plays in $\sigma|_\tau$.

We sketch now a first attempt at simulating an alternating B-$[1,2]$ automaton with a nondeterministic hB-$[1,2]$ automaton. We can assume (using Theorem 3.15) that $\mathcal{A}$ is an alternating hB-$[1,2]$ automaton.

On input t, the nondeterministic version guesses an annotation of t over the extended alphabet $\mathbb{A}'$ with a finite memory strategy σ. The output from each branch τ of a run is set to be the word w_σ^τ, and the value of this output is the maximum value over all plays described by w_σ^τ that stay on τ. Because finite memory strategies suffice in hB-$[1,2]$ games by Theorem 4.9 and the value of a strategy is the maximum value over all plays in the strategy, this automaton computes an $\approx$-equivalent value.

This automaton uses a special objective that can correctly assign the value to output words w_σ^τ. Recall that we can use Theorem 3.12 to translate between objectives, as long as the valuation is recognized by a history deterministic cost automaton. There is a B-$[1,2]$ automaton that reads the output from a single play described in w_σ^τ that stays on τ, so by Theorem 3.3 there is an S-$[1,2]$ automaton that recognizes the same function. From this, we can construct an S-$[1,2]$ automaton that guesses a play described by w_σ^τ and then computes its value, so there is an S-$[1,2]$ automaton that computes the maximum value of the plays that stay on τ described in the word w_σ^τ. By Theorem 3.3, we can conclude that there is a history deterministic hB-parity automaton (but not necessarily an hB-Büchi automaton) recognizing this valuation, which is enough to conclude that there is a nondeterministic hB-parity automaton simulating $\mathcal{A}$.

We seek a more refined result, showing that alternating B-$[1,2]$ and alternating S-$[1,2]$ automata can be simulated by nondeterministic hB-$[1,2]$ and hS-$[1,2]$ automata, respectively. To obtain this more refined result, the nondeterministic version will guess a safety game based on $\mathcal{A} \times t$ (as described in Section 4.2.3), guess a strategy in this safety game, check that it is a valid strategy, and then compute its value. The structure of the safety game makes it possible to show that there is a history-deterministic hB-$[1,2]$ or hS-$[1,2]$ automaton computing the value of words w_σ^τ.

4.3.1 Alternating B-[1,2] to nondeterministic hB-[1,2]

For the first part of Theorem 4.28, we aim to simulate an alternating B-$[1,2]$ automaton with a nondeterministic hB-$[1,2]$ automaton. By Theorem 3.15, we can assume that we are starting with a hB-$[1,2]$ automaton $\mathcal{A} = \langle Q, \mathbb{A}, q_0, \mathrm{Cost}_{hB}^{\Gamma,[1,2]}, \delta \rangle$ with k hierarchical counters. Fix some input tree t.

We first design a cost game $\mathcal{G}_{\mathrm{nd}}^t$ using $\mathcal{A}$ and t, which will serve as an intermediate object on the way to a nondeterministic hB-$[1,2]$ automaton. This game is played on the input tree t. The idea is that Eve guesses an hB-safety game based on $\mathcal{A} \times t$ as well as a finite memory strategy σ within this hB-safety game. Adam guesses a path τ through the tree and the output is a word w_σ^τ that describes the set of plays in σ that stay on τ. Formally, we define $\mathcal{G}_{\mathrm{nd}}^t := \langle V, v_0, O, \delta' \rangle$ as follows.

- The set of positions is $V := \mathcal{T}$ with initial position $v_0 := \epsilon$.
- At position $v \in V$, Eve chooses some separator $c \in \{\epsilon, \$\}$ and for each $q \in Q$ and $m \in M$, Eve selects a single disjunct $(d_1, c_1, q_1) \wedge \cdots \wedge (d_j, c_j, q_j)$ of $\delta(q, t(v))$. This induces a label $a' := (t(v), E(v)) \in \mathbb{A}'$ in the extended alphabet such that

  $$E(v) = \{((q,m), c_i \cdot c, (q_i, m_i), d_i) : i \in [1,j], m \in [0,1]\}$$

  where m_i is updated to reflect the new memory state (so m_i is 0 if $c = \$$, 1 if $c \neq \$$ but the priority output in c_i is 2, and m otherwise).
- Adam selects a direction $d \in [0,1]$.
- The play moves to position $v \cdot d$ and the output is (a', d). This means that a play induces a word of the form w_σ^τ and the value is assigned according to $O := \langle \mathbb{A}' \times [0,1], \text{max-play}, \min \rangle$ where

  $$\text{max-play}(w_\sigma^\tau) = \sup \left\{ \text{safety-cost}_{hB}^{\Gamma,[1,2]}(\pi) : \pi \in \sigma|_\tau \right\}.$$

This game is designed such that value($\mathcal{G}^t_{\text{nd}}$) is equivalent to value($\mathcal{A} \times t$).

Lemma 4.29. value($\mathcal{G}^t_{\text{nd}}$) $\approx_\alpha$ value($\mathcal{A} \times t$) *where* $\alpha_k(n) = (n+1)^k$.

Proof. Assume value($\mathcal{G}^t_{\text{nd}}$) is bounded by n. A strategy σ in $\mathcal{G}^t_{\text{nd}}$ induces an hB-safety game based on $\mathcal{A} \times t$ and a finite memory strategy σ in this hB-safety game such that safety-cost$^{\Gamma,[1,2]}_{hB}(\pi) \leq n$ for all $\pi \in \sigma$. By Lemma 4.21, value($\mathcal{A} \times t$) $\leq$ value(σ) $= n$.

Now assume that value($\mathcal{A} \times t$) is bounded by n. By Lemma 4.21, there is some hB-safety game based on $\mathcal{A} \times t$ with the same value. Moreover, by Corollary 4.20, there is a finite memory strategy σ in this hB-safety game guaranteeing value at most $(n+1)^k$. Since the *goal* in an hB-safety game is min, this means that

$$\text{value}(\sigma) = \sup\left\{\text{safety-cost}^{\Gamma,[1,2]}_{hB}(\pi) : \pi \in \sigma\right\} \leq (n+1)^k.$$

Hence, if Eve plays this strategy in $\mathcal{G}^t_{\text{nd}}$, then any output word is of the form w^τ_σ and has value at most $(n+1)^k$ via max-play, witnessing value($\mathcal{G}^t_{\text{nd}}$) $\leq \alpha_k(n)$. □

Unfortunately, this game does not use the desired hB-$[1,2]$ objective. We can show, however, that the valuation max-play in the objective O for $\mathcal{G}^t_{\text{nd}}$ is recognizable by a history deterministic B-$[1,2]$ automaton.

Lemma 4.30. *There is a history-deterministic* hB-$[1,2]$ *automaton* $\mathcal{D}_{\max}$ *such that*

$$[\![\mathcal{D}_{\max}]\!]^\vartheta(w^\tau_\sigma) \approx_\alpha \text{max-play}(w^\tau_\sigma) = \sup\left\{\text{safety-cost}^{\Gamma,[1,2]}_{hB}(\pi) : \pi \in \sigma|_\tau\right\}.$$

Proof. Let safety-cost$^\Gamma_{hB}$ map a finite word $u \in (\mathbb{C}')^*$ to ∞ if it does not end in \$ or if $u = u_0\$u_1\$\cdots\$u_j\$$ for $u_i \in \mathbb{C}^*$ and there is some subword u_i of u that does not contain priority 2; otherwise safety-cost$^\Gamma_{hB}$ maps u to its usual B-value (ignoring priorities). This is similar to the safety-cost valuation defined earlier, but adapted to finite words.

Given $w := w^\tau_\sigma$, the function max-play can be rewritten as

$$\text{max-play}(w) = \inf\{n : \exists \text{ infinitely many prefixes } v \text{ of } w \text{ s.t. } g(v) \leq n\}$$

where g maps a finite prefix of w to the maximum safety-cost$^\Gamma_{hB}$ values over the partial plays described in this prefix. In particular, if I describes an infinite number of positions where \$ occurs in w, then

$$\text{max-play}(w) = \inf\left\{n : \forall i \in I. g(w(0)\cdots w(i)) \leq n\right\}.$$

By Lemma 3.4, it suffices to show that g is a regular cost function over finite words in order to conclude that max-play is recognizable by a history deterministic hB-$[1,2]$ automaton.

Given a partial play $\pi \in \sigma|_\tau$, there is a B-automaton recognizing safety-cost$^\Gamma_{hB}(\pi)$ (it outputs the actions as described in π, unless the word does not end in \$ or priority 2 is not visited at least once between each pair of signals, in which case it rejects and assigns value ∞). By Theorem 2.8, there is a history deterministic S-automaton recognizing this same function.

We can then construct a new S-automaton that when reading a prefix of w, nondeterministically selects a partial play π described in this word, checks that it is a valid partial play in the game, and then computes safety-cost$^\Gamma_{hB}(\pi)$. Because nondeterminism in an S-automaton resolves into taking a maximum, this automaton recognizes the maximum of safety-cost$^\Gamma_{hB}$ over the partial plays described in v, which is exactly g. □

Because max-play is recognizable by a history-deterministic hB-$[1,2]$ automaton, Lemma 3.12 implies that $\mathcal{D}_{\max} \circ \mathcal{G}^t_{\text{nd}}$ uses an hB-$[1,2]$ objective and satisfies value$(\mathcal{G}^t_{\text{nd}}) \approx_\alpha$ value$(\mathcal{D}_{\max} \circ \mathcal{G}^t_{\text{nd}})$. Notice that, similar to a nondeterministic automaton, in the game $\mathcal{D}_{\max} \circ \mathcal{G}^t_{\text{nd}}$, Adam only chooses the direction while Eve controls every other choice. Hence, it is straightforward to construct a nondeterministic hB-$[1,2]$ automaton $\mathcal{A}_{\text{nd}}$ such that $\mathcal{A}_{\text{nd}} \times t$ is isomorphic to $\mathcal{D}_{\max} \circ \mathcal{G}^t_{\text{nd}}$ for all t. Moreover, since the correction function α from Lemma 4.30 did not depend on the particular input t, we have $[\![\mathcal{A}]\!] \approx_\beta [\![\mathcal{A}_{\text{nd}}]\!]$ where $\beta(n) = \alpha((n+1)^k)$.

4.3.2 Alternating S-[1,2] to nondeterministic hS-[1,2]

We use a similar method as in the previous case to construct a nondeterministic hS-$[1,2]$ automaton when starting with an alternating S-$[1,2]$ automaton.

Here we start by converting the alternating S-$[1,2]$ automaton to an alternating hB-$[0,1]$ automaton $\mathcal{A}$ using Theorem 3.15, and consider trees annotated by a strategy σ in a $\overline{hB}$-safety game based on the $\overline{hB\text{-}[0,1]}$ game $\overline{\mathcal{A} \times t}$ (let $\mathbb{A}'$ be the extended alphabet). By converting to a game in this form, we can apply the strategy results from Theorem 4.9.

We define a game $\mathcal{G}^t_{\text{nd}}$ that is played on the input tree t. The idea is that Eve guesses an $\overline{hB}$-safety game based on the $\overline{hB\text{-}[0,1]}$ game $\overline{\mathcal{A} \times t}$ as well as a finite memory strategy σ within this $\overline{hB}$-safety game. Adam guesses a path τ through

the tree and the output is a word w_σ^τ that describes the set of plays in the $\overline{hB}$-safety game that stay on τ. This time, the value is assigned according to objective $O := \langle \mathbb{A}' \times [0,1], \text{min-play}, \max \rangle$ where

$$\text{min-play}(w_\sigma^\tau) = \inf \left\{ \text{safety-cost}_{hB}^{\Gamma,[1,2]}(\pi) : \pi \in \sigma|_\tau \right\}$$

because in this dual case, we need to be able to calculate the minimum over all plays described by this word (since the objective in a $\overline{hB\text{-}[0,1]}$ game is max).

This valuation can be recognized by a history deterministic hS-$[1,2]$ automaton.

Lemma 4.31. *There is a history deterministic hS-Büchi automaton $\mathcal{D}_{\min}$ such that $[\![\mathcal{D}_{\min}]\!]^\vartheta(w_\sigma^\tau) \approx_\alpha \text{min-play}(w_\sigma^\tau) = \inf\{\text{safety-cost}_{\overline{hB}}^{\Gamma,[1,2]}(\pi) : \pi \in \sigma|_\tau\}$.*

Proof. Let $\text{safety-cost}_{\overline{hB}}^{\Gamma}$ denote the $\overline{hB}$-safety valuation on finite words u that assigns value 0 if u does not end in \$, $\text{value}_B(u_0u_1\cdots u_i)$ if $u = u_0\$u_1\$\cdots\$u_j\$$ where each $u_{i'}$ for $i' \leq j$ contains no \$ and u_i for $i \leq j$ is the first subword that does not contain priority 1, and value ∞ otherwise.

Let $w := w_\sigma^\tau$. Then the function min-play can be rewritten as $\text{min-play}(w) = \sup\{n : \exists \text{ infinitely many prefixes } v \text{ of } w \text{ such that } g'(v) \geq n\}$ where g' maps a finite prefix of w to the minimum safety-cost over the partial plays described in this prefix (in particular, if I describes an infinite number of positions where \$ occurs in w, then $\text{min-play}(w) = \sup\{n : \forall i \in I. g'(w(0)\cdots w(i)) \geq n\}$). By Lemma 3.4, it suffices to show that g' is a regular cost function.

Given a partial play π, it is straightforward to construct a B-automaton $\mathcal{C}'$ that recognizes $\text{safety-cost}_{\overline{hB}}^{\Gamma}(\pi)$ (just copy the output actions from the input word that describes a play, and use the state to track whether priority 1 has been visited between signals in order to determine acceptance). The desired B-automaton recognizing g' nondeterministically selects a partial play in a prefix of w, checks that it is a valid partial play in the $\overline{hB}$-safety game, and simulates $\mathcal{C}'$ on it; this computes the minimum of $\text{safety-cost}_{\overline{hB}}^{\Gamma}$ over the partial plays described in a prefix of w. □

As before, the desired hS-$[1,2]$ automaton $\mathcal{A}_{\text{nd}}$ is constructed such that it is isomorphic to $\mathcal{D}_{\min} \circ \mathcal{G}_{\text{nd}}^t$, and satisfies $[\![\mathcal{A}_{\text{nd}}]\!] \approx [\![\mathcal{A}]\!]$. We omit the formal proof of correctness since it is similar to the previous case.

4.4 Decidability of the domination preorder

The main decidability question for regular cost functions f_1 and f_2 is the domination preorder ($f_1 \preccurlyeq f_2$?) and boundedness relation ($f_1 \approx f_2$?). In the classical setting

(for regular languages L_1 and L_2 instead of regular cost functions f_1 and f_2), this corresponds to deciding language inclusion ($L_2 \subseteq L_1$) and language equality ($L_1 = L_2$) (see Remark 2.2).

We now summarize the usual approach to deciding the language inclusion problem, which will parallel the method used to prove the decidability of $\preccurlyeq$. Let $\mathcal{A}'_1 = \langle Q'_1, \mathbb{A}, q'^1_0, O'_1, \Delta'_1 \rangle$ and $\mathcal{A}_2 = \langle Q_2, \mathbb{A}, q^2_0, O_2, \Delta_2 \rangle$ be nondeterministic parity automata (without counters) that recognize L_1 and L_2, respectively.

Note that regular languages L_1 and L_2 satisfy $L_2 \subseteq L_1$ iff

$$\forall t.\ \forall \text{run } \rho_2 \text{ of } \mathcal{A}_2 \text{ on } t.\ \exists \text{run } \rho'_1 \text{ of } \mathcal{A}'_1 \text{ on } t.\ \rho_2 \text{ satisfies } O_2 \Rightarrow \rho'_1 \text{ satisfies } O'_1.$$

We first make some transformations of the problem based on quantifier considerations. We can immediately simplify the statement by considering $L_2 \not\subseteq L_1$ instead:

$$\exists t.\ \exists \text{run } \rho_2 \text{ of } \mathcal{A}_2 \text{ on } t.\ \forall \text{run } \rho'_1 \text{ of } \mathcal{A}'_1 \text{ on } t.\ \rho_2 \text{ satisfies } O_2 \wedge \rho'_1 \text{ does not satisfy } O'_1.$$

This is easier to work with since it starts with a block of existential quantifiers. The universal quantification over the runs of $\mathcal{A}'_1$ can also be eliminated by using the nondeterministic parity automaton $\mathcal{A}_1 = \langle Q_1, \mathbb{A}, q^1_0, O_1, \Delta_1 \rangle$ accepting the complement of L_1 instead of $\mathcal{A}'_1$. Thus, $L_2 \not\subseteq L_1$ iff

$$\exists t.\ \exists \text{run } \rho_2 \text{ of } \mathcal{A}_2 \text{ on } t.\ \exists \text{run } \rho_1 \text{ of } \mathcal{A}_1 \text{ on } t.\ \rho_2 \text{ satisfies } O_2 \wedge \rho_1 \text{ satisfies } O_1.$$

This means that in order to witness $L_2 \not\subseteq L_1$, we need to exhibit a single tree t and accepting runs of $\mathcal{A}_2$ and $\mathcal{A}_1$ on t (rather than some infinite set of trees and runs). Since we can effectively find $\mathcal{A}_1$ from $\mathcal{A}'_1$ (see, e.g. [Löd09]), this means we have reduced the decidability of the language inclusion problem $L(\mathcal{A}_2) \not\subseteq L(\mathcal{A}'_1)$ to the problem of deciding $L(\mathcal{A}_1) \cap L(\mathcal{A}_2) \neq \emptyset$.

We now capture this *intersection emptiness problem* $L(\mathcal{A}_1) \cap L(\mathcal{A}_2) \neq \emptyset$ in terms of an *intersection game* $\mathcal{G}_{\mathcal{A}_1 \cap \mathcal{A}_2} := \langle V, v_0, O, \delta \rangle$.

- The set of positions is $V := Q_1 \times Q_2$ and the initial position is $v_0 := (q^1_0, q^2_0)$.
- The objective O ensures that the play from $\mathcal{A}_1$ must satisfy the parity condition from O_1 and the play from $\mathcal{A}_2$ must satisfy the parity condition from O_2 (formally, $O := \langle P_1 \times P_2, u \mapsto \max(\mathrm{cost}^{P_1}_{\mathrm{parity}}(\mathrm{pr}_1(u)), \mathrm{cost}^{P_2}_{\mathrm{parity}}(\mathrm{pr}_2(u))), \min \rangle$ where pr_i projects $(p_1, p_2) \in P_1 \times P_2$ to p_i).

- The control function is defined by

$$\delta((q_1, q_2)) := \bigvee_{a \in \mathbb{A}} \bigvee_{\substack{(q_1, a, (p_1^0, r_1^0), (p_1^1, r_1^1)) \in \Delta_1 \\ (q_2, a, (p_2^0, r_2^0), (p_2^1, r_2^1)) \in \Delta_2}} \bigwedge_{i \in [0,1]} ((p_1^i, p_2^i), (r_1^i, r_2^i)).$$

Note that we are using the notation of a cost game even though there are no counters.

The idea is that at each turn, Eve chooses a label for the current position in the tree and transitions from Δ_1 and Δ_2 consistent with the current state and the label just chosen. Then Adam selects a direction, and the next game position corresponds to the destination states in the transitions selected by Eve in the direction chosen by Adam. Eve's goal is to show that there is a nonempty intersection (and therefore the parity condition is satisfied on any branch in the chosen runs) while Adam is trying to show otherwise by witnessing a branch in the runs selected that is not accepting for both automata. In order for a winning strategy for Eve in this game to correspond to nonempty intersection of $L(\mathcal{A}_2)$ and $L(\mathcal{A}_1)$, it is crucial that the automata are nondeterministic. This correspondence between a winning strategy for Eve in the game and a nonempty intersection is no longer true if the automata are alternating, since there would be no way to ensure that Eve selected the same labelled tree for each part of the alternating automaton.

By Theorem 4.2, solving parity games on a finite game graph like $Q_1 \times Q_2$ is decidable. Hence, the language inclusion problem is decidable for regular languages of infinite trees.

We now turn to the main result in this section, the decidability of $\preccurlyeq$ for certain cost functions over infinite trees. As before, we consider $f_1 \not\preccurlyeq f_2$ due to quantifier considerations. We have $f_1 \not\preccurlyeq f_2$ iff

$$\exists U \subseteq \mathcal{T}_{\mathbb{A}}.\ \mathcal{A}_2 \text{ is bounded on } U \wedge \mathcal{A}_1 \text{ is unbounded on } U.$$

This points to the desired form for $\mathcal{A}_1$ and $\mathcal{A}_2$. B-automata are good at expressing boundedness (because a single run of a B-automaton can witnesses a low value for the automaton) whereas S-automata are good at expressing unboundedness (because a single run of an S-automaton can witness a high value). Hence, we require that $\mathcal{A}_1$ (respectively, $\mathcal{A}_2$) is given by a nondeterministic S-parity (respectively, nondeterministic B-parity) automaton. The fact that these two different forms are used is not surprising when the classical proof is recalled: this is in analogy to the classical case where $\mathcal{A}_1$ (the complement of $\mathcal{A}_1'$) was used instead of $\mathcal{A}_1'$.

We aim to prove the following proposition.

Theorem 4.32. *The relation $f_1 \preccurlyeq f_2$ is decidable for cost functions f_1 and f_2 over infinite trees if f_1 is given by a nondeterministic S-parity automaton and f_2 is given by a nondeterministic B-parity automaton.*

We adapt the proof from [CL10, Theorem 13] for the decidability of $\preccurlyeq$ for regular cost functions over finite trees, which also incorporates ideas from the language inclusion problem described above. We proceed in three stages, reducing the decidability of $f_1 \not\preccurlyeq f_2$ to the existence of strategies in games $\mathcal{G}$, $\widetilde{\mathcal{G}}$ and $\widetilde{\mathcal{G}}'$.

Let $\mathcal{A}_1 = \langle Q_1, \mathbb{A}, q_0^1, O_1, \Delta_1 \rangle$ be a nondeterministic S-parity automaton (so $O_1 = \mathrm{Cost}_S^{\Gamma_1, P_1}$) and let $\mathcal{A}_2 = \langle Q_2, \mathbb{A}, q_0^2, O_2, \Delta_2 \rangle$ be a nondeterministic B-parity automaton (so $O_2 = \mathrm{Cost}_B^{\Gamma_2, P_2}$). We describe how to decide $[\![\mathcal{A}_1]\!] \not\preccurlyeq [\![\mathcal{A}_2]\!]$.

Stage 1: $\mathcal{G}$

Let $\mathcal{G}$ be the game $\langle Q_1 \times Q_2, (q_0^1, q_0^2), O_1, O_2, \delta \rangle$ where δ is defined as in the intersection game described earlier (see page 101). Notice that we have included two separate objectives (taken from $\mathcal{A}_1$ and $\mathcal{A}_2$), so this is different than the original definition of cost game. However, it is straightforward to adapt the definition of a cost game appropriately such that it now has two objectives, and a strategy σ (for Eve) yields two values based on the valuation from each objective.

Recall that at each turn in the game, Eve selects a label for the tree and transitions in Δ_1 and Δ_2, and then Adam chooses a direction. Eve is the player trying to show that $[\![\mathcal{A}_1]\!] \not\preccurlyeq [\![\mathcal{A}_2]\!]$ and Adam is trying to show that $[\![\mathcal{A}_1]\!] \preccurlyeq [\![\mathcal{A}_2]\!]$.

Stage 2: $\widetilde{\mathcal{G}}$

We now use composition and dualization (as described in Section 3.2) to transform the objectives in $\mathcal{G}$ to a form that is easier to work with (namely a form that is easier to translate into an ω-regular winning condition, see Stage 3).

Let $\widetilde{\mathcal{G}}$ be the game $\overline{(\mathcal{A}_{\mathrm{cost}_S^{\Gamma_1,P_1}}, \mathrm{id}) \circ \overline{\mathcal{G}}}$ where $\mathcal{A}_{\mathrm{cost}_S^{\Gamma_1,P_1}}$ converts the action from the run of $\mathcal{A}_1$ to B-parity actions (see Lemma 3.5), and id leaves the second component coming from the run of $\mathcal{A}_2$ unchanged.

First, notice the B-parity O_2 objective is unchanged: in $\overline{\mathcal{G}}$ it becomes the $\overline{B\text{-parity}}$ objective $\overline{O_2}$, it is unchanged in the composition, and then taking the dual again to get $\widetilde{\mathcal{G}}$ yields O_2. We write value_2 for the value using O_2.

The original O_1 objective is dualized to become an $\overline{S\text{-parity}}$ objective, then converted to a B-parity objective by composition with $\mathcal{A}_{\mathrm{cost}_S^{\Gamma_1,P_1}}$, and with the final dualization becomes a $\overline{B\text{-parity}}$ objective. We will call this new $\overline{B\text{-parity}}$ objective O_1' and

denote by P_1' the set of priorities used in O_1'. We write value_1 for the value obtained using O_1'.

In other words, at each turn in the game $\widetilde{\mathcal{G}}$, Eve selects a label for the tree and transitions in Δ_1 and Δ_2 as before. Adam chooses a direction and a transition in $\mathcal{A}_{\text{cost}_S^{\Gamma_1,P_1}}$ consistent with output from the transition from Δ_1 selected by Eve in the direction selected by Adam. This means that both components of the output are now B-parity actions, but Eve is trying to maximize the value of the first component and minimize the value of the second component.

By Proposition 3.11 and Lemma 3.12, the values of $\mathcal{G}$ and $\widetilde{\mathcal{G}}$ are equivalent. Moreover, we have the following characterization of the decidability of $\preccurlyeq$.

Lemma 4.33. *$[\![\mathcal{A}_1]\!] \not\preccurlyeq [\![\mathcal{A}_2]\!]$ if and only if there exists $n \in \mathbb{N}$ and a family $(\tau_j)_{j\in\mathbb{N}}$ of strategies for Eve in $\widetilde{\mathcal{G}}$ with $\text{value}_1(\tau_j) \geq j$ and $\text{value}_2(\tau_j) \leq n$ for all j.*

Proof. Assume $[\![\mathcal{A}_1]\!] \not\preccurlyeq [\![\mathcal{A}_2]\!]$. Then there is some set U and some $n \in \mathbb{N}$ such that $[\![\mathcal{A}_2]\!](U)$ is bounded by n but $[\![\mathcal{A}_1]\!](U)$ is unbounded on this set of input trees. Let $t_j \in U$ satisfy $[\![\mathcal{A}_1]\!](t_j) \geq j$, so there is some run ρ_j^1 of $\mathcal{A}_1$ on t_j witnessing value at least j and some run ρ_j^2 of $\mathcal{A}_2$ on t_j witnessing value at most n. Let σ_j be the strategy for Eve in $\mathcal{G}$ that selects labels according to t_j and runs according to ρ_j^1 and ρ_j^2. Note that Eve's role in $\mathcal{G}$ and $\widetilde{\mathcal{G}}$ is the same since Adam controls the operation of $\mathcal{A}_{\text{cost}_S^{\Gamma_1,P_1}}$ in $\widetilde{\mathcal{G}}$. This means we can let τ_j be the strategy in $\widetilde{\mathcal{G}}$ corresponding to σ_j (i.e. each play in σ_j becomes a set of plays in τ_j that correspond to the possible runs of $\mathcal{A}_{\text{cost}_S^{\Gamma_1,P_1}}$ on the original play from σ_j). Since the actions taken from ρ_j^2 are not changed at all when converting from $\mathcal{G}$ to $\widetilde{\mathcal{G}}$, we have $\text{value}_2(\tau_j) \leq n$. The actions from ρ_j^1 are modified by $\mathcal{A}_{\text{cost}_S^{\Gamma_1,P_1}}$, but since $[\![\mathcal{A}_{\text{cost}_S^{\Gamma_1,P_1}}]\!] = \text{cost}_S^{\Gamma_1,P_1}$ by Lemma 3.5, any run of $\mathcal{A}_{\text{cost}_S^{\Gamma_1,P_1}}$ that Adam can play in $\widetilde{\mathcal{G}}$ cannot force a value lower than j for τ_j. This means that $\text{value}_1(\tau_j) \geq j$ and $\text{value}_2(\tau_j) \leq n$.

Now assume there is a family $(\tau_j)_{j\in\mathbb{N}}$ of strategies for Eve in $\widetilde{\mathcal{G}}$ with $\text{value}_1(\tau_j) \geq j$ and $\text{value}_2(\tau_j) \leq n$ for all j. Because $\mathcal{A}_{\text{cost}_S^{\Gamma_1,P_1}}$ is id-history-deterministic, there is a family of translation strategies $(\vartheta_n)_{n\in\mathbb{N}}$ which Adam can use to direct $\mathcal{A}_{\text{cost}_S^{\Gamma_1,P_1}}$. Given τ_j, let τ_j' be the set of plays in τ_j where Adam plays like ϑ_j when controlling $\mathcal{A}_{\text{cost}_S^{\Gamma_1,P_1}}$. We have $\text{value}_1(\tau_j') \geq j$ by the definition of history determinism. Since Adam's remaining choices in $\widetilde{\mathcal{G}}$ only concern the direction, this strategy τ_j' induces a tree t_j and runs of $\mathcal{A}_1$ and $\mathcal{A}_2$ on t_j witnessing $[\![\mathcal{A}_2]\!](t_j) \leq n$ and $[\![\mathcal{A}_1]\!](t_j) \geq j$. This means that $\mathcal{A}_2$ is bounded on $\{t_j : j \in \mathbb{N}\}$ but $\mathcal{A}_1$ is unbounded, so $[\![\mathcal{A}_1]\!] \not\preccurlyeq [\![\mathcal{A}_2]\!]$. □

Stage 3: $\widetilde{\mathcal{G}}'$

Finally, we construct a Muller game $\widetilde{\mathcal{G}}'$ based on $\widetilde{\mathcal{G}}$. In order to do this, we define two ω-regular winning conditions $\mathcal{F}$ and $\mathcal{F}'$ and then describe how $\widetilde{\mathcal{G}}'$ is constructed from $\widetilde{\mathcal{G}}$ based on these conditions. The idea is that we capture some properties of the counters in terms of an ω-regular winning condition that can then be used to synthesize the strategies τ_j.

Let $\mathcal{F}$ be the winning condition defined by

> the parity condition from O_2 is satisfied, and every counter in Γ_2 is incremented finitely often or reset infinitely often.

Likewise, let $\mathcal{F}'$ be the winning condition

(a) the parity condition from O_1' is not satisfied and condition $\mathcal{F}$ is satisfied; or

(b) at least one counter in Γ_1 is incremented infinitely often and reset finitely often, and every counter in Γ_2 is incremented finitely often or reset infinitely often.

We briefly explain why $\mathcal{F}$ and $\mathcal{F}'$ can be expressed as Muller conditions. We refer the reader to the acceptance types in Table 3.1 as well as [Tho97, GTW02].

The condition "at least one counter in Γ_1 is incremented infinitely often and reset finitely often" can be translated into a Rabin condition (the structure of this statement fits exactly the structure of a Rabin condition). Likewise, the condition "every counter in Γ_2 is incremented finitely often or rest infinitely often" can be translated into a Streett condition (the negation of a Rabin condition). Moreover, Rabin, Streett, and parity conditions are special cases of Muller conditions, and the conjunction or disjunction of Muller conditions can be written as a Muller condition. Thus, $\mathcal{F}$ and $\mathcal{F}'$ can be represented as Muller conditions.

Note that none of these transformations between types of winning conditions required any change to the underlying game graph.

The new game $\widetilde{\mathcal{G}}'$ uses the winning condition $\mathcal{F}'$, but is played on a restriction of the game graph from $\widetilde{\mathcal{G}}$ to the winning region for Eve based on condition $\mathcal{F}$, i.e. the positions from which Eve has a strategy satisfying condition $\mathcal{F}$. Theorem 4.3 implies that the winning region for Eve can be computed and there is a finite memory strategy for Eve from each position v in this winning region which we denote by $\sigma_{\mathcal{F}}(v)$.

A winning strategy in this Muller game $\widetilde{\mathcal{G}}'$ is related to the existence of the desired family of strategies in the cost game $\widetilde{\mathcal{G}}$.

Lemma 4.34. *There exists $n \in \mathbb{N}$ and a family $(\tau_j)_{j\in\mathbb{N}}$ of strategies for Eve in $\widetilde{\mathcal{G}}$ with* $\text{value}_1(\tau_j) \geq j$ *and* $\text{value}_2(\tau_j) \leq n$ *for all j if and only if Eve has a winning strategy in the Muller game $\widetilde{\mathcal{G}}'$ with winning condition $\mathcal{F}'$.*

Proof. Assume that Eve has a winning strategy in the Muller game $\widetilde{\mathcal{G}}'$ with winning condition $\mathcal{F}'$. Let $\sigma_{\mathcal{F}'}$ be a finite memory winning strategy for Eve in $\widetilde{\mathcal{G}}'$, which exists by Theorem 4.3.

We first observe a property of any *looping segment* (a segment that begins and ends at the same game position and in the same memory state) in the finite memory strategies $\sigma_{\mathcal{F}'}$ and $\sigma_{\mathcal{F}}(v)$.

Property 4.35. Let π be a play in the finite memory strategies $\sigma_{\mathcal{F}'}$ or $\sigma_{\mathcal{F}}(v)$ for any position v in $\widetilde{\mathcal{G}}'$. Then every looping segment ℓ in π satisfies the property

if there is a counter from Γ_2 that is incremented in ℓ, then it is reset in ℓ.

Proof. Notice that any play that satisfies $\mathcal{F}$ or $\mathcal{F}'$ must satisfy the property that every counter in Γ_2 is incremented finitely often or reset infinitely often. If there were a looping segment in a play from $\sigma_{\mathcal{F}'}$ or some $\sigma_{\mathcal{F}}(v)$ that witnesses an increment for a counter from Γ_2 but no reset, then we could pump a play which is consistent with the strategy and has infinitely many increments for some counter in Γ_2 but only finitely many resets, which is no longer winning, and therefore contradicts the assumption. □

Property 4.35 implies there is some bound n' for the maximum number of increments without a reset for a counter from Γ_2 on any play in $\sigma_{\mathcal{F}'}$ or $\sigma_{\mathcal{F}}(v)$, where the bound depends on size of the arena in $\widetilde{\mathcal{G}}$ and the size of the finite memory strategy $\sigma_{\mathcal{F}'}$ and the size of the (finitely many) finite memory strategies $\sigma_{\mathcal{F}}(v)$, but does not depend on the particular play.

Let $n := 2n'$. Fix $j \in \mathbb{N}$. We describe the construction of a strategy τ_j in $\widetilde{\mathcal{G}}$ with $\text{value}_1(\tau_j) \geq j$ and $\text{value}_2(\tau_j) \leq n$. The strategy τ_j starts by playing like $\sigma_{\mathcal{F}'}$. If there is some counter from Γ_1 that reaches value j at some position v, then we switch to the strategy $\sigma_{\mathcal{F}}(v)$ (which exists because of the restriction on the game graph in $\widetilde{\mathcal{G}}'$).

Consider some play $\pi \in \tau_j$.

If π always plays like $\sigma_{\mathcal{F}'}$ (i.e. no counter from Γ_1 ever reaches value j), then $\pi \in \sigma_{\mathcal{F}'}$ and π' satisfies $\mathcal{F}'$. Since no counter from Γ_1 ever reaches value j, condition (b) is not satisfied, so it must be the case that (a) is satisfied. In particular, this means that the parity condition from O'_1 is not satisfied on π, so $\text{value}_1(\pi) = \infty > j$.

It also means that the parity condition from O_2 is satisfied. Combined with the fact that any counter from Γ_2 can achieve value at most n' on a play from $\sigma_{\mathcal{F}'}$ as observed above, $\text{value}_2(\pi) \leq n' \leq n$.

If there is some v where π switches to playing like $\sigma_{\mathcal{F}}(v)$, then $\text{value}_1(\pi) \geq j$ from some counter in Γ_1 regardless of whether the parity condition from O'_1 is satisfied or not. The strategy $\sigma_{\mathcal{F}}(v)$ ensures that the parity condition from O_2 is satisfied. As observed above, any counter from Γ_2 can achieve value at most n' on a play from $\sigma_{\mathcal{F}'}$ or $\sigma_{\mathcal{F}}(v)$. Since π consists of a partial play from $\sigma_{\mathcal{F}'}$ and a play from $\sigma_{\mathcal{F}}(v)$, this means that $\text{value}_2(\pi) \leq 2n' = n$.

Now assume for contradiction that there is an $n \in \mathbb{N}$ and a family $(\tau_j)_{j\in\mathbb{N}}$ of strategies for Eve in $\widetilde{\mathcal{G}}$ such that $\text{value}_1(\tau_j) \geq j$ and $\text{value}_2(\tau_j) \leq n$ for all j, but Eve does not have a winning strategy in $\widetilde{\mathcal{G}}'$ with winning condition $\mathcal{F}'$. Finite memory determinacy of Muller games (Theorem 4.3) implies that Adam must consequently have a finite memory winning strategy $\overline{\sigma}'$ in $\widetilde{\mathcal{G}}'$.

Fix $j \gg n \cdot m$ where m is the product of the size of memory for $\overline{\sigma}'$ and the size of the arena $\widetilde{\mathcal{G}}'$. Let π_j be the play induced by Eve's strategy τ_j and Adam's strategy $\overline{\sigma}'$. Because $\overline{\sigma}'$ is winning for Adam, the following conditions hold for π_j:

(a′) the parity condition from O'_1 is satisfied, or the parity condition from O_2 is not satisfied, or at least one counter in Γ_2 is incremented infinitely often and reset finitely often; and

(b′) every counter in Γ_1 is incremented finitely often or reset infinitely often, or at least one counter in Γ_2 is incremented infinitely often and reset finitely often.

Because $\text{value}_2(\tau_j) \leq n$, the parity condition from O_2 is satisfied and it is not possible for some counter in Γ_2 to be incremented infinitely often and reset finitely often. Since (a′) must be satisfied, this means the parity condition from O'_1 is satisfied. But $\text{value}_1(\tau_j) \geq j$, so there must be some segment β of π_j with at least j increments and no resets of some counter γ from Γ_1.

We have the following property for any looping segment (based on the game position and memory state from $\overline{\sigma}'$) on π_j.

Property 4.36. For all looping segments on π_j,

- every counter from Γ_1 is reset or not incremented, or
- there is a counter from Γ_2 that is incremented and not reset.

Proof. Assume there were a looping segment that does not satisfy this property. Then pumping this segment results in a play consistent with $\overline{\sigma}'$ in which there is a counter from Γ_1 that is incremented infinitely often and reset finitely often, and every counter in Γ_2 is incremented finitely often or reset infinitely often. But this pumped play does not satisfy (b′), contradicting the fact that $\overline{\sigma}'$ is winning for Adam. □

As described in [CL10, Lemma 19], we can derive a contradiction using Property 4.36 as follows. Since we are assuming j very large, there is a looping segment β' in β on which counter $\gamma \in \Gamma_1$ is incremented $j' = \frac{j}{m}$ times but not reset (recall m is the product of the size of the memory for $\overline{\sigma}'$ and the size of the arena for $\widetilde{\mathcal{G}}'$). By Property 4.36, this implies that there is some counter $\zeta \in \Gamma_2$ that is incremented and not reset. But ζ cannot have been incremented more than n times (by the property of τ_j), hence there is a subsegment β'' that has $j'' = \frac{j}{nm}$ increments of γ but no increments of ζ. Now we repeat this process, starting with β'', and find a subsegment that has at least $\frac{j}{(nm)^2}$ increments of γ but no increments of two counters from Γ_2. If we continue this process $|\Gamma_2|$ times we will have found a looping segment that has $\frac{j}{(nm)^{|\Gamma_2|}}$ increments of γ but no increment of *any* counter from Γ_2. This contradicts Property 4.36. □

Proof of Theorem 4.32

Putting these lemmas together, we have $f_1 \preccurlyeq f_2$ if and only if Adam has a winning strategy in the Muller game $\widetilde{\mathcal{G}}'$. Since the game graphs are finite, Theorem 4.3 implies that the game $\widetilde{\mathcal{G}}'$ can be constructed effectively from the original nondeterministic cost-parity automata for f_1 and f_2 and that it is decidable whether Adam has a winning strategy from the initial position in $\widetilde{\mathcal{G}}'$. Hence, the relation $f_1 \preccurlyeq f_2$ is decidable when f_1 is given by a nondeterministic S-parity automaton and f_2 is given by a nondeterministic B-parity automaton.

4.5 Discussion

This chapter forms the technical core of this thesis. We have proven that certain cost-parity games admit finite memory strategies and shown how this can be exploited to simulate alternating cost-Büchi automata with nondeterministic cost-Büchi automata. Finally, we have shown that the relation $f_1 \preccurlyeq f_2$ is decidable when the cost functions over infinite trees f_1 and f_2 are given in a certain form (namely, when f_1 is given as a nondeterministic S-parity automaton and f_2 as a nondeterministic B-parity automaton).

This last result points to the fact that, given some alternating cost-parity automaton, we want to be able to convert it to both a nondeterministic B-parity and a nondeterministic S-parity form. In the next two chapters, we will see classes of cost-parity automata for which we are able to do this.

What is missing is a stronger result showing that all alternating cost-parity automata can be simulated by both nondeterministic B-parity and nondeterministic S-parity automata. This comes down to proving that finite memory strategies suffice in other types of cost-parity games (besides those mentioned in Theorem 4.9).

Unfortunately, it appears that the slicing technique utilized in parts (c) and (d) of Theorem 4.9 has reached its limit. For instance, even in an hB-$[0,1]$ game (another simple type of cost-parity game with only two priorities), slicing no longer ensures the parity condition is satisfied (it can only guarantee that 0 occurs infinitely often, but cannot guarantee that the play stabilizes in priority 0). Indeed, it appears that the interaction between the parity condition and the requirements on the counters makes proving finite memory determinacy much more difficult in the cost setting, since there is no fixed rule for when to give the cost or parity condition precedence over the other. Thus, proving that finite memory strategies suffice in arbitrary cost-parity games remains an interesting and challenging open problem.

Chapter 5

Weak Cost Automata

A fundamental result in the theory of regular languages is the equivalence between monadic second-order logic (MSO) and finite state automata, which Büchi and Rabin exploited in order to provide a decision procedure for the logic over infinite words and infinite trees [Büc60, Rab69]. Rabin [Rab70] also studied *weak monadic second-order logic* (WMSO), a variant of MSO in which second-order quantification is interpreted only over finite sets. Muller, Saoudi, and Schupp [MSS86] later showed that this weak form of the logic is equivalent to an alternating automaton with a special restriction on the structure of the transition function.

We extend these classical results about WMSO and weak automata to the cost setting. In particular, we show the equivalence of a weak automaton with counters and cost weak monadic second-order logic (cost WMSO), and then apply the results from previous chapters in order to derive a decidability result for cost WMSO over infinite trees. This work was first published in [VB11].

5.1 Weak cost automata

Let $\mathcal{A} = \langle Q, \mathbb{A}, q_0, O, \delta \rangle$ be a cost-$[i, i+1]$ automaton. Following [Par87], an *alternating chain* in $\mathcal{A}$ is a sequence of states $q_0 \cdots q_m$ from Q such that there is some $p \in [i, i+1]$ with q_1 reachable from q_0 using a sequence of transitions of priority p, and for all $j \in [1, m-1]$, q_{j+1} reachable from q_j using a sequence of transitions of priority p (respectively, $\overline{p}$) if j is even (respectively, odd) (where $\overline{i} = i+1$ and $\overline{i+1} = i$). The *length of the alternating chain* $q_0 q_1 \cdots q_m$ is m.

We say $\mathcal{A}$ is a *weak cost automaton* if there is a bound m on the length of alternating chains in $\mathcal{A}$ or (equivalently) if there is no cycle in the transition function

using both priorities. We say a cost-$[i, i+1]$ game $\mathcal{G}$ is *weak* if $\mathcal{G} = \mathcal{A} \times t$ for some weak cost automaton $\mathcal{A}$.

This definition corresponds to the standard conditions for a weak automaton as introduced by Muller, Saoudi, and Schupp [MSS86] but adapted to the case when priorities label transitions rather than states.

Remark 5.1. We can always convert to more traditional automata where each state is labelled by a priority (although this may increase the number of states). When the priorities label states, there is a priority mapping $\Omega : Q \longrightarrow [i, i+1]$, and this weak condition is often stated in terms of a partition of the state set into $Q_1, \ldots, Q_m$ satisfying:

- for all $j \in [1, m]$ and for all $q, q' \in Q_j$, $\Omega(q) = \Omega(q')$;
- for all $q \in Q$ and $a \in \mathbb{A}$, if some (d, c, q') appears in $\delta(q, a)$ with $q \in Q_j$ and $q' \in Q_{j'}$, then $j \geq j'$.

In other words, the states in a given partition share the same priority, and there is a partial order on the partitions such that the transitions are non-increasing with respect to this ordering.

We now describe some weak cost automata over infinite trees.

Example 5.2. Let $\mathbb{A} = \{a, b, c\}$. Consider the function that, for trees with infinitely many a's, outputs the maximum number of b's along a single branch (and otherwise assigns value ∞). We describe informally a one-counter B-weak automaton $\mathcal{A} = \langle \{q_0, q_a, q_b, q_\top\}, \mathbb{A}, q_0, \mathrm{Cost}_B^{\{1\},[1,2]}, \delta \rangle$ recognizing this function.

The idea is that Adam can either count the number of b's on some branch (incrementing and checking the counter while in state q_b), or prove there are only finitely many a's in the tree. If there are infinitely many a's then there is some branch τ such that an a-labelled node is reachable from each position on τ (but this a-labelled node does not need to be on τ itself). Eve picks out such a branch (marking it with q_0). At any point on this branch, Adam can move to state q_a and force Eve to witness a reachable a-labelled node. If she can, then the play stabilizes in $q_\top$.

The transitions while in state q_a have priority 1; otherwise, the transitions have priority 2. The automaton can reach q_a only from q_0; once in q_a it can only stay in q_a or move to $q_\top$. Thus, there is no cycle in the transition function that visits both priorities, so $\mathcal{A}$ is weak. The maximum length of alternating chain is 2 (witnessed by $q_0 q_a q_\top$).

Example 5.3. Consider the B-$[1,2]$ automaton recognizing $f(t) = |t|_a$ described in Example 3.14. There is no cycle that visits both priorities, so this automaton is weak.

5.1.1 Simulation and decidability

Because of the requirement that there is no cycle with both priorities, any play in $\mathcal{A} \times t$ must stabilize in a single priority. This means that if a weak cost automaton uses priorities $[i, i+1]$ it can be transformed into an automaton using priorities $[i-1, i]$ simply by replacing priority $i+1$ with priority $i-1$. We will often refer to the odd priority in $[i, i+1]$ as the *rejecting priority* and the even priority as the *accepting priority*. We get the following duality result.

Lemma 5.4. *It is effectively equivalent for a cost function f over infinite trees to be recognizable by a weak cost automaton with the following objectives: B-$[i, i+1]$, S-$[i, i+1]$, hB-$[i, i+1]$, hS-$[i, i+1]$ for all $i \in \mathbb{N}$.*

Proof. We can apply Theorem 3.15 to switch between the objectives. Recall that the original automaton is composed with an automaton from Lemma 3.5 or Lemma 3.6. Notice that composition with these automata does not introduce any cycles that visit both priorities, so the weakness of the automaton is preserved during the transformation. Because of the weakness condition, the automaton can be adjusted to use any two priorities (as long as the parity of each priority is preserved in the transformation). □

As a result, we say a cost function f is a *weak cost function* if it is recognizable by a weak cost automaton with any of the objectives described above. We will normally think of these automata as using priorities $[1,2]$ or $[0,1]$, however. We will say that an automaton is *B-weak* or *S-weak* if we want to emphasize the objective being used.

This immediately implies simulation and decidability results using Theorem 4.28 and Theorem 4.32.

Corollary 5.5. *Let $\mathcal{A}$ be a weak cost automaton. Then there exists effectively a nondeterministic B-$[1,2]$ automaton $\mathcal{B}$ and a nondeterministic S-$[1,2]$ automaton $\mathcal{S}$ such that $[\![\mathcal{A}]\!] \approx [\![\mathcal{B}]\!] \approx [\![\mathcal{S}]\!]$.*

Corollary 5.6. *The relation $f_1 \preccurlyeq f_2$ is decidable for weak cost functions f_1 and f_2 over infinite trees.*

5.1.2 Closure properties

In addition to the simulation and decidability results, weak cost automata also have good closure properties. As mentioned in Chapter 2, the natural closure properties for cost-parity automata are different than classical parity automata since the automata define functions instead of languages. We prove these closure properties (and state their classical correspondences) now. Note that all of these closure properties are effective.

In place of closure under complementation, weak cost automata are closed under dualization (switching between B and S forms) as shown in Lemma 5.4.

Instead of closure under union and intersection, weak cost automata are closed under min and max.

Lemma 5.7. *Weak cost functions are closed under* min *and* max*.*

Proof. By Lemma 5.4, we can assume that we are starting with B-weak automata $\mathcal{A}$ and $\mathcal{B}$, and want to construct B-weak automata for $\min([\![\mathcal{A}]\!], [\![\mathcal{B}]\!])$ and $\max([\![\mathcal{A}]\!], [\![\mathcal{B}]\!])$.

For $\mathrm{op} \in \{\vee, \wedge\}$, let $\mathcal{A}_{\mathrm{op}} = \langle Q_{\mathcal{A}} \cup Q_{\mathcal{B}} \cup \{q_0\}, \mathbb{A}, q_0, \mathrm{Cost}_B^{\Gamma_1 \cup \Gamma_2, [1,2]}, \delta \rangle$ where

$$\delta(q_0, a) = \delta_{\mathcal{A}}(q_0^{\mathcal{A}}, a) \text{ op } \delta_{\mathcal{B}}(q_0^{\mathcal{B}}, a)$$

and $\delta(q, a)$ is $\delta_{\mathcal{A}}(q, a)$ if $q \in Q_{\mathcal{A}}$ and $\delta_{\mathcal{B}}(q, a)$ if $q \in Q_{\mathcal{B}}$.

It is straightforward to see that $\mathcal{A}_\vee$ recognizes $\min([\![\mathcal{A}]\!], [\![\mathcal{B}]\!])$ and $\mathcal{A}_\wedge$ recognizes $\max([\![\mathcal{A}]\!], [\![\mathcal{B}]\!])$. □

Corresponding to closure of regular languages under inverse morphism, is closure of weak cost functions under *composition with morphism*. Let $h : \mathbb{A}' \longrightarrow \mathbb{A}$ be a map between alphabets $\mathbb{A}'$ and $\mathbb{A}$. We write $h(t') = t$ for the natural extension to trees that relabels each $\mathbb{A}'$-labelled vertex of t' according to h. If f is a cost function over infinite $\mathbb{A}$-labelled trees, then the composition of f under the morphism h is the cost function $f \circ h$ over infinite $\mathbb{A}'$-labelled trees.

Lemma 5.8. *Weak cost functions are closed under composition with morphism.*

Proof. Again, assume we start with a B-weak automaton $\mathcal{A}$ recognizing f and some morphism h. Then the B-weak automaton $\mathcal{A}'$ recognizing $f \circ h$ simply simulates $\mathcal{A}$ on the tree that results from replacing each label a' with $h(a')$ (that is, $\delta'(q, a') := \delta(q, h(a'))$). □

More interesting is closure under *weak* inf-*projection* and *weak* sup-*projection*. These operations correspond to finite projection in the classical setting, and are related to the inf-projection and sup-projection described in Section 2.2.2.

Let $h : \mathbb{A}' \longrightarrow \mathbb{A}$ be a map between alphabets $\mathbb{A}'$ and $\mathbb{A}$ such that $\mathbb{A}' \supseteq \mathbb{A}$ and $h(a) = a$ for all $a \in \mathbb{A}$. If t' contains only finitely many vertices with labels from $\mathbb{A}' \setminus \mathbb{A}$, then we write $h_{\mathrm{fin}}(t') = t$. The *weak* op-*projection* of some cost function $g : \mathcal{T}_{\mathcal{A}'} \longrightarrow \mathbb{N}_\infty$ over $h : \mathbb{A}' \longrightarrow \mathbb{A}$ is the function $g_{\mathrm{op},h_{\mathrm{fin}}} : \mathcal{T}_{\mathcal{A}} \longrightarrow \mathbb{N}_\infty$ such that

$$g_{\mathrm{op},h_{\mathrm{fin}}}(t) := \mathrm{op}\,\{g(t') : h_{\mathrm{fin}}(t') = t\}$$

where op is inf or sup. On input t, the weak op-projection of g over h combines (using the operation op) all of the values of g on trees t' that could be finitely projected to t via h.

The proof is a generalization of [MSS86, Lemma 1]. The idea is that given a weak cost automaton for g and a tree t, we simulate it in *nondeterministic mode* on an initial finite subtree of t, and then switch to *alternating mode* and run the original weak cost automaton on the remainder of t. While in nondeterministic mode, nodes labelled $a \in \mathbb{A}$ can be relabelled with some $a' \in h^{-1}(a)$. It is essential that the automaton is in nondeterministic mode for this part, otherwise there would be no way to guarantee that the guesses about the relabelling coincide in the different copies of the alternating automaton $\mathcal{A}$. We use B-weak (respectively, S-weak) automata in order to take the infimum (respectively, supremum) of the values of $g(t')$ for t' satisfying $h_{\mathrm{fin}}(t') = t$.

We first state a lemma that will allow us to simulate an alternating automaton by running an equivalent nondeterministic automaton on some *initial finite subtree* of an infinite tree (the restriction of t to a *prefix* up to some frontier E) and then switch seamlessly back to the original alternating automaton. Given a infinite tree t, a frontier E, and a strategy σ in $\mathcal{A} \times t$, we write $t|_E$ for the finite tree up to and including the frontier E and $\sigma|_E$ for the restriction of σ to the positions in $t|_E$ (i.e. finite prefixes of the plays in σ up to positions in E). The value$(\sigma|_E)$ is defined as usual, but ignores the priorities since the plays are finite.

Lemma 5.9. *Let $\mathcal{A}$ be a hB-weak automaton with state set Q. Then we can effectively construct a nondeterministic hB-automaton $\mathcal{A}_{\mathrm{nd}}$ over finite trees with state set Q_{nd} such that $[\![\mathcal{A}]\!] \approx [\![\mathcal{A}_{\mathrm{nd}}]\!]$ over the domain of finite trees. Moreover, there is a correction function α_{nd} and mapping $\eta : Q_{\mathrm{nd}} \longrightarrow \mathscr{P}(Q)$ such that for all t and for all frontiers E,*

- for every finite memory strategy σ of size two in $\mathcal{A} \times t$, if value$(\sigma) \leq n$, then there is an accepting run R of $\mathcal{A}_{\mathrm{nd}}$ on $t|_E$ such that value$(R) \leq \alpha_{\mathrm{nd}}(n)$ and there is a play $\pi \in \sigma|_E$ ending in position (r, x) if and only if $r \in \eta(R(x))$;

- for every accepting run R of $\mathcal{A}_{\mathrm{nd}}$ on $t|_E$, if $\mathrm{value}(R) \leq n$, then there is a partial finite memory strategy $\sigma|_E$ of size two in $\mathcal{A} \times t$ such that $\mathrm{value}(\sigma|_E) \leq \alpha_{\mathrm{nd}}(n)$ and there is a play $\pi \in \sigma|_E$ ending in position (r, x) if and only if $r \in \eta(R(x))$.

Proof. The proof idea is much like the simulation theorems in Section 4.3: the nondeterministic version guesses a finite memory strategy, and then a history deterministic B-automaton (on finite words) is run on every branch in order to compute the supremum over the values of the plays on that branch described by the strategy. This is also similar to the simulation theorem for cost automata over finite trees [CL10, Theorem 12], but here we allow a finite memory strategy of size 2 (rather than a positional strategy) and make explicit the states of the copies of the alternating automaton in the state of the nondeterministic version in order to define the mapping η. □

Lemma 5.10. *Weak cost functions are closed under weak* inf*-projection and weak* sup*-projection.*

Proof. Fix some hB-weak automaton $\mathcal{A} = \langle Q_{\mathcal{A}}, \mathbb{A}', q^0_{\mathcal{A}}, \mathrm{Cost}^{\Gamma,P}_{hB}, \delta_{\mathcal{A}} \rangle$ with k hierarchical counters in Γ and some mapping $h : \mathbb{A}' \to \mathbb{A}$ where $\mathbb{A} \subseteq \mathbb{A}'$. Our goal is to construct a B-weak automaton $\mathcal{B}$ recognizing $[\![\mathcal{A}]\!]_{\mathrm{inf},h_{\mathrm{fin}}}$.

We first use Lemma 5.9 to construct $\mathcal{A}_{\mathrm{nd}} = \langle Q_{\mathrm{nd}}, \mathbb{A}', q^{\mathrm{nd}}_0, \Gamma_{\mathrm{nd}}, F_{\mathrm{nd}}, \Delta_{\mathrm{nd}} \rangle$ and a mapping $\eta : Q_{\mathrm{nd}} \to \mathcal{P}(Q_{\mathcal{A}})$ for $\mathcal{A}$. The new automaton $\mathcal{B}$ for the weak inf-projection uses states from $Q_{\mathrm{nd}} \cup Q_{\mathcal{A}} \cup \{q_\top\}$ and counters from $\Gamma_{\mathrm{nd}} \cup \Gamma_{\mathcal{A}}$ (we assume these are disjoint unions). The computation proceeds as follows, using ideas from [MSS86, Lemma 1].

Given an infinite tree t over the alphabet $\mathbb{A}$, $\mathcal{B}$ begins in nondeterministic mode. In this mode, it only outputs priority 1, but it can nondeterministically replace a label $a \in \mathbb{A}$ with an element of $h^{-1}(a) \subseteq \mathbb{A}'$ and simulate the nondeterministic hB-automaton $\mathcal{A}_{\mathrm{nd}}$ on this $\mathbb{A}'$-labelled part. If the simulated automaton $\mathcal{A}_{\mathrm{nd}}$ is in an accepting state, then $\mathcal{B}$ can switch to alternating mode and use the original hB-weak automaton $\mathcal{A}$ from that point onward.

Formally, this means that the initial state is $q^0_{\mathcal{B}} := q^0_{\mathrm{nd}}$ and for $q \in Q_{\mathrm{nd}}$, $\delta_{\mathcal{B}}(q, a)$ is a disjunction of statements of the form

$$\bigwedge_{d \in [0,1]} (d, (c_d, 1), q_d) \qquad \text{if } q \notin F_{\mathrm{nd}}, \text{ or}$$

$$\bigwedge_{d \in [0,1]} (d, (c_d, 1), q_d) \vee \bigwedge_{d \in [0,1]} \bigwedge_{r \in (\eta(q_d) \cup q_\top)} (d, (c_d, 2), r) \qquad \text{if } q \in F_{\mathrm{nd}}$$

for each transition $(q, a', (c_0, q_0), (c_1, q_1)) \in \Delta_{\mathrm{nd}}$ and each $a' \in h^{-1}(a)$. Note that if $\eta(q_d) = \emptyset$ (no copies of the automaton were sent in direction d), then the automaton

enters state $q_\top$ and $\delta_{\mathcal{B}}(q_\top, a) := \bigwedge_{d\in[0,1]}(d, (\varepsilon, 2), q_\top)$. Otherwise, for $q \in Q_{\mathcal{A}}$, we have $\delta_{\mathcal{B}}(q, a) := \delta_{\mathcal{A}}(q, a)$.

Assume there is some U such that $[\![\mathcal{B}]\!](U)$ is bounded by some $n \in \mathbb{N}$. Let $t \in U$. Then there is a strategy τ in $\mathcal{B} \times t$ such that value$(\tau) \leq n$. This implies that every play in τ satisfies the parity condition, so every play must have moved from nondeterministic mode to alternating mode. By König's lemma, there is a frontier E such that all plays have moved to alternating mode before reaching E. Hence, τ induces a tree t' such that $h_{\text{fin}}(t') = t$ (with t' identical to t at positions beyond E) and an accepting run R of $\mathcal{A}_{\text{nd}}$ on $t'|_E$ with value at most n. Moreover, there are partial plays $\pi_{(r,x)}$ starting from (r, x) in $\mathcal{A}\times t'$ for all $x \in E$ and $r \in \eta(R(x))$ such that value$(\pi_{(r,x)}) \leq n$. By Lemma 5.9, there is a strategy $\sigma|_E$ in $\mathcal{A} \times t'$ with value$(\sigma|_E) \leq \alpha_{\text{nd}}(n)$. This can be extended to a strategy σ in $\mathcal{A} \times t'$ by using the extensions $\pi_{(r,x)}$ described above. Because $\mathcal{B}$ used a different set of counters for computing the value of the plays up to E and after E, each play in σ can have value most $\alpha_{\text{nd}}(n) + n$, so $[\![\mathcal{A}]\!](t') \leq \alpha_{\text{nd}}(n) + n$. This implies that $[\![\mathcal{A}]\!]_{\inf,h_{\text{fin}}}$ is bounded on U.

Assume that $[\![\mathcal{A}]\!]_{\inf,h_{\text{fin}}}$ is bounded by $n \in \mathbb{N}$ on some U. Then for any $t \in U$, there is a tree t' such that $h_{\text{fin}}(t') = t$ and $[\![\mathcal{A}]\!](t') \leq n$. By Theorem 4.9 there is a finite memory strategy σ in $\mathcal{A}\times t'$ with value$(\sigma) \leq (n+1)^k$. Let E be a frontier such that t' and t are identical at positions outside E. Then by Lemma 5.9, there is an accepting run R of $\mathcal{A}_{\text{nd}}$ on $t'|_E$ such that value$(R) \leq \alpha_{\text{nd}}((n+1)^k)$. The frontier E, tree t', and the run R can be used to construct a strategy in $\mathcal{B}\times t|_E$. This can be extended into a full strategy in $\mathcal{B} \times t$ by using the suffixes of plays in σ starting from positions (r, x) for $x \in E$ and $r \in \eta(R(x))$. Overall, this means that $[\![\mathcal{B}]\!](t) \leq \alpha_{\text{nd}}((n + 1)^k)$, so $[\![\mathcal{B}]\!]$ is bounded on U as desired.

Closure under weak sup-projection is similar, but more technical. This time we construct a S-weak automaton, which is more suitable for the maximizing that is required in weak sup-projection. Here we want to allow Eve to guess a finite memory strategy in $\overline{\mathcal{A} \times t}$. This strategy on an initial finite subtree of t could have an artificially low value from plays that do not witness a high value inside this subtree. Hence, we need to allow the automaton to guess that some plays have a high value on the extensions, and ignore the value on the initial finite subtree. We do this by adding additional markings \$ to some plays on the frontier of this initial finite subtree; the idea is that plays with this mark have already witnessed a high value in this initial finite subtree.

Using a slight modification of the simulation proof for cost automata over finite trees [CL10, Theorem 12], we can show that there is an S-automaton $\mathcal{V}$ that reads a finite tree annotated with a partial finite memory strategy σ for Eve in some game $\overline{\mathcal{A} \times t}$ with the addition of the markings \$ added at the frontier such that $\mathcal{V}$ recognizes $\inf \{\text{value}(\pi) : \pi \in \sigma \text{ ends in } \$\}$ up to some correction function α_{nd}.

We now design an S-weak automaton $\mathcal{S}$ recognizing $[\![\mathcal{A}]\!]_{\sup,h_{\text{fin}}}$. The idea is that Eve guesses a frontier E of t, and an annotation of $t|_E$ with a finite memory strategy for Eve in $\overline{\mathcal{A} \times t'}$ where $h_{\text{fin}}(t') = t$ and there are no labels from $\mathbb{A}' \setminus \mathbb{A}$ outside of E. Eve also guesses which plays should be marked with \$ at the frontier E. The automaton $\mathcal{V}$ is controlled by Eve and run on $t|_E$ in order to compute the value (up to α_{nd}) of the plays on initial finite subtree that end in \$, and the priority output is 1 (rejecting priority to ensure that Eve eventually switches to alternating mode). For plays marked by \$ at the frontier, $\mathcal{S}$ sends a copy to $q_\top$ (an accepting sink state). Otherwise, $\mathcal{S}$ runs the dual of $\mathcal{A}$ (obtained from switching conjunctions and disjunctions in the transition function) starting from states that are not marked with \$ at E, and runs the history deterministic S-$[1, 2]$ automaton from Lemma 3.5 computing the exact value of these plays starting from the frontier E.

Assume $[\![\mathcal{S}]\!](t) > \alpha_{\text{nd}}(n)$ with strategy σ such that $\text{value}(\sigma) > \alpha_{\text{nd}}(n)$. This implies that there is a tree t' such that $h_{\text{fin}}(t') = t$. Moreover, a play in σ induces a (potentially partial) play in $\overline{\mathcal{A} \times t'}$ with value greater than n. Let $\pi \in \sigma$ be a play that stabilizes in $q_\top$. Then π restricted to the portion in $t|_E$ must already witness value greater than n. Any strategy for Eve for extensions of this play yields a set of plays in $\overline{\mathcal{A} \times t'}$ that also have high values (indeed, in such a game, any extension can only increase the value). If π does not stabilize in $q_\top$, then π is already a play in $\overline{\mathcal{A} \times t'}$ with value over n. Using these plays, we can construct a strategy $\overline{\sigma}$ in $\overline{\mathcal{A} \times t'}$ with $\text{value}(\overline{\sigma}) > n$, so $[\![\mathcal{A}]\!](t') > n$ and hence $[\![\mathcal{A}]\!]_{\sup,h_{\text{fin}}}(t) > n$.

For the other direction, assume there is a tree t' such that $h_{\text{fin}}(t') = t$ and $[\![\mathcal{A}]\!](t') > 2 \cdot \alpha_{\text{nd}}(n)$. By Theorem 4.9, there is a finite memory strategy $\overline{\sigma}$ for Eve in $\overline{\mathcal{A} \times t'}$ such that $\text{value}(\overline{\sigma}) > 2 \cdot \alpha_{\text{nd}}(n)$. This means that every play $\pi \in \overline{\sigma}$ either stabilizes in priority 1 or witnesses counter value at least $2 \cdot \alpha_{\text{nd}}(n)$. Let E be the frontier such that there are no labels from $\mathbb{A}' \setminus \mathbb{A}$ outside of E. A partial play up to E may not have reached a high value, since the high value is witnessed on the infinite extensions of the play. Likewise, there may be plays that witness a high counter value before E, but do not witness a high value afterward. This is where \$ are used. The strategy for Eve in $\mathcal{S} \times t$ should select the frontier E, marking those plays that have already witnessed value $\alpha_{\text{nd}}(n)$ by E with \$. Hence, on these plays the value from $\mathcal{S}$

will be n, assigned based only on the portion up to E. Any play not marked with \$ must have value at least $\alpha_{\mathrm{nd}}(n)$ after E, so these plays are assigned value at least n by $\mathcal{S}$. Hence, the corresponding plays in $\mathcal{S} \times t$ will have at least value n overall. □

5.2 Cost weak monadic second-order logic

Cost monadic second-order logic, or *cost MSO*, is a quantitative extension of MSO introduced by Colcombet [Col09b, Col09a] (see, e.g. [Tho97] for an introduction to classical monadic second-order logic). As usual, the logic can be defined over any relational structure, but we describe here the logic over $\mathbb{A}$-labelled binary trees.

In addition to *first-order variables* which range over nodes (and which we denote by $x, y, \ldots$) and *second-order monadic variables* which range over sets of nodes (denoted by $X, Y, \ldots$), cost MSO uses a single additional variable N called the *boundedness variable* which ranges over $\mathbb{N}$. The atomic formulas in cost MSO are the usual atomic formulas from MSO (namely, the membership relation $x \in X$ and relations $a(x, x_0, x_1)$ that assert $a \in \mathbb{A}$ is the label at position x with left successor x_0 and right successor x_1), as well as new predicates $|X| \leq N$ where X is any second-order variable and N is the boundedness variable. Arbitrary cost MSO formulas can be built in the usual way by applying boolean connectives or by quantifying (existentially or universally) over first- or second-order variables. We additionally require that predicates of the form $|X| \leq N$ appear *positively*, which means they are within the scope of an even number of negations.

If we fix a value n for N, then the meaning of $|X| \leq N$ is what one would expect: the predicate is satisfied if and only if the valuation of X has cardinality at most n. We write $t, n \models \varphi$ if t satisfies φ when all occurrences of N take value n. Because of the positivity requirement, if $t, n \models \varphi$ then $t, n' \models \varphi$ for all $n' \geq n$.

If this value for N is not specified, then a sentence φ in cost MSO defines a function $[\![\varphi]\!] : \mathcal{T}_{\mathbb{A}} \longrightarrow \mathbb{N}_\infty$ such that

$$[\![\varphi]\!](t) := \inf \{n : t, n \models \varphi\}.$$

We say that φ recognizes the cost function $[\![\varphi]\!]$. Recall that $\inf \emptyset = \infty$, so in case φ is a pure MSO sentence (with no instances of the predicates $|X| \leq N$), $[\![\varphi]\!](t)$ is 0 if t satisfies the sentence φ and ∞ otherwise.

Cost weak monadic second-order logic (written cost WMSO) has the same syntax as cost MSO but the semantics are changed to restrict the second-order quantification to finite sets, as usual. WMSO (and consequently cost WMSO) is still a very

expressive logic (e.g. the well known temporal logic CTL embeds into it). We pause to give an example of a function definable in cost WMSO.

Example 5.11. Let $\mathbb{A} = \{a, b, c\}$. We seek to define the function from Example 5.2 that, for trees with infinitely many a's, computes the maximum number of b's along a single branch (and otherwise assigns value ∞). A suitable cost WMSO sentence φ is

$$\forall X.\exists x.\Big(\neg(x \in X) \wedge a(x)\Big) \wedge \forall Z.\Big((\forall z.(z \in Z \longrightarrow b(z)) \wedge \mathit{chain}(Z)) \longrightarrow |Z| \leq N\Big).$$

where $\mathit{chain}(Z)$ is a WMSO formula asserting Z is totally ordered (and hence the nodes are on the same branch). We write $a(x)$ for $\exists x_0.\exists x_1.a(x, x_0, x_1)$.

The first conjunct is a typical WMSO formula: "infinitely many a's" is expressed as "for all finite sets of nodes, there is an a-labelled node outside". The least n that can be substituted for N to satisfy the second conjunct is exactly the bound on the number of b's along a single branch (∞ if there is no bound).

To simplify some proofs in this section, we switch to an equivalent variant of the logic in which only monadic second-order variables are allowed (called MSO_0 in [Tho97]). This means that the inclusion relation $X \subseteq Y$ is used instead of the membership relation, and each relation $a(x, x_0, x_1)$ over first-order variables is raised to the relation $a(X, X_0, X_1)$ over monadic variables that holds if X,X_0,X_1 are singleton sets, the letter at the position given by the singleton set X is a, and its children are the positions represented by the singleton sets X_0 and X_1. Pushing negations to the leaves, it means that a cost WMSO formula is generated by the grammar

$$R(X_1, \ldots, X_k) \mid \neg R(X_1, \ldots, X_k) \mid |X| \leq N \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists X.\varphi \mid \forall X.\varphi$$

where X,X_1,...,X_k are monadic second-order variables, N is the boundedness variable, and R is some relation of arity k (the inclusion relation $\subseteq$ of arity 2, or the relation $a(X, X_0, X_1)$ of arity 3).

If $\varphi(X_1, \ldots, X_k)$ is a formula with free variables $X_1, \ldots, X_k$ (excluding the boundedness variable), then a *valuation* ν for φ is a mapping from each free variable X_i to a set $V_i \subseteq \mathcal{T}$ of positions. If ν is a valuation, then $\nu[X \mapsto V]$ denotes the new valuation that maps X to V and all other variables $Y \neq X$ to $\nu(Y)$. We write $t, \nu, n \models \varphi$ if t satisfies φ when free variables are evaluated according to ν and when n takes value N, and $[\![\varphi(X_1, \ldots, X_k)]\!](t, \nu) := \inf\{n : t, \nu, n \models \varphi\}$. Examining this definition for the particular constructs in the logic we have the following straightforward results.

Lemma 5.12. *For all infinite trees t and valuations ν,*

$$[\![R(X_1,\dots,X_k)]\!](t,\nu) = \begin{cases} 0 & \text{if } t \models R(\nu(X_1),\dots,\nu(X_k)) \\ \infty & \text{otherwise} \end{cases} \tag{5.1}$$

$$[\![\neg R(X_1,\dots,X_k)]\!](t,\nu) = \begin{cases} \infty & \text{if } t \models R(\nu(X_1),\dots,\nu(X_k)) \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

$$[\![|X| \leq N]\!](t,\nu) = |\nu(X)| \tag{5.3}$$

$$[\![\varphi \vee \psi]\!](t,\nu) = \min([\![\varphi]\!](t,\nu),[\![\psi]\!](t,\nu)) \tag{5.4}$$

$$[\![\varphi \wedge \psi]\!](t,\nu) = \max([\![\varphi]\!](t,\nu),[\![\psi]\!](t,\nu)) \tag{5.5}$$

$$[\![\exists X.\varphi]\!](t,\nu) = \inf\{[\![\varphi]\!](t,\nu[X \longmapsto V]) : V \subseteq \mathcal{T} \text{ is finite}\} \tag{5.6}$$

$$[\![\forall X.\varphi]\!](t,\nu) = \sup\{[\![\varphi]\!](t,\nu[X \longmapsto V]) : V \subseteq \mathcal{T} \text{ is finite}\} \tag{5.7}$$

We now seek to prove that cost WMSO recognizes exactly the class of weak cost functions. We prove the two directions of this theorem in Sections 5.2.1 and 5.2.2.

Theorem 5.13. *It is effectively equivalent for a cost function to be recognized by a weak cost automaton and a cost WMSO sentence.*

5.2.1 Logic to automata

The strategy for the translation between logic and automata is standard, so we only summarize the proof here. We identify an $\mathbb{A}$-labelled infinite tree t and valuation ν with domain $X_1,\dots,X_k$ with the $\mathbb{A} \times [0,1]^k$ labelled tree t^ν such that $t^\nu(x) = (t(x), b_1(x),\dots,b_k(x))$ with $b_i(x) = 1$ if and only if $x \in \nu(X_i)$. We must show that for any formula $\varphi(X_1,\dots,X_k)$ with free variables $X_1,\dots,X_k$, there is a weak cost automaton $\mathcal{A}_{\varphi(X_1,\dots,X_k)}$ and a correction function α such that for all t and all valuations ν with domain $X_1,\dots,X_k$, $[\![\mathcal{A}_{\varphi(X_1,\dots,X_k)}]\!](t^\nu) \approx_\alpha [\![\varphi(X_1,\dots,X_k)]\!](t,\nu)$.

This comes down to showing that the functions corresponding to the atomic formulas are recognizable using a weak cost automaton, and proving that weak cost automata are closed under operations corresponding to the other logical constructs as listed in Lemma 5.12.

- It is straightforward to construct weak automata without counters recognizing the languages corresponding to standard predicates of WMSO over infinite trees ($a(X, X_0, X_1)$ and $X \subseteq Y$) and their negations. Hence, the functions in (5.1) and (5.2) are recognizable by weak cost automata.

- The new predicate $|X| \leq N$ corresponds to the function that computes the cardinality of the valuation of some set X of nodes (5.3). Up to a change of alphabet, which is possible due to closure under morphism from Lemma 5.8, this automaton is equivalent to the weak cost automaton given in Example 5.3.
- Disjunction (5.4) and conjunction (5.5) in the logic correspond to min and max, respectively. Weak cost automata are closed under these operations by Lemma 5.7.
- Existential and universal second-order quantification (5.6, 5.7) correspond to weak inf-projection and weak sup-projection, respectively. Lemma 5.4 shows that weak cost automata are closed under these operations.

5.2.2 Automata to logic

For the next part, we seek to write a cost WMSO sentence that describes the operation of the weak cost automaton, and consequently recognizes the same cost function. If we were using cost MSO, then we could write a formula that describes exactly the value of a run of the nondeterministic B-$[1,2]$ automaton equivalent to the weak cost automaton. Here we need to find another way to describe the value using information only about finite partial runs of the nondeterministic B-automaton from Lemma 5.9.

We fix a weak cost automaton $\mathcal{A} = \langle Q, \mathbb{A}, q_0, O, \delta \rangle$ with counters Γ. We assume that all transitions leading from a given state $q \in Q$ are labelled with the same priority so there is some mapping $\Omega : Q \to [1,2]$ (see Remark 5.1). We also assume that $\mathcal{A}$ has objective $O = \mathrm{Cost}_{hB}^{\Gamma,[1,2]}$ (using Lemma 5.4 if necessary).

For the purposes of an inductive argument, we actually write a formula with one free variable that represents the starting position in the tree. For notational simplicity, we write this as $\varphi(x)$, knowing that we could switch to the variant of the logic described above that only uses second-order variables. Given $\varphi(x)$ and some valuation $\nu : \{x\} \to \mathcal{T}$, we write $[\![\varphi(\nu(x))]\!](t)$ as shorthand for $[\![\varphi_q(x)]\!](t, \nu)$.

We first consider a *trivial acceptance condition* so $\Omega : Q \to \{1\}$ or $\Omega : Q \to \{2\}$ and the underlying weak automaton either rejects all trees or accepts all trees.

Lemma 5.14. *Let $\mathcal{A}$ be a hB-weak automaton with k hierarchical counters and a trivial acceptance condition. Then for all $q \in Q$, there is a formula $\varphi_q(x)$ of cost WMSO such that $[\![\mathcal{A}_q]\!](t_{\nu(x)}) \approx_{\beta_0} [\![\varphi_q(x)]\!](t, \nu)$ for all trees t and for all valuations $\nu : \{x\} \to \mathcal{T}$ where $\beta_0(n) := \alpha_k(\alpha_{\mathrm{nd}}(n)) + \alpha_{\mathrm{nd}}(\alpha_k(n))$ and $\alpha_k(n) = (n+1)^k$ and α_{nd} is the correction function from Lemma 5.9.*

Proof. If $\Omega(q) = 1$ for all $q \in Q$, then $\varphi_q(x)$ is set to some false statement. This means that $[\![\mathcal{A}_q]\!](t_{\nu(x)}) = [\![\varphi_q(x)]\!](t, \nu) = \infty$ for all trees t and valuations ν. Otherwise, $\Omega(q) = 2$ for all $q \in Q$, and all runs must satisfy the weak acceptance condition, so we can focus strictly on the counters when defining φ_q. We utilize the nondeterministic cost automaton on finite trees $\mathcal{A}_{\mathrm{nd}}$ given by Lemma 5.9 (which satisfies $[\![\mathcal{A}_{\mathrm{nd}}]\!] \approx [\![\mathcal{A}_q]\!]$).

Formula $\varphi_q(x)$ asserts that

$$\begin{aligned} &\forall \text{ frontier } E.\ \exists \text{ accepting run } R \text{ of } \mathcal{A}_{\mathrm{nd}} \text{ on } t_x|_E. \\ &\quad \forall \text{ branches } \pi \text{ in } R \text{ from } x \text{ to } y \in E.\ \mathrm{value}(\pi) \leq N \end{aligned}$$

where N is the boundedness variable. This means that we are approximating the value of $[\![\mathcal{A}_q]\!](t_x)$ by taking the supremum of $[\![\mathcal{A}_{\mathrm{nd}}]\!](t_x|_E)$ over all frontiers E. Formally, $\varphi_q(x)$ would take the form

$$\forall X.\exists \overline{Y}.\exists \overline{Z}.\forall Z'.\Big(\mathrm{Run}_B(x, X, \overline{Y}, \overline{Z}) \wedge (\mathrm{Check}_B(Z', \overline{Z}) \rightarrow |Z'| \leq N)\Big)$$

where $\overline{Y}$ is a tuple of $|Q_{\mathrm{nd}}|$ sets where each set represents the nodes in a particular state, $\overline{Z}$ is a tuple of sets where each set represents nodes where a particular counter operation was performed, and $\mathrm{Run}_B(x, X, \overline{Y}, \overline{Z})$ expresses the fact that X is a frontier of t_x, $\overline{Y}$ and $\overline{Z}$ describe a finite accepting run of $\mathcal{A}_{\mathrm{nd}}$ (starting in state q) on $t_x|_X$. Finally, $\mathrm{Check}_B(Z', \overline{Z})$ ensures that Z' is a series of positions along a single branch in the run where a particular counter is incremented and checked, and there are no intermediate positions where the counter is reset. It is straightforward to see that Run_B and Check_B are expressible in WMSO.

Fix some tree t and $\nu(x) = v \in \mathcal{T}$.

Assume that $[\![\mathcal{A}_q]\!](t_v) \leq n \in \mathbb{N}$. Then there is a finite memory strategy σ with $\mathrm{value}(\sigma) \leq \alpha_k(n)$. This means that for every frontier E of t_v, $\mathrm{value}(\sigma|_E) \leq \alpha_k(n)$. Hence, by Lemma 5.9, there is a run R of $\mathcal{A}_{\mathrm{nd}}$ on $t|_E$ such that $\mathrm{value}(R) \leq \alpha_{\mathrm{nd}}(\alpha_k(n))$. This is enough to conclude that $[\![\varphi_q(v)]\!] \leq \beta_0(n)$.

The other direction is more interesting. The fear is that there are runs of $\mathcal{A}_{\mathrm{nd}}$ of a low value on any initial finite subtree of t, but no infinite run of $\mathcal{A}_q$ on t with a low value. Assume for the sake of contradiction that there is some t such that $[\![\varphi_q(v)]\!](t) < n$ but $[\![\mathcal{A}_q]\!](t_v) \geq \beta_0(n)$. The underlying weak automaton without counters accepts all trees, so value $\beta_0(n)$ must be witnessed by the counters. By Proposition 3.11, this means there is a strategy $\overline{\sigma}$ for Eve in $\overline{\mathcal{A}_q \times t_v}$ such that every play witnesses counter values at least $\beta_0(n)$. A high B-value coming from the counters can always be witnessed by a finite prefix of the play, so there is a frontier E of t_v such that all plays in $\overline{\sigma}$ witness value at least $\beta_0(n)$ by E (if not, König's lemma

would imply that there is an infinite play in $\overline{\sigma}$ that never witnesses value at least $\beta_0(n)$). Then the finite duration game $\overline{\mathcal{A}_q \times t_v|_E}$ satisfies value($\overline{\mathcal{A}_q \times t_v|_E}$) $\geq \beta_0(n)$, so value($\mathcal{A}_q \times t_v|_E$) $\geq \beta_0(n) \geq \alpha_k(\alpha_{\mathrm{nd}}(n))$ by Proposition 3.11. But this means there is no finite memory strategy in $\mathcal{A}_q \times t_v|_E$ with value less than $\alpha_{\mathrm{nd}}(n)$. By Lemma 5.9, this implies that $[\![\mathcal{A}_{\mathrm{nd}}]\!](t_v|_E) \geq n$ and contradicts $[\![\varphi_q(v)]\!](t) < n$ since the formula requires value less than n for all frontiers, including this particular E. □

We now need to define a formula when the acceptance condition is not trivial. The proof will proceed by induction on the maximum length of alternating chain in $\mathcal{A}$.

Lemma 5.15. *Let $\mathcal{A}$ be a hB-weak automaton with alternating chains of length at most m. Then there is a cost WMSO sentence φ such that $[\![\mathcal{A}]\!] \approx [\![\varphi]\!]$.*

Proof. We prove a stronger result for the purposes of induction:

> for any state $q \in Q$, there is a formula $\varphi_q(x)$ such that for all trees t and all positions $v \in \mathcal{T}$, $[\![\varphi_q(v)]\!](t) \approx_{\beta_m} [\![\mathcal{A}_q]\!](t_v)$ where $\mathcal{A}_q$ denotes the automaton $\mathcal{A}$ starting from state q and m is the maximum length of alternating chains in $\mathcal{A}_q$.

The correction functions β_m are chosen to help the induction go through. We already defined the correction function $\beta_0(n) := \alpha_k(\alpha_{\mathrm{nd}}(n)) + \alpha_{\mathrm{nd}}(\alpha_k(n))$. We now define recursively $\beta_j(n) := \beta_{m-1}(\alpha_k(n)) + \beta_{m-1}(\alpha_{\mathrm{nd}}(n))$. Notice that $\beta_{j'}(n) \geq \beta_j(n)$ for all $j' \geq j$.

The desired sentence is then $\varphi := \varphi_{q_0}(\epsilon)$. The proof of the stronger result is by induction on the maximum length of alternating chains of $\mathcal{A}_q$. As before, we assume that we are starting with an hB-weak automaton $\mathcal{A}_q$. We write $\mathrm{alt}(q) = m$ if $\mathcal{A}_q$ has alternating chains of length at most m.

If $\mathrm{alt}(q) = 0$, then $\mathcal{A}_q$ satisfies the conditions for Lemma 5.14 and we are done.

Otherwise, assume $\mathrm{alt}(q) = m > 0$ and the induction hypothesis holds for $\mathcal{A}_r$ with $\mathrm{alt}(r) < m$. Let $\mathcal{A}_{\mathrm{nd}}$ be the automaton from Lemma 5.9 based on $\mathcal{A}_q$.

If $\Omega(q) = 1$, then $\varphi_q(x)$ expresses

$$\begin{aligned}
&\exists \text{ frontier } E \text{ of } t_x.\ \exists \text{ accepting run } R \text{ of } \mathcal{A}_{\mathrm{nd}} \text{ on } t_x|_E.\\
&\quad \forall \text{ branches } \pi \text{ in } R \text{ from } x \text{ to } y \in E.\\
&\qquad \mathrm{value}(\pi) \leq N \text{ and } \forall \text{ states } r \in \eta(R(y)).\ \mathrm{alt}(r) < m \wedge \varphi_r(y)
\end{aligned}$$

where N is the boundedness variable. It is straightforward to see that this is expressible in cost WMSO. Notice the formulas $\varphi_r(y)$ are well-defined by the inductive hypothesis because $\mathrm{alt}(r) < m$.

Fix some tree t and position v. We first prove that $[\![\varphi_q(v)]\!](t) \preccurlyeq_{\beta_m} [\![\mathcal{A}_q]\!](t_v)$. Assume $[\![\mathcal{A}_q]\!](t_v) \leq n \in \mathbb{N}$. There must be a finite memory strategy σ in $\mathcal{A}_q \times t_v$ such that every play satisfies the weak acceptance condition and has checked counter value at most $\alpha_k(n)$. Moreover, there must be a frontier E of t_v such that every play in σ has passed through a transition of priority 2 before reaching E (if not, then König's lemma would imply the existence of an infinite play in σ that never stabilizes to priority 2, a contradiction). Thus by Lemma 5.9 there is a partial run R of $\mathcal{A}_{\mathrm{nd}}$ on $t|_E$ such that for all branches π in R from v to $w \in E$ and for all states $r \in \eta(R(w))$, $\mathrm{alt}(r) < m$. The checked counter values on R must be bounded by $\alpha_{\mathrm{nd}}(\alpha_k(n))$ by Lemma 5.9. Moreover, on the extensions of this run from positions w in E, the inductive hypothesis implies $[\![\varphi_r(w)]\!](t) \leq \beta_{m-1}(\alpha_k(n))$. Therefore, overall, $[\![\varphi_q(v)]\!](t) \leq \max\{\alpha_{\mathrm{nd}}(\alpha_k(n)), \beta_{m-1}(\alpha_k(n))\}$, so $[\![\varphi_q(v)]\!](t) \preccurlyeq_{\beta_m} [\![\mathcal{A}_q]\!](t_v)$.

Next we must show $[\![\mathcal{A}_q]\!](t_v) \preccurlyeq_{\beta_m} [\![\varphi_q(v)]\!](t)$. Assume $[\![\varphi_q(v)]\!](t) \leq n$. Then there is a frontier E and a run R of $\mathcal{A}_{\mathrm{nd}}$ on $t_v|_E$ satisfying the conditions described by the formula. By Lemma 5.9, there is a finite memory strategy $\sigma|_E$ in $\mathcal{A}_q \times t_v|_E$ such that $\mathrm{value}(\sigma|_E) \leq \alpha_{\mathrm{nd}}(n)$. Moreover, the inductive hypothesis implies that there is a strategy $\sigma_{(r,w)}$ in $\mathcal{A}_r \times t_w$ such that $\mathrm{value}(\sigma_{(r,w)}) \leq \beta_{m-1}(n)$ for all positions $w \in E$ and $r \in \eta(R(w))$. From all such $\sigma_{(r,w)}$ and $\sigma|_E$ we can construct a strategy σ in $\mathcal{A}_q \times t_v$. The overall value of the strategy is at most the sum of $\alpha_{\mathrm{nd}}(n)$ (the value of a play in $\sigma|_E$) and $\beta_{m-1}(n)$ (the value of a play in some $\sigma_{(r,w)}$), so $\mathrm{value}(\sigma) \leq \alpha_{\mathrm{nd}}(n) + \beta_{m-1}(n) \leq \beta_m(n)$ as desired.

In the classical case, if $\Omega(q) = 2$, then we could dualize the automaton to get a weak automaton accepting the complement language and starting in a state with priority 1, use the previous case to write a formula for the complement, and then negate this formula. Here, dualization would result in an S-weak automaton that is difficult to describe directly using a cost WMSO sentence, so we continue working with an hB-weak automaton $\mathcal{A}_q$ but utilize the dual game $\overline{\mathcal{A}_q \times t_v}$ in the proof.

If $\Omega(q) = 2$ then $\varphi_q(x)$ expresses

$$\begin{aligned}
&\forall \text{ frontier } E \text{ of } t_x.\ \exists \text{ accepting run } R \text{ of } \mathcal{A}_{\mathrm{nd}} \text{ on } t_x|_E.\\
&\quad \forall \text{ branches } \pi \text{ in } R \text{ from } x \text{ to } y \in E.\\
&\qquad \mathrm{value}(\pi) \leq N \text{ and } \forall r \in \eta(R(y)).\ \mathrm{alt}(r) < m \longrightarrow \varphi_r(y)
\end{aligned}$$

where N is the boundedness variable.

First assume that $[\![\mathcal{A}_q]\!](t_v) \leq n \in \mathbb{N}$. Then there is a finite memory strategy σ for Eve in $\mathcal{A}_q \times t_v$ such that $\mathrm{value}(\sigma) \leq \alpha_k(n)$. By Lemma 5.9, for all frontier E,

$[\![\mathcal{A}_{\mathrm{nd}}]\!](t_v|_E) \leq \alpha_{\mathrm{nd}}(\alpha_k(n))$ and for all $w \in E$ and $r \in \eta(R(w))$ with $\mathrm{alt}(r) < m$, $[\![\mathcal{A}_r]\!](t_w) \leq \alpha_k(n)$ so the inductive hypothesis implies that $[\![\varphi_r(w)]\!](t) \leq \beta_{m-1}(\alpha_k(n))$. Hence, $[\![\varphi_q(v)]\!](t) \leq \max\{\alpha_{\mathrm{nd}}(\alpha_k(n)), \beta_{m-1}(\alpha_k(n))\} \leq \beta_m(n)$.

Next, assume that $[\![\varphi_q(v)]\!](t) \leq n$ but $[\![\mathcal{A}_q]\!](t_v) > \beta_m(n)$. Then Eve has a strategy $\overline{\sigma}$ in $\overline{\mathcal{A}_q \times t_v}$ such that every play in $\overline{\sigma}$ either stabilizes in priority 1 or has counter value exceeding $\beta_m(n)$. In fact, because $\Omega(q) = 2$, there is a frontier E such that all plays in $\overline{\sigma}$ either witness counter value exceeding $\beta_m(n)$ or visit priority 1 at least once by E (if not, then König's lemma would imply that there is an infinite play in $\overline{\sigma}$ which never witnesses counter values exceeding $\beta_m(n)$ and is always in priority 2, contradicting $\mathrm{value}(\overline{\sigma}) > \beta_m(n)$).

Since $[\![\varphi_q(v)]\!](t) \leq n$, there is an accepting run of R of $\mathcal{A}_{\mathrm{nd}}$ on $t_v|_E$ with value at most n, and every extension from positions $w \in E$ and $r \in \eta(R(w))$ with $\mathrm{alt}(r) < m$ have value at most n. By Lemma 5.9, the run R induces a strategy σ in $\mathcal{A}_q \times t_v|_E$ with $\mathrm{value}(\sigma|_E) \leq \alpha_{\mathrm{nd}}(n)$. Consider the partial play (up to some position $w \in E$) resulting from Eve playing from σ and Adam playing from $\overline{\sigma}$. This partial play has counter value at most $\alpha_{\mathrm{nd}}(n)$, so by the choice of E, it must be the case that this play visits priority 1 by E. Hence, we can apply the inductive hypothesis from all $r \in \eta(R(w))$ to ensure that $[\![\mathcal{A}_r]\!](t_w) \leq \beta_{m-1}(n)$. Hence there is a play in $\overline{\mathcal{A}_q \times t_v}$ consistent with $\overline{\sigma}$ that has value at most $\alpha_{\mathrm{nd}}(n) + \beta_{m-1}(n) \leq \beta_m(n)$. But this contradicts the fact that $\mathrm{value}(\overline{\sigma}) > \beta_m(n)$. □

5.2.3 Expressivity and decidability

Rabin [Rab70] showed there is a language of infinite trees definable in MSO that is not definable in WMSO. The separating language L consists of infinite trees over the alphabet $\{a, b\}$ on which every branch has finitely many b's. A similar result holds in the cost setting (and the separating function is the characteristic function for L).

Proposition 5.16. *There is a cost function over infinite trees definable in MSO (and hence cost MSO) that is not definable in cost WMSO.*

Proof. Consider the language L defined above that separates MSO and WMSO. Assume for the sake of contradiction that χ_L were recognized by a cost WMSO sentence φ. Then $[\![\varphi]\!] \approx_\alpha \chi_L$, so $t \in L$ if and only if $[\![\varphi]\!](t) \leq n$, for $n = \alpha(0)$.

But this means that we can replace any predicate $|X| \leq N$ in φ with the WMSO formula

$$\exists x_1, \ldots, x_n.(x \in X \leftrightarrow \bigvee_{i \in [1,n]} x = x_i)$$

to yield an equivalent classical WMSO formula φ' with $[\![\varphi']\!] = \chi_L$. This implies that the language defined by the WMSO formula φ' is L, contradicting the separation of traditional WMSO and MSO from Rabin [Rab70]. □

In fact, cost WMSO is incomparable with MSO, since there are also cost functions definable in cost WMSO not definable in MSO.

Proposition 5.17. *There is a cost function over infinite trees definable in cost WMSO that is not definable in MSO.*

Proof. Take any cost function that counts some behaviour in the input, say, the function $f(t) = |t|_a$. Assume there were some MSO sentence recognizing f. Then $f(t) \approx_\alpha \chi_L$ for some regular language L of infinite trees, and some correction function α. Take a tree t such that $\alpha(0) < f(t) < \infty$. Since $f(t) > \alpha(0)$, t cannot be in L. But since $f(t) < \infty$ and $t \notin L$, $f(t)$ is bounded but $\chi_L(t)$ is unbounded, a contradiction. □

Most importantly, cost WMSO represents a fragment of cost MSO for which the domination preorder is known to be decidable.

Theorem 5.18. *Given cost WMSO sentences φ and ψ over infinite trees, it is decidable whether or not $[\![\varphi]\!] \preccurlyeq [\![\psi]\!]$.*

Proof. We simply convert to equivalent weak cost automata $\mathcal{A}_\varphi$ and $\mathcal{A}_\psi$ as described in Section 5.2.1 and then apply Corollary 5.6. □

5.3 Discussion

We have seen in this chapter that cost WMSO has strong closure properties and decidability like classical WMSO. In particular, we have shown that cost WMSO is equivalent to weak cost automata in terms of recognizing cost functions (Theorem 5.13), and therefore $\preccurlyeq$ is decidable for the weak cost functions definable using cost WMSO (Theorem 5.18).

Another variant of WMSO that has received attention is WMSO+$\mathbb{U}$, where $\mathbb{U}$ is the unbounding quantifier described in Section 2.4. Although the satisfiability problem for full MSO+$\mathbb{U}$ remains open (even over infinite words), Bojańczyk and Toruńczyk have shown that the satisfiability problem for WMSO+$\mathbb{U}$ is decidable over infinite words [Boj09] and infinite trees [BT12]. The decidability proof goes via automata (*deterministic max automata* in the case of infinite words, and *nested lim*

sup automata in the case of infinite trees) which is similar to the classical approach by Büchi and Rabin, as well as the approach taken in this chapter. However, it appears that the decidability results for WMSO+$\mathbb{U}$ and cost WMSO are incomparable (that is, it appears that there is no way to derive the decidability of cost WMSO from the decidability of WMSO+$\mathbb{U}$, and vice versa). The connections between these logics remain to be explored in future work.

In [Lan11b], another cost logic called *first-order logic with resource relations* (FO+RR) is introduced. The syntax is similar to traditional first-order logic over some relational structure, except negation is not allowed. Like the cost logic in this chapter, however, formulas evaluated in a structure are assigned a value, and the values come from $\mathbb{N}_\infty$ because the relations, called *resource relations*, are evaluated as functions with range $\mathbb{N}_\infty$.

The decidability of this logic is studied over pushdown systems that are enriched with a finite set of counters that can be incremented, reset, or left unchanged (similar to the B-automata in this thesis). These pushdown systems can be viewed as a model of recursive programs with resource consumption, where the resource could be memory, time, energy, etc. The logic FO+RR can then be used to express desired properties of the system, e.g. that the resource usage is bounded across all runs of the program.

Lang shows that the configuration graphs of resource pushdown systems are first-order-interpretable in the binary tree (so there is a formula of first-order logic that can describe the domain and edges of the configuration graph within the binary tree). When restricted to systems with one counter and without resets, a formula ψ in FO+RR can be converted to a formula φ in cost WMSO such that the value of ψ over the configuration graph of the resource pushdown system is equivalent to the value of φ in the interpretation of the configuration graph in the binary tree [Lan11b]. By the decidability of cost WMSO over infinite trees in Theorem 7.4, this implies that FO+RR is decidable over the configuration graphs of resource pushdown systems with one counter without reset. An alternative proof for systems with multiple counters and with reset is given in [Lan11b], but it is now conjectured that the approach using cost WMSO should work in this extended case as well [Lan11a].

This is a nice application of the decidability of cost WMSO, and may point to further applications of the results in this thesis for verification of systems with some quantitative features.

Chapter 6

Quasi-Weak Cost Automata

In this chapter, we explore quasi-weak cost automata, a new variant of weak automata introduced by Kuperberg and the author [KVB11]. Roughly speaking, quasi-weak cost automata are an extension of weak cost automata that share the property that "good" plays must stabilize in accepting priorities.

In Section 6.1, we show that quasi-weak cost automata are strictly more expressive than weak cost automata over infinite trees, but can still be simulated by both nondeterministic B-$[1, 2]$ and nondeterministic S-$[1, 2]$ automata. This implies decidability of the domination preorder for quasi-weak cost functions. In Section 6.2, we show a form of converse: given a nondeterministic B-$[1, 2]$ and nondeterministic S-$[1, 2]$ automaton defining the same cost function, we construct a quasi-weak cost automaton recognizing it. The proof of this result is technical, so we give both an informal description of the construction as well as a formal proof.

These results can be viewed as a generalization of Rabin's famous characterization of weakly definable languages which stated that a language is weakly definable if and only if the language and its complement are recognizable by nondeterministic Büchi automata [Rab70]. The fact that quasi-weak cost automata, rather than the more traditional weak cost automata, admit this Rabin-style characterization represents an interesting divergence from the classical theory.

This chapter is based on joint work with Kuperberg that was first presented in [KVB11].

6.1 Expressivity

Quasi-weak cost automata are an extension of B-weak automata that share the property that "good" plays (plays with a low value) must stabilize in accepting priorities.

Recall that in a weak automaton, there is a hard coded bound on the number of alternations between accepting and rejecting priorities because of the restriction that no cycle contains both an accepting and rejecting priority. Here we have another tool available to bound the number of such alternations: the counters. We know that in a B-automaton, an accepting play of finite value n does at most n increments between resets, but this number is not known a priori by the automaton. If we guarantee that there is correction function α such that in any play π of value n, $\alpha(n)$ is greater than the number of alternations between accepting and rejecting priorities in π, then we know that any play of finite value must stabilize in an accepting priority. Otherwise, infinitely many alternations would give value ∞ to the cost function computed by the automaton. This is what we mean by quasi-weak.

Definition 6.1. A B-$[1,2]$ or B-$[0,1]$ automaton is *quasi-weak* if there is a correction function α such that for all t, in any play of $\mathcal{A} \times t$ of value $n < \infty$, the number of alternations between accepting priorities and rejecting priorities is at most $\alpha(n)$. We say a cost function is *quasi-weak* if it is recognized by some B-quasi-weak automaton. A cost game $\mathcal{G}$ is quasi-weak if it is of the form $\mathcal{A} \times t$ for a quasi-weak automaton $\mathcal{A}$.

Remark 6.2. In particular, any weak automaton $\mathcal{A}$ is quasi-weak since we can take $\alpha(n) = l$ for all n, where l is the maximum length of alternating chains in $\mathcal{A}$. Also, if $\mathcal{A}$ has no counters, quasi-weakness implies $\mathcal{A}$ is weak with alternating chains of length at most $\alpha(0)$ since $[\![\mathcal{A}]\!](t) \in \{0, \infty\}$ for all t.

We can also give a structural characterization of quasi-weakness which we will call the *quasi-weak cycle condition*: in any reachable cycle containing both accepting and rejecting priorities, there is some counter that is incremented but not reset.

Proposition 6.3. *A B-$[1,2]$ or B-$[0,1]$ automaton $\mathcal{A}$ is quasi-weak if and only if it satisfies the quasi-weak cycle condition: in any reachable cycle containing both accepting and rejecting priorities, some counter is incremented but not reset.*

Proof. Let $\mathcal{A} = \langle Q, \mathbb{A}, q_0, O, \delta \rangle$ be a B-$[1,2]$ automaton, and let $k = |\Gamma|$.

We first assume $\mathcal{A}$ does not satisfy the quasi-weak cycle condition, i.e. $\mathcal{A}$ contains a reachable cycle c in the transition function with both priorities and such that for all $\gamma \in \Gamma$, if there is `ic` for γ in c, then there is `r` for γ in c.

Since c is reachable, there exists an input tree t and a play π of $\mathcal{A}$ on t that reaches c after a finite partial play u, and then repeats c forever: $\pi = u(c)^\omega$. The play π is accepting, but has infinitely many alternations between priorities. Moreover, its value is bounded by $\text{value}_B(uc)$, since c performs a reset for any counter that is

incremented. This means that there can be no α bounding the number of alternations between priorities as a function of the value of the play, so we can conclude that $\mathcal{A}$ cannot be quasi-weak. By contraposition, any quasi-weak automaton satisfies the quasi-weak cycle condition.

Now assume $\mathcal{A}$ satisfies the quasi-weak cycle condition. Let π be an accepting play of $\mathcal{A}$ of finite value n. Let m be the number of alternations between accepting and rejecting priorities in π. We want to show that if m is sufficiently high, then $\text{value}(\pi) > n$, which would be a contradiction.

We will use Ramsey's theorem in order to define a large number, depending only on $\mathcal{A}$ (in particular, on the number of counters k used by $\mathcal{A}$) and n. Let $\alpha_k(n)$ be the bound given by Ramsey's theorem, ensuring that if a graph G has size $\alpha_k(n)$ and has edges coloured with $2^k - 1$ colours, it contains a one-colour clique of size $n + 2$. Notice that $\alpha_k(n)$ only depends on (k, n). It means that if k is fixed, we can view α_k as a correction function that maps n to the value $\alpha_k(n)$ given by Ramsey's theorem.

Assume m is strictly greater than $2|Q|\alpha_k(n)$. We can find states $(q_i)_{1\leq i\leq m}$ such that π visits q_1 to q_m in this order, and the priority output when leaving q_i is accepting if i is even and rejecting if i is odd. Let u be the finite infix of π, from q_1 to q_m. For all i, let x_i be the position of q_i in u.

Then there exists a set $I \subset [1, m]$ and a state $q \in Q$ such that $|I| \geq \alpha_k(n)$, and for all $i \in I$, $q_i = q$. Thus, u contains $|I| - 1$ consecutive cycles from q to q, with both accepting and rejecting priorities in each cycle. By the quasi-weak cycle condition, each of these cycles (and any concatenation of several of them) must increment a counter without resetting it.

Consider the complete graph G with vertices $\{x_i : i \in I\}$, of size at least $\alpha_k(n)$. We define the set of colours $K = \{A \subseteq \Gamma : A \neq \emptyset\}$. We colour edges of G in the following way: for any $i < j$ in I, the colour of the edge between x_i and x_j is A, where A is the set of counters that are incremented but not reset in the path from x_i to x_j in u. The quasi-weak cycle condition ensures that $A \neq \emptyset$.

By choice of $\alpha_k(n)$, and since $|K| = 2^k - 1$, there is a clique C of size $n + 2$ in G, entirely coloured by some $A \in K$. Let $\gamma \in A$. We can write $C = \left\{x_{i_1}, \ldots, x_{i_{n+2}}\right\}$, with $i_1 < \cdots < i_{n+2}$. For all $j \in [1, n + 1]$, there is an increment of γ and no reset of γ in u, between x_{i_j} and $x_{i_{j+1}}$. It means that between x_{i_1} and $x_{i_{n+2}}$, γ is incremented at least $n + 1$ times and never reset. This implies $\text{value}(\pi) \geq n + 1$, which is absurd.

Since assuming the number of alternations $m > 2|Q|\alpha_k(n)$ leads to a contradiction, we must have $m < 2|Q|\alpha_k(n)$. We can conclude that any accepting play of $\mathcal{A}$ of value n

B-weak automaton	B-quasi-weak automaton
alternating B-$[1,2]$ or B-$[0,1]$	alternating B-$[1,2]$ or B-$[0,1]$
$\exists M.\forall t.\forall \sigma$ for Eve in $\mathcal{A} \times t$. any play in σ has at most M alternations between priorities	$\forall N.\exists M.\forall t.\forall \sigma$ for Eve in $\mathcal{A} \times t$. $\text{value}(\sigma) \leq N \longrightarrow$ any play in σ has at most M alternations between priorities
there is no cycle with both priorities	if there is a cycle with both priorities, then there is some `ic` without `r`

Table 6.1. Comparison of weak and quasi-weak cost automata.

does at most $2|Q|\alpha_k(n)$ alternations between accepting and rejecting states, so $\mathcal{A}$ is a B-quasi-weak automaton. □

This quasi-weak cycle condition means that one can think of a quasi-weak cost automaton as an automaton that has an extra counter that is incremented and checked each time the priority changes (and never reset). Indeed, a counter behaving like this could always be added to a B-quasi-weak automaton without changing the cost function recognized by it. Table 6.1 shows how this compares with the characteristics of weak cost automata.

Over infinite trees, the class of cost functions definable by quasi-weak cost automata is strictly more expressive than the class of cost functions definable using weak cost automata.

Proposition 6.4. *There exists a cost function over infinite trees that is recognized by a nondeterministic B-quasi-weak automaton, but not by any B-weak automaton. Consequently, B-quasi-weak automata are strictly more expressive than B-weak automata over infinite trees.*

Proof. We will give an explicit cost function f and for each $n \in \mathbb{N}$ an infinite tree t_n. These trees include labels that dictate which player controls each position in the game (inspired by [AN07]) and are designed such that any alternating B-automaton recognizing f is forced to do $\alpha_{\text{alt}}(n)$ alternations between accepting and rejecting priorities on t_n for some correction function α_{alt}. This shows f cannot be computed by a B-weak automaton. On the other hand, we give an explicit nondeterministic B-quasi-weak automaton for f.

The function f acts on trees over $\mathbb{A} = \{\vee, \wedge\} \times \{e, a, b\}$. We write $\mathrm{pr}_1 : \mathbb{A} \longrightarrow \{\vee, \wedge\}$ and $\mathrm{pr}_2 : \mathbb{A} \longrightarrow \{e, a, b\}$ for the projections of $\mathbb{A}$ onto its components.

If t is an $\mathbb{A}$-labelled tree, we will say a subset C_t of $\mathcal{T}$ is a *choice tree* for t if $\epsilon \in C_t$ and for all $x \in C_t$,

- if $\mathrm{pr}_1(t(x)) = \vee$ then C_t contains either the left or right child of x;
- if $\mathrm{pr}_1(t(x)) = \wedge$ then C_t contains both children of x;
- if $\mathrm{pr}_2(t(x)) = a$, then all paths of C_t starting in x contain a b-labelled position.

Moreover, if C_t is a choice tree of t, we define $\mathrm{value}(C_t)$ as the maximum number of a's on some path in C_t according to the labelling of t.

We now define the desired cost function

$$f(t) = \inf \{\mathrm{value}(C_t) : C_t \text{ is a choice tree of } t\}.$$

In particular, if there is no choice tree of t, then $f(t) = \infty$.

It is easy to show that f is recognized by a nondeterministic B-quasi-weak automaton: it has two states, and guesses a choice tree while counting the number of a's. If an a is seen, then the automaton outputs a rejecting priority until b is seen; otherwise, it outputs accepting priorities. This means the number of alternations for C_t is bounded by $2\,\mathrm{value}(C_t)$. Since $\mathrm{value}(C_t)$ is the value of the run corresponding to the choice of C_t, this describes a nondeterministic B-quasi-weak automaton.

Now assume for the sake of contradiction that f is recognized by a B-weak automaton $\mathcal{A}$ with N states, up to some correction function α. Because $\mathcal{A}$ is weak, there is some bound m on the maximum length of alternating chains in $\mathcal{A}$.

We inductively build a family of $\mathbb{A}$-labelled trees $(t_n)_{n\in\mathbb{N}}$ (positions are labelled by $[0,1]^*$). The tree t_0 is entirely labelled by $(\wedge, e)$. If t_{n-1} is built, we build t_n in the following way (see Figure 6.2):

- $t_n(0^*) = (\wedge, e)$;
- $t_n(0^*1) = (\vee, a)$;
- $t_n(0^*11^+) = (\vee, e)$;
- $t_n(0^*1^N1^+0) = (\wedge, b)$;
- Subtrees rooted in nodes of $0^*1^N1^+00^+$ are t_{n-1};
- Other subtrees (to complete the binary tree) are t_0.

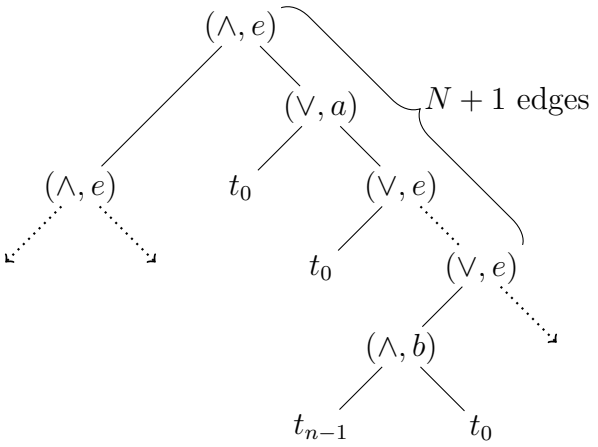

Figure 6.2. The tree t_n forcing alternations between accepting and rejecting priorities (see Proposition 6.4).

We have $f(t_n) = n$ for all $n \in \mathbb{N}$. We prove by induction on n that a B-$[1, 2]$ automaton computing f must do $2n$ alternations while processing input t_n. This is trivial for $n = 0$.

Fix $n \in \mathbb{N}$, and consider a strategy for Eve witnessing value $\alpha(n)$ on t_n. Every play consistent with this strategy must stabilize in an accepting priority. In particular, on the branch 0^ω, the play must stabilize starting at some node 0^d. Since the label is $\wedge$, the subtree rooted in 0^d1 needs to be accepting. Eve is forced to reach node 0^d1^{N+1} in order to witness an accepting choice tree for t_n. By choice of N, there must be a cycle between 0^d and 0^d1^{N+1}.

Assume Adam can enforce an increment without reset during this cycle. Then consider the infinite sequence of trees generated by repeating the cycle a finite (but increasing) number of times. On this sequence, the value of f is unchanged, but Adam has a family of strategies witnessing unbounded value for $\mathcal{A}$. This is absurd so we can consider that the strategy of Eve enforces a reset for any increment in the cycle.

Now if this cycle is accepting, repeating it infinitely many times would create a tree accepted by $\mathcal{A}$ with a bounded cost, but with value ∞ for f, since the a at position 0^d1 will never be followed by a b on any path. Since $\mathcal{A}$ recognizes f, we get a contradiction, so this cycle has to include a rejecting priority. Then Eve can read a b by choosing any node $0^d1^N1^+0$, and this eventually returns to an accepting priority on some node of $0^d1^N1^+00^{d'}$ (otherwise a partial version of t_n would not be accepted by $\mathcal{A}$, but would have value 1 by f). We then need to accept t_{n-1} at node $0^d1^N1^+00^{d'}$, which by the inductive hypothesis requires at least $2(n-1)$ alternations.

Overall we can conclude that a B-weak automaton computing f needs to alternate between accepting and rejecting priorities at least $2n$ times when reading t_n, for any $n \in \mathbb{N}$. This is absurd since $\mathcal{A}$ has a fixed number of alternations m across all inputs, so f cannot be computed by a B-weak automaton. □

The constructions in Section 6.2 and Section 7.2 also provide examples where quasi-weak, rather than weak, cost automata are required because we are unable to bound the number of alternations between accepting and rejecting priorities upfront.

6.2 Rabin-style characterization

Rabin [Rab70] showed an interesting characterization of WMSO in terms of nondeterministic automata. Recast in terms of weak automata rather than WMSO (based on [MSS86]), the characterization can be stated as follows.

Theorem 6.5 ([Rab70, MSS86]). *A regular language L of infinite trees is recognizable by a weak automaton $\mathcal{W}$ (equivalently, definable in WMSO) if and only if there are nondeterministic Büchi automata $\mathcal{U}$ and $\mathcal{U}'$ such that*

$$L = L(\mathcal{U}) = \overline{L(\mathcal{U}')}.$$

Moreover, the conversion between $\mathcal{W}$ and $\mathcal{U}/\mathcal{U}'$ (and vice versa) is effective.

Recall that in the cost setting, complementation corresponds to switching between the B and S semantics (see Example 2.3). With this in mind, a natural conjecture for the cost setting would be "a cost function is definable using a weak cost automaton if and only if it is definable by both a nondeterministic B-Büchi and S-Büchi automaton". As we saw in the last chapter, one direction is true: weak cost automata can be simulated by both nondeterministic B-Büchi and S-Büchi automata. However, the other direction turns out to be false. Instead, it is the larger class of quasi-weak cost automata that admits this Rabin-style characterization.

Theorem 6.6. *A cost function f over infinite trees is recognizable by a B-quasi-weak automaton $\mathcal{B}$ if and only if there is a nondeterministic B-Büchi automaton $\mathcal{U}$ and a nondeterministic S-Büchi automaton $\mathcal{U}'$ such that*

$$f \approx [\![\mathcal{U}]\!]_B \approx [\![\mathcal{U}']\!]_S.$$

Moreover, the conversion between $\mathcal{B}$ and $\mathcal{U}/\mathcal{U}'$ (and vice versa) is effective.

Remark 6.7. When restricted to languages, Theorem 6.6 implies Theorem 6.5 since

- if there are nondeterministic Büchi automata $\mathcal{U}$ and $\mathcal{U}'$ (without counters) recognizing a language and its complement, respectively, then $[\![\mathcal{U}]\!]_B = [\![\mathcal{U}']\!]_S$ (see Example 2.3), and
- quasi-weak and weak automata coincide when the automata have no counters (see Remark 6.2).

6.2.1 Simulation and decidability

Because plays in a weak automaton stabilize in a single priority, weak automata can be viewed as parity automata using either priorities $[1, 2]$ or $[0, 1]$. It is this property that allowed us to apply the results from Chapter 4 to show that weak cost automata can be simulated by both nondeterministic B-$[1, 2]$ and nondeterministic S-$[1, 2]$ automata (Corollary 5.5), thereby proving that the domination preorder is decidable for weak cost functions over infinite trees. As discussed in the previous section, quasi-weak cost automata share this property, so we immediately get the following result as a corollary to Theorem 4.28.

Corollary 6.8. *Let $\mathcal{B}$ be a B-quasi-weak automaton. Then $\mathcal{B}$ can be simulated by a nondeterministic B-$[1, 2]$ and a nondeterministic S-$[1, 2]$ automaton.*

Notice that this is one direction of the Rabin-style characterization in Theorem 6.6. Combined with Theorem 4.32, we get the following decidability result for quasi-weak cost functions.

Corollary 6.9. *The relation $f_1 \preccurlyeq f_2$ is decidable for quasi-weak cost functions f_1 and f_2 over infinite trees.*

Based on the expressivity result in Proposition 6.4, this means that quasi-weak cost functions are the largest class of cost functions over infinite trees for which $\preccurlyeq$ is known to be decidable.

We remark that for both weak and quasi-weak cost automata it is possible to improve slightly the complexity (in terms of number of states) required for simulation with a nondeterministic B-$[1, 2]$ automaton. Recall that the nondeterministic version guesses a finite memory strategy in the corresponding cost game, so the number of states depends on the size of the memory required. Theorem 4.9 and Proposition 4.23 showed that in general B-$[1, 2]$ games require memory of size two. This can be improved to a positional strategy for the special case of B-weak and B-quasi-weak

games. The proof was presented in [VB11] for B-weak cost games, but can be easily extended to B-quasi-weak games. The price one pays is that the correction function increases from $\alpha(n) = (n+1)^k$ in the case of the finite memory strategy to $\alpha(n) = 2 \cdot \alpha_{\text{alt}}(n) \cdot (n+1)^k$ in the case of the positional strategy, where k is the number of counters and $\alpha_{\text{alt}}(n)$ is the maximum number of alternations between accepting and rejecting priorities for a play of value n.

6.2.2 Quasi-weak construction

We now turn to the more difficult direction of Theorem 6.6, moving from nondeterministic cost-Büchi automata for some cost function to a quasi-weak automaton defining the same cost function. We start by describing the classical proof (mainly following Kupferman and Vardi [KV99]) constructing a weak automaton from nondeterministic Büchi automata for the language and its complement. We then proceed to the more technical construction required in the cost setting which results in a quasi-weak cost automaton.

To improve readability, we switch to a more classical notation for Büchi automata, using a set of accepting states that must be visited infinitely often (instead of a parity condition with priorities $[1,2]$ or $[0,1]$); see Section 3.4 for more information.

Classical construction

Let $\mathcal{U} = \langle Q_{\mathcal{U}}, \mathbb{A}, q_0^{\mathcal{U}}, F_{\mathcal{U}}, \Delta_{\mathcal{U}} \rangle$ and $\mathcal{U}' = \langle Q_{\mathcal{U}'}, \mathbb{A}, q_0^{\mathcal{U}'}, F_{\mathcal{U}'}, \Delta_{\mathcal{U}'} \rangle$. The classical result states that starting from nondeterministic Büchi automata $\mathcal{U}$ and $\mathcal{U}'$ such that $L(\mathcal{U})$ is the complement of $L(\mathcal{U}')$, a weak automaton $\mathcal{W}$ can be constructed such that $L(\mathcal{W}) = L(\mathcal{U})$.

The proofs in [Rab70, KV99] begin with an analysis of composed runs of $\mathcal{U}$ and $\mathcal{U}'$. Let $m := |Q_{\mathcal{U}}| \cdot |Q_{\mathcal{U}'}|$. Recall that a frontier E is a set of nodes of t such that for any branch π of t, $E \cap \pi$ is a singleton. Kupferman and Vardi [KV99] define a *trap* for $\mathcal{U}$ and $\mathcal{U}'$ to be a strictly increasing sequence of frontiers $E_0 = \{\epsilon\}, E_1 \ldots, E_m$ such that there exists a tree t, a run R of $\mathcal{U}$ on t, and a run R' of $\mathcal{U}'$ on t satisfying the following properties: for all $0 \leq i < m$ and for all branches π in t, there exists $x, x' \in [e_i^\pi, e_{i+1}^\pi)$ such that $R(x) \in F_{\mathcal{U}}$ and $R'(x') \in F_{\mathcal{U}'}$ where $e_0^\pi < \cdots < e_m^\pi$ is the set of nodes from $E_0, \ldots, E_m$ induced by π. The set of positions $[e_i^\pi, e_{i+1}^\pi)$ can be viewed as an *accepting block* that witnesses an accepting state from both $\mathcal{U}$ and $\mathcal{U}'$.

This is called a trap because $L(\mathcal{U}')$ is the complement of $L(\mathcal{U})$, but a trap implies $L(\mathcal{U}) \cap L(\mathcal{U}') \neq \emptyset$ using a pumping argument on blocks.

The weak automaton $\mathcal{W}$ built in [KV99] has Eve (respectively, Adam) select a run of $\mathcal{U}$ (respectively, $\mathcal{U}'$). The acceptance condition requires that any time an accepting state from $\mathcal{U}'$ is seen, an accepting state from $\mathcal{U}$ is eventually seen. Because of the trap condition, these accepting blocks only need to be counted up to m times before the automaton is allowed to enter an accepting sink state, so $\mathcal{W}$ is weak. The idea is that if $t \in L(\mathcal{U})$, then Eve has a strategy to play in the game $\mathcal{W} \times t$ (i.e. play the accepting run of $\mathcal{U}$). On the other hand, if $t \in L(\mathcal{U}')$ and we assume by contradiction that Eve has a winning strategy in $\mathcal{W} \times t$, then this strategy and the accepting run of $\mathcal{U}'$ can be used to build a trap, which is impossible.

Cost trap

We want to extend these ideas to the cost setting where $\mathcal{U} = \langle Q_{\mathcal{U}}, \mathbb{A}, q_0^{\mathcal{U}}, \Gamma_B^{\mathcal{U}}, F_B^{\mathcal{U}}, \Delta_{\mathcal{U}} \rangle$ (respectively, $\mathcal{U}' = \langle Q_{\mathcal{U}'}, \mathbb{A}, q_0^{\mathcal{U}'}, \Gamma_S^{\mathcal{U}'}, F_S^{\mathcal{U}'}, \Delta_{\mathcal{U}'} \rangle$) is a nondeterministic B-Büchi (respectively, S-Büchi) automaton and $[\![\mathcal{U}]\!]_B \approx [\![\mathcal{U}']\!]_S$. Our goal is to construct a B-quasi-weak automaton $\mathcal{B}$ that is equivalent to $\mathcal{U}$.

We seek a notion of "cost trap" that implies a contradiction with $[\![\mathcal{U}]\!]_B \approx [\![\mathcal{U}']\!]_S$. More specifically, we want a notion of blocks and traps that will help witness a bounded B-value from $\mathcal{U}$ on some set of trees but an unbounded S-value for $\mathcal{U}'$ on the same set, showing $[\![\mathcal{U}']\!]_S \not\preccurlyeq [\![\mathcal{U}]\!]_B$ and therefore $[\![\mathcal{U}']\!]_S \not\approx [\![\mathcal{U}]\!]_B$.

The definition of a block when using arbitrary B- and S-counter actions coming from $\mathcal{U}$ and $\mathcal{U}'$ would be very intricate because it would have to deal with the interaction of the B- and S-actions. To avoid these complications, we first package $\mathcal{U}$ and $\mathcal{U}'$ into a single nondeterministic BS-Büchi automaton $\mathcal{U} \times \mathcal{U}'$ in an obvious way: a run of this automaton is simply the composition of a run of $\mathcal{U}$ and a run of $\mathcal{U}'$ on the same input. We then consider a hierarchical BS-Büchi automaton $\mathcal{A} = \langle Q_{\mathcal{A}}, \mathbb{A}, q_0^{\mathcal{A}}, \Gamma_B, F_B, \Gamma_S, F_S, \delta_{\mathcal{A}} \rangle$ that satisfies $\mathcal{A} \cong \mathcal{U} \times \mathcal{U}'$ but uses hierarchical BS-counter actions as described in Section 3.4. This is possible by Lemma 3.18.

A *block* based on hierarchical BS-actions from $\mathcal{A}$ has accepting states from both F_B and F_S (corresponding to accepting states for $\mathcal{U}$ and $\mathcal{U}'$), but it also has a reset for B-counter γ if γ is incremented in that block in order to ensure pumping does not inflate the B-value. The number of blocks we are required to count is dependent on the size of $\mathcal{A}$ and the number of S-counters in $\mathcal{A}$ and is increased to $m := (|Q_{\mathcal{A}}|+2)^{|\Gamma_S|+1}$.

A *cost trap* for $\mathcal{A}$ is a frontier E_m and for every branch π up to E_m a strictly increasing set of nodes $e_0^\pi < \cdots < e_m^\pi \in E_m$ such that there exists a tree t and a run R of $\mathcal{A}$ on t with $\text{value}_S(R) > |Q_{\mathcal{A}}|$ satisfying the following properties: for all $0 \leq i < m$ and for all branches π, $[e_i^\pi, e_{i+1}^\pi)$ is a block; and if branches π_1 and π_2 share

some prefix up to position y and $x < y$ is the first position with $e_i^{\pi_1} = x$ and $e_i^{\pi_2} \neq x$ then $e_i^{\pi_2} > y$ (we call this the *nesting condition*, which ensures that pumping blocks from π_2 does not damage blocks from π_1).

An intricate, but not difficult, pumping argument shows that the existence of a cost trap for $\mathcal{A}$ implies $\mathcal{U}$ and $\mathcal{U}'$ are not equivalent.

Proposition 6.10. *Let $\mathcal{U}$ (respectively, $\mathcal{U}'$) be a nondeterministic B-Büchi automaton (respectively, S-Büchi automaton). Let $\mathcal{A} \approx \mathcal{U} \times \mathcal{U}'$ be a nondeterministic hBS-automaton. If there exists a cost trap for $\mathcal{A}$, then $[\![\mathcal{U}']\!] \not\preccurlyeq [\![\mathcal{U}]\!]$.*

The idea for the proof is that by pumping blocks, we first construct an infinite tree and a run of $\mathcal{A}$ on this tree that is B-accepting and S-accepting, has a low B-value, and also has an S-value at least $|Q_{\mathcal{A}}|$. From this single tree and run, we then construct a family of trees and runs that are still B-accepting, S-accepting and have a low B-value, but have unbounded S-value. This is done by pumping segments with increments before check-resets a finite, but increasing, number of times. We delay the proof of this result until after we describe the construction.

This result holds for any $\mathcal{A}$ that is BS-equivalent to $\mathcal{U} \times \mathcal{U}'$. For the construction, however, we fix a particular $\mathcal{A}$. Specifically, if $\mathcal{H}$ is the transducer $\mathcal{H}(\Gamma_B^{\mathcal{U}}, \Gamma_S^{\mathcal{U}'})$ from Theorem 3.17, then we let $\mathcal{A}$ be the nondeterministic BS-Büchi automaton $\mathcal{H} \circ (\mathcal{U} \times \mathcal{U}')$ and let $m := (|Q_{\mathcal{A}}| + 2)^{|\Gamma_S^{\mathcal{H}}|+1}$ where $\Gamma_S^{\mathcal{H}}$ is the set of S-counters used by $\mathcal{H}$.

Overview of construction

We now describe informally the construction of a B-quasi-weak automaton $\mathcal{B}$. On an input tree t, $\mathcal{B}$ is designed to:

- simulate in parallel a run of $\mathcal{U}$ (selected by Eve) and a run of $\mathcal{U}'$ (selected by Adam) on the input t;
- run the hBS-transducer $\mathcal{H}$ over the composed actions from $\mathcal{U}$ and $\mathcal{U}'$;
- analyse the hierarchical counter actions output by $\mathcal{H}$ together with the accepting states of $\mathcal{U}$ and $\mathcal{U}'$, keeping track of blocks (described below);
- output the B-actions from the run of $\mathcal{U}$.

We want to ensure that the run of $\mathcal{U}$ that Eve plays is actually a run with a low B-value and is accepting for the Büchi condition. We ensure that it has a low B-value by copying exactly the B-actions from the run played by Eve. The difficulty is that

we cannot directly check whether the run that Eve plays satisfies the Büchi condition, because we only have the power of a quasi-weak automaton. This is where the block counting and acceptance condition comes in.

The acceptance condition is similar to the classical construction: while the blocks are being counted, it requires that if an accepting state from $\mathcal{U}'$ is seen, then eventually an accepting state from $\mathcal{U}$ is seen (i.e. a block is never left open because we are waiting for an accepting state from $\mathcal{U}$). That means while we are counting blocks, we are ensuring that accepting states from $\mathcal{U}$ are being visited.

In [KV99], the block number only increases and it suffices to count blocks up to a fixed bound. This means that there is no cycle containing both accepting and rejecting states, so the resulting automaton is weak.

Here the definition of a block also forbids the presence of an increment for some B-counter γ that does not have a reset for γ. It may be the case that on a branch of a run of $\mathcal{U}$, some counter is incremented but is never reset. This could disrupt the block counting, because a block could remain open in an accepting state simply because it is waiting for a reset for γ, and this reset may not necessarily exist. Because this block is open, Eve could cheat and play a run with a low B-value but is not accepting for the Büchi condition since, while the block is open, $\mathcal{B}$ is not checking for additional accepting states from $\mathcal{U}$.

One idea would be to allow Eve to guess whether or not there are infinitely many resets for each B-counter in the run that she wants to play. If this is the case, then the block counting can start immediately and proceed as in the classical construction since there will always be a reset for each B-counter γ that is incremented. If not, then Eve could guess when the last increment without a reset is seen, and then start counting the blocks after this point. This sort of construction is possible over words (in fact, this idea can be used to give an improved construction over words [KVB12] that generalizes [KV01]). However, consider a tree and a nondeterministic B-Büchi automaton that at positions 0^*1 outputs `ic` (and everywhere else ε). At every position 0^d along the spine, Eve would never be able to start counting blocks because 0^d1 is a position where there is an increment without a reset. This means that the block counting would never start on the spine, allowing Eve to cheat by playing a run that is not accepting for the Büchi condition on the spine.

Because of this issue with the branching, we must employ a more sophisticated block counting technique. Indeed, $\mathcal{B}$ should be able allowed to restart the block counting if an increment is seen that does not have a later reset. The crucial point here is that any decrease in the block number when $\mathcal{B}$ restarts the counting is prompted

by an increase in the cost of the play due to an increment from a B-counter. Consequently, the bound on the number of blocks and the number of alternations between accepting and rejecting states depends on the B-value of the selected run, which is exactly the property of a quasi-weak automaton.

With this block counting with restarts in place, the proof of correctness starts by assuming that $[\![\mathcal{U}]\!]_B \approx [\![\mathcal{U}']\!]_S$ and proves that $[\![\mathcal{B}]\!]_B \approx [\![\mathcal{U}]\!]_B$. If $\mathcal{U}$ accepts some t with low value, then it gives Eve a strategy of the same value in $\mathcal{B} \times t$, so $[\![\mathcal{B}]\!]_B(t) \leq [\![\mathcal{U}]\!]_B(t)$. On the other hand, assuming (for the sake of contradiction) that Eve has a low-value strategy in $\mathcal{B} \times t$ but $\mathcal{U}$ actually assigns t a high value, means that one could construct a cost trap, which is absurd by Proposition 6.10. This implies that $[\![\mathcal{U}]\!]_B \preccurlyeq [\![\mathcal{B}]\!]_B$. Overall, $[\![\mathcal{B}]\!]_B \approx [\![\mathcal{U}]\!]_B \approx [\![\mathcal{U}']\!]_S$ as desired.

Formal construction

Recall $\mathcal{A}$ is $\mathcal{H} \circ (\mathcal{U} \times \mathcal{U}')$ and $m = (|Q_{\mathcal{A}}| + 2)^{|\Gamma_S^{\mathcal{H}}|+1}$. Let $K = |\Gamma_B^{\mathcal{H}}|$.

Formally, the B-quasi-weak automaton $\mathcal{B}$ has states

$$Q := Q_{\mathcal{U}} \times Q_{\mathcal{U}'} \times Q_{\mathcal{H}} \times ((I \times J \times Z) \cup \{q_\top\})$$

where

$$I := [1, m]^{[0,K]} \quad , \quad J := \{\epsilon, \bot, \top\}^{[0,K]} \quad , \quad Z := \{\varepsilon, \mathtt{ic}, \mathtt{r}\}^{[0,K]}.$$

This means a state is of the form $(q_{\mathcal{U}}, q_{\mathcal{U}'}, q_{\mathcal{H}}, (i, j, z))$ where i, j, and z are vectors used while the automaton is counting blocks, or $(q_{\mathcal{U}}, q_{\mathcal{U}'}, q_{\mathcal{H}}, q_\top)$ when the automaton is no longer counting blocks but is still simulating a run of $\mathcal{U}$ and $\mathcal{U}'$. We write, e.g. $i(k)$ for the kth component of the vector i. In fact, there is only a subset Q_{use} of Q that is used: an element $(q_{\mathcal{U}}, q_{\mathcal{U}'}, q_{\mathcal{H}}, (i, j, z))$ is in Q_{use} if for all $k, k' \in [0, K]$, $k < k'$ implies $j(k) \leq j(k')$, for the order $\epsilon < \bot < \top$; also, any $(q_{\mathcal{U}}, q_{\mathcal{U}'}, q_{\mathcal{H}}, q_\top)$ is in Q_{use}.

Consider $\mathcal{B}$ acting on a tree t. The idea is that Eve selects a run of $\mathcal{U}$ on t and Adam selects a run of $\mathcal{U}'$ on t (technically this is done one move at a time). The B-actions output are exactly those from the run of $\mathcal{U}$ chosen by Eve. However, at the same time, the transducer $\mathcal{H}$ is simulated by Adam on the composed BS-actions, and this output is analysed for the presence of blocks. The reason why Adam is in charge of the nondeterministic choices of the transducer $\mathcal{H}$ is that these choices aim at maximizing the S-values (Theorem 3.17 ensures that Adam cannot adversely affect the B-value by his choice of the run of $\mathcal{H}$). Adam also selects a branch in the tree. This means that each play in $\mathcal{B} \times t$ describes a branch of the binary tree t, a run of $\mathcal{U}$

and a run of $\mathcal{U}'$ on this branch, and Adam's choices in the simulation of $\mathcal{H}$ on the output from these runs.

A subpath π of a play is called a *block* if the following hold:

- a state from $F_B^{\mathcal{U}}$ occurs after a state from $F_S^{\mathcal{U}'}$ during π;
- for every counter $\gamma \in \Gamma_B^{\mathcal{H}}$, if there is an increment for γ in π, there is also a reset for the same γ in π.

We will call a *block of level k* a block that occurs in an $\{\varepsilon, \mathtt{r}\}$-sequence of B-counter $k{+}1$ (a block of level K is just a normal block). Given a state $(q_{\mathcal{U}}, q_{\mathcal{U}'}, q_{\mathcal{H}}, (i, j, z)) \in Q$ the components (i, j, z) track information about blocks at each level. For $k \in [0, K]$,

- $i(k) \in [1, m]$ is the number of the current block of level k;
- $j(k)$ is ϵ if no $F_S^{\mathcal{U}'}$ state has been seen in the current block of level k, $\bot$ if $F_S^{\mathcal{U}'}$ has been seen but no $F_B^{\mathcal{U}}$ following it, and $\top$ if $F_B^{\mathcal{U}}$ has been seen following $F_S^{\mathcal{U}'}$;
- $z(k)$ is $\mathtt{ic}$ if there was an increment for counter k or lower but no reset for counter k in the current block of level k, $\mathtt{r}$ if there was a reset for counter k in the block, and ε otherwise.

Notice that B-counters are indexed $[1, K]$ but the block levels are taken from $[0, K]$, so $z(0)$ is never $\mathtt{ic}$. A block of level k can be closed when an $F_S^{\mathcal{U}'}$ and then an $F_B^{\mathcal{U}}$ have been seen (so $j(k) = \top$), and if an $\mathtt{ic}$ for counter k has been seen, then an $\mathtt{r}$ has also been seen (so $z(k) \in \{\varepsilon, \mathtt{r}\}$).

The initial state is $(q_0^{\mathcal{U}}, q_0^{\mathcal{U}'}, q_0^{\mathcal{H}}, (i_1, j_\epsilon, z_\varepsilon))$ where $i_1(k) = 1$, $j_\epsilon(k) = \epsilon$ and $z_\varepsilon(k) = \varepsilon$ for all $k \in [0, K]$. Table 6.3 describes the sequence of update rules performed by $\mathcal{B}$ to determine in a deterministic way the new vectors i, j, z depending on the counter actions and states coming from the choices of Eve and Adam. In these update rules, *restarting level k* means that for all $k' \leq k$, $i(k')$ is set to 1, $j(k')$ to ϵ, and $z(k')$ to ε (this represents restarting the block counting for level k and below because an increment has been seen for counter $k+1$). Note that the counter actions referenced in the update rules come from the transducer $\mathcal{H}$ and are BS-hierarchical.

A state of the form $(q_{\mathcal{U}}, q_{\mathcal{U}'}, q_{\mathcal{H}}, (i, j, z))$ is accepting if $\mathcal{H}$ is in its rejecting sink state or $j(k) \neq \bot$ for all k. Any state $(q_{\mathcal{U}}, q_{\mathcal{U}'}, q_{\mathcal{H}}, q_\top)$ is accepting.

It is straightforward to verify that according to these updates, all reachable states are in Q_{use}.

1. If a state from $F_B^{\mathcal{U}}$ is seen, all $j(k) = \bot$ are updated to $\top$.
2. If a state from $F_S^{\mathcal{U}'}$ is seen, all $j(k) = \epsilon$ are updated to $\bot$.
3. If action `r` is seen for counter k (with ε for $k' > k$ and `r` for $k' \leq k$), then the automaton sets $z(k)$ to `r` for all $k' \leq k$.
4. If action `ic` is seen for counter k (with ε for $k' > k$ and `r` for $k' < k$), then the automaton restarts level $k - 1$. Moreover, for all $k' \geq k$ such that $z(k') = \varepsilon$, $z(k')$ is set to `ic`.
5. After updates 1 to 4, if there is a k with $j(k) = \top$, $z(k) \in \{\varepsilon, \texttt{r}\}$ and $i(k) = l$, then for all $k' \leq k$, $i(k')$ is set to $l + 1$, $j(k')$ is set to ϵ, and $z(k')$ is set to ε.
6. If some $i(k)$ is supposed to increase by rule 5 but is already at m, then the automaton moves to $q_\top$.

Table 6.3. Update rules for (i, j, z) components of $\mathcal{B}$.

Example 6.11. We give an example showing how the vectors (i, j, z) would be updated given a sequence of states and counter actions. We assume there is only one B-counter, so each vector has two components. We also assume $q_B \in F_B^{\mathcal{U}}$ and $q_S \in F_S^{\mathcal{U}'}$. Each column shows the values of (i, j, z) after the action in the column header is read and update rules 1–6 have been applied to the vectors described by the previous column.

	init	q_S	`r`	q_B	q_B	q_S	`ic`	q_S	q_B	`r`
$i(0)$	1	1	1	2	2	2	1	1	2	3
$j(0)$	ϵ	$\bot$	$\bot$	ϵ	ϵ	$\bot$	ϵ	$\bot$	ϵ	ϵ
$z(0)$	ε	ε	`r`	ε	ε	ε	ε	ε	ε	ε
$i(1)$	1	1	1	2	2	2	2	2	2	3
$j(1)$	ϵ	$\bot$	$\bot$	ϵ	ϵ	$\bot$	$\bot$	$\bot$	$\top$	ϵ
$z(1)$	ε	ε	`r`	ε	ε	ε	`ic`	`ic`	`ic`	ε

Note that (i, j, z) would usually be updated based on both the state and counter action at each position in the run (it would not be separated as in this example).

6.2.3 Technical proofs

We now proceed with the technical proofs showing correctness and quasi-weakness of $\mathcal{B}$. The proof of correctness utilizes Proposition 6.10 (which is given at the end of the section).

Proof of correctness

We break the proof that $[\![\mathcal{B}]\!]_B \approx [\![\mathcal{U}]\!]_B$ into two directions. The easier direction is captured in the following lemma.

Lemma 6.12. $[\![\mathcal{B}]\!]_B \leq [\![\mathcal{U}]\!]_B$.

Proof. Let t be a tree and $M \in \mathbb{N}$ such that $[\![\mathcal{U}]\!]_B(t) = M$. We want to show $[\![\mathcal{B}]\!]_B(t) \leq M$. Let R be a B-accepting run of $\mathcal{U}$ on t of value M.

Let σ be the strategy of Eve in the game $\mathcal{B} \times t$ that consists in playing from the run R. We have $\text{value}_B(\sigma) = \text{value}_B(R) = M$ (the value according to the B-counter actions, not considering the accepting states visited), since $\mathcal{B}$ just copies the B-actions of $\mathcal{U}$.

It remains to show that the strategy σ actually satisfies the acceptance condition. Assume for the sake of contradiction that there is a play $\pi \in \sigma$ that stabilizes in rejecting states. Recall that each rejecting state is of the form $(q_{\mathcal{U}}, q_{\mathcal{U}'}, q_{\mathcal{H}}, (i, j, z))$ where there is some k' such that $j(k') = \bot$. Let k be the maximum such that there is $j(k) = \bot$ infinitely often in π.

We show that after some point, $j(k)$ must stabilize in $\bot$. If k is the highest counter, then level k can never be restarted, so rule 1, 5, and 2 (which could result in $j(k)$ cycling between $\bot$, $\top$, and ϵ) can only be applied m times. Hence, $j(k)$ stabilizes in $\bot$. Otherwise, assume $j(k)$ does not stabilize in $\bot$. There must be infinitely many updates of $j(k)$ from ϵ to $\bot$, and infinitely many restarts of level k. Let $k' > k$. By choice of k, $j(k')$ must stabilize in ϵ or $\top$ (otherwise, $j(k')$ is $\bot$ infinitely often, contradicting the choice of k). If $j(k')$ stabilizes in ϵ, every update of $j(k)$ from ϵ to $\bot$ is simultaneously done on $j(k')$ by update rule 2 so this is absurd. If $j(k')$ stabilizes in $\top$, then the infinitely many restarts of level k must come from infinitely many `ic` for counter k' (since if level k' was also restarted infinitely many times, $j(k')$ cannot stabilize in $\top$). Since the B-value of any play is at most M, this means there must be infinitely many resets of counter k'. But this is absurd, since by update rule 3, action `r` for counter k' would result in $j(k') = \top$ and $z(k') = \texttt{r}$, which results in an increment of $i(k')$ and sets $j(k')$ back to ϵ (or m is reached and the automaton moves to $q_\top$).

Hence $j(k)$ stabilizes in $\bot$. But this is absurd, because there are infinitely many accepting states $F_B^{\mathcal{U}}$ of $\mathcal{U}$ on π, so update rule 1 implies that $j(k)$ cannot stabilize in $\bot$ on π.

From this, we can conclude that $\text{value}(\sigma) = M$, so $[\![\mathcal{B}]\!]_B \leq [\![\mathcal{U}]\!]_B$. □

The other direction is the more challenging part of the proof of correctness, and requires the notion of a cost trap and Proposition 6.10.

Lemma 6.13. $[\![\mathcal{U}]\!]_B \preccurlyeq [\![\mathcal{B}]\!]_B$.

Proof. Assume for the sake of contradiction that $[\![\mathcal{U}]\!]_B \not\preccurlyeq [\![\mathcal{B}]\!]_B$, i.e. there exists a set U such that $\{[\![\mathcal{B}]\!]_B(t) : t \in U\}$ is bounded by some $M \in \mathbb{N}$, but $\{[\![\mathcal{U}]\!]_B(t) : t \in U\}$ is unbounded. Note that since $[\![\mathcal{U}]\!]_B \approx [\![\mathcal{U}']\!]_S$, we have $\{[\![\mathcal{U}']\!]_S(t) : t \in U\}$ also unbounded.

By Theorem 3.17, $\mathcal{H} \cong_\alpha^\beta \mathcal{I}$ (recall $\mathcal{H}$ outputs hierarchical actions, but $\mathcal{I}$ outputs the actions unchanged). This means that $[\![\mathcal{I}]\!]_S^B(M) \preccurlyeq_{\beta_M} [\![\mathcal{H}]\!]_S^B(\alpha(M))$. Let $(\vartheta_n)_{n\in\mathbb{N}}$ be the translation strategies witnessing the history determinism of $\mathcal{H}$. Let $\mathcal{A}$ be the nondeterministic BS-Büchi automaton $\mathcal{H} \circ (\mathcal{U} \times \mathcal{U}')$ from Section 3.4, which satisfies $\mathcal{A} \cong_\alpha^\beta \mathcal{U} \times \mathcal{U}'$ by Lemma 3.18.

Let σ be an accepting strategy for Eve in $\mathcal{B} \times t$, witnessing value$(\sigma) \leq M$. We look in particular at the set P of plays compatible with σ where Adam chooses to play from a run R' of $\mathcal{U}'$ on some $t \in U$ with $[\![\mathcal{U}']\!]_S(t) > N := \beta_M(|Q_\mathcal{A}| + 1)$, and uses strategy $\vartheta_{|Q_\mathcal{A}|+1}$ to drive $\mathcal{H}$. Notice that the only remaining choice for Adam concerns the branching, so P describes a binary tree of possible plays, which agree on common prefixes.

We know that in order for value$(\sigma) \leq M$, Eve must play a run R of $\mathcal{U}$ such that value$_B(R) \leq M$ on every branch of P and Adam must choose an accepting run of $\mathcal{H}$ on each branch. Let π be a branch of the binary tree described by P.

Consider $v = (\text{out}_B(R, \pi), \text{out}_S(R', \pi))$ with out$_B(R, \pi)$ (respectively, out$_S(R', \pi)$) the infinite word describing the actions output by run R (respectively, R') on the branch π. By Theorem 3.17, any run of $\mathcal{H}$ on v has value$_B(R_\mathcal{H}) \leq \alpha(M)$. Moreover, there is a run $R_\mathcal{H}^\pi$ of $\mathcal{H}$ on v driven by $\vartheta_{|Q_\mathcal{A}|+1}$ such that value$_B(R_\mathcal{H}^\pi) \leq \alpha(M)$ and $\beta_M(\text{value}_S(R_\mathcal{H}^\pi)) \geq N = \beta_M(|Q_\mathcal{A}| + 1)$, and for any π and π', $R_\mathcal{H}^\pi$ and $R_\mathcal{H}^{\pi'}$ agree on any shared prefix. This means that every play of P represents an accepting run of $\mathcal{H}$ over v and outputs hierarchical BS-actions with S-value at least $|Q_\mathcal{A}|+1$ and B-value at most M.

Next we show that $\mathcal{B}$ must witness $m = (|Q_\mathcal{A}|+2)^{|\Gamma_S^\mathcal{H}|+1}$ blocks on every play in P. These blocks will eventually be used to build a cost trap. Let π be a play in P.

If the automaton $\mathcal{B}$ reaches the accepting sink state $q_\top$ on π, it means that for some level k, m blocks have been witnessed. We assume by contradiction that it is not the case. Let k be the least level such that $i(k)$ does not change infinitely many

times in π, so it must stabilize to some value (this is always the case for $i(K)$, which can only increase).

If $i(k)$ stabilizes, then $j(k)$ must stabilize as well. It is impossible for $j(k)$ to stabilize to ϵ: R' is accepting so any of the infinitely many states from $F_S^{\mathcal{U}'}$ would make $j(k)$ change to $\perp$ by update rule 2. It also cannot stabilize in $\perp$: $\text{value}(\sigma) \leq M$ implies that every play in σ is accepting, which is not possible if $j(k)$ stabilizes in $\perp$. Hence $j(k)$ must stabilize in $\top$. This means that there is a finite number of resets for counter k, otherwise $i(k)$ would be incremented by rule 5. Since the play has bounded value, we can conclude that the number of increments for any counter $k' \geq k$ on π is also finite. But $i(k-1)$ is supposed to change infinitely many times, and this can only be done by incrementing infinitely many times higher counters (rule 4). This is absurd, so π must witness m consecutive blocks, and stabilize in $q_\top$.

Now we build a cost trap for the automaton $\mathcal{A} \cong \mathcal{U} \times \mathcal{U}'$. Recall that the transition function of $\mathcal{B}$ is defined such that Eve selects a run of $\mathcal{U}$, Adam selects a run of $\mathcal{U}'$, and Adam controls $\mathcal{H}$ which outputs the hierarchical BS-actions corresponding to these composed runs. Let $R_\mathcal{A}$ be the run of $\mathcal{A}$ on t corresponding to the tree of plays P (ignoring the components of the state dealing with the blocks). We have $\text{value}_B(R_\mathcal{A}) \leq M$ and $\text{value}_S(R_\mathcal{A}) > |Q_\mathcal{A}|$.

We showed that for any branch π of $R_\mathcal{A}$, state $q_\top$ is reached on π, so m consecutive blocks are witnessed along this branch. If π is a branch of $R_\mathcal{A}$, for all $l \in [0, m]$, we take for e_l^π the last position of π where some level k has $i(k) = l$. By the definition of the transitions of $\mathcal{B}$ (rules 1 to 6), the path $[e_l^\pi, e_{l+1}^\pi)$ is always a block.

It remains to show the nesting condition of the cost trap definition. Let π_1, π_2 be two branches of $R_\mathcal{A}$, sharing some prefix up to position y, and let $x < y$ be the first position where they disagree: $e_l^{\pi_1} = x$ but $e_l^{\pi_2} > x$. By definition of the e_l^π's, we have that x is the last position on π_1 where some $i(k)$ has value l. Assume $e_l^{\pi_2} \leq y$, then $e_l^{\pi_2}$ is on π_1, but some $i(k)$ has value l, which is absurd since $e_l^{\pi_2} > x$. We can conclude that $e_l^{\pi_2} > y$, so the nesting condition is satisfied.

We have completed the proof that $R_\mathcal{A}$ is a cost trap for $\mathcal{A}$. By Proposition 6.10, this implies $[\![\mathcal{U}']\!]_S \not\preccurlyeq [\![\mathcal{U}]\!]_B$, which is a contradiction. □

Quasi-weakness of $\mathcal{B}$

Next, we show that $\mathcal{B}$ is a B-quasi-weak automaton.

Fix some t and assume there is a strategy σ in $\mathcal{B} \times t$ with $\text{value}(\sigma) \leq n$. Let $\pi \in \sigma$. We define a *configuration* C of the automaton $\mathcal{B}$ to be an element of the set $([0, n]^{[0,K]} \times I \times J \times Z) \cup \{q_\top\}$ where I, J, and Z are as defined in the construction.

For a configuration of the form (v, i, j, z), the i, j, z components are as before. The new element v stores the value of B-counter k for all $k \in [1, K]$. Since level 0 does not correspond to a counter, we fix the value of $v(0)$ to be always 0. Note that for the other components of v, the set of values $[0, n]$ is sufficient because counter values never go above n in π if value$(\sigma) \leq n$.

We now analyse the evolution of C on the path π, according to the update rules 1 to 6. An *alternation* is a switch from accepting states to rejecting states (or vice-versa). We show that if a subpath of π begins and ends in the same configuration C, then it is alternation-free. We call such a subpath a *configuration cycle* to distinguish it from a true cycle based on the state of the automaton.

Lemma 6.14. *If a subpath of π begins and ends in the same configuration C, then it is alternation-free.*

Proof. Fix some configuration cycle. Notice that the update rules 1–3 only increase $j(k)$ and $z(k)$ for all k (for the ordering $\epsilon < \bot < \top$ and $\varepsilon < \texttt{ic} < \texttt{r}$), while update rules 4 and 5 can decrease these components.

If no $i(k)$ is modified during the configuration cycle, then update rule 4 cannot have been applied, so the configuration cycle must contain only one configuration, and hence is alternation-free.

Otherwise let k be maximal such that $i(k)$ changes during the configuration cycle. In order to return to the same configuration, the configuration cycle must restart level k. By rule 4, this means some `ic` is seen for counter $k' > k$. Level k' is never restarted (otherwise k would not be maximal), so according to rules 1 and 2, $j(k')$ can only increase during the configuration cycle. Since the configuration C is the same in the beginning and in the end, we get that $j(k')$ must not change during the whole configuration cycle.

We consider the possible cases for $j(k')$. If $j(k')$ is $\bot$, then the whole configuration cycle is non-accepting, so there is no alternation.

We know that $v(k')$ is incremented at some point by action `ic` for counter k', so counter k' has to be reset in the configuration cycle in order to match the original value. Thus if the value of $j(k')$ during the configuration cycle is $\top$, it means that we will have $z(k') = \texttt{r}$ and $j(k') = \top$ at some point in the configuration cycle. Rule 5 implies that $i(k')$ would then be incremented, which is absurd because $k' > k$ implies that $i(k')$ does not change during the configuration cycle (by choice of k).

The only remaining case is $j(k') = \epsilon$ during the entire configuration cycle. By definition of Q_{use}, for all $k'' \leq k'$, $j(k'') = \epsilon$ during the whole configuration cycle.

Moreover, for $k'' > k'$, $j(k'')$ can only increase (since levels k' and above are never restarted), so in order to return to the initial configuration, these components cannot change at all. Since acceptance is determined by the $j(k'')$ for $k'' > k'$ and these components never change, the configuration cycle is alternation-free. □

Any configuration cycle is alternation-free, so the number of alternations in π is bounded by the number of configurations, which is at most $((n+1)(m+1)9)^{K+1}+1$. Hence the length of an alternating chain in a play π in $\mathcal{B} \times t$ with value$(\pi) \leq n$ is bounded by $\alpha(n) = ((n+1)(m+1)9)^{K+1}+1$, where K and m are fixed by $\mathcal{U}$ and $\mathcal{U}'$ and do not depend on the input tree.

This is enough to conclude that $\mathcal{B}$ is a B-quasi-weak alternating automaton.

Proof of Proposition 6.10

We start by assuming that there is a cost trap for some BS-Büchi automaton $\mathcal{A} = \langle Q_\mathcal{A}, \mathbb{A}, q_0^\mathcal{A}, \Gamma_B, F_B, \Gamma_S, F_S, \delta_\mathcal{A} \rangle$ such that $\mathcal{A} \cong \mathcal{U} \times \mathcal{U}'$. We aim to show that $[\![\mathcal{U}']\!] \not\leq [\![\mathcal{U}]\!]$.

Let $m := (|Q_\mathcal{A}|+2)^{|\Gamma_S|+1}$. By the definition of cost trap, there is a frontier E_m, a set of nodes $\{e_i^\pi : 0 \leq i \leq m \text{ and } \pi \in [0,1]^\omega\}$, an input tree, and a run R of $\mathcal{A}$ on t such that value$_S(R) > |Q_\mathcal{A}|$, and for all branches π, $e_0^\pi < \cdots < e_m^\pi \in E_m$ are nodes of π such that for all $0 \leq i < m$ the portion of R corresponding to $[e_i^\pi, e_{i+1}^\pi)$ is a block, i.e. contains states from both F_B and F_S, and for all $\gamma \in \Gamma_B$, if γ is incremented in the block then it is also reset in the block.

Moreover, if branches π_1 and π_2 share some prefix up to position y and $x < y$ is the first position with $e_i^{\pi_1} = x$ and $e_i^{\pi_2} \neq x$ then $e_i^{\pi_2} > y$ (the nesting condition). In this case we will say that $\pi_1 <_{\text{nest}} \pi_2$ and y is a *nesting node*. If π_1 and π_2 do not disagree on the e_i's on their common prefix, we will say that $\pi_1 \approx_{\text{nest}} \pi_2$. The idea is that if $\pi_1 <_{\text{nest}} \pi_2$, then pumping the blocks from π_2 will not damage any of the blocks from π_1.

The first step will be to build a single tree t' such that there is some n with $[\![\mathcal{A}]\!]_B(t') \leq n < \infty$, and also $[\![\mathcal{A}]\!]_S^B(n)(t') > |Q_\mathcal{A}|$.

Lemma 6.15. *There exists a tree t' such that there exists $n \in \mathbb{N}$ with $[\![\mathcal{A}]\!]_B(t') \leq n$ and $[\![\mathcal{A}]\!]_S^B(n)(t') > |Q_\mathcal{A}|$.*

Proof. Let t_m be the finite tree obtained from t by removing all nodes after the frontier E_m. The tree t' will be obtained from t_m by an infinite pumping of some blocks on every branch.

Let G be the set of all branches of t_m, a finite set that is partially ordered by $<_{\text{nest}}$. We inductively build sets $(G_j)_{j \in \mathbb{N}}$ in the following way: for all j, G_j is the set of

branches of t_m that are maximal for $<_{\text{nest}}$ in $G \setminus (\bigcup_{j'<j} G_{j'})$. Let $p \in \mathbb{N}$ be such that $G_p \neq \emptyset$ and for all $j > p$, $G_j = \emptyset$. Then $G_1, \ldots, G_p$ form a partition of G, and for all $j \in [1, p]$ and for all $\pi_1, \pi_2 \in G_j$, we have $\pi_1 \approx_{\text{nest}} \pi_2$. Notice that for all $j \in [1, p]$ and all $i \in [1, m]$, the set $\{e_i^\pi : \pi \in G_j\}$ is a partial frontier of t_m (more precisely, it is the intersection of a frontier of t_m with G_j).

We can now adapt the technique of Rabin [Rab70], but applying it only on the branches of G_1 (for now). Let $N := |Q_{\mathcal{A}}|$. In the following, S-values will be approximated by assigning value $N + 1$ for any value larger than N. Unlike [Rab70], when we pump we must be careful that we do not reduce the S-value below N.

We define sets $H_m, \ldots, H_0$ by induction: we set $H_m := Q_{\mathcal{A}} \times [0, N + 1]^{\Gamma_S}$, and $(q, \eta) \in H_{i-1}$ if $(q, \eta) \in H_i$ and there is a finite tree t_{f} contained in t_m and a partial run R_{f} of $\mathcal{A}$ over t_{f} starting from state q and valuation η for the S-counters, such that

- every branch of R_{f} is a block,
- on every leaf of t_{f}, the state and S-counters values configuration (q', η') of $\mathcal{A}$ belongs to H_i.

Notice that some nodes of t_{f} are allowed to have only arity 1 (the nodes corresponding to nesting nodes of t_m). The idea is that when starting from a node in H_{i-1} (which is also in H_i), we can reach a H_i-frontier with blocks.

We have $H_0 \subseteq H_1 \subseteq \cdots \subseteq H_m$, and moreover, the run R on branches G_1 witnesses the fact that for all branches $\pi \in G_1$ and for all $i \in [0, m]$, the configuration (q, η) of $\mathcal{A}$ in the run R at node e_i^π always belongs to H_i. Moreover, there is a finite tree t_0 such that $\mathcal{A}$ has a run over t_0 starting in $(q_0, 0)$ and ending in a configuration belonging to H_0 on every leaf (t_0 is a prefix of t_m, and this run is a prefix of the run R).

But $|H_m| = N \cdot (N + 2)^{|\Gamma_S|} \leq m$, so there must be $i \in [0, m - 1]$ such that $H_i = H_{i+1}$. It means that from any configuration of H_i, we can find a finite tree t_{f} contained in t_m, and a partial run of $\mathcal{A}$ over t_{f} (contained in R), such that any branch of this run is a block, and the configuration on any leaf is again in H_i. This will allow us to construct an infinite tree and a run with infinitely many H_i-frontiers by pumping.

Pumping cannot result in a run with S-value less than N since no block from R can check a S-value less than N and the value of the S-counter is stored as part of this configuration (up to value N).

Pumping can distort the B-value slightly. The definition of block includes a constraint on B-actions (for all B-counters γ, if a block increments γ, then it also resets γ) which ensures that pumping does not result in an unbounded B-value.

However, when appending two blocks, the worst case that can happen is when the first block starts with a reset and then make some increments, and the second block makes some increments and then resets. The maximum value resulting from this type of error is $n_{\text{blocks}} := \max\{\text{value}_B(uv) : u, v \text{ are blocks in } R \text{ from } H_i \text{ to } H_i\}$, the maximal B-value obtained by appending two blocks (appending more than two blocks cannot increase this value further). We also define n_{pref} to be the B-value of the partial run reaching the first H_i-frontier from the root in the run R. Thus, the B-value after pumping is at most $n_1 := n_{\text{blocks}} + n_{\text{pref}}$.

This allows us to build an infinite (but not complete) tree t_1, and a partial run R_1 of $\mathcal{A}$ over t_1, by using R to reach the H_i-frontier (nodes e_i^π for $\pi \in G_1$), and then repeating infinitely many finite trees witnessing partial runs from H_i-nodes to H_i-frontiers. Note that every infinite play π of R_1 is B-accepting with $\text{value}_B(\pi) \leq n_1$, and S-accepting with $\text{value}_S(\pi) > N$, by the block condition and the fact that no S-value less than N is checked during R.

The tree t_1 also contains infinitely many finite branches, which are duplicated from $G_2, \ldots, G_p$. Let us call G_2' the infinite set of finite branches in t_1 coming from G_2. The nesting condition implies that pumping branches of G_1 did not split blocks of the other G_i's, so we can now define the sets $H_0, \ldots, H_m$ as before, and pump all the branches of G_2' simultaneously, building a tree t_2 satisfying the same properties as t_1, but with additional infinite branches.

Iterating this process p times yields a complete binary infinite tree $t' := t_p$, and some $n := n_1 + n_2 + \cdots + n_p$ (only depending on the finite set of blocks from the run R on t_m) such that there is a run R' of $\mathcal{A}$ on t' witnessing both $[\![\mathcal{A}]\!]_B(t') \leq n$, and $[\![\mathcal{A}]\!]_S^B(n)(t') > N$ (i.e. the run R' is B-accepting with B-value less than n, and S-accepting with S-value strictly above N). □

We will now build from t' an infinite sequence of infinite trees that will witness bounded B-value but unbounded S-value.

Lemma 6.16. *There is an infinite sequence of infinite trees $(t'_s)_{s\in\mathbb{N}}$, such that for all $s \in \mathbb{N}$, $[\![\mathcal{A}]\!]_B(t'_s) \leq n$, and $[\![\mathcal{A}]\!]_S^B(n)(t'_s) > |Q_\mathcal{A}| + s$.*

Proof. Let $N := |Q_\mathcal{A}|$ as in the previous lemma. Consider the run R' on t' from Lemma 6.15. In order for R' to have S-value more than N, any action `cr` for $\gamma \in \Gamma_S$ must occur when the value of γ is at least $N + 1$, so the `cr` must be immediately preceded by a sequence of at least $N + 1$ increments of γ. This path may contain ε but cannot contain any `r` or `cr` for γ.

By the choice of N, there must be positions x and y on this path that both output action `i` for γ and correspond to the same state $q \in Q_{\mathcal{A}}$, and such that there are no `r` or `cr` for γ between x and y. Let u_γ be the subpath $[x, y]$.

Recall that we are using hierarchical BS-actions. Consider the behaviour of the other counters during u_γ.

- If γ' is a B- or S-counter higher than γ in the hierarchy, any non-ε action for γ' resets γ, which is not possible in u_γ. Hence u_γ contains only action ε for $\gamma' > \gamma$.
- If γ' is a B- or S-counter lower than γ, then u_γ starts with action `r` for γ'.

This means that if there is some $\gamma' < \gamma$, if $v_{\gamma'}$ (based on some other path) intersects u_γ, then $v_{\gamma'}$ must occur entirely within u_γ (because increments for γ result in resets for γ'). If $\gamma = \gamma'$ and u_γ and $v_{\gamma'}$ intersect, then iterating one could increase the value of the other, but cannot change the properties we are interested in: it would still result in a path from some $q \in Q_{\mathcal{A}}$ to q, starting with an action `i` for γ and without any reset for γ. In any case, repeating u_γ several times does not adversely affect the values of other B- and S-counters, or prevent iterating some other $v_{\gamma'}$.

Let t'_s be the infinite tree obtained from t' where for all S-counters γ and for all position z where action `cr` on γ is done in R', the path u_γ relative to this position z is repeated s times. The run R' induces a run R'_s for each t'_s that is still B-accepting and S-accepting (accepting states still appear infinitely often since t' was B-accepting and S-accepting and we are only iterating parts of the tree finitely many times), has B-value less than n, but where at least s increments have been added before each action `cr` of any counter, so $\text{value}_S(R_s) > N + s$. □

We are now ready to observe the contradiction required to prove Proposition 6.10. Let α and $(\beta_i)_{i\in\mathbb{N}}$ be corrections functions witnessing $\mathcal{A} \cong \mathcal{U} \times \mathcal{U}'$. In particular $[\![\mathcal{A}]\!]_B \approx_\alpha [\![\mathcal{U} \times \mathcal{U}']\!]_B = [\![\mathcal{U}]\!]_B$ ($\mathcal{U}'$ does not play any role in the B-semantics of $\mathcal{U} \times \mathcal{U}'$). This implies that $[\![\mathcal{U}]\!]_B(t'_s) \leq \alpha(n)$ for any t'_s from Lemma 6.16.

But we also have $[\![\mathcal{A}]\!]^B_S(n) \preccurlyeq_{\beta_n} [\![\mathcal{U}\times\mathcal{U}']\!]^B_S(\alpha(n))$, so $N+s \leq \beta_n([\![\mathcal{U}\times\mathcal{U}']\!]^B_S(\alpha(n))(t'_s))$ for all $s \in \mathbb{N}$. But $[\![\mathcal{U} \times \mathcal{U}']\!]^B_S(\alpha(n))(t'_s)$ is

$$\sup\left\{\text{value}_S(R'') : \begin{array}{l} R'' \text{ is an } S\text{-accepting run of } \mathcal{U}\times\mathcal{U}' \text{ on } t'_s \\ \text{with } \text{value}_B(R'') \leq \alpha(n) \end{array}\right\}.$$

Since we know that $[\![\mathcal{U} \times \mathcal{U}']\!]_B(t'_s) \leq \alpha(n)$, there must be an S-accepting run R'' of $\mathcal{U}\times\mathcal{U}'$ on t'_s witnessing $\beta_n([\![\mathcal{U}\times\mathcal{U}']\!]^B_S(t'_s)) \geq N+s$. This means $\beta_n(\text{value}_S(R'')) \geq N+s$. But R'' induces an S-accepting run $R_{\mathcal{U}'}$ of $\mathcal{U}'$ over t'_s with $\beta_n(\text{value}_S(R_{\mathcal{U}'})) \geq N + s$.

Therefore, for all $s \in \mathbb{N}$, $\beta_n([\![\mathcal{U}']\!]_S(t'_s)) \geq N + s$. This means $[\![\mathcal{U}']\!]_S$ is unbounded over the set $\{t'_s : s \in \mathbb{N}\}$ while $[\![\mathcal{U}]\!]_B$ is bounded over the same set, which shows that a cost trap implies a contradiction with $[\![\mathcal{U}']\!]_S \preccurlyeq [\![\mathcal{U}]\!]_B$.

6.3 Discussion

One of the goals in the theory of regular cost functions introduced by Colcombet in [Col09c] was to develop a theory that retained the robust equivalences, closure properties, and decidability of the classical theory of regular languages. As described in Chapter 2, this goal was achieved over finite words and finite trees.

Chapter 5 extended the theory to weak cost functions over infinite trees. The correspondence between weak cost automata and cost WMSO, and the decidability result that follows from this, parallels the classical theory nicely. The new quasi-weak automata that arise in this setting, and the fact that quasi-weak cost automata, rather than weak cost automata, admit the Rabin-style characterization is an interesting divergence from the classical theory. Figure 6.4 (courtesy of Kuperberg) compares the classical theory over trees compared to the known results in the cost setting.

A major difference between the classical picture and the cost version shown in Figure 6.4 is that the relationship between the full cost MSO logic and arbitrary cost-parity automata (and the decidability of this class) is not known. A major open question is to prove the equivalence of cost MSO and cost-parity automata, and derive a decidability result for the corresponding cost functions. As explained in Chapter 4, this comes down to showing that finite memory strategies suffice in the cost-parity games that come out of cost-parity automata acting on infinite trees.

If we restrict to cost functions over infinite words rather than infinite trees, however, the correspondence with classical results holds again. For instance, Kuperberg and the author have recently shown that the classical equivalence of WMSO and MSO over infinite words holds in the cost setting [KVB12]. Moreover, over infinite words, cost functions definable by B-quasi-weak, B/S-weak, and nondeterministic B/S-Büchi automata are actually equivalent and capture all regular cost functions over infinite words. This implies that the domination preorder is decidable for all regular cost functions over infinite words (although this result was already known by Colcombet [Col12b] using algebraic methods).

In fact, additional classical results can be lifted to the cost setting over infinite words. It turns out that a cost version of first-order logic (FO), linear temporal logic (LTL), and very-weak automata (weak automata with a further restriction that any

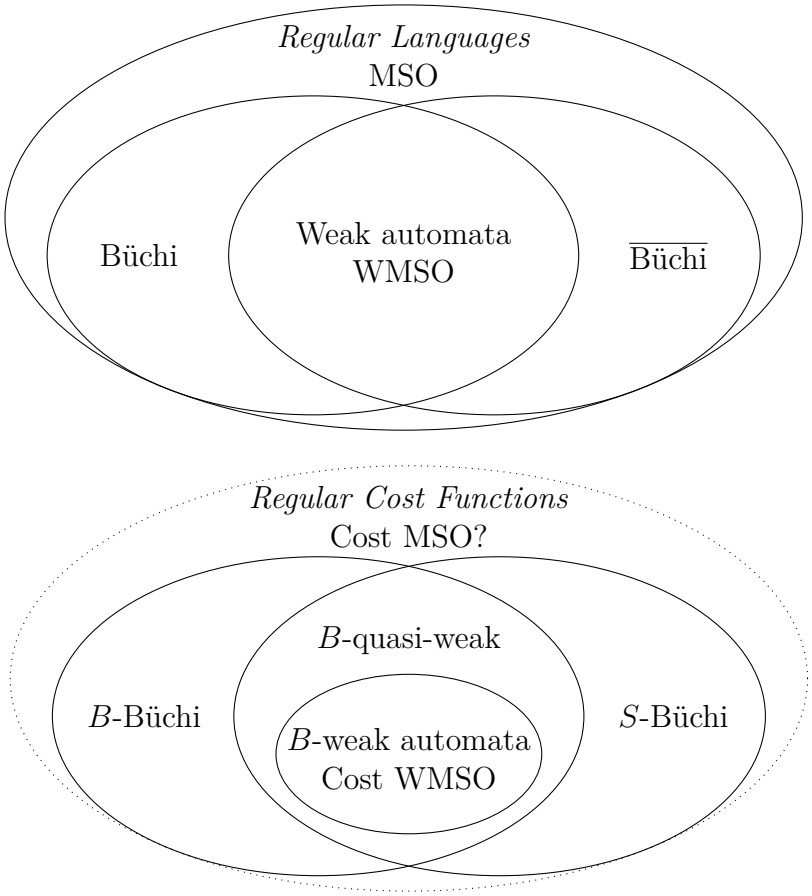

Figure 6.4. Comparison of the theory of regular languages and the theory of weak cost functions over infinite trees.

cycle consists of only one state) are equivalent, which parallels well-known classical results. We refer the interested reader to [KVB12] for details. A comparison of the classical theory compared to the cost theory over infinite words is given in Figure 6.5.

These results over infinite words and infinite trees give additional evidence that the theory of regular cost functions is a robust quantitative extension to the theory of regular languages.

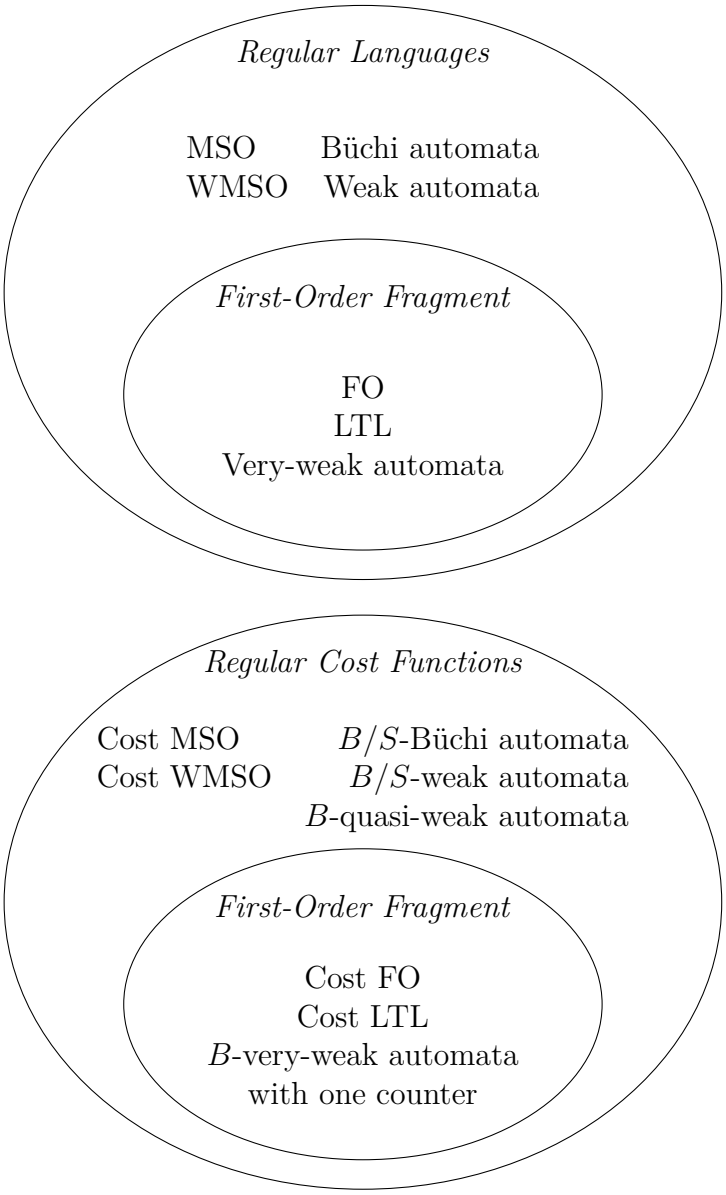

Figure 6.5. Comparison of the theory of regular languages and the theory of regular cost functions over infinite words (see [KVB12]).

Chapter 7

Application to Parity Index Problem

As mentioned in Chapter 1, one of the earliest motivations for studying automata with counters came from problems in formal language theory, such as the star height problem. In this chapter, we describe two problems from language theory called the parity index problem (Section 7.1) and weak definability problem (Section 7.2). After reviewing the known decidability results, we apply the results on cost automata over infinite trees developed in this thesis to decide special cases of these problems that were previously open. The result on the special case of the weak definability problem in Section 7.2 is based on joint work with Kuperberg [KVB11].

7.1 Parity index problem

The range of priorities $[i, j]$ used in a parity automaton is a useful measure of the complexity of the automaton, even more so than the number of states. Indeed, all known algorithms for solving parity games (and hence deciding the emptiness problem for parity automata) are essentially exponential in the number of priorities but only polynomial in the number of the states [Jur00].

Of course, if a language is definable using an $[i, j]$ automaton (a parity automaton using only priorities in $[i, j]$), then the language is definable using an automaton with a larger range of priorities, i.e. an $[i', j']$ automaton with $j' - i' > j - i$. Likewise, if the priorities used by a parity automaton are shifted by steps of 2 or -2, then the language defined by the automaton is unaffected. Hence, we can restrict attention to $[i, j]$ with $i \in [0, 1]$. We call $[i, j]$ the *parity index* of the automaton.

We can consider the hierarchy of indices of parity automata, and the languages expressible using automata of a given index. This is known as the *Mostowski hierarchy* or *Mostowski-Rabin hierarchy* after the first researchers who studied automata

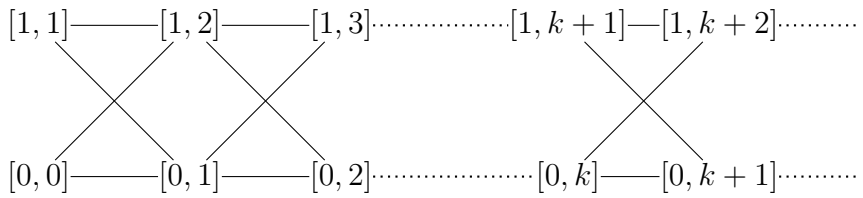

Figure 7.1. The Mostowski hierarchy of parity indices.

using a parity acceptance condition (see Figure 7.1). This hierarchy comes in several flavours depending on the structure of the transition relation used by the automata, i.e. whether the automata are deterministic, nondeterministic, or alternating. We will refer to, e.g. the nondeterministic Mostowski hierarchy, if we want to make this clear.

Over infinite words, the nondeterministic hierarchy collapses to the $[1, 2]$ level since every regular language of infinite words is expressible using a nondeterministic Büchi automaton (see Theorem 3.1 and [Tho97]). Likewise, the alternating hierarchy for automata over infinite words collapses to the intersection of the $[0, 1]$ and $[1, 2]$ levels. This follows from the fact that regular languages are closed under complement [Büc60], and an alternating co-Büchi automaton for a regular language L can be obtained from the nondeterministic Büchi automaton for $\overline{L}$ by switching the conjunctions and disjunctions in the transition relation and replacing priority $p \in [1, 2]$ with $p - 1 \in [0, 1]$.

Over infinite trees, however, the Mostowski hierarchy is strict for deterministic[1] [Wag77] (as cited in [NW03]), nondeterministic [Niw86], and alternating models [Bra98, Arn99].

Given a regular language of infinite trees in the form of an arbitrary parity automaton for the language, it is helpful to know if there is a simpler (lower index) automaton defining the same language. The *parity index problem* is:

> Given an arbitrary parity automaton $\mathcal{A}$ and a range of priorities $[i, j]$, is there an $[i, j]$ automaton $\mathcal{A}_{[i,j]}$ such that $L(\mathcal{A}) = L(\mathcal{A}_{[i,j]})$?

7.1.1 Known decidability results

The status of the parity index problem over trees depends on which version of the hierarchy is being used (i.e. whether $\mathcal{A}_{[i,j]}$ is required to be deterministic, nondeterministic, or alternating). It also depends on the types of languages allowed as input to the problem. We refer the reader to [NW05] for a nice introduction to this problem.

[1] In fact, the deterministic hierarchy is strict even over words.

The lowest levels $[0,0]$ and $[1,1]$ of the nondeterministic and alternating hierarchies coincide and are known to be decidable. Level $[0,0]$ consists of the regular languages of infinite trees that are closed under the standard prefix topology [JL02] (building on earlier work in [Mos91]). This means that membership in the $[0,0]$ level can be decided by constructing the closure $\mathcal{A}_{\mathrm{cl}}$ of $\mathcal{A}$ and testing whether $L(\mathcal{A}) \cap \overline{L(\mathcal{A}_{\mathrm{cl}})} = \emptyset$ (see [Löd09, Theorem 5.1]). Level $[1,1]$ consists of regular languages of finite trees (since no infinite structure can be accepted when using only priority 1). Deciding this level can be reduced to determining whether $L(\mathcal{A}) \cap L_{\mathrm{inf}} = \emptyset$ where L_{inf} is the set of all infinite trees (a regular language). As described in Section 4.4, this is decidable.

Decidability of the parity index problem has also been established when the input automaton $\mathcal{A}$ is deterministic. Deterministic tree languages form a proper, decidable subclass of all regular tree languages [NW05]. Unlike nondeterministic automata, it is natural to identify a deterministic tree automaton $\mathcal{A}$ over alphabet $\mathbb{A}$ with a deterministic word automaton $\mathcal{A}_{\mathrm{w}}$ over $\mathbb{A} \times [0,1]$ such that

$$t \in L(\mathcal{A}) \text{ iff for all branches } \pi,\ \pi \in L(\mathcal{A}_{\mathrm{w}}).$$

This has been exploited to show the decidability of the deterministic [NW98] and nondeterministic [NW05] hierarchies. The proof proceeds by reducing membership in a given level to the detection of special patterns (called *flowers*) in the graph of $\mathcal{A}_{\mathrm{w}}$. Since the search for these patterns is effective, the parity index problem is decidable. The alternating hierarchy is actually easier since all deterministic tree languages are contained in the $[0,1]$ level: there is a nondeterministic Büchi automaton for $\overline{L(\mathcal{A})}$ that guesses a branch rejected by $\mathcal{A}$, so this automaton can be dualized as described above to get an alternating co-Büchi automaton for $L(\mathcal{A})$ [Mur08a].

7.1.2 Reduction to boundedness for cost-parity automata

For the remainder of this section, we concentrate on the case of the nondeterministic hierarchy. Unfortunately, the techniques mentioned above cannot be generalized to the nondeterministic Mostowski hierarchy when the input is an alternating or nondeterministic automaton. Recently, Colcombet and Löding have suggested another route, by providing a reduction of the decidability of the nondeterministic hierarchy to the decidability of $\approx$ for cost-parity automata.[2]

[2]The notation and terminology used in [CL08b] is different than in this thesis. The "distance-parity" automata in that work are actually hB-parity automata with priorities and counter operations labelling states rather than transitions. The reduction is to the uniform universality problem (see Remark 2.2) of an automaton $\mathcal{U}_{ij}$. The construction of $\mathcal{U}_{ij}$ uses an automaton $\mathcal{A}_{ij}$, which corresponds to $\mathcal{E}$ here.

Proposition 7.1 ([CL08b]). *Given a parity automaton $\mathcal{A}$ and an index $[i,j]$, there is a nondeterministic B-$[i,j]$ automaton $\mathcal{E}$ and a correction function α such that $[\![\mathcal{E}]\!] \approx_\alpha \chi_{L(\mathcal{A})}$ if and only if $L(\mathcal{A})$ is recognizable by a nondeterministic $[i,j]$ automaton.*

One direction of the proof of Proposition 6.1 is straightforward. If there is some $\mathcal{E}$ that satisfies $[\![\mathcal{E}]\!] \approx_\alpha \chi_{L(\mathcal{A})}$ then $t \in L(\mathcal{A})$ if and only if there is an accepting run ρ of $\mathcal{E}$ on t with $\text{value}(\rho) \leq \alpha(0)$. Hence, there is a nondeterministic $[i,j]$ automaton $\mathcal{E}'$ based on $\mathcal{E}$ but without counters that recognizes $L(\mathcal{A})$. The automaton $\mathcal{E}'$ simulates the counters in the state up to value $\alpha(0)$, and enters a special rejecting sink state as soon as a counter would exceed this bound. This $\mathcal{E}'$ is the witness to the fact that $L(\mathcal{A})$ has index $[i,j]$.

The other direction requires us to examine the construction of $\mathcal{E}$. In some sense, the goal of $\mathcal{E}$ is to take an accepting run of $\mathcal{A}$ using some set of priorities P and optimize it so it uses only the desired priorities $[i,j]$. That is, $\mathcal{E}$ must guess a run of $\mathcal{A}$ and a mapping from the original set of priorities P to $[i,j]$. In order to preserve correctness (i.e. to ensure that a tree is accepted by $\mathcal{E}$ only if $\mathcal{A}$ accepted it), these mappings must satisfy some properties: odd priorities must be mapped to odd priorities, and the ordering of the priorities should be preserved such that the image of any odd priority dominates the image of all lower priorities.

Unfortunately, it is not true in general that there is a connection between the priorities in an arbitrary automaton such as $\mathcal{A}$ and the priorities in an $[i,j]$ automaton defining the same language. However, *guidable parity automata* (introduced in [CL08b] and studied in more detail in [Löd09]) do have this good property. A parity automaton is *guidable* if for any $\mathcal{B}$ such that $L(\mathcal{B}) = L(\mathcal{A})$ (in fact, even $L(\mathcal{B}) \subseteq L(\mathcal{A})$), there is a deterministic transducer with state set $Q_\mathcal{A}$ that reads a run $\rho_\mathcal{B}$ and outputs a run $\rho_\mathcal{A}$ of $\mathcal{A}$ such that if $\rho_\mathcal{B}$ is accepting for $\mathcal{B}$, then $\rho_\mathcal{A}$ is accepting for $\mathcal{A}$. Formally, this means that there exists a mapping $g : Q_\mathcal{A} \times \Delta_\mathcal{B} \longrightarrow \Delta_\mathcal{A}$ satisfying

- $g(p,(q,a,q',q'')) = (p,a,p',p'')$ for $p',p'' \in Q_\mathcal{A}$, and
- if $\rho_\mathcal{B}$ is an accepting run of $\mathcal{B}$ over some tree t, then $\rho_\mathcal{A} := g(\rho_\mathcal{B})$ is an accepting run of $\mathcal{A}$ over t where $g(\rho_\mathcal{B})$ is the unique run such $g(\rho_\mathcal{B})(\epsilon) = q^0_\mathcal{A}$, and for all $x \in [0,1]^*$,

$$(\rho_\mathcal{A}(x), t(x), \rho_\mathcal{A}(x0), \rho_\mathcal{A}(x1)) = g(\rho_\mathcal{A}(x), (\rho_\mathcal{B}(x), t(x), \rho_\mathcal{B}(x0), \rho_\mathcal{B}(x1))).$$

The key property of guidable automata is that in any cycle in a composed run of $\mathcal{B}$ and the guided run of $\mathcal{A}$, if the maximum priority in this cycle in the run of $\mathcal{B}$ is even, then the maximum priority in the run of $\mathcal{A}$ in this cycle is also even.

Lemma 7.2 ([CL08b, Lemma 2]). *Let $\mathcal{A}$ be a guidable parity automaton and let $\mathcal{B}$ be a nondeterministic parity automaton with $L(\mathcal{B}) = L(\mathcal{A})$. Set $\rho_{\mathcal{A}} := g(\rho_{\mathcal{B}})$ to be the run of $\mathcal{A}$ guided by some run $\rho_{\mathcal{B}}$ of $\mathcal{B}$. Let $x, y \in [0,1]^*$ with $x < y$. If $\rho_{\mathcal{A}}(x) = \rho_{\mathcal{A}}(y)$, $\rho_{\mathcal{B}}(x) = \rho_{\mathcal{B}}(y)$, and the maximum priority in $\rho_{\mathcal{B}}$ between x and y is even, then the maximum priority in $\rho_{\mathcal{A}}$ between x and y is even.*

We use the term *loop* to describe a cycle between positions x and y in the composed runs of some $\rho_{\mathcal{B}}$ and $\rho_{\mathcal{A}} := g(\rho_{\mathcal{B}})$ such that $\rho_{\mathcal{A}}(x) = \rho_{\mathcal{A}}(y)$ and $\rho_{\mathcal{B}}(x) = \rho_{\mathcal{B}}(y)$. The previous lemma implies that if there is some $[i,j]$ automaton $\mathcal{B}$ with $L(\mathcal{A}) = L(\mathcal{B})$, then there will be a nice relationship between the priorities $[i,j]$ and the priorities in the guidable automaton $\mathcal{A}$ in loops.

It turns out that if a parity automaton $\mathcal{A}$ has been constructed from its complement $\mathcal{A}'$ using a standard complementation procedure (see, e.g. [Löd09, Section 1.4]), then it is guidable [CL08b, Theorem 1].

This is not surprising once the idea behind the complementation procedure for parity automata over trees is recalled: on input t, $\mathcal{A}$ guesses a positional strategy for Adam in $\mathcal{A}' \times t$ and then runs a deterministic parity automaton on each branch to check that this is a winning strategy for Adam. This positional strategy for Adam is essentially a labelling of positions with a function $\Delta_{\mathcal{A}'} \longrightarrow [0,1]$, so selecting a transition of $\mathcal{A}$ corresponds to selecting a mapping $\Delta_{\mathcal{A}'} \longrightarrow [0,1]$ for a given position. The precise definition of the function g witnessing the guidability of $\mathcal{A}$ is not necessary for our work here, but the idea is that given some transition from $\mathcal{B}$, g selects an element from $\Delta_{\mathcal{A}'} \longrightarrow [0,1]$ by using a positional strategy for Adam in the intersection game $\mathcal{G}_{\mathcal{B} \cap \mathcal{A}'}$ (see Section 4.4), which can be viewed as a mapping $\Delta_{\mathcal{B}} \longrightarrow (\Delta_{\mathcal{A}'} \longrightarrow [0,1])$.

The important point is that every regular language has a guidable parity automaton recognizing it, and this guidable version can be constructed from an arbitrary parity automaton by using two complementations.

Returning to the description of $\mathcal{E}$, we can assume without loss of generality that $\mathcal{A}$ is guidable. Then $\mathcal{E}$ guesses a run of the guidable parity automaton $\mathcal{A}$ and mappings $s : P \longrightarrow ([i,j] \cup \{\bot\})$. We must allow these mappings to use the undefined value $\bot$ since there may be no connection between priorities P and $[i,j]$ in some parts of the run (e.g. outside of loops). The automaton $\mathcal{E}$ has $|P|$ counters and each counter is responsible for ensuring that eventually $\mathcal{E}$ makes a guess for the corresponding

priority (i.e. counter p is incremented to penalize $\mathcal{E}$ if priority p is used in the run of $\mathcal{A}$ but $s(p) = \bot$). The counters also catch other mistakes in the mappings (when the automaton is forced to change its mind about the mapping because it changes strongly connected component). These errors may be unavoidable because the nice connection between priorities P and $[i, j]$ may only hold in the context of loops. However, the automaton can only change strongly connected component a finite number of times, so the number of errors and hence the counter values are bounded. Therefore, it can be shown that $[\![\mathcal{E}]\!] \approx \chi_{L(\mathcal{A})}$.

This is an interesting reduction to a question of boundedness which is quite different than the star height reduction. Unfortunately, it reduces the decidability of the challenging parity index problem to another challenging problem, namely deciding $\approx$ for regular cost functions over infinite trees. It is not known in general whether the boundedness relation $\approx$ is decidable for cost functions over infinite trees definable by arbitrary cost-parity automata. However, the results in this thesis can be combined with Proposition 7.1 in order to show the decidability of the $[0, 1]$ level.

7.1.3 Decidability of the [0,1] level

As mentioned above, the only levels of the nondeterministic Mostowski hierarchy that are known to be decidable when the input is a nondeterministic parity automaton are the lowest levels [0,0] and [1,1]. We can apply the simulation and duality result from Chapter 4 to prove the decidability of the co-Büchi level (index $[0, 1]$). This is a corollary of Theorem 4.28 and Proposition 7.1 (with thanks to Löding for pointing out this application [Löd11]).

Proposition 7.3. *Given a parity automaton $\mathcal{A}$ over infinite trees, it is decidable whether or not there is a co-Büchi automaton $\mathcal{A}'$ such that $L(\mathcal{A}) = L(\mathcal{A}')$.*

Proof. By Proposition 7.1, given a parity automaton $\mathcal{A}$, there is a nondeterministic B-$[0, 1]$ automaton $\mathcal{E}$ such that $[\![\mathcal{E}]\!] \approx \chi_{L(\mathcal{A})}$ if and only if $L(\mathcal{A})$ is recognizable by a co-Büchi automaton.

Recall by Example 2.3 that for an automaton without counters such as $\mathcal{A}$, we have $[\![\mathcal{A}]\!]_B = \chi_{L(\mathcal{A})} = [\![\mathcal{A}']\!]_S$ where $\mathcal{A}'$ accepts the complement of $L(\mathcal{A})$. Hence, in order to test whether or not $\chi_{L(\mathcal{A})} \preccurlyeq [\![\mathcal{E}]\!]$, $\mathcal{A}'$ can be constructed from $\mathcal{A}$ using a standard complementation procedure, and the algorithm from Theorem 4.32 can be run to test $[\![\mathcal{A}']\!]_S \preccurlyeq [\![\mathcal{E}]\!]_B$.

The more difficult test is for $[\![\mathcal{E}]\!] \preccurlyeq \chi_{L(\mathcal{A})}$. By applying the duality result (Theorem 3.15) and the simulation result (Theorem 4.28), we can construct a nondeterministic S-$[1,2]$ automaton $\mathcal{E}'$ such that $[\![\mathcal{E}]\!] \approx [\![\mathcal{E}']\!]$. Hence, it suffices to test $[\![\mathcal{E}']\!]_S \preccurlyeq [\![\mathcal{A}]\!]_B$ which is decidable by Theorem 4.32. □

Based on the structure of the proof of Proposition 7.3, one route to proving the decidability of the other levels $[i,j]$ of the Mostowski hierarchy would be to prove a simulation result similar to Theorem 4.28 for arbitrary alternating S-$[i+1,j+1]$ automata. This would likely rely on showing that finite memory strategies suffice in $\overline{B\text{-}[i,j]}$ games. As discussed in Chapter 4, this remains a challenging open problem.

7.2 Weak definability problem

The weakly definable languages are a proper subclass of the regular languages of infinite trees. This subclass is well-studied since it is expressive (e.g. the temporal logic CTL embeds in it) but still admits efficient model checking procedures [KV99]. As a result, it is desirable to know when a language lies in this class. The *weak definability problem* is the problem of deciding whether a given language of regular trees is weakly definable (i.e. definable in weak monadic second-order logic or, equivalently, using a weak alternating automaton).

Since Rabin [Rab70] characterized weak languages L as exactly those languages for which both L and $\overline{L}$ are recognizable by nondeterministic Büchi automata (see Theorem 6.5), one route to proving the decidability of the weak definability problem would be to show the decidability of level $[1,2]$ in the nondeterministic Mostowski hierarchy. Indeed, this means that the weak definability problem is decidable when the input is a deterministic tree language [NW05]. Because the decidability of the [1,2] level in the nondeterministic hierarchy is open in general, an alternative method is used here. We have been unable to prove the general decidability of the weak definability problem, but we can solve some special cases when it is possible to obtain a nondeterministic Büchi automaton for L or $\overline{L}$ as input to the problem.

7.2.1 Decidability for Büchi input

If a Büchi automaton $\mathcal{U}$ for L is given as input (instead of an arbitrary parity automaton), there is a reduction of the weak definability problem to the decidability of $\approx$ for quasi-weak automata. This can also be viewed as a decidability result for the $[1,2]$

level in the Mostowski hierarchy in the case when the input is a Büchi automaton for the complement language.

Theorem 7.4. *Given a Büchi automaton $\mathcal{U}$ with $L = L(\mathcal{U})$, there exists effectively a quasi-weak B-automaton $\mathcal{B}$ such that $[\![\mathcal{B}]\!] \approx \chi_{\overline{L}}$ if and only if L is weakly definable.*

Proof. Given the Büchi automaton $\mathcal{U}$ without counters, we view this as an S-Büchi automaton with $[\![\mathcal{U}]\!]_S = \chi_{\overline{L}}$. Using Proposition 7.1 we get a B-Büchi automaton $\mathcal{E}$ that satisfies $[\![\mathcal{E}]\!]_B \approx \chi_{\overline{L}}$ if and only if $\overline{L}$ is Büchi. We run the construction in Theorem 6.6 with $\mathcal{E}$ and $\mathcal{U}$ which yields a B-quasi-weak automaton $\mathcal{B}$ with the property that $[\![\mathcal{B}]\!] \approx \chi_{\overline{L}}$ if $[\![\mathcal{E}]\!]_B \approx [\![\mathcal{U}]\!]_S$.

Assume that $[\![\mathcal{B}]\!] \approx \chi_{\overline{L}}$. Then there exists N such that for all $t \in \overline{L}$, there is an accepting run ρ of $\mathcal{B}$ on t with $\text{value}(\rho) \leq N$. Hence, there is an alternating Büchi automaton $\mathcal{B}'$ based on $\mathcal{B}$ that simulates the counters in the state up to value N, and enters a special rejecting state as soon as a counter would exceed this bound. It is not hard to see that $L(\mathcal{B}') = \overline{L}$.

Moreover, because $\mathcal{B}$ is quasi-weak, this bound N on the value also implies that there is a bound $\beta(N)$ on the number of alternations between accepting and rejecting states needed in order to recognize $t \in \overline{L}$ (and consequently, in $\mathcal{B}'$, a bound $\beta(N)+1$ on the number of alternations between accepting and rejecting states for $t \in L$). Hence, $\mathcal{B}'$ is a weak automaton recognizing $\overline{L}$, and by closure of weak languages under complementation, L and $\overline{L}$ must be weakly definable.

In the other direction, we assume that L is weak and need to prove that $[\![\mathcal{B}]\!] \approx \chi_{\overline{L}}$. But if L is weak, then L and $\overline{L}$ are also Büchi, so $[\![\mathcal{E}]\!] \approx \chi_{\overline{L}}$ by Proposition 7.1. Hence by Theorem 6.6, $[\![\mathcal{B}]\!] \approx [\![\mathcal{E}]\!] \approx \chi_{\overline{L}}$ as desired. □

Corollary 7.5. *Given an alternating Büchi automaton, alternating co-Büchi automaton, or deterministic parity automaton $\mathcal{A}$, it is decidable whether there is a weak automaton $\mathcal{W}$ such that $L(\mathcal{W}) = L(\mathcal{A})$.*

Proof. Because $[\![\mathcal{B}]\!] \approx \chi_{\overline{L}}$ is decidable when $\mathcal{B}$ is quasi-weak and $\overline{L}$ is a regular language (by Corollary 6.8 and Theorem 4.32), Theorem 7.4 implies the decidability of the weak definability problem when the input is a nondeterministic Büchi automaton.

The other inputs can all be transformed into a nondeterministic Büchi automaton recognizing the language or its complement.

If the input is an alternating Büchi automaton, then an equivalent nondeterministic Büchi automaton can be constructed ([MS95]), and Theorem 7.4 can be applied.

If the input is an alternating co-Büchi automaton, then the complement is an alternating Büchi automaton and we can use the previous case to decide whether $\overline{L}$ is weak. Since weakly definable languages are closed under complement, $\overline{L}$ is weak if and only if L is weak.

Finally, if the input is a deterministic parity automaton (not necessarily Büchi), then an alternating co-Büchi automaton can be constructed for the language [Mur08a, Proposition 2], so we reduce to the other cases. As mentioned earlier, the decidability of this case was known already from [NW05] using different methods. □

7.2.2 An alternative construction

We now describe another construction for $\mathcal{B}$ in Theorem 7.4 that does not use the involved construction from Theorem 6.6. This alternative construction is much simpler (both in terms of the informal description and the number of states) compared to the automaton that comes out of applying the construction from Chapter 6. It also serves as another nice example where quasi-weak automata (rather than weak automata) arise naturally.

It will be useful to refer to Section 6.2.2 and Section 7.1.2 for a summary of [KV99] and [CL08b], since details of these constructions are required in the definition and proof of correctness for $\mathcal{B}$.

We are given a Büchi automaton $\mathcal{U} = \langle Q_{\mathcal{U}}, \mathbb{A}, q_0^{\mathcal{U}}, F_{\mathcal{U}}, \Delta_{\mathcal{U}} \rangle$ (without counters), with $L = L(\mathcal{U})$. Using a standard complementation procedure starting from $\mathcal{U}$, we construct a parity automaton $\mathcal{A}'$ using priorities P such that $L(\mathcal{A}') = \overline{L}$ (this means $\mathcal{A}'$ is guidable, but not necessarily Büchi). From $\mathcal{A}'$, we construct the B-Büchi automaton $\mathcal{E} = \langle Q_{\mathcal{E}}, \mathbb{A}, q_0^{\mathcal{E}}, \Gamma_{\mathcal{E}}, F_{\mathcal{E}}, \Delta_{\mathcal{E}} \rangle$ using Proposition 7.1 which simulates $\mathcal{A}'$ while guessing priority mappings, and satisfies $[\![\mathcal{E}]\!]_B \approx \chi_{\overline{L}}$ if and only if $\overline{L}$ is Büchi.

We use $\mathcal{U}$ and $\mathcal{E}$ to build an alternating B-Büchi automaton $\mathcal{B} = \langle Q, \mathbb{A}, q_0, \Gamma, F, \Delta \rangle$ described below. It will turn out that $\mathcal{B}$ is quasi-weak and $[\![\mathcal{B}]\!] \approx \chi_{\overline{L}}$ if and only if L is weak.

The state set is

$$Q := Q_{\mathcal{E}} \times Q_{\mathcal{U}} \times \{\epsilon, \bot\}$$

with initial state $q_0 := (q_0^{\mathcal{E}}, q_0^{\mathcal{U}}, \epsilon)$.

We now describe the operation of $\mathcal{B}$ on some input t. The idea is that Adam (respectively, Eve) selects a run of $\mathcal{U}$ (respectively, $\mathcal{E}$) on t (technically, this is done one move at a time). Eve also selects a branch of the tree.

Acceptance is determined by an analysis of blocks. For the construction in Section 6.2.2, a complicated definition of blocks was required because both B- and S-actions were involved. Here, we can use the simpler definition of blocks that is similar to [KV99]. A complete $\mathcal{B}$-*block* must first witness an accepting state from the run of $\mathcal{U}$ followed by an accepting state from the run of $\mathcal{E}$. The last component of the state $(q_\mathcal{E}, q_\mathcal{U}, z)$ tracks these accepting states from $\mathcal{E}$ and $\mathcal{U}$: $z = \epsilon$ if no accepting states have been seen and $z = \bot$ if $F_\mathcal{U}$ has been visited but $F_\mathcal{E}$ has not. If $z = \bot$ and a state from $F_\mathcal{E}$ is seen, then the current block is closed and z is set to ϵ. The set F consists of all states $(q_\mathcal{E}, q_\mathcal{U}, z)$ where $z \neq \bot$.

Thus far, this is similar to the construction in [KV99] that takes nondeterministic Büchi automata for a language and its complement and constructs a weak automaton for the same language. The key difference is that instead of counting the blocks in the state up to some fixed bound depending on the size of the initial nondeterministic Büchi automata, in this construction the number of blocks are going to be tracked using counters because the approximate nature of $\mathcal{E}$ means that we do not know this bound upfront.

For this reason, the set of counters is $\Gamma := \Gamma_\mathcal{E} \cup \{\gamma_{\text{alt}}\}$. The counter actions are copied from the run of $\mathcal{E}$ chosen by Eve. In addition, the new counter γ_{alt} is incremented any time $\mathcal{B}$ closes a block (when the last component of the state changes from $\bot$ to ϵ).

Because of γ_{alt}, $\mathcal{B}$ satisfies the quasi-weak cycle condition and hence is quasi-weak by Proposition 6.3.

We are now ready to give an alternative proof of Theorem 7.4.

Alternative proof of Theorem 7.4. Assuming that $[\![\mathcal{B}]\!] \approx \chi_{\overline{L}}$ and then proving that L is weak is the same as the original proof of Theorem 7.4 above.

The other direction is the interesting case.

Assume that L is weak. We already have a nondeterministic Büchi automaton $\mathcal{U}$ such that $L(\mathcal{U}) = L$. By Theorem 6.5, there is a nondeterministic Büchi automaton $\mathcal{U}'$ such that $L(\mathcal{U}') = \overline{L}$. This implies that $[\![\mathcal{E}]\!] \approx_\alpha \chi_{\overline{L}}$.

By Theorem 6.5, we can construct a weak automaton $\mathcal{W}$ defining L (using the construction from [KV99] starting from $\mathcal{U}$ and $\mathcal{U}'$, see Section 6.2.2) in which at each position, Eve selects a transition of $\mathcal{U}$ and Adam selects a transition of $\mathcal{U}'$ and a direction. Blocks are counted in the resulting play as described in Section 6.2.2: complete blocks have $F_\mathcal{U}$ after $F_{\mathcal{U}'}$, and are counted up to some fixed bound $M := |Q_\mathcal{U}| \cdot |Q_{\mathcal{U}'}|$. We call these $\mathcal{W}$-blocks to distinguish from the $\mathcal{B}$-blocks described above.

We show that $[\![\mathcal{B}]\!] \approx \chi_{\overline{L}}$.

First, assume by contradiction that $\chi_{\overline{L}} \not\preccurlyeq [\![\mathcal{B}]\!]$. This means that there is some t such that Eve has a strategy σ in $\mathcal{B} \times t$ with value$(\sigma) \leq N \in \mathbb{N}$, but $t \in L$. Consider the play π in $\mathcal{B} \times t$ resulting from Eve playing according to σ and Adam playing an accepting run of $\mathcal{U}$ on t (possible since $t \in L$). Then π will have infinitely many states from $F_{\mathcal{U}}$. If there are finitely many states from $F_{\mathcal{E}}$, then π must stabilize in rejecting states of the form $(q_{\mathcal{E}}, q_{\mathcal{U}}, \bot)$ and value$(\pi) = \infty$. If there are infinitely many states from $F_{\mathcal{E}}$ then there will be infinitely many blocks, so γ_{alt} will be unbounded and value$(\pi) = \infty$. In either case, this contradicts the assumption that value$(\sigma) \leq N$.

Finally, we prove $[\![\mathcal{B}]\!] \preccurlyeq \chi_{\overline{L}}$. Let $M := |Q_{\mathcal{U}}| \cdot |Q_{\mathcal{U}'}|$ and $N := (M+1) \cdot (\alpha(0)+2)$. For all $t \in \overline{L}$, we describe a strategy σ for Eve in $\mathcal{B} \times t$ such that value$(\sigma) \leq N$, witnessing the fact that $[\![\mathcal{B}]\!]$ is bounded.

Because $t \in \overline{L}$, there is a winning strategy for Adam in $\mathcal{W} \times t$. In fact, we can fix an accepting run $\rho_{\mathcal{U}'}$ of $\mathcal{U}'$ on t and consider the variant of the game where Adam plays according to $\rho_{\mathcal{U}'}$ and only chooses a direction. Adam still has a winning strategy in this game, so there is a positional winning strategy $\sigma_{\text{dir}} : (\mathcal{T} \times \Delta_{\mathcal{U}}) \rightarrow [0,1]$. This means that any $\pi \in \sigma_{\text{dir}}$ must have at most M $\mathcal{W}$-blocks (since in the construction from [KV99], a play is immediately winning for Eve if the number of blocks exceeds M).

Let $m \leq M$ be the number of $\mathcal{W}$-blocks in some $\pi \in \sigma_{\text{dir}}$. This means π can be partitioned into sections $\pi_1 \pi_2 \cdots \pi_m \pi'$ where each partial play $\pi_i = u_i v_i$ for $i \in [1, m]$ has some (possibly empty) prefix u_i with no $F_{\mathcal{U}'}$ followed by a suffix v_i of length at least 1 that begins with a state from $F_{\mathcal{U}'}$, ends with a state from $F_{\mathcal{U}}$, but has no intermediate states from $F_{\mathcal{U}}$. Since π is winning for Adam in $\mathcal{W} \times t$, the infinite continuation π' after the m blocks has some $F_{\mathcal{U}'}$ that is not followed by any $F_{\mathcal{U}}$. This means it can be split into $\pi' = u'v'$ where u' is a possibly empty prefix with no $F_{\mathcal{U}'}$, and v' is an infinite play beginning with a state from $F_{\mathcal{U}'}$ and containing no states from $F_{\mathcal{U}}$.

We now describe a finite memory strategy σ for Eve in $\mathcal{B} \times t$ which needs to select a branch and a run of $\mathcal{E}$ given the run of $\mathcal{U}$ selected by Adam. Recall that a run of $\mathcal{E}$ consists of simulating a run of $\mathcal{A}'$ and guessing mappings $s : P \rightarrow ([1,2] \cup \{\bot\})$. Because $t \in \overline{L}$, there is a strategy (run) $\sigma_{\mathcal{E}}$ in $\mathcal{E} \times t$ with value$(\sigma_{\mathcal{E}}) \leq \alpha(0)$ that is obtained by using $\rho_{\mathcal{U}'}$ to guide a run $\rho_{\mathcal{A}'} := g(\rho_{\mathcal{U}'})$ and the corresponding mappings $s : P \rightarrow ([1,2] \cup \{\bot\})$ (see [Löd09, Section 5.3] for further explanation of how these mappings are chosen based on $\rho_{\mathcal{U}'}$).

The strategy σ plays the direction according to σ_{dir} and the run $\rho_{\mathcal{A}'} := g(\rho_{\mathcal{U}'})$ (the run of $\mathcal{A}'$ guided by $\rho_{\mathcal{U}'}$). This requires no additional memory.

The choice of mappings is determined by $\sigma_{\mathcal{E}}$ and the partitions $\pi_i = u_i v_i$ (based on the run of $\mathcal{U}$ and $\mathcal{U}'$ being played). On the u_i part, σ maps all priorities P in $\mathcal{A}'$ to 1, and we call this the *wait mode*. On the v_i part, σ maps priorities according to $\sigma_{\mathcal{E}}$, and we call this the *active mode*. This can be done in a deterministic way using finite memory (Eve remembers which of these two modes she is currently in, and changes mode based on conditions involving $F_{\mathcal{U}}$ and $F_{\mathcal{U}'}$ described above). Similarly, on $\pi' = u'v'$, σ maps all priorities P to 1 on u', and according to $\sigma_{\mathcal{E}}$ on v'.

Let $\pi \in \sigma$ and partition into $\pi_1\pi_2\cdots\pi_m\pi'$ as described above. Consider some $\pi_i = u_i v_i$ on positions $[x, y]$.

We claim that on $[x, y)$ (which represents one $\mathcal{W}$-block) there is at most one complete $\mathcal{B}$-block. Indeed, in the wait mode on u_i, $\mathcal{B}$ may move to a state with $z = \perp$ but cannot close this block since there will be no accepting state from $F_{\mathcal{E}}$ (since all priorities are mapped to 1). In the active mode on v_i, however, this block might be closed. But because y is the only position in v_i that can have a state from $F_{\mathcal{U}}$, there is no way to start/finish more $\mathcal{B}$-blocks on $[x, y)$. Since γ_{alt} is only incremented at the close of a $\mathcal{B}$-block, this means that γ_{alt} has value at most 1 on this segment.

We also claim that on $[x, y)$ the value is at most $\alpha(0) + 2$ based on the counters from $\mathcal{E}$ (where α comes from Proposition 7.1 and depends only on $\mathcal{A}'$). On the v_i part, the counters from $\mathcal{E}$ have value at most $\alpha(0)$ (since $\text{value}(\sigma_{\mathcal{E}}) \leq \alpha(0)$). Recall that $\mathcal{E}$ increments a counter if the guess about the priority mapping changes; thus the additional two increments could come from changing the guess to priority 1 at the start of u_i, and then changing back to the priority mapping according to $\sigma_{\mathcal{E}}$ at the end of u_i.

Likewise, on $\pi' = u'v'$, γ_{alt} can be incremented at most once (i.e. there can be at most one $\mathcal{B}$-block) and the value from the counters from $\mathcal{E}$ is at most $\alpha(0) + 1$. Moreover, we know that the infinite play on the v' part visits $F_{\mathcal{E}}$ infinitely many times (since $\text{value}(\sigma_{\mathcal{E}}) \leq \alpha(0)$), so this play is accepting.

Since any $\pi \in \sigma$ is of the form $\pi_1\pi_2\cdots\pi_m\pi'$ for $m \in [1, M]$, this means that γ_{alt} has a maximum value $M + 1$ and the counters from $\mathcal{E}$ have a maximum value of $(M + 1) \cdot (\alpha(0) + 2)$. Hence $\text{value}(\sigma) \leq N$. □

7.3 Discussion

Cost automata (in particular, distance and nested distance desert automata) were originally introduced as tools to help solve problems from formal language theory such as the star height problem. In this chapter, we have seen that cost automata

over infinite trees can be used to address problems about regular languages of infinite trees.

The fact that we can only decide the $[0, 1]$ level of the nondeterministic Mostowski hierarchy (as shown in Proposition 7.3) comes down to the challenge of proving finite memory determinacy in arbitrary cost-parity games, as discussed in Chapter 4.

The decidability of the weak definability problem for Büchi input is implied by the Rabin-style characterization result from Chapter 6. The alternative construction provides more insight into the problem. As described in Section 7.2.2, the idea is that Adam guesses a run of the given Büchi automaton while Eve uses a guidable automaton for the complement and guesses a mapping from the original set of priorities P to the desired priorities $[1, 2]$. The work from [CL08b] provides a way for the counters to be used to ensure that these guesses are valid and that the resulting automaton is quasi-weak. A first attempt to extend this approach to the general problem would allow both players to guess runs of guidable automata for the language and its complement. Unfortunately, thus far we are unable to design a quasi-weak cost automaton that ensures that both Adam and Eve make appropriate choices. This is related to the fact that guidable automata preserve accepting loops (loops where the maximum priority is even), but a relationship between rejecting loops (where the maximum priority is odd) would also be needed in such a construction, and this relationship is not guaranteed in guidable automata.

Despite these setbacks, it seems worthwhile to explore the cost automaton approach to this problem further. A similar problem that could also be studied is the *weak index problem*. Given a weakly definable language, the weak index is related to the length of alternating chains in the corresponding weak automaton. The weak index problem asks for the minimum weak index for a weakly definable language and is known to be decidable when the input is a deterministic tree language [Mur08b], or an extension known as a game tree language [DFM11]. It would be interesting to explore whether or not this problem (or at least special cases of this problem) can be reduced to a question of the decidability of $\preccurlyeq$ for quasi-weak cost automata.

Chapter 8

Conclusion

This thesis has developed the theory of regular cost functions over infinite trees and provides further proof that this is a robust quantitative extension to the theory of regular languages. The main results can be summarized as follows.

- Alternating B-parity and alternating S-parity automata over infinite trees, as well as versions with hierarchical counters, are effectively equivalent (Theorem 3.15).
- For cost functions f and g over infinite trees, $f \preccurlyeq g$ is decidable when f is given as a nondeterministic S-parity automaton and g is given as a nondeterministic B-parity automaton (Theorem 4.32).
- Finite memory strategies cannot always guarantee the optimal value in cost-parity games. However, finite memory strategies are sufficient (up to $\approx$) for Eve in B-$[1,2]$ and $\overline{B\text{-}[0,1]}$ games when the underlying game graph is chronological and finite branching (Corollary 4.12).
- Alternating B-$[1,2]$ automata can be simulated by nondeterministic B-$[1,2]$ automata, and alternating S-$[1,2]$ automata can be simulated by nondeterministic S-$[1,2]$ automata (Theorem 4.28).
- The logic cost WMSO is effectively equivalent to weak cost automata (Theorem 5.13). Because weak cost automata can be simulated by both nondeterministic B-$[1,2]$ and S-$[1,2]$ automata (Corollary 5.5), the logic is decidable: given cost WMSO sentences φ and ψ, it is decidable whether or not $[\![\varphi]\!] \preccurlyeq [\![\psi]\!]$ (Theorem 5.18).

- Quasi-weak automata exhibit a new variant of weakness: unlike weak automata which bound the number of alternations between accepting and rejecting priorities using the state, quasi-weak cost automata bound the number of alternations using the counters. Quasi-weak cost automata are strictly more expressive than weak cost automata over infinite trees (Proposition 6.4). Moreover, it is this larger class of quasi-weak cost functions that admits a Rabin-style characterization: a cost function over infinite trees is quasi-weak if and only if it is recognizable by both a nondeterministic B-$[1, 2]$ and nondeterministic S-$[1, 2]$ automaton (Theorem 6.6).

- These results about the theory of regular cost functions over infinite trees imply the decidability of some special cases of problems from language theory. In particular, the $[0, 1]$ level of the nondeterministic Mostowski hierarchy is decidable (Proposition 7.3), and the weak definability problem is decidable when a Büchi or co-Büchi automaton is provided as input (Theorem 7.4).

We view the proofs about finite memory strategies in certain cost-parity games from Chapter 4 and the Rabin-style characterization from Chapter 6 as the main contributions of this thesis.

Further directions

We have already mentioned some open questions in the discussion section at the end of each chapter, but we highlight three directions for further work here.

- The main open problem in the theory is whether or not further classes of cost-parity games admit finite memory strategies. Proving finite memory determinacy for arbitrary cost-parity games over acyclic game graphs would imply the decidability of $\preccurlyeq$ over infinite trees, the decidability of full cost MSO over infinite trees, and the decidability of the parity index problem and weak definability problem. Proving that finite memory strategies are not sufficient would be unexpected and very interesting as well.

- Another research direction would be to study other logics related to boundedness. The relationship between cost WMSO and WMSO+$\mathbb{U}$ (described in Sections 2.4 and 5.3) still needs to be resolved. Likewise, the study of the logic FO+RR (also described in Section 5.3) and its application in model checking seems like a promising line of work as well.

- Finally, additional applications of quasi-weak or other types of cost automata could be explored. In particular, the question remains whether there is a reduction of the general weak definability problem or weak index problem to the decidability of quasi-weak cost automata (see Section 7.3). These applications would probably benefit from a finer analysis of the complexity of the various decidability results in this thesis, which we also leave for further work.

The hope is that this thesis sets the stage for further development of this theory of regular cost functions over infinite trees, which may ultimately help settle the decidability of important problems like the parity index problem.

Bibliography

[AKY08] Parosh Aziz Abdulla, Pavel Krcál, and Wang Yi. R-automata. In Franck van Breugel and Marsha Chechik, editors, *CONCUR*, volume 5201 of *LNCS*, pages 67–81. Springer, 2008. (Cited on pages 4, 21 and 34.)

[Arn99] André Arnold. The μ-calculus alternation-depth hierarchy is strict on binary trees. *ITA*, 33(4/5):329–340, 1999. (Cited on page 156.)

[AN07] André Arnold and Damian Niwiński. Continuous separation of game languages. *Fundam. Inform.*, 81(1-3):19–28, 2007. (Cited on page 132.)

[Bal04] Sebastian Bala. Regular language matching and other decidable cases of the satisfiability problem for constraints between regular open terms. In Volker Diekert and Michel Habib, editors, *STACS*, volume 2996 of *LNCS*, pages 596–607. Springer, 2004. (Cited on page 21.)

[BPR11] Michael Benedikt, Gabriele Puppis, and Cristian Riveros. Regular repair of specifications. In *LICS*, pages 335–344. IEEE Computer Society, 2011. (Cited on page 4.)

[Boj04] Mikołaj Bojańczyk. A bounding quantifier. In Jerzy Marcinkowski and Andrzej Tarlecki, editors, *CSL*, volume 3210 of *LNCS*, pages 41–55. Springer, 2004. (Cited on pages 34 and 36.)

[Boj09] Mikołaj Bojańczyk. Weak MSO with the unbounding quantifier. In Susanne Albers and Jean-Yves Marion, editors, *STACS*, volume 3 of *LIPIcs*, pages 159–170. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009. (Cited on page 127.)

[BC06] Mikołaj Bojańczyk and Thomas Colcombet. Bounds in ω-regularity. In *LICS*, pages 285–296. IEEE Computer Society, 2006. (Cited on pages 7, 8, 16, 21, 23, 34, 35, 36 and 63.)

[BT12] Mikołaj Bojańczyk and Szymon Toruńczyk. Weak MSO+U over infinite trees. In Christoph Dürr and Thomas Wilke, editors, *STACS*, volume 14 of *LIPIcs*, pages 648–660. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012. (Cited on page 127.)

[Bra98] Julian C. Bradfield. The modal μ-calculus alternation hierarchy is strict. *Theor. Comput. Sci.*, 195(2):133–153, 1998. (Cited on page 156.)

[Büc60] J. Richard Büchi. Weak second-order arithmetic and finite automata. *Z. Math. Logik Grundlagen Math.*, 6:66–92, 1960. (Cited on pages 10, 111 and 156.)

[Büc77] J. Richard Büchi. Using determinancy of games to eliminate quantifiers. In *FCT*, pages 367–378, 1977. (Cited on page 67.)

[Col09a] Thomas Colcombet. Regular cost functions over words, 2009. Manuscript at `http://www.liafa.jussieu.fr/~colcombe/`. (Cited on pages 7, 8, 13, 17, 19, 20, 21 and 119.)

[Col09b] Thomas Colcombet. Regular cost functions, part I: logic and algebra over words. Submitted, 2009. (Cited on page 119.)

[Col09c] Thomas Colcombet. The theory of stabilisation monoids and regular cost functions. In *ICALP (2)*, volume 5556 of *LNCS*, pages 139–150. Springer, 2009. (Cited on pages 1, 5, 6, 7, 20, 22, 23, 24 and 152.)

[Col11] Thomas Colcombet. Safra-like constructions for regular cost functions over finite words. Unpublished, 2011. (Cited on page 22.)

[Col12a] Thomas Colcombet. Forms of determinism for automata (invited talk). In Christoph Dürr and Thomas Wilke, editors, *STACS*, volume 14 of *LIPIcs*, pages 1–23. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012. (Cited on page 28.)

[Col12b] Thomas Colcombet. Personal communication, 2012. (Cited on pages 19, 44, 85 and 152.)

[CKL10] Thomas Colcombet, Denis Kuperberg, and Sylvain Lombardy. Regular temporal cost functions. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors,

ICALP (2), volume 6199 of *LNCS*, pages 563–574. Springer, 2010. (Cited on pages 7, 29, 76 and 85.)

[CL08a] Thomas Colcombet and Christof Löding. The nesting-depth of disjunctive mu-calculus. In *CSL*, volume 5213 of *LNCS*, pages 416–430. Springer, 2008. (Cited on pages 4, 72, 73 and 92.)

[CL08b] Thomas Colcombet and Christof Löding. The non-deterministic Mostowski hierarchy and distance-parity automata. In Luca Aceto, Ivan Damgard, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP (2)*, volume 5126 of *LNCS*, pages 398–409. Springer, 2008. (Cited on pages 5, 10, 157, 158, 159, 163 and 167.)

[CL10] Thomas Colcombet and Christof Löding. Regular cost functions over finite trees. In *LICS*, pages 70–79. IEEE Computer Society, 2010. Online at `http://www.liafa.jussieu.fr/~colcombe/`. (Cited on pages 7, 8, 9, 13, 29, 30, 33, 34, 39, 41, 51, 56, 57, 71, 73, 95, 103, 108, 116 and 118.)

[CL12] Thomas Colcombet and Christof Löding. Personal communication, 2012. (Cited on pages 28, 29, 30, 75, 76, 77 and 80.)

[DKV09] Manfred Droste, Werner Kuich, and Heiko Vogler, editors. *Handbook of Weighted Automata*. Springer Berlin Heidelberg, 2009. (Cited on page 35.)

[DFM11] Jacques Duparc, Alessandro Facchini, and Filip Murlak. Definable operations on weakly recognizable sets of trees. In Supratik Chakraborty and Amit Kumar, editors, *FSTTCS*, volume 13 of *LIPIcs*, pages 363–374. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011. (Cited on page 167.)

[Egg63] L. C. Eggan. Transition graphs and the star-height of regular events. *Michigan Math. J.*, 10:385–397, 1963. (Cited on page 3.)

[EJ91] E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *FOCS*, pages 368–377. IEEE, 1991. (Cited on pages 67, 69 and 77.)

[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. (Cited on pages 41, 51, 69 and 105.)

[GW06] Erich Grädel and Igor Walukiewicz. Positional determinacy of games with infinitely many priorities. *Logical Methods in Computer Science*, 2(4), 2006. (Cited on pages 69 and 78.)

[GH82] Yuri Gurevich and Leo Harrington. Trees, automata, and games. In Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber, editors, *STOC*, pages 60–65. ACM, 1982. (Cited on pages 9, 29, 67 and 70.)

[Has79] Kosaburo Hashiguchi. A decision procedure for the order of regular events. *Theor. Comput. Sci.*, 8:69–72, 1979. (Cited on page 2.)

[Has82] Kosaburo Hashiguchi. Limitedness theorem on finite automata with distance functions. *J. Comput. Syst. Sci.*, 24(2):233–244, 1982. (Cited on pages 2, 3, 21, 34, 35 and 76.)

[Has88] Kosaburo Hashiguchi. Relative star height, star height and finite automata with distance functions. In Jean-Éric Pin, editor, *Formal Properties of Finite Automata and Applications*, volume 386 of *LNCS*, pages 74–88. Springer, 1988. (Cited on page 3.)

[HP06] Thomas A. Henzinger and Nir Piterman. Solving games without determinization. In Zoltán Ésik, editor, *CSL*, volume 4207 of *LNCS*, pages 395–410. Springer, 2006. (Cited on page 28.)

[HST10] Szczepan Hummel, Michał Skrzypczak, and Szymon Toruńczyk. On the topological complexity of MSO+U and related automata models. In Petr Hlinený and Antonín Kucera, editors, *MFCS*, volume 6281 of *LNCS*, pages 429–440. Springer, 2010. (Cited on page 36.)

[JL02] David Janin and Giacomo Lenzi. On the logical definability of topologically closed recognizable languages of infinite trees. *Computers and Artificial Intelligence*, 21(3), 2002. (Cited on page 157.)

[Jur98] Marcin Jurdziński. Deciding the winner in parity games is in UP ∩ co-UP. *Inf. Process. Lett.*, 68(3):119–124, 1998. (Cited on page 70.)

[Jur00] Marcin Jurdziński. Small progress measures for solving parity games. In Horst Reichel and Sophie Tison, editors, *STACS*, volume 1770 of *LNCS*, pages 290–301. Springer, 2000. (Cited on pages 69, 77 and 155.)

[Kec95] Alexander S. Kechris. *Classical Descriptive Set Theory*. Graduate Texts in Mathematics. Springer-Verlag, 1995. (Cited on page 55.)

[Kir04] Daniel Kirsten. Desert automata and the finite substitution problem. In Volker Diekert and Michel Habib, editors, *STACS*, volume 2996 of *LNCS*, pages 305–316. Springer, 2004. (Cited on pages 21 and 76.)

[Kir05] Daniel Kirsten. Distance desert automata and the star height problem. *ITA*, 39(3):455–509, 2005. (Cited on pages 3, 4, 21 and 34.)

[Kle56] Stephen C. Kleene. Representation of events in nerve nets and finite automata. In Claude Shannon and John McCarthy, editors, *Automata Studies*, pages 3–42. Princeton University Press, Princeton, N.J., 1956. (Cited on page 5.)

[Kro94] Daniel Krob. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. *International Journal of Algebra and Computation*, 4:405–425, 1994. (Cited on page 6.)

[Kup11] Denis Kuperberg. Linear temporal logic for regular cost functions. In Thomas Schwentick and Christoph Dürr, editors, *STACS*, volume 9 of *LIPIcs*, pages 627–636. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011. (Cited on page 7.)

[KVB11] Denis Kuperberg and Michael Vanden Boom. Quasi-weak cost automata: a new variant of weakness. In Supratik Chakraborty and Amit Kumar, editors, *FSTTCS*, volume 13 of *LIPIcs*, pages 66–77. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011. Online at `http://www.liafa.jussieu.fr/~dkuperbe/`. (Cited on pages 10, 11, 61, 64, 129 and 155.)

[KVB12] Denis Kuperberg and Michael Vanden Boom. On the expressive power of cost logics over infinite words. In Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer, editors, *ICALP (2)*, volume 7392 of *Lecture Notes in Computer Science*, pages 287–298. Springer, 2012. (Cited on pages 140, 152, 153 and 154.)

[KV99] Orna Kupferman and Moshe Y. Vardi. The weakness of self- complementation. In Christoph Meinel and Sophie Tison, editors, *STACS*, volume 1563 of *LNCS*, pages 455–466. Springer, 1999. (Cited on pages 10, 137, 138, 140, 161, 163, 164 and 165.)

[KV01] Orna Kupferman and Moshe Y. Vardi. Weak alternating automata are not that weak. *ACM Trans. Comput. Log.*, 2(3):408–429, 2001. (Cited on page 140.)

[Lan11a] Martin Lang. Personal communication, 2011. (Cited on page 128.)

[Lan11b] Martin Lang. Resource-bounded reachability on pushdown systems. Master's thesis, RWTH Aachen University, 2011. (Cited on page 128.)

[Leu88] Hing Leung. On the topological structure of a finitely generated semigroup of matrices. *Semigroup Forum*, 37:273–287, 1988. (Cited on page 35.)

[Löd09] Christof Löding. Automata for boundedness problems on trees. Tutorial at the Workshop on Distance Automata, Paris, 2009. Slides online at `http://www.automata.rwth-aachen.de/~loeding/`. (Cited on page 71.)

[Löd11] Christof Löding. Personal communication, 2011. (Cited on page 160.)

[Löd09] Christop Löding. Logic and automata over infinite trees. Habilitation thesis, RWTH Aachen University, 2009. (Cited on pages 101, 157, 158, 159 and 165.)

[Mar75] Donald A. Martin. Borel determinacy. *The Annals of Mathematics*, 102(2):336–371, 1975. (Cited on page 55.)

[Min67] Marvin L. Minsky. *Computation: Finite and Infinite Machines*. Englewood Cliffs, N. J., Prentice-Hall, 1967. (Cited on page 1.)

[MH84] Satoru Miyano and Takeshi Hayashi. Alternating finite automata on omega-words. *Theor. Comput. Sci.*, 32:321–330, 1984. (Cited on pages 9, 88 and 95.)

[Moh97] Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997. (Cited on page 4.)

[Mos91] Andrzej W. Mostowski. Games with forbidden positions. Technical Report 78, Uniwersytet Gdański, Instytut Matematyki, 1991. (Cited on pages 67, 69 and 157.)

[MSS86] David E. Muller, Ahmed Saoudi, and Paul E. Schupp. Alternating automata. The weak monadic theory of the tree, and its complexity. In Laurent Kott, editor, *ICALP*, volume 226 of *LNCS*, pages 275–283. Springer, 1986. (Cited on pages 111, 112, 115, 116 and 135.)

[MS84] David E. Muller and Paul E. Schupp. Alternating automata on infinite objects, determinacy and rabin's theorem. In Maurice Nivat and Dominique Perrin, editors, *Automata on Infinite Words*, volume 192 of *LNCS*, pages 100–107. Springer, 1984. (Cited on pages 67, 69 and 95.)

[MS95] David E. Muller and Paul E. Schupp. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the theorems of rabin, mcnaughton and safra. *Theor. Comput. Sci.*, 141(1&2):69–107, 1995. (Cited on pages 9, 67, 95 and 162.)

[Mur08a] Filip Murlak. Effective topological hierarchies of recognizable tree languages. PhD thesis, Warsaw University, 2008. (Cited on pages 157 and 163.)

[Mur08b] Filip Murlak. Weak index versus borel rank. In Susanne Albers and Pascal Weil, editors, *STACS*, volume 1 of *LIPIcs*, pages 573–584. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2008. (Cited on page 167.)

[Niw86] Damian Niwiński. On fixed-point clones (extended abstract). In Laurent Kott, editor, *ICALP*, volume 226 of *LNCS*, pages 464–473. Springer, 1986. (Cited on page 156.)

[NW98] Damian Niwiński and Igor Walukiewicz. Relating hierarchies of word and tree automata. In Michel Morvan, Christoph Meinel, and Daniel Krob, editors, *STACS*, volume 1373 of *LNCS*, pages 320–331. Springer, 1998. (Cited on page 157.)

[NW03] Damian Niwiński and Igor Walukiewicz. A gap property of deterministic tree languages. *Theor. Comput. Sci.*, 1(303):215–231, 2003. (Cited on page 156.)

[NW05] Damian Niwiński and Igor Walukiewicz. Deciding nondeterministic hierarchy of deterministic tree automata. *Electr. Notes Theor. Comput. Sci.*, 123:195–208, 2005. (Cited on pages 156, 157, 161 and 163.)

[Par87] Michel Parigot. Automata, games, and positive monadic theories of trees. In Kesav V. Nori, editor, *FSTTCS*, volume 287 of *LNCS*, pages 44–57. Springer, 1987. (Cited on page 111.)

[PP04] Dominique Perrin and Jean-Éric Pin. *Infinite Words: Automata, Semigroups, Logic, and Games.* Pure and Applied Mathematics Series. London: Elsevier, 2004. (Cited on page 20.)

[Rab69] Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. Amer. Math. Soc.*, 141:1–35, 1969. (Cited on page 111.)

[Rab70] Michael O. Rabin. Weakly definable relations and special automata. In *Mathematical Logic and Foundations of Set Theory (Proc. Internat. Colloq., Jerusalem, 1968)*, pages 1–23. North-Holland, Amsterdam, 1970. (Cited on pages 10, 111, 126, 127, 129, 135, 137, 149 and 161.)

[RS59] Michael O. Rabin and Dana Scott. Finite automata and their decision problems. *IBM J. Res. Dev.*, 3(2):114–125, April 1959. (Cited on page 5.)

[Sim78] Imre Simon. Limited subsets of a free monoid. In *FOCS*, pages 143–150. IEEE, 1978. (Cited on pages 2 and 35.)

[Tho97] Wolfgang Thomas. Languages, automata, and logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 389–455. Springer, 1997. (Cited on pages 40, 95, 105, 119, 120 and 156.)

[Tor11] Szymon Toruńczyk. Languages of profinite words and the limitedness problem. PhD thesis, University of Warsaw, 2011. (Cited on page 36.)

[VB11] Michael Vanden Boom. Weak cost monadic logic over infinite trees. In Filip Murlak and Piotr Sankowski, editors, *MFCS*, volume 6907 of *LNCS*, pages 580–591. Springer, 2011. (Cited on pages 8, 9, 10, 39, 68, 76, 111 and 137.)

[Wag77] Klaus W. Wagner. Eine topologische charakterisierung einiger klassen regulärer folgenmengen. *Elektronische Informationsverarbeitung und Kybernetik*, 13(9):473–487, 1977. (Cited on page 156.)

[Wal96] Igor Walukiewicz. Pushdown processes: Games and model checking. In Rajeev Alur and Thomas A. Henzinger, editors, *CAV*, volume 1102 of *LNCS*, pages 62–74. Springer, 1996. (Cited on pages 69 and 77.)

Index