Regular cost functions over finite trees

Thomas Colcombet
CNRS *and* LIAFA
Université Paris Diderot,
Paris, France
Email: thomas.colcombet@liafa.jussieu.fr

Christof Löding
Informatik 7
RWTH Aachen University
Aachen, Germany
Email: loeding@cs.rwth-aachen.de

***Abstract*—We develop the theory of regular cost functions over finite trees: a quantitative extension to the notion of regular languages of trees: Cost functions map each input (tree) to a value in $\omega + 1$, and are considered modulo an equivalence relation which forgets about specific values, but preserves boundedness of functions on all subsets of the domain.**

We introduce nondeterministic and alternating finite tree cost automata for describing cost functions. We show that all these forms of automata are effectively equivalent. We also provide decision procedures for them. Finally, following Büchi's seminal idea, we use cost automata for providing decision procedures for cost monadic logic, a quantitative extension of monadic second order logic.

***Keywords*-tree automata; games; limitedness problem; monadic-second order logic;**

I. Introduction

Since the seminal works of Kleene [1] and Rabin and Scott [2], the theory of regular languages is one of the cornerstones in computer science. The theory has then been extended to infinite words [3], to finite trees [4], and to infinite trees [5]. This latter result is of such importance that it is sometimes called the 'mother of all decidability results'.

Recently, the notion of regular cost function of words has been presented as a candidate for being a quantitative extension to the notion of regular languages [6], while retaining most of the fundamental properties of the original theory such as the equivalence with logic and decidability[1]. A cost function is an equivalence class of the functions from the domain (e.g. words or trees) to $\omega + 1$, modulo an equivalence relation $\approx$ which allows some distortion, but preserves the existence of bounds over each subset of the domain. The model of cost functions is a strict extension to the notion of languages. The objective of this paper is to extend this theory to finite trees.

Related work

The models of distance automata and their generalisations were introduced for solving difficult problems in language theory. The most famous of them is the star-height problem, which amounts to compute the nesting of Kleene stars required in order to describe a regular language of finite words by a rational expression. It was raised in 1963 by Eggan [7], and solved 25 years later by Hashiguchi [8], [9], [10], [11]. Hashiguchi used in his proof *distance automata*, i.e., a model of finite automata used for describing functions: a distance automaton is a finite state non-deterministic automaton running over words that can count the number of occurrences of some 'special' states, and hence attach a value to each input. The proof of Hashiguchi is done by reduction to the *limitedness* problem, i.e., the existence of a bound over the function computed by an automaton over its domain. The more recent and elegant proof of Kirsten relies on the same argument, but uses a more general class of automata [12]. Automata similar to the ones of Kirsten have also been introduced independently in the context of verification [13]. Further decision problems that can be reduced to limitedness questions over words are: in language theory the *finite power property* [14], [15] and the *finite substitution problem* [16], [17], and in model theory the *boundedness problem* of monadic formulas over words [18]. The notion of distance automata and its relationship with the tropical semiring has also been the source of many investigations [8], [19], [20], [21], [22]. The automata used above compute a minimum over all their runs over the input of some function. In [23], a new dual form of automata, computing a maximum over all runs of some functions, was introduced. The paper [23] was not focused on those automata by themselves, but at infinitary variants that also used asymptotic considerations. This makes the results difficult to compare (though all limitedness results can be deduced from it). However, the principle of using a dual form of automata plays a very important role in the present work.

Finally, the theory of those automata over words has been unified in [6], in which cost functions are introduced, and suitable models of automata, algebra, and logic for defining them are presented and shown to be equivalent, and furthermore corresponding decidability results are provided. The resulting theory is a neat extension of the standard theory of regular languages. All the limitedness problems

[1]Regular cost functions differ from other quantitative extensions to regular languages in the sense that they cannot be reduced to such other extensions, and that at the same time they retain very strong closure and decidability properties.

from the literature appear as special instances of those results, as well as all the central results known for regular languages.

The present paper is about the extension of this theory of regular cost functions to finite trees. There are different motivations for such an extension. First of all, some results have already been obtained for trees. In [24], the *star-height problem over trees* has been solved by a reduction to the limitedness problem of nested distance desert automata over trees. Though the general approach of the reduction is the same, it requires the introduction of new concepts (such that games) and new results. One such result is the decidability of the limitedness problem for (an extension of) alternating distance tree automata.

Another similar problem is the Mostowski hierarchy problem. The Mostowski hierarchy is the hierarchy induced by the alternation of greatest and least fixpoints in the definition of languages of infinite trees. It corresponds to the number of distinct priorities required for describing the language by means of a parity tree automaton. In [25], it is shown that the problem of deciding the level of a language inside the Mostowski hierarchy can be reduced to the limitedness problem for a form of automata combining distance automata with parity tree automata. The decidability of the later problem is open. One motivation of the present work is to gain a better understanding of cost functions over trees by providing a solid basis for the case of finite trees.

Beside these points, the theory of regular cost functions being an approach to extend the notion of regular languages with counting capabilities, we can expect that more applications are to come, in particular in the field of verification. This calls for the development of the theory in the word case as well as in the tree case.

Contributions

The present paper is the natural continuation of [6]. The objective in [6] was to raise the standard theory of regular languages of (finite) words to the level of cost functions. This paper aims at the same goal, but for (finite) trees.

We define in this paper two dual forms of alternating finite tree automata, namely B- and S-automata (B-automata were already used in [24]). We show that these models are equivalent and are also equivalent to their non-deterministic variant (simulation and duality theorem). We also provide closure and decidability results for those automata.

Establishing the duality/simulation theorem is very similar to establishing the complementation [5], [26] or simulation [27] results for automata over infinite trees. Our proofs follow similar techniques, hence, we use games together with a fine analysis of the shape of strategies in our proofs. We also use new notions such as history-determinism, that have no counterpart in the classical theory.

The remainder of the paper is organised as follows. Cost functions are presented in Section II. Cost automata over words are introduced in Section III. Games are presented in Section IV. All results are combined in Section V, where we study cost tree automata. Finally, all the theory is used in Section VI for presenting cost monadic logic, an extension of monadic logic equivalent to cost tree automata.

II. Cost functions

We are interested in representing functions f over some set E assigning to each element of E a cost in $\omega+1$, i.e., each $x \in E$ is mapped to a natural number or to the first infinite ordinal ω. When using such functions for modelling boundedness problems, the specific value of the function is not of importance. Instead, we are interested in the behaviour of such functions on subsets of the domain, namely on which subsets of the domain the functions are bounded. Using this abstract view we can compare functions: A function $f : E \to \omega+1$ is below some other function $g : E \to \omega+1$, written as $f \preccurlyeq g$, if on each set $X \subseteq E$ on which g is bounded, f is also bounded. This defines a pre-order on functions from E to $\omega+1$. The corresponding equivalence is denoted by $\approx$, i.e., $f \approx g$ if f and g are bounded on the same subsets of their domain. The equivalence classes for $\approx$ are called *cost-functions* (over E).

An alternative definition uses standard comparison of functions modulo a "stretching factor" α. Here $\alpha : \omega \to \omega$ is a non-decreasing mapping that we extend to $\omega+1$ by $\alpha(\omega)=\omega$. For such α we define:

$$f \preccurlyeq_\alpha g \quad \text{iff} \quad f \leq \alpha \circ g \ .$$

It is easy to verify that $f \preccurlyeq g$ iff $f \preccurlyeq_\alpha g$ for some α. We also compare single values using $\preccurlyeq_\alpha$ with the semantics $n \preccurlyeq_\alpha m$ if $n \leq \alpha(m)$. Throughout the paper, α, α' etc. denote such stretching factors, also called *correction functions*.

Cost functions over a set E can be seen as a refinement of the subsets of the base set E. Indeed, given some subset $A \subseteq E$, one defines its *characteristic function* χ_A which maps elements in A to 0, and other elements to ω. We then have for all $A, B \subseteq E$, $A \subseteq B$ iff $\chi_B \preccurlyeq \chi_A$. Consequently, all the information concerning a set is preserved in the cost function of its characteristic function: sets can be seen as particular cases of cost functions. Note also that $\chi_{A\cap B} = \max(\chi_A, \chi_B)$ and $\chi_{A\cup B} = \min(\chi_A, \chi_B)$.

III. Cost automata over words

We need a special notion of words in this work introduced in Section III-A. In Section III-B we introduce objectives, which are the counterpart of accepting conditions in the theory of automata over infinite objects. Word automata are presented in Section III-C and their history-deterministic variant in Section III-D.

A. Words

It is necessary in our framework to have special symbols that identify the end of words. For this reason, a *word alphabet* $\mathbb{A} = \langle \mathbb{A}_1, \mathbb{A}_0 \rangle$ consists of two disjoint sets of letters. The letters in $\mathbb{A}_0$ are called *final letters*, as opposed to *non-final letters* in $\mathbb{A}_1$. A *word* over $\mathbb{A}$ is a sequence in $\mathbb{A}_1^*\mathbb{A}_0$, i.e., consisting of a sequence of non-final letters, and terminating with a final letter. In order to avoid confusion, an element in $\mathbb{A}_1^*$ will be called a *sequence of non-final letters*. For coding words in the usual sense, we append the final letter $\Box$ to their end.

B. Objectives and basic objectives

An objective is a triple $\langle \mathbb{C}, f, \mathit{goal} \rangle$ in which:

- $\mathbb{C}$ is a word alphabet of *actions*, letters in $\mathbb{C}_0$ are called *final actions*,
- the *value mapping* f maps $\mathbb{C}_1^*\mathbb{C}_0$ to $\omega + 1$,
- $\mathit{goal} \in \{\min, \max\}$ is the *goal*.

Intuitively, an objective in the context of a game is assigned to a player, and tells which function to optimise (f), over which domain ($\mathbb{C}_1^*\mathbb{C}_0$), and whether the player's aim is to minimise or maximise this value (goal). The *dual* $\overline{O}$ of an objective O is obtained by changing the goal, i.e., exchanging $\min$ for $\max$, or $\max$ for $\min$. In a game, this represents the goal of the opponent. This notion of objective will be used for games as well as for automata.

We use some basic objectives, that can be seen as the counterpart to basic acceptance conditions, like Büchi, Muller, Streett, Rabin, or parity, in the theory of automata over infinite objects (cf. [28]). The general mechanism for defining our basic objectives is to use a counter that can be incremented by one (i), reset to zero (r) or checked (c). Elements in $\{\mathsf{i}, \mathsf{r}, \mathsf{c}\}$ are called *atomic actions*. We read a sequence of atomic actions over the counter from left to right, and the counter, starting with value 0, evolves according to the actions (the action c does not change the value of the counter). From such a sequence u, one computes the set $C(u) \subseteq \omega$ which collects all the values of the counter when checked (i.e., when the action c is encountered). For example $C(\mathsf{iriiicicri}) = \{3, 4\}$ because the counter values at the two occurrences of c are 3 and 4, respectively.

This base mechanism is instantiated in different ways depending on the situation. In particular, one uses several counters and non-atomic actions (such as ic or cr). Hence, consider a finite set of *counters* Γ, and a set of actions $\mathbb{C}_1 \subseteq \{\mathsf{i}, \mathsf{r}, \mathsf{c}\}^*$ (those can be non-atomic). For each sequence u over the alphabet $\mathbb{C}_1^\Gamma$, we define $C(u)$ as $\bigcup_{\gamma \in \Gamma} C(u_\gamma)$, in which $u_\gamma \in \{\mathsf{i}, \mathsf{r}, \mathsf{c}\}^*$ is obtained by projecting u to its γ-component. In other words, each action in $\mathbb{C}_1^\Gamma$ tells for each counter what sequence of actions has to be performed, and all the values collected by all counters along a sequence of actions u are gathered into the unique set $C(u)$.

The *B-objective* (over counters Γ) is $\mathit{Cost}_B^\Gamma = \langle \langle \{\varepsilon, \mathsf{ic}, \mathsf{r}\}^\Gamma, \{[0], [\omega]\} \rangle, \mathit{cost}_B^\Gamma, \min \rangle$ in which for all $u \in (\{\varepsilon, \mathsf{ic}, \mathsf{r}\}^\Gamma)^*$ and $[x] \in \{[0], [\omega]\}$,

$$\mathit{cost}_B^\Gamma(u[x]) = \sup\left(C(u) \cup \{x\}\right) \ .$$

This corresponds to the definition of B-automata as in [6], except for the final letter in $\{[0], [\omega]\}$. This difference comes from the fact that we use here this last letter for coding accepting and rejecting states of an automaton: The letter $[0]$ should be understood as representing an *accepting state*, while the letter $[\omega]$ corresponds to a non-accepting state because it cancels all the computation before by making the value of $\mathit{cost}_B^\Gamma(u[\omega])$ equal to ω. The examples are given in the context of one counter. The corresponding goal is written Cost_B^1, the value mapping being cost_B^1. In this case, we simplify all the notations and use the alphabet $\{\varepsilon, \mathsf{ic}, \mathsf{r}\}$ without explicit reference to the counter.

The *S-objective* (over counters Γ) is $\mathit{Cost}_S^\Gamma = \langle \langle \{\varepsilon, \mathsf{i}, \mathsf{r}, \mathsf{cr}\}^\Gamma, \{[0], [\omega]\} \rangle, \mathit{cost}_S^\Gamma, \max \rangle$ in which for all $u \in (\{\varepsilon, \mathsf{i}, \mathsf{r}, \mathsf{cr}\}^\Gamma)^*$ and $[x] \in \{[0], [\omega]\}$,

$$\mathit{cost}_S^\Gamma(u[x]) = \inf\left(C(u) \cup \{x\}\right) \ .$$

The same comment applies to the final letters. This time, $[0]$ should be understood as *rejecting*, and the final letter $[\omega]$ as *accepting*.

For technical reasons we also define a hierarchical version of the B-objective, the *hB-objective*. In this case, the set of counters Γ is totally ordered, and one sets $H_\Gamma \subseteq \{\varepsilon, \mathsf{ic}, \mathsf{r}\}^\Gamma$ to be the set of counter actions such that whenever a counter is touched, all smaller counters are reset, i.e., $c \in H_\Gamma$ if for all $\gamma' < \gamma$, $c(\gamma) \neq \varepsilon$ implies $c(\gamma') = \mathsf{r}$. We set $\mathit{Cost}_{hB}^\Gamma$ to be $\langle \langle H_\Gamma, \{[0], [\omega]\} \rangle, \mathit{cost}_B^\Gamma, \min \rangle$. The hB-objective has particularly good properties when used in the context of games, as shown by Theorem 8 below. One can see it in analogy to the parity condition in the theory of games of infinite duration and automata on infinite objects (see [28]), which also has good properties because of its hierarchical nature.

C. Cost-automata over words

A (non-deterministic word) *cost-automaton* $\mathcal{A} = \langle Q, \mathbb{A}, I, O, \Delta_1, F \rangle$ consists of a finite set of *states* Q, a word alphabet $\mathbb{A}$, a set of *initial states* I, an objective $O = \langle \mathbb{C}, f, \mathit{goal} \rangle$, a set of *non-final transitions* $\Delta_1 \subseteq Q \times \mathbb{A}_1 \times \mathbb{C}_1 \times Q$, and a *final transition* mapping $F : Q \times \mathbb{A}_0 \to \mathbb{C}_0$. The set Δ_0 of final transitions is $\{(q, b, F(q, b)) : q \in Q,\ b \in \mathbb{A}_0\}$. It is convenient to see Δ as a word alphabet consisting of Δ_1 and Δ_0. For short, we call *B-automata* (resp. S-automata, hB-automata) the cost automata using B-objectives (resp. S-objectives, hB-objectives).

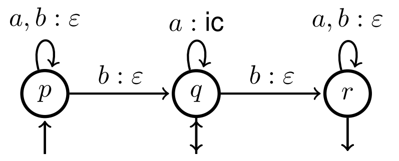
We assume that our automata are *complete* in the sense that for all states q and all letters $a \in \mathbb{A}_1$, there exists a transition of the form (q, a, c, r) in Δ_1. In the case of B-objective and S-objective, completeness can be obtained simply by adding a trap state from which every non-final transition is possible and is a loop, and only rejecting final transitions are possible (this construction is consistent with the definition of the semantics which follows).

A *run* ρ of the automaton is a word $(q_0, a_1, c_1, q_1) \dots (q_{n-1}, a_n, c_n, q_n)(q_n, b, d)$ over the alphabet Δ such that q_0 is initial. The corresponding *input word* $In(\rho)$ is $a_1 \dots a_n b$. One also says that ρ is a run over $a_1 \dots a_n b$. The *output word* $Out(\rho)$ is $c_1 \dots c_n d$. The *value* $f(\rho)$ of a run ρ is $f(Out(\rho))$. The value of a word u over $\mathbb{A}$ depends on the goal and is denoted by $[\![\mathcal{A}]\!]$:

- if *goal* $= \min$, then $[\![\mathcal{A}]\!](u)$ is the infimum[2] of $f(\rho)$ for all runs ρ over u,
- if *goal* $= \max$, then $[\![\mathcal{A}]\!](u)$ is the supremum of $f(\rho)$ for all runs ρ over u.

One also says that the function $[\![\mathcal{A}]\!]$ is *accepted* by $\mathcal{A}$. A cost function f is *accepted* by $\mathcal{A}$ if $[\![\mathcal{A}]\!] \in f$.

Example 1. The following one counter B-automaton accepts the function minseg, which associates to each word over the alphabet $\{a, b\}$ the minimal length of a maximal segment of consecutive occurrences of a. States are represented by circles, and each transition (p, a, c, q) by an edge from p to q labelled by $a : c$. Multiple transitions that differ only by the input letter, e.g., $(p, a, c, q), (p, b, c, q)$ are represented by a single edge labelled $a, b : c$. Initial states are marked by ingoing arrows.



Our automata do not have accepting states, but a final function F. In our case, it maps $\square$ to $[0]$ for the states marked by an outgoing edge, and to $[\omega]$ otherwise. Given an input word, one constructs the optimal run as follows: the automaton guesses non-deterministically the beginning of the shortest a-segment, and jumps to state q at this moment (it can be at the beginning of the word). It then proceeds by counting the length of this interval, until it reaches the end of the word, or a letter b, in which case it goes to the trap state r.

D. History-determinism

In general, the automata we consider cannot be made deterministic, even modulo $\approx$. For instance, the above Example 1 requires to guess the interval of minimal length, and this is 'unavoidable'. In replacement of determinism, we use the notion of *history-determinism* which is a semantic driven weakening of the standard (syntactic) definition of determinism. The use of these automata becomes clear in the context of games (in Lemma 7) for transforming objectives, which usually requires deterministic automata.

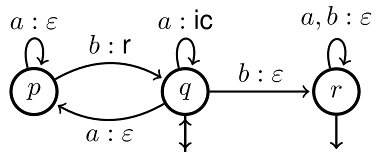
Let us fix a cost automaton $\mathcal{A} = \langle Q, \mathbb{A}, I, O, \Delta, F\rangle$. A *translation strategy*[3] for $\mathcal{A}$ is a mapping δ which maps $\mathbb{A}_1^* \times \mathbb{A}_1$ to Δ_1 and $\mathbb{A}_1^* \times \mathbb{A}_0$ to Δ_0. This mapping tells how to deterministically construct a run of $\mathcal{A}$ over a word. It is transformed into a mapping $\tilde{\delta}$ from $\mathbb{A}_1^*$ to Δ_1^* and from $\mathbb{A}_1^*\mathbb{A}_0$ to $\Delta_1^*\Delta_0$ by $\tilde{\delta}(\varepsilon) = \varepsilon$, and $\tilde{\delta}(va) = \tilde{\delta}(v)\delta(v, a)$ for all $v \in \mathbb{A}_1^*$ and $a \in \mathbb{A}$. Given a word u over $\mathbb{A}$, if $\tilde{\delta}(u)$ is a valid run of $\mathcal{A}$ over u, it is called the run *driven by* δ over u. In the following, we assume that $\tilde{\delta}(u)$ is a valid run for every word u.

An automaton is called *history-deterministic* if there exists α and a family of translation strategies $(\delta_n)_{n\in\omega}$ such that for all words u,

- if *goal* $= \min$ and $[\![\mathcal{A}]\!](u) \leq n$, $f(\tilde{\delta}_n(u)) \leq \alpha(n)$,
- if *goal* $= \max$ and $[\![\mathcal{A}]\!](u) \geq \alpha(n)$, $f(\tilde{\delta}_n(u)) \geq n$.

In other words, when restricting the possible behaviours of the automaton to the ones given by the translation strategies δ_n, then the automaton still computes a function $\approx_\alpha$-equivalent to its normal semantics. One says that the automaton is *α-history-deterministic* when one wants to make the correction function α explicit.

Example 2. The following automaton is an example of a history-deterministic B-automaton accepting the function minseg from Example 1:



We use the same convention as in the previous example concerning accepting states. Let us assume that there is a run of value at most n over the word u. This means that the counter value never exceeds n. Thus the automaton was in state q with counter value 0 after some b or at the beginning of the word, then read at most n consecutive occurrences of letter a, followed by either the end of the word $\square$ or letter b allowing it to jump to state r. This witnesses that $\mathrm{minseg}(u) \leq n$.

Conversely, assume that $\mathrm{minseg}(u) \leq n$. We describe the translation strategy δ_n as a deterministic process for constructing the accepting run reaching r of value at most n. The sole cause of non-determinism in this automaton occurs when in state q, while reading letter a. The automaton can choose either to go to state p, and skip the remaining of the a-segment (call this choice 'skip'), or to stay in state q and increment and check the counter (choice 'continue').

[2]We use infimum and supremum and not min and max for handling the case when there are no runs, using $\inf \emptyset = \omega$ and $\sup \emptyset = 0$.

[3]The name comes from a more general presentation of the notion, in which the notion of a translation strategy can be unified with the standard notion of strategy in games.

There is no freedom in the definition of the translation strategy δ_n, but in this case. The translation strategy resolves this non-determinism by choosing the option 'continue' as long as possible, i.e., as long as the value of the counter is less than n, and by choosing to 'skip' only when it is unavoidable, i.e., when the counter has value n. It is clear that following this translation strategy, the counter will never exceed value n. It is also easy to see that following this translation strategy, a run will terminate in state q or r iff it contains an a-segment of length at most n.

Theorem 3 states the equivalence of all forms of automata.

Theorem 3 (duality, Theorem 1 in [6]). *It is equivalent for a cost function to be accepted by a B-automaton, an S-automaton or an hB-automaton, as well as by their history-deterministic variants.*

We call such cost functions *regular*. The proof of Theorem 3 relies on algebraic techniques and the transformations have a very high complexity. For simpler cases it is possible to provide direct constructions with better complexity, stated in the following lemmas.

Lemma 4. *The function $cost_S^\Gamma$ (resp. $cost_B^\Gamma$) is accepted by an id-history-deterministic B-automaton (resp. S-automaton) of size $2^{|\Gamma|}+1$ (with id the identity function).*

The constructions use similar ideas as in Example 2.

Lemma 5. *The cost function $cost_B^\Gamma$ is accepted by a deterministic hB-automaton of size $|\Gamma|!$.*

The construction of the hB-automaton uses the idea of the latest appearance record construction known from the translation between acceptance conditions for ω-automata (see for instance [28]).

IV. Cost games

As it is the case in the theory of automata over infinite trees, games play a special role in this work. Games are used below both as the framework in which we define the semantics of cost tree automata, and as a theory giving us precious arguments in the proof of equivalences of the various models.

The definition of games is presented in Section IV-A. We show in Section IV-B how games can be composed with history-deterministic automata, and in Section IV-C we present results concerning the shape of winning strategies.

A. Definition

From now, $\mathcal{B}^+(X)$ represents the set of positive Boolean combinations of elements in X. Given some $\varphi \in \mathcal{B}^+(X)$ and some function h from X to $\mathcal{B}^+(Y)$, $\varphi[x \leftarrow h(x)]$ represents the formula φ in which $h(x)$ has been substituted for each occurrence of x for $x \in X$. One also denotes by $\overline{\varphi}$ the *dual* of φ, i.e., φ in which disjunctions and conjunctions have been exchanged. Given $\varphi, \varphi' \in \mathcal{B}^+(X)$, $\varphi \Rightarrow \varphi'$ holds when φ' is a consequence of φ for the usual meaning.

A cost game is a game involving two players, Adam and Eva, the result of which is a value in $\omega+1$. The standard definitions are adapted to the present context. A *cost game* $\mathcal{G} = \langle V, v_0, \delta, O\rangle$ consists of the following components:

- V is the set of *positions*.
- $v_0 \in V$ is the *initial* position.
- $O = \langle \mathbb{C}, f, goal\rangle$ is a basic *objective* (for Eva).
- $\delta : V \to \mathcal{B}^+(\mathbb{C}_1 \times V) \cup \mathbb{C}_0$ is the *control function*. The *non-final moves* in the game are the triples $(v, c, v') \in V \times \mathbb{C}_1 \times V$ such that some (c, v') appears in $\delta(v)$. M_1 is the set of non-final moves. The *final moves* are the pairs $(v, c) \in V \times \mathbb{C}_0$ such that $\delta(v) = c$. M_0 is the set of final moves. One requires that every position either has a successor in M_1, or is final. One finally assumes that the game is of finite duration, i.e., that the graph $\langle V, M_1\rangle$ does not contain any infinite path (and in particular no cycles).

The *dual* $\overline{\mathcal{G}}$ of a game $\mathcal{G}$ is obtained by dualizing the objective and the control relation. Dualization amounts to exchanging the roles of the two players. In particular, all definitions below are given for Eva, but their counterparts for Adam are obtained by dualization of the game.

As for the transitions of automata, we see M_0 and M_1 as a word alphabet. A *play* π is a word of the form $(v_0, a_1, v_1)(v_1, a_2, v_2)\ldots(v_n, a_n) \in M_1^* M_0$ (in which v_0 is indeed the initial position). The *output* of the play π is $Out(\pi) = a_1 \ldots a_n$. The *cost* $f(\pi)$ is $f(Out(\pi))$. A strict prefix of a play is called a *partial play*. Its output is defined accordingly. A *strategy for Eva* σ_{E} is a set of plays such that for every partial play $\pi \in M_1^*$ ending in a position v,

$$\bigwedge\{m \in M(v) \ :\ \pi m \in pref(\sigma_{\mathrm{E}})\} \quad \Rightarrow \quad \delta(v)\ ,$$

in which $M(v)$ is the set of moves of origin v, $pref(\sigma_{\mathrm{E}}) = \{u \ :\ uv \in \sigma_{\mathrm{E}},\ v \in M_1^* M_0\}$, and $\bigwedge S$ denotes the conjunction over all elements from the set S.

When $goal = \min$, Eva aims at minimising over all strategies the maximum value of all plays compatible with the strategy. In other words, the value $value(\sigma_{\mathrm{E}})$ of a strategy for Eva σ_{E} (with respect to objective O) is defined as the supremum of $f(\pi)$ for $\pi \in \sigma_{\mathrm{E}}$, and the value of a game is the infimum of $value(\sigma_{\mathrm{E}})$ for σ_{E} ranging over the strategies for Eva. Dually, when $goal = \max$, $value(\sigma_{\mathrm{E}})$ is defined as the infimum of $f(\pi)$ for $\pi \in \sigma_{\mathrm{E}}$, and the value of a game is the supremum of $value(\sigma_{\mathrm{E}})$ for σ_{E} ranging over the strategies for Eva.

It is well known that games of finite duration are determined, i.e., that the best value that can be obtained by one player is the same as the best value which can be obtained by its opponent. In our case, this is formalised by the following proposition.

Proposition 6. *For all cost games, $\mathit{value}(\mathcal{G}) = \mathit{value}(\overline{\mathcal{G}})$.*

The two following sections give some key arguments for working with games.

B. History-deterministic reduction

We now show how we can compose word automata with games. The goal is to transform a game into an "equivalent" one with a different objective (as it is known from the theory of infinite games, e.g., the transformation of Muller into parity games by the latest appearance record construction, cf. [28]). For this purpose we take the product of a game with the automaton. The game outputs a word, this word is read by the automaton, which in turn yields a new word. The non-determinism of the automaton is controlled by player Eva, meaning that Eva chooses a run of the automaton along the play. Hence, given a game $\mathcal{G} = \langle V, \delta, \langle \mathbb{A}, f, \mathit{goal}\rangle\rangle$, and a cost automaton $\mathcal{A} = \langle Q, \mathbb{A}, I, \langle \mathbb{C}, g, \mathit{goal}\rangle, \Delta, F\rangle$, one defines the product $\mathcal{A} \times \mathcal{G} = \langle Q \times V, \delta', \langle \mathbb{C}, g, \mathit{goal}\rangle\rangle$ in which one sets $\delta'((q, v))$ to be

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

if v is non-final, and $F(\delta(v))$ otherwise.

This construction is standard. However, it is also well known that it fails to have the correct semantics in general: it is not true that, when $\mathcal{A}$ accepts the function f, the game $\mathcal{A} \times \mathcal{G}$ has the same value as $\mathcal{G}$. It is classical that this property holds either if the automaton is deterministic, or if Adam is never allowed to play in the game (in our case if all Boolean formulas are disjunctions). However, the following lemma shows that when composing with history-deterministic automata, good properties are recovered.

Lemma 7. *Let $\mathcal{A}$ be an α-history-deterministic cost automaton over alphabet $\mathbb{A}$ and $\mathcal{G} = \langle V, \delta, \langle \mathbb{A}, [\![\mathcal{A}]\!], \mathit{goal}\rangle\rangle$ be a cost game, then $\mathit{value}(\mathcal{G}) \approx_\alpha \mathit{value}(\mathcal{A} \times \mathcal{G})$.*

Proof: Let $\mathcal{A} = \langle Q, \mathbb{A}, I, \langle \mathbb{C}, g, \mathit{goal}\rangle, \Delta, F\rangle$. Let us treat the case $\mathit{goal} = \min$. We claim that $\mathit{value}(\mathcal{G}) \leq \mathit{value}(\mathcal{A} \times \mathcal{G})$ (this part does not use the history-determinism of the automaton). For this, consider a strategy for Eva σ_{E} in the game $\mathcal{A} \times \mathcal{G}$. We would like to project this strategy into a strategy $\tilde{\sigma_{\mathrm{E}}}$ in the original game $\mathcal{G}$. For this, for each non-final move $m = ((q, v), c, (q', v'))$ in the game $\mathcal{A} \times \mathcal{G}$ one associates a move $\tilde{m} = (v, a, v')$ in $\mathcal{G}$ such that $(q, a, c, q') \in \Delta$, and to each final move $m = ((q, v), c)$, one associates a final move $\tilde{m} = (v, a)$ in $\mathcal{G}$ such that $F(q, a) = c$ (in both cases $\tilde{m}$ exists by definition of $\mathcal{A} \times \mathcal{G}$). One then constructs $\tilde{\sigma_{\mathrm{E}}}$ from σ_{E} by applying this transformation to each move occurring in the strategy. One can check that $\tilde{\sigma_{\mathrm{E}}}$ is a strategy for Eva in $\mathcal{G}$. Furthermore, it is straightforward that $\mathit{value}(\tilde{\sigma_{\mathrm{E}}}) \geq \mathit{value}(\sigma_{\mathrm{E}})$ because the plays in $\tilde{\sigma_{\mathrm{E}}}$ are combined with a run of $\mathcal{A}$, and the cost of plays in σ_{E} is the minimal value of a run of $\mathcal{A}$. Hence $\mathit{value}(\mathcal{G}) \leq \mathit{value}(\mathcal{A} \times \mathcal{G})$.

Conversely, let σ_{E} be a strategy for Eva in the game $\mathcal{G}$ such that $[\![\mathbb{A}]\!](\sigma_{\mathrm{E}}) = n$. Let δ_n be the translation strategy for $\mathcal{A}$. Let $\pi = (v_0, a_1, v_1) \dots (v_k, a_k)$ be a play in σ_{E}. Let $(p_0, a_1, c_1, p_1) \dots (p_k, a_k, c_k)$ be the run driven by δ_n over the word $a_1 \dots a_k = \mathit{Out}(\pi)$, i.e., $\tilde{\delta}_n(\mathit{Out}(\pi))$. Define $\tilde{\pi}$ to be $((p_0, v_0), c_1, (p_1, v_1)) \dots ((p_k, v_k), c_k)$, which is a play in the game $\mathcal{A} \times \mathcal{G}$. By assumption of history-determinism, one knows that $g(\tilde{\pi}) \leq \alpha([\![\mathcal{A}]\!](\pi)) \leq \alpha(n)$. Finally set $\tilde{\sigma_{\mathrm{E}}} = \{\tilde{\pi} \ : \ \pi \in \sigma_{\mathrm{E}}\}$. It is not difficult to check that $\tilde{\sigma_{\mathrm{E}}}$ is a strategy for Eva in the game $\mathcal{A} \times \mathcal{G}$. Furthermore, by the above remark, $g(\sigma_{\mathrm{E}}) \leq \alpha(n)$. Hence, overall, we have established $\mathit{value}(\mathcal{A} \times \mathcal{G}) \preccurlyeq_\alpha \mathit{value}(\mathcal{G})$.

One uses the same argument when $\mathit{goal} = \max$, replacing $\leq$ by $\geq$ and $\preccurlyeq$ by $\succcurlyeq$. ■

What is really interesting in this statement is that it is possible that every winning strategy for Eva in the game $\mathcal{G}$ may require an unbounded quantity of memory (this is the case for S-games in general), while at the same time it is possible to win the game $\mathcal{A} \times \mathcal{G}$ with a bounded quantity of memory (this is the case for B-games). In this respect, this result differs a lot from the standard composition with deterministic automata used in the literature.

C. On the shape of strategies

Given a strategy for Eva σ_{E} in some game $\mathcal{G}$, and some $u \in \mathit{pref}(\sigma_{\mathrm{E}})$, $u^{-1}\sigma_{\mathrm{E}}$ denotes the set $\{v \ : \ uv \in \sigma_{\mathrm{E}}\}$. The strategy σ_{E} is called *positional* if for all $u, v \in \sigma_{\mathrm{E}}$ ending in the same position, $u^{-1}\sigma_{\mathrm{E}} = v^{-1}\sigma_{\mathrm{E}}$. Given a stretching function α, a game $\mathcal{G}$ is *α-positional*, if there exists a positional strategy for Eva σ_{E} in $\mathcal{G}$ such that $\mathit{value}(\sigma_{\mathrm{E}}) \approx_\alpha \mathit{value}(\mathcal{G})$. In other words, by playing positionally, Eva commits an error which is bounded by α. The following result has been established in [24] for $\mathit{Cost}^\Gamma_{hB}$-objectives and the argument can easily be adapted for $\overline{\mathit{Cost}^\Gamma_{hB}}$-objectives.

Theorem 8. *For all finite hierarchical sets of counters Γ, the $\mathit{Cost}^\Gamma_{hB}$- and $\overline{\mathit{Cost}^\Gamma_{hB}}$-games of finite duration are α-positional, in which $\alpha(n) = n^{|\Gamma|}$.*

Note finally an asymmetry here: this result does only hold for the hierarchical B-condition. Using Lemma 5 one can also show that strategies with finite memory are sufficient in B-games (where the size of the memory depends on the number of counters), whereas winning strategies in S-games may require an unbounded quantity of memory in general.

V. Cost tree automata

We introduce here our models of cost automata over trees, and study their properties.

A. Trees

A *ranked alphabet* $\mathbb{A}$ is a finite set of *letters*, each of them having a rank in ω. For $r \in \omega$, $\mathbb{A}_r$ is the set of letters in $\mathbb{A}$ of rank r. Remark that this notation is compatible with the notion of word alphabet, which is equivalent to a ranked alphabet that uses only ranks 0 and 1. The set $\mathcal{T}_{\mathbb{A}}$ of *trees* over the ranked alphabet $\mathbb{A}$ is the least set containing $\mathbb{A}_0$, and such that if $t_1,\dots,t_r$ are trees, and $a \in \mathbb{A}_r$, $a(t_1,\dots,t_r)$ is a tree. A *position* in a tree is a sequence in ω^* such that ε is a position in every tree, and ix is a position in a tree $a(t_1,\dots,t_r)$ iff $1 \le i \le r$ and x is a position in t_i. Given a position x in a tree $t = a(t_1,\dots,t_r)$, $t(x)$ denotes the letter at position x, i.e., a when $x=\varepsilon$, and $t_i(y)$ when $x = iy$. A position x with $t(x)$ of rank 0 is called a leaf. The set of all positions of t is denoted by $pos(t)$.

B. Cost alternating tree automata

A *cost alternating tree automaton* $\mathcal{A} = \langle Q, \mathbb{A}, q_{in}, O, \delta\rangle$ consists of a finite set of states Q, a ranked alphabet $\mathbb{A}$, an *initial state* $q_{in} \in Q$, an objective $O = \langle \mathbb{C}, f, goal\rangle$ and a transition function $\delta \in \left[\bigcup_{i>0} Q \times \mathbb{A}_i \to \mathcal{B}^+([1,i]\times\mathbb{C}_1\times Q)\right] \cup [Q\times\mathbb{A}_0 \to \mathbb{C}_0]$, where $[1,i]$ denotes the set $\{1,\dots,i\}$. The semantics of cost alternating automata is defined in terms of games. Given a tree t over $\mathbb{A}$, one defines the game $\mathcal{A}\times t = \langle Q\times pos(t), (q_{in},\varepsilon), \delta', O\rangle$ by setting $\delta'(p,x) = \delta(p,t(x))$ for $t(x)\in\mathbb{A}_0$, and

$$\delta'((p,x)) = \delta(p,t(x))\,[(n,c,q) \leftarrow (c,(q,xn))]$$

otherwise. One defines $[\![\mathcal{A}]\!](t)$ to be $value(\mathcal{A}\times t)$.

A *cost (non-deterministic) tree automaton* is a cost alternating tree automaton $\mathcal{A} = \langle Q, \mathbb{A}, q_{in}, O, \delta\rangle$ such that there exists $\Delta \subseteq \cup_{i>0} Q\times\mathbb{A}_i\times(\mathbb{C}_1\times Q)^i$ such that for all states q and all $a\in\mathbb{A}_i$ with $i>0$,

$$\delta(q,a) = \bigvee_{(q,a,(c_1,q_1),\dots,(c_i,q_i))\in\Delta} \ \bigwedge_{n\in[1,i]} (n,c_n,q_n)\ .$$

An equivalent definition of $[\![\mathcal{A}]\!]$ that is used in the example below can be given when $\mathcal{A}$ is non-deterministic, say defined by the transition relation Δ. A *run over* t is a pair $\sigma = (r,c)$ consisting of the mapping r from $pos(t)$ to Q and the mapping c from $pos(t)\setminus\{\varepsilon\}$ to $\mathbb{C}$, and such that for all non-leaves $x\in pos(t)$, $(r(x), t(x), (c(x1), r(x1)),\dots,(c(xr), r(r)))\in\Delta$, $r(\varepsilon) = q_0$, and for all leaf $x\in pos(t)$, $c(x) = \delta(r(x),t(x))$. Given a *branch* $x_0 < x_1 < \dots < x_k$, i.e., a maximal sequence of positions ordered by prefix, its *cost* for σ is $f(c(x_1)\dots c(x_k))$. The cost of a run σ is the supremum if $goal = \min$ (otherwise the infimum) of $f(\tau)$ when τ ranges over all branches of the tree. The value $[\![\mathcal{A}]\!](t)$ is the infimum (resp. the maximum if $goal=\max$) over the values of all runs over t.

Example 9. Consider the alphabet $\mathbb{A}$ consisting of $\mathbb{A}_0 = \{a,b\}$ and $\mathbb{A}_2 = \{f\}$ (all other $\mathbb{A}_i$'s are empty). One aims at counting the number of occurrences of leaves labelled by a using a non-deterministic B-automaton. Our automaton uses two states p and q, and the following set of transitions (the $\star$ is for later reference to the transition):

$$\Delta = \left\{\begin{array}{ll} (p,f,(\varepsilon,p),(\varepsilon,p)) & \\ (q,f,(\varepsilon,q),(\varepsilon,p)) & \\ (q,f,(\varepsilon,p),(\varepsilon,q)) & \\ (q,f,(\mathsf{ic},q),(\mathsf{ic},q)) & \star \end{array}\right\} \quad \begin{array}{ll} \delta(p,a) &= [\omega] \\ \delta(p,b) &= [0] \\ \delta(q,a) &= [0] \\ \delta(q,b) &= [\omega] \end{array}$$

We assume that both states are initial (formally this is not covered by our definition, but can be simulated in an easy way by introducing a new initial state).

This automaton is simpler to read as a bottom-up deterministic one. The objective of Eva is to minimise the maximum value over all branches. As a consequence, the state p must necessarily be used for every b-labelled leaf, and the state q over every a-labelled leaf (otherwise the $cost_B^1$ value of the corresponding branch is immediately ω). Then, the transitions force the state p to be used if and only if the subtree rooted in the corresponding position does not contain any a-labelled leaf. Conversely, q is used iff there exist an a-labelled leaf below. Now, the cost of a branch of the run is exactly the number of occurrences of the transition $\star$. This transition is used iff state q is assumed by the run at both children. In other words, the value $S(t)$ computed by the automaton over a tree t is the maximal number of separating positions in a branch, where a separating position is a position below which both subtrees contain an a. It is easy to check that $S(t) \le |t|_a \le 2^{S(t)}$ where $|t|_a$ is the number of occurrences of a-leaves.

Lemma 10. *The classes of functions computed by alternating B-automata, alternating S-automata, and alternating hB-automata are effectively equivalent.*

Proof: From alternating S-automata to alternating B-automata: Consider an S-automaton $\mathcal{A}$ over counters Γ. By Lemma 4, let $\mathcal{S}$ be an id-history-deterministic B-automaton which accepts $cost_S^\Gamma$. One then easily constructs by dualizing $\mathcal{A}$ and product with $\mathcal{S}$, an automaton $\mathcal{B}$ such that for all trees t, $\mathcal{B}\times t$ is a game isomorphic to $\mathcal{S}\times\overline{(\mathcal{A}\times t)}$. We then directly get that for all trees t, $[\![\mathcal{B}]\!](t) = value(\mathcal{B}\times t) = value(\mathcal{S}\times\overline{(\mathcal{A}\times t)}) \overset{(1)}{=} value(\overline{\mathcal{A}\times t}) \overset{(2)}{=} value(\mathcal{A}\times t) = [\![\mathcal{A}]\!](t)$, where (1) is by Lemma 7 for $\alpha = id$, and (2) is by Proposition 6. The same composition principle allows similarly to go from B-automata to S-automata and from B-automata to hB-automata (using Lemma 5). ■

The non-deterministic automata have different 'natural' closure properties. In particular, one uses the operations of inf-projection and sup-projection. Given two ranked alphabets $\mathbb{A}$ and $\mathbb{B}$, a translation from $\mathbb{A}$ to $\mathbb{B}$ is a mapping h from $\mathbb{A}_n$ to $\mathbb{B}_n$ for each n. It is naturally extended into a mapping $\tilde{h}$ from $\mathcal{T}_{\mathbb{A}}$ to $\mathcal{T}_{\mathbb{B}}$ by $\tilde{h}(a(t_1,\dots,t_r)) = h(a)(\tilde{h}(t_1),\dots,\tilde{h}(t_r))$. Given a mapping f from $\mathcal{T}_{\mathbb{A}}$ to $\omega+1$, the $(\inf,h)$*-projection* of f is the mapping $f_{\inf,h}$ from $\mathcal{T}_{\mathbb{B}}$

to $\omega+1$ defined by:

$$f_{\inf,h}(t) = \inf\left\{f(t') \ : \ \tilde{h}(t') = t\right\} \quad (= \inf f(\tilde{h}^{-1}(t)))$$

The $(\sup, h)$-*projection* of f is defined similarly by:

$$f_{\sup,h}(t) = \sup\left\{f(t') \ : \ \tilde{h}(t') = t\right\} \quad (= \sup f(\tilde{h}^{-1}(t)))$$

Lemma 11. *B-automata are closed under* $\min, \max$ *and* $\inf$*-projection, hB-automata are closed under* $\min$ *and* $\inf$*-projection, and S-automata are closed under* $\min, \max$ *and* $\sup$*-projection.*

C. Simulation and duality result

We are now able to state and prove the main result of the paper. It shows the simulation result, i.e., that alternating automata can be transformed into equivalent non-deterministic automata, as well as the duality result, which states that non-deterministic B-automata and non-deterministic S-automata are equivalent. The proof method is inspired from modern presentations (see e.g., [28]) of similar results for automata on infinite trees: the simulation theorem of Muller and Schupp [27] and Rabin's complementation lemma [5].

Theorem 12 (simulation and duality)**.** *It is effectively equivalent for a cost function to be accepted by a tree B-automaton, S-automaton, or hB-automaton, as well as their alternating versions.*

Proof sketch: According to Lemma 10, alternating tree S-, B-, and hB-automata are effectively equivalent. Furthermore, hB-automata are B-automata over a restricted output alphabet. Therefore it is sufficient for us to show how to transform an alternating tree hB-automaton into (1) a non-deterministic tree hB-automaton and (2) a non-deterministic tree S-automaton. We sketch the proof of (1), the proof of (2) uses the same technique.

Consider an alternating tree hB-automaton $\mathcal{A} = \langle Q, \mathbb{A}, q_{in}, Cost_{hB}, \delta\rangle$. Given a tree t, the value $[\![\mathcal{A}]\!](t)$ is defined as the infimum over the values of all strategies σ_{E} for Eva in $\mathcal{A} \times t$. According to Theorem 8 it is sufficient to consider positional strategies. Now note that we can code such a positional strategy by annotating t at each inner node x with all the tuples (p, c, q, n) such that $(c, (q, xn))$ is a possible move from (p, x) according to σ_{E}, and similarly the leaf nodes with tuples (p, c) for the possible σ_{E}-moves from (p, x). Denote this annotated tree by $t_{\sigma_{\mathrm{E}}}$. We construct a tree hB-automaton $\mathcal{B}$ such that $[\![\mathcal{B}]\!](t_{\sigma_{\mathrm{E}}}) \approx_\alpha \mathit{value}(\sigma_{\mathrm{E}})$ for some correction function α. We obtain the desired automaton by applying an $\inf$-projection to $\mathcal{B}$ defined by the mapping that removes the strategy annotations.

The construction of $\mathcal{B}$ works as follows: Consider some path τ through $t_{\sigma_{\mathrm{E}}}$ and define its cost to be the supremum over the costs of all σ_{E}-plays that stay on this path. This defines a cost function over words. It is not very difficult to see that this cost function is regular and therefore there exists a history-deterministic hB-automaton $\mathcal{D}$ computing it (according to Theorem 3). The automaton $\mathcal{B}$ is constructed by simulating $\mathcal{D}$ over all branches of the tree (in each direction $\mathcal{B}$ takes a transition that $\mathcal{D}$ could have taken when reading the corresponding path coded as a word). Since $\mathcal{D}$ is history-deterministic, we obtain that $[\![\mathcal{B}]\!](t_{\sigma_{\mathrm{E}}})$ is the supremum over the costs of the paths through τ computed by $\mathcal{D}$ (formally we apply Lemma 7). This corresponds to the value of σ_{E}, as desired. ■

D. Decidability

In the spirit of algorithms for automata on infinite trees ([28]), we can use games to decide $\preccurlyeq$.

Theorem 13. *The relation* $\preccurlyeq$ *is decidable over regular cost functions of finite trees.*

In particular, the uniform universality problem (whether the function is bounded on the whole domain) is decidable, since it amounts to test whether $f \preccurlyeq 0$. This result was already known from [24] for alternating tree hB-automata.

VI. Cost monadic logic

In this section, we briefly state/recall the consequences of our results in logical terms. Let us recall that monadic second-order logic (monadic logic for short) is the extension of first-order logic with the ability to quantify over sets (i.e., monadic relations). Formally monadic formulae use *first-order variables* ($x, y, \ldots$, ranging over elements of the structure), and *monadic variables* ($X, Y, \ldots$ ranging over sets of elements), existential and universal quantification over both first-order and monadic variables, boolean connectives, the membership predicate ($x \in X$), as well as all the predicates in the relational structure.

In cost monadic logic, one uses a single extra variable N of a new kind, called the *bound variable*, which ranges over non-negative integers. *Cost monadic* logic is obtained from monadic logic by allowing the extra predicate $|X| \leq N$ – in which X is some monadic variable and N is the bound variable – iff it appears *positively* in the formula (i.e., under the scope of an even number of negations). The semantics of $|X| \leq N$ is, as one may expect, to be satisfied if (the valuation of) X has cardinality at most (the valuation of) N. If we push negations to the leaves, one obtains the following syntax:

$$\begin{aligned}\phi ::= {} & \exists x.\phi \quad | \quad \forall x.\phi \quad | \quad \exists X.\phi \quad | \quad \forall X.\phi \\ | \quad & \phi \vee \phi \quad | \quad \phi \wedge \phi \quad | \quad x \in X \quad | \quad x \notin X \\ | \quad & R(x_1, \ldots, x_r) \quad | \quad \neg R(x_1, \ldots, x_r) \quad | \quad |X| \leq N\end{aligned}$$

in which $x, x_1, \ldots, x_r$ are first-order variables, X is a monadic variable, and R is some predicate symbol of arity r.

Given a sentence ϕ of cost monadic logic (i.e., with N as sole free variable), let us write $\mathcal{S}, n \models \phi$ when the formula ϕ holds over the relational structure $\mathcal{S}$ when the bound variable N takes value n. From the positivity requirement on the occurrences of the predicates $|X| \leq N$, it is clear that $\mathcal{S}, n \models \phi$ implies $\mathcal{S}, m \models \phi$ for all $m \geq n$. We use sentences of cost monadic logic for defining values over structures as follows. Given a cost monadic sentence ϕ and a relational structure $\mathcal{S}$, one defines $[\![\phi]\!](\mathcal{S}) \in \omega + 1$ as follows:

$$[\![\phi]\!](\mathcal{S}) = \inf\{n \ : \ \mathcal{S}, n \models \phi\} \ .$$

Example 14. When representing a digraph as a structure, the elements are the vertices of the digraph, and the predicate $\mathrm{edge}(x, y)$ expresses the existence of an edge of source x and target y. The monadic formula $\mathtt{reach}(x, y, X)$:

$$\begin{aligned} &\mathtt{reach}(x, y, X) ::= \forall Z. \\ &\quad (x \in Z \wedge \forall z \in Z.\forall z' \in X.\ \mathrm{edge}(z, z') \rightarrow z' \in Z) \\ &\qquad\qquad\qquad\qquad\qquad\qquad \rightarrow \quad y \in Z \end{aligned}$$

describes the existence of a path in a (directed) graph from vertex x to vertex y such that all vertices appearing in the path, but the first one, belong to X. Indeed, it expresses that every set Z containing x and closed under taking edges ending in X, also contains y. Consider now the following cost monadic sentence:

$$\mathtt{diameter} ::= \forall x, y.\exists X.|X| \leq N \wedge \mathtt{reach}(x, y, X).$$

It defines the diameter of a graph: the diameter of a graph is the least n such that for all pair of states x, y, there exists a set of size at most N allowing to reach y from x. Remark that the formula produces value ω if the graph is not strongly connected.

We are interested in using cost monadic logic for defining values over finite trees. In the case of trees over a ranked alphabet $\mathbb{A}$, the elements of the structure are the positions in the tree, and there is a predicate a of arity $r + 1$ for each letter a of rank r. The statement $a(x, x_1, \ldots x_r)$ holds if the letter at position x is a, and its children are, from left to right, $x_1, \ldots, x_r$.

Over (finite or infinite) words as well as (finite or infinite) trees, the expressiveness of monadic logic coincide with standard forms of automata [29], [3], [4], [5]. Those fundamental results are all established in the same way (cf. [28]). In our case, we obtain the following result.

Theorem 15. *A cost function over finite trees is regular if and only if it is definable in cost monadic logic.*

Proof: From logic to automata. As in the case of monadic logic, one shows that to each connector of the logic corresponds an operation under which regular cost functions are closed. For instance, consider a cost monadic formula $\phi \vee \psi$, then one easily shows that $[\![\phi \vee \psi]\!] = \min([\![\phi]\!], [\![\psi]\!])$. It follows that disjunction corresponds to the $\min$ operation over cost functions. Pushing further this relationship, one gets that conjunction corresponds to the $\max$ operation, monadic existential quantification (and also first-order existential quantification as a particular case) corresponds to $\inf$-projection, and universal quantification corresponds to $\sup$-projection. Concerning the constants, the only novelty compared to standard monadic logic is the predicate $|X| \leq N$. However, by definition, $[\![|X| \leq N]\!]$ evaluates to the cardinal of X. The corresponding cost function $|\cdot|_{\mathbb{B}}$ associates to each $\mathbb{A}$-tree the number of positions labelled by letters in $\mathbb{B} \subseteq \mathbb{A}$. This cost function is regular, using a slight extension of Example 9.

From automata to logic. One writes a formula guessing a run and computing its value, as usual. ■

Corollary 16. *The relation $\preccurlyeq$ is decidable over cost monadic definable functions over finite trees.*

VII. Conclusion

In this paper we have extended the theory of regular cost functions to the case of finite trees, showing all equivalence, closure and decidability results we could expect. The techniques involved are game-theoretic (in a way similar to the theory of languages of infinite trees), and require the use of new notions such as history-determinism. A challenging continuation would be the extension of those results to infinite trees. This would imply the decidability of the Mostowski hierarchy by [25].

References

[1] S. C. Kleene, "Representation of events in nerve nets and finite automata," in *Automata Studies*, C. E. Shannon and J. McCarthy, Eds. Princeton University Press, Princeton, New Jersey, 1956, pp. 3–42.

[2] M. O. Rabin and D. Scott, "Finite automata and their decision problems," *IBM J. Res. and Develop.*, vol. 3, pp. 114–125, Apr. 1959.

[3] J. R. Büchi, "On a decision method in restricted second order arithmetic," in *Proceedings of the International Congress on Logic, Methodology and Philosophy of Science*. Stanford Univ. Press, 1962, pp. 1–11.

[4] J. W. Thatcher and J. B. Wright, "Generalized automata theory with an application to a decision problem in second-order logic," *Math. Syst. Theory*, vol. 2, pp. 57–81, 1968.

[5] M. O. Rabin, "Decidability of second-order theories and automata on infinite trees," *Trans. Amer. Math. soc.*, vol. 141, pp. 1–35, 1969.

[6] T. Colcombet, "The theory of stabilisation monoids and regular cost functions," in *36th ICALP*, ser. Lecture Notes in Comput. Sci., no. 5556. Rhodos: Springer, Jul. 2009, pp. 139–150.

[7] L. C. Eggan, "Transition graphs and the star-height of regular events," *Michigan Math. J.*, vol. 10, pp. 385–397, 1963.

[8] K. Hashiguchi, "Regular languages of star height one," *Information and Control*, vol. 53, no. 3, pp. 199–210, 1982.

[9] ——, “Limitedness theorem on finite automata with distance functions,” *J. Comput. Syst. Sci.*, vol. 24, no. 2, pp. 233–244, 1982.

[10] ——, “Representation theorems on regular languages,” *J. Comput. Syst. Sci.*, vol. 27, no. 1, pp. 101–115, 1983.

[11] ——, “Relative star height, star height and finite automata with distance functions,” in *Formal Properties of Finite Automata and Applications*, 1988, pp. 74–88.

[12] D. Kirsten, “Distance desert automata and the star height problem,” *RAIRO – Theor. Inform. Appl.*, vol. 3, no. 39, pp. 455–509, 2005.

[13] P. A. Abdulla, P. Krcál, and W. Yi, “R-automata,” in *Proceedings of the 19th International Conference on Concurrency Theory, CONCUR’08*, ser. Lecture Notes in Comput. Sci., vol. 5201. Springer, 2008, pp. 67–81.

[14] I. Simon, “Limited subsets of a free monoid,” in *FOCS*. IEEE, 1978, pp. 143–150.

[15] K. Hashiguchi, “A decision procedure for the order of regular events,” *Theoret. Comput. Sci.*, vol. 8, pp. 69–72, 1979.

[16] S. Bala, “Regular language matching and other decidable cases of the satisfiability problem for constraints between regular open terms,” in *STACS*, ser. Lecture Notes in Comput. Sci., vol. 2996. Springer, 2004, pp. 596–607.

[17] D. Kirsten, “Desert automata and the finite substitution problem,” in *STACS*, ser. Lecture Notes in Comput. Sci., vol. 2996. Springer, 2004, pp. 305–316.

[18] A. Blumensath, M. Otto, and M. Weyer, “Boundedness of monadic second-order formulae over finite words,” in *36th ICALP*, ser. Lecture Notes in Comput. Sci. Springer, Jul. 2009, pp. 67–78.

[19] K. Hashiguchi, “New upper bounds to the limitedness of distance automata,” *Theor. Comput. Sci.*, vol. 233, no. 1–2, pp. 19–32, 2000.

[20] H. Leung and V. Podolskiy, “The limitedness problem on distance automata: Hashiguchi’s method revisited,” *Theoret. Comput. Sci.*, vol. 310, no. 1-3, pp. 147–158, 2004.

[21] I. Simon, “On semigroups of matrices over the tropical semiring,” *ITA*, vol. 28, no. 3-4, pp. 277–294, 1994.

[22] A. Weber, “Finite-valued distance automata,” *Theoret. Comput. Sci.*, vol. 134, no. 1, pp. 225–251, 1994.

[23] M. Bojańczyk and T. Colcombet, “Bounds in ω-regularity,” in *Proceedings of the 21th IEEE Symposium on Logic in Computer Science (LICS 2006)*, pp. 285–296.

[24] T. Colcombet and C. Löding, “The nesting-depth of disjunctive μ-calculus for tree languages and the limitedness problem,” in *CSL*, ser. Lecture Notes in Comput. Sci., no. 5213. Bertinoro: Springer, Sep. 2008, pp. 416–430.

[25] ——, “The non-deterministic Mostowski hierarchy and distance-parity automata,” in *35th ICALP*, ser. Lecture Notes in Comput. Sci., no. 5126. Reykjavik: Springer, Jul. 2008, pp. 398–409.

[26] Y. Gurevich and L. Harrington, “Trees, automata and games,” in *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, STOC ’82*, 1982, pp. 60–65.

[27] D. Muller and P. E. Schupp, “Alternating automata on infinite objects, determinacy and Rabin’s theorem,” in *Automata on Infinite Words*, ser. Lecture Notes in Comput. Sci., M. Nivat and D. Perrin, Eds., vol. 192. Springer, 1985, pp. 100–107.

[28] W. Thomas, “Languages, automata and logic,” in *Handbook of language theory*, G. Rozenberg and A. Salomaa, Eds. Springer Verlag, 1997, vol. 3, ch. 7, pp. 389–455.

[29] J. R. Büchi, “Weak second-order arithmetic and finite automata,” *Z. Math. Logik Grundl. Math.*, vol. 6, pp. 66–92, 1960.