

FEASIBILITY OF LEARNING WEIGHTED AUTOMATA ON A SEMIRING

LAURE DAVIAUD¹ AND MARIANNE JOHNSON²

ABSTRACT. Since the seminal work by Angluin and the introduction of the L^* -algorithm, active learning of automata, by membership and equivalence queries, has been extensively studied and several generalisations have been developed to learn various extensions of automata. For weighted automata, restricted cases have been tackled in the literature and in this paper we chart the boundaries of the Angluin approach (using a class of hypothesis automata constructed from membership and equivalence queries) applied to learning weighted automata over a general semiring. We show precisely the theoretical limitations of this approach and classify functions with respect to how ‘guessable’ they are (corresponding to the existence and abundance of solutions of certain systems of equations). We provide a syntactic description of the boundary condition for a correct hypothesis of the prescribed form to exist. Of course, from an algorithmic standpoint, knowing that (many) solutions exist need not translate into an effective algorithm to find one; we conclude with a discussion of some known conditions (and variants thereof) that suffice to ensure this, illustrating the ideas over several familiar semirings (including the natural numbers) and pose some open questions for future research.

1. INTRODUCTION

Imagine the following situation: you want to model a computer system (for verification purposes for example) with some mathematical abstraction, but the internal state of the system cannot be accessed. Think for example of very complex systems such as AI systems, that can realistically only be viewed as black-boxes: the system can be tested, you can feed it some chosen inputs and observe the outputs. Given these (finite set of) pairs input-output, you create a model. Suppose you also have a way to decide whether your model matches (or, more realistically in the application fields, is close enough to) the system’s behaviour. If this is not the case, the system gives you an input that behaves incorrectly in your model and you can try again. This is active learning.

Automata learning. In her seminal paper [4], Angluin introduced the L^* -algorithm where a learner tries to guess a rational language L known only by an oracle. The learner is allowed to ask two types of queries: (1) membership queries where the learner chooses a word w and asks the oracle whether w belongs to L , and (2) equivalence queries where the learner chooses an automaton \mathcal{A} and asks the oracle whether \mathcal{A} recognises L ; if this is not the case, the oracle gives a word (called counter-example) witnessing this fact. The L^* -algorithm allows the learner to correctly guess the minimal deterministic automaton recognising L in a number of membership and equivalence queries polynomial in its size and the length of counter-examples given by the oracle. An overview of applications of automata learning can be found in [22, 19, 27]. In particular, the L^* -algorithm has been used in [35] to extract automata from recurrent neural networks.

Weighted automata. Weighted automata are a quantitative extension of automata where transitions are weighted in a semiring, allowing to model probabilities, costs or running time of programs [26, 16, 17]. They find applications in image compression [20, 14], language and speech processing [24, 25], bioinformatics [1], formal verification [12, 3], analysis of on-line algorithms [2] and probabilistic systems [31]. Notably, they have also been used recently to model recurrent neural networks [34].

¹UNIVERSITY OF EAST ANGLIA, UK

²UNIVERSITY OF MANCHESTER, UK

Key words and phrases. Weighted Automata, Learning, Semiring, Angluin algorithm.

Extending automata learning. Active learning has been extended to weighted automata but only for restricted classes: it has first been studied mainly for semirings that are fields [8, 7], and then principal ideal domains [28]. The Angluin framework has also been generalised to other extensions of automata and transducers [32, 33, 11, 9, 6, 5, 10, 15, 23, 30]. In [13] (and at first [29]), a categorical approach was proposed to encompass many of these cases. However, although this encompasses the case of weighted automata over fields, this framework does not include weighted automata in general. As often with weighted automata, finding a unified approach, working for all semirings, is not easy.

In [28], the authors set a general framework for learning weighted automata, presenting a general learning algorithm, and focus on specifying two conditions on the semirings (the ascending chain condition and the progress measure), ensuring termination of the algorithm - this allows them to develop a learning algorithm à la Angluin for automata weighted on a principal ideal domain. They also present one specific example of automata over the non-negative integers for which their algorithm does not terminate.

Contribution. In the present paper, we somehow take the reverse approach: we look globally at all functions computed by finite state weighted automata and investigate whether they can be learnt. For an arbitrary semiring, we define levels of learnability and a hierarchy of functions with respect to these levels: the ones that definitely cannot be learned à la Angluin, the ones that might be, provided we are lucky or have a good algorithmic strategy, and the ones that definitely will be. In the literature, focus is (reasonably) on algorithmic issues. In this paper, we focus less on algorithmics but rather on picturing the landscape for active learning in arbitrary semirings, and showing the limitations. We do discuss a general algorithm (the one from [28]) and give some conditions on the target functions to ensure termination. In doing so, we refine the ascending chain condition and progress measure from [28] to pinpoint exactly what is needed.

Within this framework, we then consider specific well-studied examples of semirings in this context and provide a detailed picture for these. Note that the learning algorithm is based on equivalence queries, and one might ask whether the undecidability of equivalence for certain semirings [21] is problematic. Since in the application fields, equivalence can rarely be used as such (and often testing or approximation might be used instead), we believe having a framework that applies even in these cases is still of interest.

Structure of the paper. In Section 2, we recall the definition of weighted automata, some specific examples, and outline the type of learning we are considering. We end this section with a set of questions that arise naturally and that we will try to answer in the rest of the paper. In Section 3, we explain what we mean by levels of learnability and define classes of functions depending on them. We give several characterisations for these functions, and refine the notions of ascending chain condition and progress measure from [28]. In Section 4, we picture the hierarchy based on the previous definitions, prove its strictness and give conditions for collapse. Finally, in Section 5, we address algorithmic issues.

2. LEARNING WEIGHTED AUTOMATA

In this section, we recall the definition of weighted automata (on a semiring) and the type of learning we are considering.

2.1. Weighted automata and specific semirings. A finite alphabet is a finite set of symbols, called letters. A finite word is a finite sequence of letters and ε denotes the empty word. We will also denote by $|w|$ the length of a word w and by $|w|_a$ the number of occurrences of the letter a in w .

A monoid $(M, +, 0)$ is a set M equipped with a binary operation $+$ that is associative and has a neutral element 0 . A **semiring** $(S, \oplus, \otimes, 0_S, 1_S)$ is a set S equipped with two operations such that $(S, \oplus, 0_S)$ and $(S, \otimes, 1_S)$ are monoids, \oplus is commutative, \otimes distributes on the left and the right of \oplus and 0_S is a zero for \otimes . A semiring is said to be commutative if \otimes is commutative. Given a semiring $(S, \oplus, \otimes, 0_S, 1_S)$, two matrices A and B over S can be multiplied as usual by $(AB)_{i,j} = \bigoplus_{k=1,\dots,n} (A_{i,k} \otimes B_{k,j})$, provided the number, n , of columns of A is equal to the number of rows of B .

Definition 2.1. A weighted automaton \mathcal{A} on a semiring $(S, \oplus, \otimes, 0_S, 1_S)$ over alphabet Σ is defined by: a finite set of states Q ; initial-state vector $\alpha \in S^Q$ (which we view as a row vector); for all letters a of Σ , transition matrices $M(a) \in S^{Q \times Q}$; and final-state vector $\eta \in S^Q$ (which we view as a column vector). The automaton \mathcal{A} computes a function $f_{\mathcal{A}} : \Sigma^* \rightarrow S$ via $f_{\mathcal{A}}(w_1 \cdots w_k) = \alpha M(w_1) \cdots M(w_k) \eta$. We say that $f_{\mathcal{A}}(w)$ is the value computed by \mathcal{A} on input w .

A weighted automaton can be seen as a graph with set of vertices Q , and transitions $p \xrightarrow{a:m} q$ if $M(a)_{p,q} = m$ where m is called the weight of the transition. A run from a state p to a state q is a path in the graph, its weight is the product of α_p by the product of the weights on the transitions (taken in the order traversed) by η_q . The value computed on a word w is the sum of all weights of runs labelled by w . A state p such that α_p (resp. η_p) is not 0_S might be referred to as an initial (resp. final) state with initial (resp. final) weight α_p (resp. η_p). We use both the matrix and graph vocabulary in this paper, as convenient.

Throughout the paper, Σ denotes a finite alphabet and S a semiring.

Specific semirings and examples. Besides giving a general framework valid for any semiring, we consider several specific well-studied semirings that arise frequently in application areas. We will consider the semiring $(\mathbb{R}, +, \times, 0, 1)$ and its restriction to non-negative reals $\mathbb{R}_{\geq 0}$, integers \mathbb{Z} , and non-negative integers \mathbb{N} . We also consider the semiring $(\mathbb{R} \cup \{-\infty\}, \max, +, -\infty, 0)$ and its restrictions to $\mathbb{Z} \cup \{-\infty\}$ and $\mathbb{N} \cup \{-\infty\}$, denoted by \mathbb{R}_{\max} , \mathbb{Z}_{\max} and \mathbb{N}_{\max} and the Boolean semiring \mathbb{B} . Note that **all these semirings are commutative**. Finally, we will consider the non-commutative semiring with domain the finite subsets of Σ^* , with operations given by union and concatenation ($XY = \{xy \mid x \in X, y \in Y\}$) and neutral elements \emptyset and $\{\varepsilon\}$. This semiring will be denoted $\mathcal{P}_{\text{fin}}(\Sigma^*)$.

Example 2.2. We consider $\Sigma = \{a, b\}$. Figure 1 depicts weighted automata where the initial (resp. final) states with weight 1_S are indicated with an ingoing (resp. outgoing) arrow, the other states have initial or final weights 0_S . All six examples will be used in Section 4 and the example in Figure 1a will be used as a running example to illustrate the different notions we introduce. The automaton in Figure 1a is considered over \mathbb{N}_{\max} and computes the number of a 's if the word starts with an a , the number of b 's if the words starts with a b , and $-\infty$ on the empty word. The automaton depicted in Figure 1b will be considered over \mathbb{R}_{\max} and computes the length of the longest block of consecutive a 's in a word. The automaton in Figure 1c is considered over \mathbb{N} and on input w computes: $2^{|w|_a}$ if w starts with an a ; $2^{|w|_b}$ if w starts with a b ; and 0 if w is the empty word. The automaton from Figure 1d is viewed over $\mathbb{R}_{\geq 0}$ and computes $2^n - 1$ on the words a^n for all positive integers n , and 0 on any other word. Finally the automata in Figures 1e and 1f are over $\mathcal{P}_{\text{fin}}(\Sigma^*)$. The first one has outputs: \emptyset on the empty word; $\{a^{|w|_a}\}$ on all words w starting with an a ; and $\{b^{|w|_b}\}$ on all words w starting with a b . The second one outputs $\{a^{|w|_a}, b^{|w|_b}\}$ on all input words w .

Linear combinations. Since we are going to consider non-commutative semirings, care needs to be taken when talking about linear combinations. Given a set Y , an element x of S^Y and λ in S , $\lambda \otimes x$ denotes the element of S^Y computed from x by multiplying every component of x by λ on the left. Given a subset $X \subseteq S^Y$, a left-linear combination over X is one of the form $\bigoplus_{x \in X} \lambda_x \otimes x$ for some λ_x in S , with only finitely many λ_x different from 0_S . The **left-semimodule** generated by X is then the subset of S^Y containing those elements that can be written as a left-linear combination over X . An element of the left-semimodule generated by X will simply be said to be left-generated by X .

Mirror. Finally, we will use the mirror operation on functions $\Sigma^* \rightarrow S$. The mirror of a word $w = w_1 w_2 \cdots w_n$ where for all $i = 1, \dots, n$, $w_i \in \Sigma$ is the word $\bar{w} = w_n \cdots w_2 w_1$. For $f : \Sigma^* \rightarrow S$, the mirror of f , denoted by $\bar{f} : \Sigma^* \rightarrow S$ is defined as $\bar{f}(w) = f(\bar{w})$. Given a weighted automaton \mathcal{A} , the mirror of \mathcal{A} is obtained by reversing its transitions and swapping the initial-state vector and the final-state vector. Note that if S is commutative and f is computed by a weighted automaton \mathcal{A} over S then \bar{f} is computed by the mirror of \mathcal{A} . However, there are cases when S is non-commutative where a function can be computed by a weighted automaton over S but its mirror cannot (for example the function $w \mapsto \{w\}$ over $\mathcal{P}_{\text{fin}}(\Sigma^*)$ is computed by a weighted automaton over $\mathcal{P}_{\text{fin}}(\Sigma^*)$ but the mirror of this function ($w \mapsto \{\bar{w}\}$) is not).

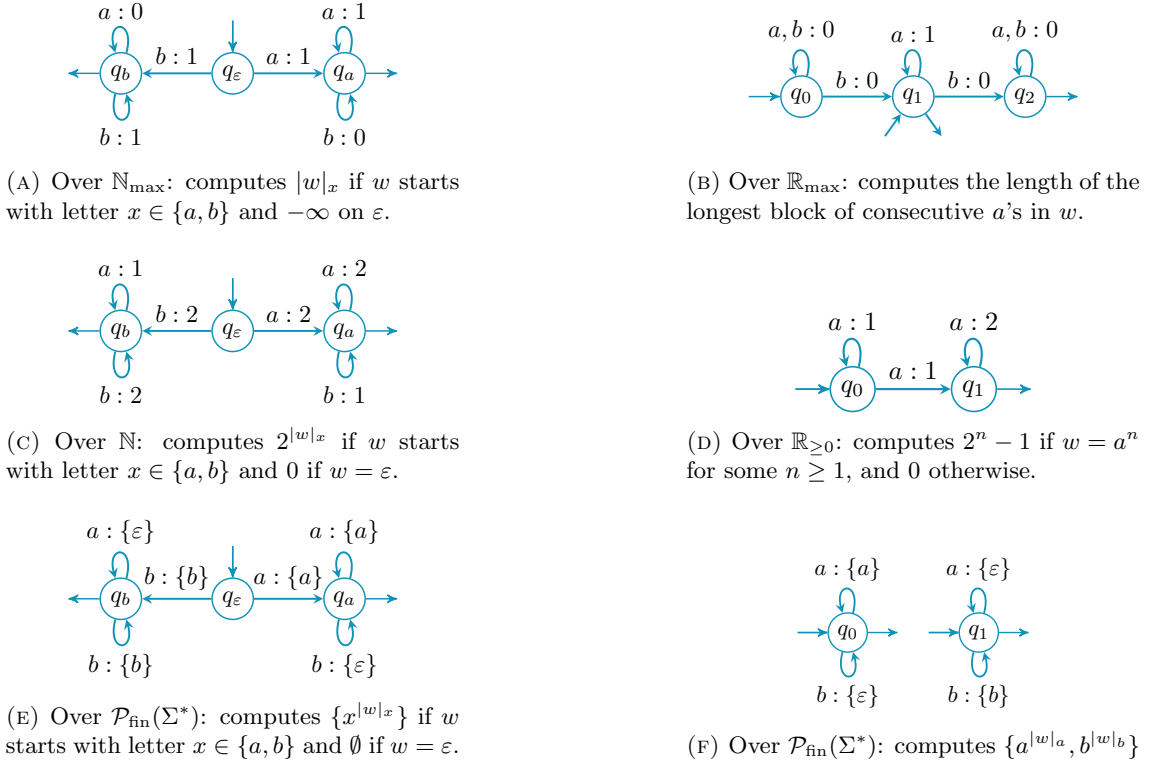


FIGURE 1. Examples of weighted automata and the values computed on input word w .

2.2. Learning with membership and equivalence queries. We are investigating one type of learning, by membership and equivalence queries [4, 8, 7]. In certain circumstances (i.e. by placing conditions on the weighted automaton and the semiring) it is known that one can learn efficiently a weighted automaton computing the function f when provided with finitely many data points $f(w)$ and some steering. Specifically, suppose that there is an oracle who can answer two types of questions about f :

- **Membership** - when provided with a word w : what is the value of f on w ?
- **Equivalence** - when provided with a weighted automaton \mathcal{H} : is f the function computed by \mathcal{H} ? If the oracle answers “no”, it will also give an input word z on which \mathcal{H} performs incorrectly.

A ‘membership’ query is so-named since in the case of the Boolean semiring, this question amounts to asking whether a given word belongs to the rational language recognised by the automaton computing f . More generally, a membership query provides a mechanism for a learner to gather data points $f(w)$ from which they may attempt to build an hypothesis automaton \mathcal{H} consistent with these data points. An equivalence query provides a mechanism for a learner to test their hypothesis and receive useful feedback, in the form of success or counterexample. Throughout this paper we consider hypothesis automata constructed in a prescribed manner from a partial Hankel matrix, as we shall now explain. What follows has been introduced and studied in the literature and we give the proofs of the results for completeness only. Our presentation differs slightly and leads to questions that are answered in Sections 3 and 4 and represent our main contribution.

Hankel matrix. We will use the notation $\langle x \rangle$ for elements of S^{Σ^*} viewed as infinite rows of elements of S , each column being indexed by a word of Σ^* ; thus for a word $w \in \Sigma^*$ we write $\langle x \rangle_w$ to denote the w -th component of $\langle x \rangle$. The restriction of an infinite row to entries indexed by a subset $Z \subseteq \Sigma^*$ will be denoted by $\langle x \rangle_Z$. Given an infinite row $\langle x \rangle \in S^{\Sigma^*}$, and a letter a of Σ , we denote by $\langle xa \rangle$ the infinite row defined by $\langle xa \rangle_w = \langle x \rangle_{aw}$ for all words w . Given a function $f : \Sigma^* \rightarrow S$, the infinite matrix F whose rows and columns are indexed by Σ^* and with entries $F_{w,w'} = f(w w')$ is called the Hankel matrix of f . If x is a word and f and F are clear from context,

we shall write $\langle x \rangle$ to denote the infinite row indexed by x in F . Note that if x is a word, f a function and F its Hankel matrix, this definition is consistent: $\langle xa \rangle_u = F_{xa,u} = f(xau) = F_{x,au} = \langle x \rangle_{au}$. For a set of rows X and $\langle x \rangle$ in X , we will sometimes simplify notation by writing $x \in X$.

Example 2.3. Consider f the function computed by our running example from Figure 1a. Its Hankel matrix is given by $F_{\varepsilon,\varepsilon} = -\infty$, $F_{\varepsilon,aw} = |aw|_a$, $F_{\varepsilon,bw} = |bw|_b$, $F_{aw,w'} = |aww'|_a$ and $F_{bw,w'} = |bww'|_b$ for all words w, w' .

Closed sets and generating sets. Let $Z \subseteq \Sigma^*$ and $f : \Sigma^* \rightarrow S$. A subset $X \subseteq S^{\Sigma^*}$ is said to be: (i) row-closed if for all $\langle x \rangle$ in X and $a \in \Sigma$, $\langle xa \rangle$ is left-generated by X ; (ii) row-closed on Z if for all $\langle x \rangle$ in X and $a \in \Sigma$, $\langle xa \rangle_Z$ is left-generated by the elements of X restricted to Z ; (iii) row-generating for f if every row of the Hankel matrix of f is left-generated by X . By an abuse of notation, we say that a set of words¹ Q is row-closed (resp. row-closed on Z , row-generating for f) if the set $X = \{\langle q \rangle : q \in Q\}$ has this property. Observe that if X is row-closed and left-generates the row $\langle \varepsilon \rangle$ of the Hankel matrix of f then X is row-generating for f .

Example 2.4. Consider again the function f computed by our running example from Figure 1a over the semiring \mathbb{N}_{\max} whose multiplicative operation \otimes is given by usual addition. The singleton set $\{\langle \varepsilon \rangle\}$ is not row-closed: indeed, $\langle a \rangle$ is not left-generated by this element since $\langle \varepsilon \rangle_\varepsilon = -\infty$ but $\langle a \rangle_\varepsilon = 1$, and hence there is no left-linear combination. The two element set $\{\langle \varepsilon \rangle, \langle a \rangle\}$ is row-closed on $\{\varepsilon\}$, since $\langle b \rangle_\varepsilon = \langle ab \rangle_\varepsilon = 1 = 0 + 1 = 0 \otimes \langle a \rangle_\varepsilon$ and $\langle aa \rangle_\varepsilon = 2 = 1 + 1 = 1 \otimes \langle b \rangle_\varepsilon$. The set $\{\langle \varepsilon \rangle, \langle a \rangle, \langle b \rangle\}$ is row-closed and also row-generating since for $x \in \{a, b\}$, $\langle xw \rangle = |w|_x \otimes \langle x \rangle$.

If one can find a finite set of infinite rows that is row-closed and such that the row $\langle \varepsilon \rangle$ is left-generated, then one can construct an automaton computing the correct function. We call this a Hankel automaton and describe its construction below. However, in practice, one does not have access to infinite rows, but (using finitely many membership queries) a finite submatrix of the Hankel matrix can be obtained. An hypothesis automaton is defined in the same way as a Hankel automaton, replacing the finite set of infinite rows by a finite set of *rows of the Hankel matrix* (i.e. indexed by a set of words) *restricted to a finite set of columns*. For Boolean automata, it is always possible to find a finite set amongst the rows of the Hankel matrix and a finite set of columns such that the corresponding hypothesis automaton computes the target function. The L^* -algorithm is based on this idea.

Hankel automata and hypothesis automata. Given a function $f : \Sigma^* \rightarrow S$, a finite subset Q of S^{Σ^*} and a subset T of Σ^* , we define $\Lambda_{Q,T} \subseteq S^Q \times S^{Q \times \Sigma \times Q}$ to be the set of tuples $\lambda = (\lambda_q^\varepsilon, \lambda_{q,a,p})_{q \in Q, a \in \Sigma, p \in Q}$ such that:

- $\langle \varepsilon \rangle_T = \bigoplus_{q \in Q} \lambda_q^\varepsilon \otimes \langle q \rangle_T$, and
- for all $q \in Q$ and all $a \in \Sigma$, $\langle qa \rangle_T = \bigoplus_{p \in Q} \lambda_{q,a,p} \otimes \langle p \rangle_T$.

Each element λ thus represents a set of possible coefficients making $\langle \varepsilon \rangle_T$ and the $\langle qa \rangle_T$ (for all $q \in Q$ and all $a \in \Sigma$) left-linear combinations of the $\langle q \rangle_T$ for q in Q . Note that $\Lambda_{Q,T}$ is non-empty if and only if Q is row-closed on T and $\langle \varepsilon \rangle_T$ is left-generated by Q restricted to T . Note also that if $T \subseteq T'$, then $\Lambda_{Q,T} \supseteq \Lambda_{Q,T'}$. If $T = \Sigma^*$, we simply write Λ_Q .

Definition 2.5. Given a finite subset Q of S^{Σ^*} and a subset T of Σ^* , for each $\lambda \in \Lambda_{Q,T}$, we define a finite weighted automaton $\mathcal{H}_{Q,T,\lambda}$ with:

- finite set of states Q ;
- initial-state vector $(\lambda_q^\varepsilon)_{q \in Q}$;
- final-state vector $(\langle q \rangle_\varepsilon)_{q \in Q}$; and
- for all $q, p \in Q$ and all $a \in \Sigma$, a transition from q to p labelled by a with weight $\lambda_{q,a,p}$.

If $T = \Sigma^*$, we say $\mathcal{H}_{Q,T,\lambda}$ is a Hankel automaton and write simply $\mathcal{H}_{Q,\lambda}$ to reduce notation. If Q is a finite set of rows of the Hankel matrix and T is finite, we say that $\mathcal{H}_{Q,T,\lambda}$ is a hypothesis automaton, and view Q as a finite set of words by identifying with an appropriate index set.

Example 2.6. Consider the function f over \mathbb{N}_{\max} computed by our running example from Figure 1a. The set $\{\langle \varepsilon \rangle, \langle a \rangle, \langle b \rangle\}$ clearly left-generates $\langle \varepsilon \rangle$ and, as shown previously, is row-closed. Let

¹Using Q as the label for a set of words might seem odd at this point, but the reason behind this will become apparent very shortly.

q_ε, q_a and q_b denote these rows. Then considering the coefficients $\lambda_{q_\varepsilon}^\varepsilon = 0$, $\lambda_{q_\varepsilon, a, q_a} = \lambda_{q_\varepsilon, b, q_b} = \lambda_{q_a, b, q_a} = \lambda_{q_b, a, q_b} = 0$, $\lambda_{q_a, a, q_a} = \lambda_{q_b, b, q_b} = 1$ and all non-specified coefficients $-\infty$, the corresponding Hankel automaton would be obtained from the one in Figure 1a by replacing the weight of the transitions starting at q_ε by 0, and replacing the final weights by 1. For $Q = \{q_\varepsilon, q_a\}$ and $T = \{\varepsilon\}$ we have also seen that Q is row-closed on T . By taking $\lambda_{q_\varepsilon}^\varepsilon = 0$, $\lambda_{q_\varepsilon, a, q_a} = \lambda_{q_\varepsilon, b, q_a} = \lambda_{q_a, b, q_a} = 0$, $\lambda_{q_a, a, q_a} = 1$ and all remaining coefficients $-\infty$, one obtains an hypothesis automaton with two states. Notice that this automaton does not compute f ; it computes $|w|_a$ if w starts with an a , $|w|_a + 1$ if w starts with a b , and $-\infty$ if $w = \varepsilon$. Note that if $\{\varepsilon, aw\} \subseteq T'$ for some word w , then $Q = \{q_\varepsilon, q_a\}$ is not row-closed on T' .

Theorem 2.7 (see also [18]). *Let $f : \Sigma^* \rightarrow S$ be a function. There exists a finite set of rows $Q \subseteq S^{\Sigma^*}$ that is row-closed and left-generates $\langle \varepsilon \rangle$ if and only if f is computed by a finite-state weighted automaton \mathcal{A} over S . If the first condition holds, then $\mathcal{H}_{Q, \lambda}$ computes f for all $\lambda \in \Lambda_Q \neq \emptyset$, whilst if the second condition holds, the set of all $\langle q \rangle$ (where q is a state of \mathcal{A} and $\langle q \rangle_w$ is defined as the value of w in \mathcal{A} if q was considered the unique initial state with weight 1_S and all other weights are the same) is row-closed and left-generates every row of the Hankel matrix of f .*

Proof. We note that for the case where S is a field this result can be found in [18]. However, this can easily be adapted to the general case and since the details are instructive, we include a short proof. Suppose that $Q \subseteq S^{\Sigma^*}$ is a finite set that is row-closed and left-generates $\langle \varepsilon \rangle$. Note that we do not assume that the elements of Q are rows of the Hankel matrix of f . By definition, Λ_Q is non-empty. We prove that if λ belongs to Λ_Q then $\mathcal{H}_{Q, \lambda}$ computes f . We first show that for all q in Q and w in Σ^* , $\langle q \rangle_w$ is equal to the “weight of w from q ” that is, the weight computed on input w by the automaton obtained from $\mathcal{H}_{Q, \lambda}$ by imposing the condition that q is the unique initial state with weight 1_S , and keeping all other weights the same. The proof is by induction on the length of w . If $w = \varepsilon$, $\langle q \rangle_\varepsilon$ is (by construction) equal to the final weight of q in $\mathcal{H}_{Q, \lambda}$ which proves the property. For a in Σ and w in Σ^* , $\langle q \rangle_{aw} = \langle qa \rangle_w = \bigoplus_{p \in Q} \lambda_{q, a, p} \otimes \langle p \rangle_w$ since $\lambda \in \Lambda_Q$. By induction hypothesis, $\langle p \rangle_w$ is the weight of w from p , and moreover $\lambda_{q, a, p}$ is (by construction) the weight of the transition from q to p labelled by a . So $\langle q \rangle_{aw}$ is indeed the weight of aw from q . To conclude the proof, recall that $\langle \varepsilon \rangle$ denotes the row of ε in the Hankel matrix of f , and so by definition of the Hankel matrix of f and the fact that $\lambda \in \Lambda_Q$ we have $f(w) = \langle \varepsilon \rangle_w = \bigoplus_{q \in Q} \lambda_q^\varepsilon \otimes \langle q \rangle_w$; by our previous observation the latter is then exactly the value computed by $\mathcal{H}_{Q, \lambda}$ on input w .

Now suppose that $f : \Sigma^* \rightarrow S$ is computed by a finite-state weighted automaton \mathcal{A} over S with set of states Q . For each $q \in Q$, let ${}_q\mathcal{A}$ (resp. \mathcal{A}_q) denote the automaton obtained from \mathcal{A} by making q the unique initial (resp. final) state with weight 1_S . Define $\langle q \rangle \in S^{\Sigma^*}$ where $\langle q \rangle_u$ is equal to the value computed by ${}_q\mathcal{A}$ on u . Let $R = \{\langle q \rangle \mid q \in Q\}$. First R is row-closed. Indeed, by definition, for all words u , $\langle qa \rangle_u = \langle q \rangle_{au} = \bigoplus_{p \in Q} (\nu_p \otimes \langle p \rangle_u)$ where ν_p is the weight of the transition labelled by a in \mathcal{A} from q to p . Let w be a word in Σ^* and let $\langle w \rangle$ denote the infinite row of the Hankel matrix of f indexed by w . Let $\mu_q(w)$ be the value computed by \mathcal{A}_q on input w . Then, by definition, $\langle w \rangle = \bigoplus_{q \in Q} (\mu_q(w) \otimes \langle q \rangle)$, showing that R is row-generating for f and hence in particular $\langle \varepsilon \rangle$ is left-generated by R . \square

The previous result together with its dual (see Theorem A.3 of the appendix) give the following:

Corollary 2.8. *Let S be a semiring and $f : \Sigma^* \rightarrow S$. The following are equivalent:*

- (1) *f is computed by a finite-state weighted automaton \mathcal{A} over S ;*
- (2) *the rows of the Hankel matrix of f lie in a finitely generated left-subsemimodule of S^{Σ^*} ;*
- (3) *the columns of the Hankel matrix of f lie in a finitely generated right-subsemimodule of S^{Σ^*} .*

Remark 2.9. *Notice that if S is a field then the second condition of the previous result is that the rows of the Hankel matrix must lie in a finitely generated vector space, which in turn forces the vector space spanned by the rows of the Hankel matrix to be finitely generated. Thus if f is a function computed by a finite automaton over a field S , one can find finite sets of words $Q, P \subseteq \Sigma^*$ such that the rows of the Hankel matrix indexed by Q are row-generating and the columns of the Hankel matrix indexed by P are column-generating. However, in complete generality, a subsemimodule of a finitely generated semimodule need not be finitely generated, and so there is no reason to suspect that the left-semimodule generated by the rows of the Hankel matrix is finitely*

generated. *Moreover, it can happen that the left-semimodule generated by the rows of the Hankel matrix is finitely generated, whilst the right-semimodule generated by the columns is not (or vice versa).*

Arising questions and structure of the paper. It is known that for all functions computed by weighted automata over fields there is an efficient way to find Q , T and λ , by membership and equivalence queries such that the hypothesis automaton $\mathcal{H}_{Q,T,\lambda}$ computes the target function (see for example [8]). For general semirings, the following questions arise:

- For which functions $f : \Sigma^* \rightarrow S$ does there *exist* a hypothesis automaton $\mathcal{H}_{Q,T,\lambda}$ computing f ? We will call the class of all such functions weakly guessable. Clearly, if f is not weakly guessable then any learning algorithm based around making hypotheses of this kind will fail to learn f .
- For the class of weakly guessable functions, is there a learning algorithm using membership and equivalence queries that allows a learner to construct a finite sequence of hypothesis automata, where the last in the sequence computes the target function? In other words: when there exists at least one correct hypothesis, can we devise a strategy to find one? A possible issue here is that even when we have settled upon a suitable set Q , it may be that a “correct” λ cannot be found from the coefficient sets in finite time.
- For which functions f do there exist finite sets Q and T such that *every* hypothesis automaton $\mathcal{H}_{Q,T,\lambda}$ with $\lambda \in \Lambda_{Q,T}$ computes f ? We call such functions guessable. There is a naive algorithm (by enumerating all the words for Q and T) that permits us to learn the class of guessable functions, and so a natural question is whether there is a “fast” algorithm that permits us to learn this class. Additionally, we define strongly guessable functions as the ones for which for every suitable set Q , there exist a finite set T , such that every associated hypothesis automaton $\mathcal{H}_{Q,T,\lambda}$ with $\lambda \in \Lambda_{Q,T}$ computes f .

In Section 3, we define the classes of weakly guessable, guessable and strongly guessable functions and give several equivalent characterisations for them. In Section 4, we give a general hierarchy for these classes, explain where some examples of weighted automata fall in it and give conditions for this hierarchy to collapse. This allows us to have a better picture of what happens for several well-studied examples of semirings arising in application areas, where limitations of this approach may be of interest. Finally, in Section 5, we discuss algorithmic issues and guarantees for termination. This opens many questions for further research that are stated in the concluding Section 6.

3. GUESSABLE FUNCTIONS

In this section, we define weakly guessable, guessable, and strongly guessable functions and give characterisations for them.

3.1. Weakly guessable functions.

Definition 3.1. A function $f : \Sigma^* \rightarrow S$ is said to be weakly guessable if there exist a finite set Q of rows of the Hankel matrix of f such that Λ_Q is non empty.

Observe that by Theorem 2.7, this implies that for all $\lambda \in \Lambda_Q$ the automaton $\mathcal{H}_{Q,\lambda}$ computes f and hence for any non-empty T , there exists a hypothesis automaton $\mathcal{H}_{Q,T,\lambda}$ computing f since $\Lambda_Q \subseteq \Lambda_{Q,T}$. So this definition means that there is “some hope” that an algorithm à la Angluin can find a correct automaton, but does not really give a handle on what these functions look like. The following definition turns out to provide a syntactic characterisation of these functions in terms of automata.

Definition 3.2. A weighted automaton over S with set of states Q is said to be literal if it has a unique initial state with weight 1_S and there exists a prefix-closed set of words $W \subseteq \Sigma^*$ and a bijection $\sigma : Q \rightarrow W$ such that for all $q \in Q$:

- there is a unique run, γ_q , starting at the initial state and labelled by $\sigma(q)$,
- each transition in γ_q has weight 1_S ,
- γ_q terminates at state q .

This means that in a literal automaton \mathcal{A} , each state q can be characterised by a word w_q that leads uniquely to that state (and with weight 1_S), and hence the rows $\{\langle q \mid q \text{ is a state of } \mathcal{A} \rangle$ defined in Theorem 2.7 are exactly the rows of the Hankel matrix of \mathcal{A} indexed by the words w_q .

Finally, we link these functions to a property given in [28], giving a weaker variant that corresponds to the weakly guessable functions.

Definition 3.3. *Given a function $f : \Sigma^* \rightarrow S$ and F its Hankel matrix, f satisfies the weak ascending chain condition if for all sequences of left-semimodules $(X_i)_{i \in \mathbb{N}}$ generated by finite sets of rows of F , such that all rows of F belong to $\cup_{i \in \mathbb{N}} X_i$ and $X_0 \subseteq X_1 \subseteq X_2 \subseteq \dots$ there is n such that for all $m \geq n$, $X_m = X_n$.*

Proposition 3.4. *Given a function $f : \Sigma^* \rightarrow S$, the following assertions are equivalent:*

- (1) f is weakly guessable;
- (2) there exists a literal automaton computing f ;
- (3) there exists a finite subset of Σ^* that is row-generating for f ;
- (4) f satisfies the weak ascending chain condition.

Proof. (1. implies 2.) Since f is weakly guessable there is a finite set of rows Q of the Hankel matrix such that Λ_Q is non-empty; by a slight abuse of notation, we shall view Q as a subset of Σ^* , identifying a row of a Hankel matrix with the word labelling that row. Fix $\lambda \in \Lambda_Q$ and recall that (by Theorem 2.7) $\mathcal{H}_{Q,\lambda}$ computes f . Let Q' denote the set of all prefixes of words in Q , and define $\mu = (\mu_q^\varepsilon, \mu_{q,a,p})_{p,q \in Q', a \in \Sigma}$ as follows:

- $\mu_q^\varepsilon = 1_S$ if and only if $q = \varepsilon$ and 0_S otherwise,
- if $q \in Q'$, $a \in \Sigma$ and $qa \in Q'$, then $\mu_{q,a,qa} = 1_S$, and $\mu_{q,a,p} = 0_S$ for all other $p \in Q'$;
- if $q \in Q$, $a \in \Sigma$ and qa is not in Q' , then $\mu_{q,a,p} = \lambda_{q,a,p}$ for all $p \in Q$ and $\mu_{q,a,p} = 0_S$ for all $p \in Q' \setminus Q$;
- if q in $Q' \setminus Q$, a in Σ and qa is not in Q' , then for each $p \in Q'$ we define $\mu_{q,a,p}$ by induction on the length of q , as follows:

If $q = \varepsilon$, then for all words w we have $\langle \varepsilon \rangle_w = \bigoplus_{r \in Q} \lambda_r^\varepsilon \otimes \langle r \rangle_w$ and

$$\langle a \rangle_w = \langle \varepsilon \rangle_{aw} = \bigoplus_{r \in Q} \lambda_r^\varepsilon \otimes \langle r \rangle_{aw} = \bigoplus_{r \in Q} \lambda_r^\varepsilon \otimes \langle ra \rangle_w = \bigoplus_{r,p \in Q} \lambda_r^\varepsilon \otimes \lambda_{r,a,p} \otimes \langle p \rangle_w.$$

In this case we set: $\mu_{\varepsilon,a,p} = \bigoplus_{r \in Q} \lambda_r^\varepsilon \otimes \lambda_{r,a,p}$ for all $p \in Q$ and $\mu_{\varepsilon,a,p} = 0_S$ for all $p \in Q' \setminus Q$. If $q = ra$ and $\mu_{r,a,p}$ has already been defined for all $p \in Q'$ in such a way that expresses $\langle ra \rangle$ as a left-linear combination of the rows Q (i.e. such that $\mu(r,a,p) = 0_S$ if p is not in Q), then for all words w and all $b \in \Sigma$ we have

$$\langle qb \rangle_w = \langle rab \rangle_w = \langle ra \rangle_{bw} = \bigoplus_{p \in Q} \mu_{r,a,p} \otimes \langle p \rangle_{bw} = \bigoplus_{p \in Q} \mu_{r,a,p} \otimes \langle pb \rangle_w = \bigoplus_{p,s \in Q} \mu_{r,a,p} \otimes \lambda_{p,b,s} \otimes \langle s \rangle_w.$$

In this case we set $\mu_{ra,b,p} = \bigoplus_{s \in Q} \mu_{r,a,p} \otimes \lambda_{p,b,s}$ for all $p \in Q$ and $\mu_{ra,b,p} = 0_S$ for all $p \in Q' \setminus Q$.

By construction we then have that μ belongs to $\Lambda_{Q'}$ and hence by Lemma 2.7, $\mathcal{H}_{Q',\mu}$ computes f . The automaton $\mathcal{H}_{Q',\mu}$ is also literal by construction, which concludes the proof.

(2. implies 3.) Let \mathcal{A} be a literal automaton computing f and Q its set of states. By Theorem 2.7, the set $R = \{\langle q \rangle \mid q \in Q\}$, where $\langle q \rangle_w$ is defined as the weight of w from q , is row-generating for f . Moreover, by definition of literal automaton, there is a finite set of words $\{w_q \mid q \in Q\}$ such that for all q , $\langle q \rangle = \langle w_q \rangle$.

(3. implies 4.) Let $W \subseteq \Sigma^*$ such that $\{\langle w \rangle \mid w \in W\}$ is row-generating and let a sequence of left-semimodules $(X_i)_{i \in \mathbb{N}}$ as in the definition of the weak ascending chain condition. Then there is n such that for all $m \geq n$, all the rows $\langle w \rangle$ for w in W are in X_m , and since they are row-generating, $X_m = X_n$.

(4. implies 1.) For all positive integers i , let X_i be the left-semimodule generated by the rows indexed by words of length at most i . Then the X_i satisfy the hypothesis of the weak ascending chain condition and hence there is n such that for all m , $X_m = X_n$. Let Q be the set of words of length at most n . Necessarily the set of rows indexed by Q is row-generating (and hence row-closed, and left-generates $\langle \varepsilon \rangle$) so there exists some $\lambda \in \Lambda_Q$, which concludes the proof. \square

Remark 3.5. *In semirings where all functions computed by finite state weighted automata satisfy the weak ascending and co-ascending chain conditions, we have that such functions are both weakly guessable and weakly co-guessable (by Proposition 3.4 and its dual, Proposition A.7). This means that over such a semiring all functions computed by a finite weighted automaton have the potential*

to be learnt, provided one can make a lucky guess for Q , T and λ . This is in particular the case for \mathbb{B} (or indeed any finite semiring), \mathbb{R} (or indeed, any field), and \mathbb{Z} (or indeed, any principal ideal domain c.f. [28]).

3.2. Guessable and strongly guessable functions. The definition of weakly guessable functions ensures the existence of some finite sets Q, T and with corresponding coefficients λ that together determine a correct hypothesis automaton. In other words, functions that are *not weakly guessable* are those for which *all hypothesis automata are incorrect*, and so (in fairness to the learner!) one should immediately exclude all such functions from consideration if one seeks to learn by membership and equivalence in the style of Angluin. For weakly guessable functions, the problem then becomes: how to find a suitable Q, T and λ . Clearly Q must be a row-generating set for the full Hankel matrix; but is it always possible to find such a set from finitely many membership and equivalence queries? Also, even if a suitable row generating set Q is found, there is no guarantee that accompanying T and λ can be found in finite time. Guessable functions are those for which there exist finite subsets $Q, T \subseteq \Sigma^*$ such that all the associated hypothesis automata compute the correct function. Strongly guessable functions are those for which for *every* finite row-generating set Q there exists a finite subset $T \subseteq \Sigma^*$ such that all the associated hypothesis automata compute the correct function.

Definition 3.6. A function $f : \Sigma^* \rightarrow S$ is guessable if there exist finite subsets $Q, T \subseteq \Sigma^*$ such that $\Lambda_{Q,T} = \Lambda_Q \neq \emptyset$. We say that the pair (Q, T) witnesses that f is guessable. Furthermore, we say that f is strongly guessable if for all finite sets $Q \subseteq \Sigma^*$ with $\Lambda_Q \neq \emptyset$ there exists a finite set $T \subseteq \Sigma^*$ such that (Q, T) witnesses that f is guessable.

It is clear from the definitions that every strongly guessable function is guessable, and every guessable function is weakly guessable. Moreover, observe that, by Theorem 2.7, if the pair (Q, T) witnesses that f is guessable, then all hypothesis automata over Q and T compute f . We now give a weaker variant of another property given in [28] to characterise guessable functions.

Definition 3.7. We say that $(T_i)_{i \geq 0}$ is a total sequence of words if $T_0 \subseteq T_1 \subseteq \dots \subseteq \Sigma^*$ and $\bigcup_{i \in \mathbb{N}} T_i = \Sigma^*$. We say that a function $f : \Sigma^* \rightarrow S$ is row-bound if there exists a finite set of words Q such that $\Lambda_Q \neq \emptyset$ and there is no infinite descending chain of the form $\Lambda_{Q,T_0} \supsetneq \Lambda_{Q,T_1} \supsetneq \Lambda_{Q,T_2} \supsetneq \dots$ where $(T_i)_{i \geq 0}$ is a total sequence of words. We say that $f : \Sigma^* \rightarrow S$ is strongly row-bound if for every finite set of words Q such that $\Lambda_Q \neq \emptyset$ there is no infinite descending chain of the form $\Lambda_{Q,T_0} \supsetneq \Lambda_{Q,T_1} \supsetneq \Lambda_{Q,T_2} \supsetneq \dots$ where $(T_i)_{i \geq 0}$ is a total sequence of words.

The above definition is to be linked with the progress measure defined in [28, Definition 11]. The existence of a progress measure for an Hankel matrix implies that the corresponding function is strongly row-bound. However, the converse might not be true. A progress measure implies a uniform bound, for all Q , on the number of elements in a descending chain of the form $\Lambda_{Q,T_0} \supsetneq \Lambda_{Q,T_1} \supsetneq \Lambda_{Q,T_2} \supsetneq \dots$ where $(T_i)_{i \geq 0}$ is a total sequence of words, but for a (strongly) row-bound function, even if no such chain can be infinite, its size might depend on the set of words Q . The progress measure is a stronger assumption, and one of the sufficient condition for termination of a learning algorithm [28], while (strongly) row-boundedness pinpoints more precisely a characterisation of (strongly) guessable functions.

Proposition 3.8. Let $f : \Sigma^* \rightarrow S$ be a weakly guessable function. Then f is (strongly) guessable if and only if f is (strongly) row-bound.

Proof. Suppose that Q, T are finite sets of words such that $\Lambda_{Q,T} = \Lambda_Q$. Let $(T_i)_{i \geq 0}$ be a total sequence of words. By definition, there exists some set T_j such that $T \subseteq T_j$. Then for all $i \geq j$ we have $T \subseteq T_j \subseteq T_i$. Since $\Lambda_{Q,T} = \Lambda_Q$ it then follows that $\Lambda_{Q,T_i} = \Lambda_{Q,T_j} = \Lambda_{Q,T}$ for all $i \geq j$. Thus there can be no infinite descending chains of the Λ_{Q,T_i} where the T_i form a total sequence of words. It now follows that if f is guessable then f is row-bound, whilst if f is strongly guessable, then it must be strongly row-bound.

Suppose now that f is (strongly) row-bound. Let Q be a finite set of words such that $\Lambda_Q \neq \emptyset$ and there is no infinite descending chain of the form $\Lambda_{Q,T_0} \supsetneq \Lambda_{Q,T_1} \supsetneq \Lambda_{Q,T_2} \supsetneq \dots$ where the T_i form a total sequence of words. For instance, taking for each positive integer i , T_i to be the finite set of words of length at most i , we have that $(T_i)_{i \geq 0}$ is a total sequence of words and so there must exist n such that for all $m \geq n$, $\Lambda_{Q,T_m} = \Lambda_{Q,T_n}$. Since the T_i enumerate all words,

we must therefore also have $\Lambda_{Q,T_n} = \Lambda_Q$. Hence for any λ in Λ_{Q,T_n} , the hypothesis automaton over Q, T_n and λ computes f . Thus, if f is (strongly) row-bound, we conclude that f is (strongly) guessable. \square

Remark 3.9. Let S be a field and $f : \Sigma^* \rightarrow S$ a function computed by a finite automaton with weights from S . By Remark 2.9 there exist finite sets of words $Q, P \subseteq \Sigma^*$ such that Q is row-generating for f and P is column generating for f . It is straightforward to check that for any such sets Q and P we have $\Lambda_{Q,P} = \Lambda_Q \neq \emptyset$ (see Lemma 4.5 below for a generalisation of this fact) and hence f is strongly guessable (and by a dual argument, f is strongly co-guessable).

4. HIERARCHY OF LEARNABLE FUNCTIONS

In the previous section, we have defined the classes of strongly guessable, guessable, and weakly guessable functions. In a dual way, we can define the strongly co-guessable, co-guessable, and weakly co-guessable functions, by re-writing the previous section swapping rows and columns of the Hankel matrix, left and right multiplications and concatenations, and initial and final states: we give full details in Appendix A, sketching only the main ideas here. This leads us to define a co-hypothesis automaton constructed from a finite set of columns of the Hankel matrix with entries restricted to a finite set of rows, where we use right-linear combinations to define the weights. The weakly co-guessable functions are those for which there exists a co-hypothesis automaton computing them. They are characterised equivalently by a co-literal automaton (an automaton whose mirror is literal) and a weak co-ascending chain condition. This version can be found in Appendix A to avoid any ambiguity. We obtain a general hierarchy depicted in Figure 2. We do not have a precise description of whether all of the containments shown can be strict, however, in the next two subsections we show that parts of this hierarchy certainly collapse in the case of commutative semirings (see Figure 3c below) and give examples to demonstrate that certain of these containments can be strict. The examples are drawn from various semirings discussed in Section 2, but one could obtain a unified semiring by considering their product and component-wise operations. We then prove that under some conditions (and hence in particular, for some well-studied examples) further parts of this hierarchy collapse.

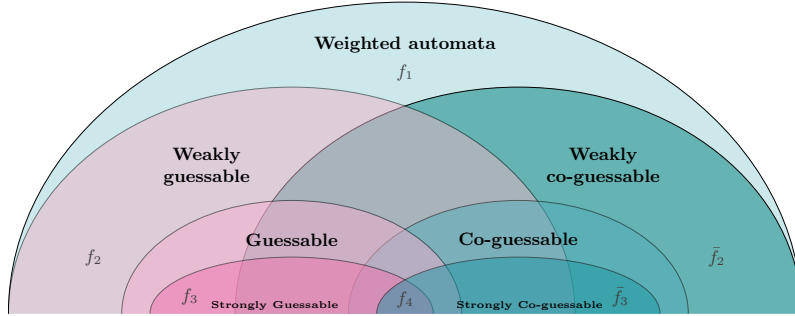


FIGURE 2. General hierarchy.

4.1. Three conditions for collapse. In this section, we give three conditions under which certain classes of the hierarchy collapse. The various results we obtain are depicted in Figure 3.

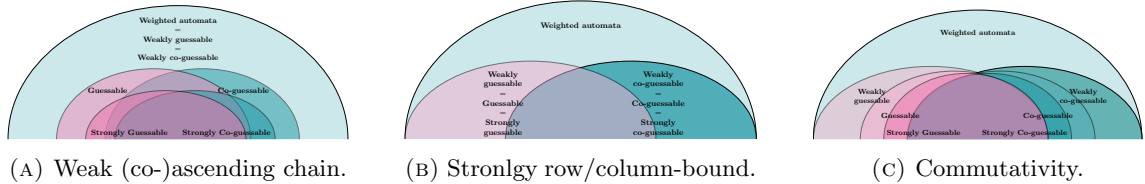


FIGURE 3. Hierarchies.

As noted in Remark 3.5, in semirings where all functions satisfy the weak ascending and weak co-ascending chain condition, we have that all functions computed by a finite weighted automaton

are both weakly guessable and weakly co-guessable, giving the hierarchy depicted in Figure 3a. Similarly, by Proposition 3.8, it follows that in semirings where all weakly guessable functions are strongly row-bound and column-bound, we have that all weakly guessable functions computed by a finite weighted automaton are both strongly guessable and strongly co-guessable, giving the hierarchy depicted in Figure 3b. This is in particular the case for \mathbb{N}_{\max} , \mathbb{N} , \mathbb{Z} , \mathbb{R} , \mathbb{B} and $\mathcal{P}_{\text{fin}}(\Sigma^*)$. For \mathbb{Z} , \mathbb{R} and \mathbb{B} , this is shown in [28] (as special cases of principal ideal domains, fields and finite semiring respectively).

Lemma 4.1. *Every weakly guessable (respectively, weakly co-guessable) function over \mathbb{N}_{\max} , \mathbb{N} , or $\mathcal{P}_{\text{fin}}(\Sigma^*)$ is strongly row-bound (respectively, column-bound).*

Proof. We give the proof for rows. The proof for columns is symmetric. Let S be one of the semirings \mathbb{N}_{\max} , \mathbb{N} , or $\mathcal{P}_{\text{fin}}(\Sigma^*)$. Let f be a weakly guessable function computed over S , and F its Hankel matrix. Let Q be such that Λ_Q is non-empty. First, note that, for some fixed T , if no row indexed by Q and restricted to T is fully 0_S, then in the three cases, $\Lambda_{Q,T}$ is finite. Let $Z_{Q,T} = \{q \in Q \mid \langle q \rangle_T \text{ is a } 0_S \text{ row}\}$, $Q' = Q \setminus Z_{Q,T}$ and $N_{Q',T}$ the set of tuples $(\lambda_p^e, \lambda_{q,a,p})_{q \in Q', a \in \Sigma, p \in Q'}$ from $\Lambda_{Q,T}$ where p is restricted to Q' . Clearly $Z_{Q,T}$ is finite and its size is bounded above by the one of Q . Notice that $N_{Q',T}$ is also a finite set, since each $\lambda_{q,a,p}$ where $p \in Q'$ can only take a finite numbers of different values. Suppose that $T \subsetneq T'$ with $\Lambda_{Q,T} \supsetneq \Lambda_{Q,T'}$. It is clear from the definition that $Z_{Q,T} \supseteq Z_{Q,T'}$, and either this containment is strict, or if not, then $N_{Q',T} \supsetneq N_{Q',T'}$. Since both sets have a finite size, there cannot be an infinite descending chain as in the definition. \square

Clearly if both the conditions depicted in Figures 3a and 3b hold, we have that all functions computed by finite weighted automata are strongly guessable. We remark that in semirings where the concepts of weakly guessable and strongly guessable (resp. weakly co-guessable and strongly co-guessable) coincide (as is the case for fields and finite semirings, for example), this is inherited by subsemirings:

Proposition 4.2. *Let S' be a subsemiring of S . If every weakly guessable function over S is strongly guessable, then every weakly guessable function over S' is also strongly guessable.*

Proof. Suppose that $f : \Sigma^* \rightarrow S'$ is weakly guessable. Thus there is a literal automaton \mathcal{A} with weights in S' that computes f . Viewed as a function over S , we also have that f is weakly guessable (since \mathcal{A} is a literal automaton with weights in S computing f) and hence by assumption, strongly guessable.

Suppose for contradiction that f not strongly guessable over S' . Then there exists a finite set Q such that $\Lambda_Q \neq \emptyset$ and for all finite sets T there exists $\lambda_T \in \Lambda_{Q,T} \setminus \Lambda_Q$. However, noting that the coefficients λ_T lie in S , this directly contradicts that f is strongly guessable over S . \square

Corollary 4.3. *If S is a subsemiring of a field, then every weakly guessable function over S is strongly guessable.*

Proof. This follows immediately from Proposition 4.2 and Remark 3.9. \square

Remark 4.4. *The previous result applies in particular to all integral domains, since every integral domain embeds into a field (its field of fractions).*

Finally, for commutative semirings we have the following result, giving hierarchy depicted in Figure 3c.

Lemma 4.5. *Let S be a commutative semiring and let $f : \Sigma^* \rightarrow S$ be computed by a weighted automaton over S . If f is weakly guessable and weakly co-guessable, then f is strongly guessable and strongly co-guessable.*

Proof. Suppose that f is weakly guessable and weakly co-guessable and let F be the Hankel matrix of f . We first show that f is strongly guessable. Since f is weakly co-guessable, we may fix a finite set of words T such that the columns indexed by T are column-generating. We will prove that for any finite set of words Q with $\Lambda_Q \neq \emptyset$ we have $\Lambda_Q = \Lambda_{Q,T}$.

Thus let Q be arbitrary with the property that $\Lambda_Q \neq \emptyset$ and let $\lambda \in \Lambda_{Q,T}$. Then for all $q, p \in Q$, all $a \in \Sigma$ and all $t \in T$ we have: $F_{qa,t} = \bigoplus_{p \in Q} \lambda_{q,a,p} \otimes F_{p,t}$. By our assumption on T we know

that for all words w and all $t \in T$ there exist $\mu_{w,t} \in S$ such that $F_{u,w} = \bigoplus_{t \in T} \mu_{w,t} \otimes F_{u,t}$ holds for all words u . In particular, for all $w \in \Sigma^*$, we have:

$$\begin{aligned} F_{qa,w} &= \bigoplus_{t \in T} \mu_{w,t} \otimes F_{qa,t} = \bigoplus_{t \in T} \mu_{w,t} \otimes \left(\bigoplus_{p \in Q} \lambda_{q,a,p} \otimes F_{p,t} \right) = \bigoplus_{p \in Q} \lambda_{q,a,p} \otimes \left(\bigoplus_{t \in T} \mu_{w,t} \otimes F_{p,t} \right) \\ &= \bigoplus_{p \in Q} \lambda_{q,a,p} \otimes F_{p,w}. \end{aligned}$$

In other words, $\langle qa \rangle = \bigoplus_{p \in Q} \lambda_{q,a,p} \otimes \langle p \rangle$, giving $\lambda \in \Lambda_Q$, as required. This demonstrates that f is strongly guessable.

Since f is weakly guessable and co-guessable, then \bar{f} is also weakly guessable and weakly co-guessable. The above argument therefore yields that \bar{f} is strongly guessable and so f is strongly co-guessable. \square

We next aim to show that for commutative semirings almost all inclusions depicted in Figure 3c can be strict.

4.2. Examples for the strict hierarchy. Before getting into technical details, we give a quick overview of what we aim to show in this section. For some semirings S , there are functions $f : \Sigma^* \rightarrow S$ that are neither weakly guessable, nor weakly co-guessable. We will show that this is the case for f_1 computed by the automaton given in Figure 1b, whether viewed over \mathbb{N}_{\max} , \mathbb{Z}_{\max} or \mathbb{R}_{\max} , f'_1 computed by the automaton given in Figure 1d, whether viewed over \mathbb{N} or $\mathbb{R}_{\geq 0}$, and f''_1 computed by the automaton in Figure 1f. There are also functions that are weakly guessable but neither guessable nor weakly co-guessable. We will show that this is the case for f_2 computed by the automaton given in Figure 1a, whether viewed as an automaton over \mathbb{Z}_{\max} or \mathbb{R}_{\max} . It then follows that the mirror function \bar{f}_2 is weakly co-guessable but neither co-guessable nor weakly guessable (since a function over a commutative semiring is (weakly) guessable if and only if its mirror is (weakly) co-guessable.). Finally, we will show that the functions f_3 computed by the automaton depicted in Figure 1a (over \mathbb{N}_{\max}), f'_3 computed by the automaton from Figure 1c and f''_3 computed by the automaton from Figure 1e are all strongly guessable but not weakly co-guessable. The functions \bar{f}_3 and \bar{f}'_3 are strongly co-guessable but not weakly guessable. It is in general easy to find examples at the intersection of all the classes: a trivial example of such function would be the constant function f_4 mapping all words to 1_S . For commutative semirings, any function computed by such an automaton that is both literal and co-literal lies in this intersection.

We now prove the claims above.

Neither weakly guessable nor weakly co-guessable (f_1). We will give examples of functions that are neither weakly guessable nor weakly co-guessable for \mathbb{N}_{\max} , \mathbb{Z}_{\max} , \mathbb{R}_{\max} , \mathbb{N} , $\mathbb{R}_{\geq 0}$ and $\mathcal{P}_{\text{fin}}(\Sigma^*)$. We give the proofs for weakly guessable. The proofs for weakly co-guessable are dual.

Let S be one of \mathbb{N}_{\max} , \mathbb{Z}_{\max} , or \mathbb{R}_{\max} , and consider the function f_1 computed by the automaton depicted in Figure 1b and F its Hankel matrix over S . Let $w = ua^n$ and $w' = a^{n'}v$ where $u, v \in \Sigma^*$ are such that u does not end with an a and v does not start with an a . The corresponding entry of the Hankel matrix is given by $F_{w,w'} = f(ww') = \max\{f_1(u), n + n', f_1(v)\}$. We claim that f_1 is neither weakly guessable nor weakly co-guessable. Suppose for contradiction that f_1 is weakly guessable. By Proposition 3.4 there exists a finite set of words $W = \{w_1, \dots, w_k\}$ such that the rows of the Hankel matrix indexed by W are row-generating. Let us write $w_i = u_i a^{n_i}$ for $i = 1, \dots, k$, where u_i does not end with an a and $n_i \geq 0$, and choose a positive integer N such that $N \geq \max_{i \in \{1, \dots, k\}} \{f_1(u_i), n_i\}$. Consider the row of the Hankel matrix indexed by $w = a^{N+1}b$. Since the rows indexed by W are assumed to be row-generating, we have: $\langle w \rangle = \bigoplus_{i=1}^k \lambda_i \otimes \langle w_i \rangle$ for some $\lambda_i \in S$. Now set $u = ba^N$ and $v = ba^{N+1}$. We have:

$$\begin{aligned} f_1(wu) = F_{w,u} = \langle w \rangle_u &= \max_i \{\lambda_i + \langle w_i \rangle_u\} \\ &= \max_i \{\lambda_i + F_{w_i,u}\} \\ &= \max_i \{\lambda_i + \max(f_1(u_i), n_i, N)\} \\ &= \max_i \{\lambda_i + N\} = \max_i \{\lambda_i\} + N \end{aligned}$$

and, similarly, $f_1(wv) = \max_i \{\lambda_i\} + N + 1$. But this immediately gives a contradiction, since $f_1(wu) = f_1(a^{N+1}b^2a^N) = N + 1 = f_1(a^{N+1}b^2a^{N+1}) = f_1(wv)$.

Let S be one of \mathbb{N} , $\mathbb{R}_{\geq 0}$ and consider the function f'_1 computed by the automaton depicted in Figure 1d and F its Hankel matrix. Suppose that f'_1 is weakly guessable. Then there exist a finite set of words $Q = \{w_1, w_2, \dots, w_k\}$ such that the rows indexed by Q left-generate the rows of F . Without loss of generality, we may assume that $w_1 = \varepsilon$ and that w_i is a power of a for $i = 1, \dots, m \leq k$. Let $w = a^n$ for some positive integer n . Then by assumption, considering columns ε and a of F , there exist $\lambda_1, \dots, \lambda_k$ in S such that:

$$2^n - 1 = \sum_{i=1}^k \lambda_i f'_1(w_i) \text{ and } 2^{n+1} - 1 = \sum_{i=1}^k \lambda_i f'_1(w_i a),$$

which gives $\sum_{i=1}^k \lambda_i f'_1(w_i a) = 1 + \sum_{i=1}^k 2\lambda_i f'_1(w_i)$. Moreover, for each w_i that is a power of a , we have $f'_1(w_i a) = 2f'_1(w_i) + 1$, whilst if w_i is not a power of a , we have $f'_1(w_i a) = 0 = f'_1(w_i)$. Thus when restricting attention to columns $\{\varepsilon, a\}$ it suffices to consider only the w_i with $i \leq m$, giving:

$$\sum_{i=1}^m \lambda_i (2f'_1(w_i) + 1) = 1 + \sum_{i=1}^m 2\lambda_i f'_1(w_i)$$

giving $\sum_{i=1}^m \lambda_i = 1$.

Suppose now that n is (strictly) larger than the lengths of all the w_i . We have proved that there are $\lambda_i \geq 0$ such that $2^n - 1 = \sum_{i=1}^m \lambda_i f'_1(w_i)$ and $\sum_{i=1}^m \lambda_i = 1$. But $2^n - 1$ is strictly greater than any of the $f'_1(w_i)$, so cannot be obtained as a convex combination of them.

For $S = \mathcal{P}_{\text{fin}}(\Sigma^*)$, consider the function f'_1 computed by the automaton depicted in Figure 1f and F its Hankel matrix. If a row $\langle w \rangle$ of F is left-generated by a finite set of rows, this in particular means (by restricting attention to column ε) that $\{a^{|w|_a}, b^{|w|_b}\}$ can be written as $\bigcup_{i=1}^k X_i \{a^{n_i}, b^{m_i}\}$ for some finite sets X_i and non-negative integers n_i, m_i . This is only possible if exactly one of the sets X_i is non-empty, and for this value i we have that $X_i = \{\varepsilon\}$ and $n_i = |w|_a$ and $m_i = |w|_b$. It follows from this that for all pairs of non-negative integers (n, m) a row-generating set of words must contain at least one word with $|w|_a = n$ and $|w|_b = m$.

Weakly guessable but neither guessable nor weakly co-guessable (f_2). Let $S = \mathbb{Z}_{\max}$ or $S = \mathbb{R}_{\max}$ and consider the function f_2 computed by the automaton depicted in Figure 1a, but viewed as an automaton over S and let F denote its Hankel matrix.

First, f_2 is weakly guessable. Indeed, it is easy to see that one can transform the automaton given in the figure into an equivalent literal automaton, by pushing the weights from the first transitions to final weights.

Second, f_2 is not weakly co-guessable. For all words w , we have $f_2(aw) = |aw|_a$, and $f_2(bw) = |bw|_b$. Suppose for contradiction that f_2 is weakly co-guessable. Then there exists a finite set of words $W = \{w_1, \dots, w_k\}$ such that the columns of the Hankel matrix labelled by W are column-generating. Let $N > \max_i |w_i|_a + 1$. Since for all i , we have $F_{a, w_i} = |w_i|_a + 1$ and $F_{b, w_i} = |w_i|_b + 1$, it is easy to see that the column indexed by $w = a^N$ (for which $F_{a, w} = N + 1$ and $F_{b, w} = 1$) cannot be expressed as a right-linear combination of the columns indexed by W .

Finally, f_2 is not guessable. Suppose for contradiction that f_2 is guessable. Then there exist finite sets Q, T such that $\Lambda_{Q, T} = \Lambda_Q \neq \emptyset$. Without loss of generality, we may assume that for a fixed constant N (to be chosen below) we have $n > N$ where n is the length of the longest word in T (since for all $T' \supseteq T$ we also have $\Lambda_{Q, T'} = \Lambda_Q \neq \emptyset$). Hence for all λ in $\Lambda_{Q, T}$, we have that $\mathcal{H}_{Q, T, \lambda}$ computes f_2 . If Q only contains words starting with an a , then for all $q \in Q$, $\langle qa \rangle_T = \langle q \rangle + 1$ and $\langle qb \rangle_T = \langle q \rangle$, and it is easy to see that an hypothesis automaton constructed from Q, T and a λ of this shape would not compute f_2 . A symmetric argument holds if Q only contains words starting with b . If Q contains a word q starting with a and a word q' starting with b , then, we may write $\langle qa \rangle_T = \max(\langle q \rangle_T + 1, \langle q' \rangle_T - n)$, $\langle qb \rangle_T = \max(\langle q \rangle_T, \langle q' \rangle_T - n)$, $\langle q'a \rangle_T = \max(\langle q \rangle_T - n, \langle q' \rangle_T)$ and $\langle q'b \rangle_T = \max(\langle q \rangle_T - n, \langle q' \rangle_T + 1)$. It is then easy to see that there exists a hypothesis automaton constructed from sets Q, T and a λ using the values from the linear combinations above that would not compute f_2 ; it would for example misbehave on a word of the form $qa^n bb^{2n}$. Indeed, since q starts with a the target function computes $|q|_a + n$. If there is no run from any initial state to q in the hypothesis automaton, then the automaton computes $-\infty$ and we are done. Otherwise, there is a run of weight $x + 2n$, where x is the maximal weight of a run from an initial state to q . Taking $N = |q|_a - x$ we see that $x + 2n > x + n + N = |q|_a + n = f_2(qa^n bb^{2n})$.

Weakly co-guessable but neither co-guessable nor weakly guessable (\bar{f}_2). Since \mathbb{Z}_{\max} and \mathbb{R}_{\max} are commutative, and f_2 is weakly guessable but neither guessable nor weakly co-guessable, it follows that \bar{f}_2 is weakly co-guessable but neither co-guessable nor weakly guessable (since a function over a commutative semiring is (weakly) guessable if and only if its mirror is (weakly) co-guessable).

Strongly guessable but not weakly co-guessable (f_3 and f_5). We will give examples of functions that are strongly guessable but not weakly co-guessable for \mathbb{N}_{\max} , \mathbb{N} , $\mathcal{P}_{\text{fin}}(\Sigma^*)$ and $\mathbb{R}_{\geq 0}$. In Lemma 4.1, it is shown that all the functions from the first three semirings are strongly row-bound and hence, any weakly guessable function is strongly guessable. It is also the case for $\mathbb{R}_{\geq 0}$ by Corollary 4.3. To prove that the functions under consideration are strongly guessable, it is then enough to prove that they are weakly guessable, or equivalently that they are computed by a literal automaton.

For $S = \mathbb{N}_{\max}$, consider the function f_3 computed by the automaton depicted in Figure 1a and F its Hankel matrix. It is easy to see that f_3 is computed by a literal automaton, by pushing the weights on the first transitions in the automaton depicted in the figure, as final weights. So f_3 is weakly guessable and hence strongly guessable. Let us prove now that it is not weakly co-guessable. For all words w , we have $f_3(aw) = |aw|_a$, and $f_3(bw) = |bw|_b$. Suppose for contradiction that f_3 is weakly co-guessable. Then there exists a finite set of words $W = \{w_1, \dots, w_k\}$ such that the columns of the Hankel matrix labelled by W are column-generating and let $N > \max_i |w_i|_a + 1$. Since for all i , we have $F_{a,w_i} = |w_i|_a + 1$ and $F_{b,w_i} = |w_i|_b + 1$, it is easy to see that the column indexed by $w = a^N$ (for which $F_{a,w} = N + 1$ and $F_{b,w} = 1$) cannot be expressed as a right-linear combination of the columns indexed by W .

For $S = \mathbb{N}$, consider the function f'_3 computed by the automaton depicted in Figure 1c and F its Hankel matrix. It is easy to see that f'_3 is computed by a literal automaton, by pushing the weights on the first transitions in the automaton depicted in the figure, as final weights. So f'_3 is weakly guessable and hence strongly guessable. Let us prove now that it is not weakly co-guessable. For all words w , we have $f'_3(aw) = 2^{|aw|_a}$, and $f'_3(bw) = 2^{|bw|_b}$. Suppose for contradiction that f'_3 is weakly co-guessable. Then there exists a finite set of words $W = \{w_1, \dots, w_k\}$ such that the columns of the Hankel matrix labelled by W are column-generating and let $N > \max_i |w_i|_a + 1$. Since for all i , we have $F_{a,w_i} = 2^{|w_i|_a + 1}$ and $F_{b,w_i} = 2^{|w_i|_b + 1}$, it is easy to see that the column indexed by $w = a^N$ (for which $F_{a,w} = 2^{N+1}$ and $F_{b,w} = 2$) cannot be expressed as a right-linear combination of the columns indexed by W .

For $S = \mathcal{P}_{\text{fin}}(\Sigma^*)$, consider the function f''_3 computed by the automaton depicted in Figure 1e and F its Hankel matrix. It is easy to see that f''_3 is computed by a literal automaton, by pushing the weights on the first transitions in the automaton depicted in the figure, as final weights. So f''_3 is weakly guessable and hence strongly guessable. Let us prove now that it is not weakly co-guessable. For all words w , we have $f''_3(aw) = a^{|aw|_a}$, and $f''_3(bw) = b^{|bw|_b}$. Suppose for contradiction that f''_3 is weakly co-guessable. Then there exists a finite set of words $W = \{w_1, \dots, w_k\}$ such that the columns of the Hankel matrix labelled by W are column-generating and let $N > \max_i |w_i|_a + 1$. Since for all i , we have $F_{a,w_i} = a^{|w_i|_a + 1}$ and $F_{b,w_i} = b^{|w_i|_b + 1}$, it is easy to see that the column indexed by $w = a^N$ (for which $F_{a,w} = a^{N+1}$ and $F_{b,w} = b$) cannot be expressed as a right-linear combination of the columns indexed by W .

Finally, we give an example of a function over $S = \mathbb{R}_{\geq 0}$ that is strongly guessable but not weakly co-guessable. Consider the automaton obtained from that in Figure 1c by replacing all weights labelled by b by 1. In a similar manner to the previous cases, it is easy to see that the function computed, which we denote by f_5 , is computed by a literal automaton and hence is weakly guessable. For all words w , we have $f_5(aw) = 2^{|aw|_a}$, and $f_5(bw) = 1$. Taking $Q = T = \{\varepsilon, a, b\}$ it is straightforward to verify that $\Lambda_{Q,T} = \Lambda_Q \neq \emptyset$ (indeed, for each of $\langle \varepsilon \rangle, \langle qa \rangle, \langle qb \rangle$ where $q \in Q$ we find that there is exactly one way to write the given vector as a linear combination of the $\langle q' \rangle$ with $q' \in Q$). Hence f_5 is strongly guessable. Suppose then for contradiction that f_5 is weakly co-guessable. Then there exists a finite set of words $W = \{w_1, \dots, w_k\}$ such that the columns of the Hankel matrix labelled by W are column-generating and let $N > \sum_i 2^{|w_i|_a + 1}$. Since for all i , we have $F_{a,w_i} = 2^{|w_i|_a + 1}$ and $F_{b,w_i} = 1$, it is easy to see that the column indexed by $w = a^N$ (for which $F_{a,w} = 2^{N+1}$ and $F_{b,w} = 1$) cannot be expressed as a right-linear combination of the columns indexed by W .

Strongly co-guessable but not weakly guessable (\bar{f}_3). Since \mathbb{N}_{\max} and \mathbb{N} are commutative, and f_3 and f'_3 are strongly guessable but not weakly co-guessable, then \bar{f}_3 and \bar{f}'_3 are strongly co-guessable but not weakly guessable (using once more the fact that a function over a commutative semiring is (weakly/strongly) guessable if and only if its mirror is (weakly/strongly) co-guessable).

	A	B	C
\mathbb{B}	Yes	Yes	Yes
\mathbb{R}	Yes	Yes	Yes
\mathbb{Z}	Yes	Yes	Yes
\mathbb{N}	No (f'_1)	Yes	No (f'_3)
$\mathbb{R}_{\geq 0}$	No (f'_1)	Yes	No (f_5)
$\mathcal{P}_{\text{fin}}(\Sigma^*)$	No (f''_1)	Yes	No (f''_3)
\mathbb{N}_{\max}	No (f_1)	Yes	No (f_3)
\mathbb{Z}_{\max}	No (f_1)	No (f_2)	???
\mathbb{R}_{\max}	No (f_1)	No (f_2)	???

A: all functions computed by finite state automata are weakly guessable and weakly co-guessable.
 B: weakly guessable functions are strongly guessable and weakly co-guessable functions are strongly co-guessable.
 C: strongly guessable functions are weakly co-guessable and strongly co-guessable functions are weakly guessable.

FIGURE 4. Summary of collapse of classes in familiar semirings.

Property A holds if and only if every function satisfies the weak ascending chain condition and co-ascending chain condition (see Proposition 3.4). Property B holds if and only if every function satisfying the weak (co)-ascending chain condition is strongly row-bound (column-bound) (see Proposition 3.8). Clearly if A and B both hold, then C must also hold.

4.3. Summary for some familiar semirings. The table in Figure 4 demonstrates that there are interesting and important semirings (including the natural numbers) where the properties of weakly guessable and strongly guessable coincide (i.e. property B holds) whilst there exist functions that are not weakly guessable (i.e. property A does not hold). Since the property of being weakly guessable is characterised in terms of automata by Proposition 3.4, over such semirings it is possible for the ‘teacher’ to select a guessable function to begin with.

5. ALGORITHMIC CONSIDERATIONS

As mentioned earlier, if one knows that the target function is weakly guessable, that is to say that there is an hypothesis automaton that computes it, the question becomes to find some suitable Q, T and λ . We reproduce, in Figure 5, the general algorithm given in [28], and that under certain conditions will find these Q, T and λ . In [28], three conditions are requested to ensure termination and correctness of the algorithm: the ascending chain condition, the progress measure and the solvability condition. We state slight variants of them. It appears that functions satisfying these conditions are enforced to be guessable. In [28], an example showing that this algorithm does not always terminate is given. This example corresponds to the automaton given in Figure 1d, that we have shown to be not weakly guessable. Since the algorithm can only guess hypothesis automata, it is clear that it will not terminate on any function that is not weakly guessable function.

Learning Algorithm

Initialisation: Set $Q = T = \{\varepsilon\}$.

Closure: While Q is not row-closed on T , do:

 | Pick $q \in Q, a \in \Sigma$ such that $\langle qa \rangle_T$ is not a left-linear combination of the rows in Q .
 | Set $Q := Q \cup \{qa\}$.

Test: Pick λ in $\Lambda_{Q,T}$. Construct the hypothesis automaton $\mathcal{H}_{Q,T,\lambda}$ and make an equivalence query. If the oracle gives a counter-example z , set $T = T \cup \{\text{suffixes of } z\}$, and go back to the Closure step. Else terminate with success.

FIGURE 5. Learning algorithm.

In the algorithm, membership queries are used to obtain the relevant parts of the Hankel matrix. An additional requirement for the algorithm is to be able to determine whether left-linear combinations exist, and when this is the case, to be able to find at least one such. This is given in the solvability condition.

Definition 5.1. A function f is solvable if there is an algorithm that, for any finite sets Q, T , can determine whether $\Lambda_{Q,T} \neq \emptyset$ and compute at least one element of $\Lambda_{Q,T}$ in that case.

Clearly, if S is a semiring where *there exists* an algorithm which can take as input an arbitrary system of linear equations over S and compute a solution, then all functions over the semiring are solvable. We note that in [28] such semirings were called *solvable*. It is clear that all finite semirings are solvable (a very naive algorithm would be to try all possibilities) and fields where the basic operations are computable are also solvable (via Gaussian elimination). In [28] the authors provide a sufficient condition for a principal ideal domain to be solvable. Turning to examples listed in Figure 4 which are neither finite, nor principal ideal domains, we note for example that over \mathbb{Z}_{\max} there is a straightforward (and well-known) method to determine whether a system of the form $A \otimes x = b$ has a solution. Indeed, if x is to be a solution, we must in particular have that $A_{i,j} + x_j \leq b_i$ for all i, j . If $A_{i,j} = -\infty$, then this inequality holds automatically, otherwise, we require that $x_j \leq b_i - A_{i,j}$. In order to solve the inequality, for each j such that there exists at least one i with $A_{i,j} \neq -\infty$ we could set $x_j = \min\{b_i - A_{i,j} : A_{i,j} \neq -\infty\}$, and set all remaining x_j equal to 0. It is clear that this is a computable (we use finitely many operations of subtraction and comparison on finitely many entries drawn from \mathbb{Z}_{\max}) solution to the inequality $A \otimes x \leq b$. In some sense it can be viewed as a “greatest” solution since the entries x_j where there exists $A_{i,j} \neq -\infty$ cannot be taken any larger (of course, if there are any columns of A where the entries are all equal to $-\infty$, the corresponding entry of x is irrelevant and our choice of taking $x_j = 0$ was completely arbitrary). Furthermore, it can also be seen that either the solution to the inequality is a solution to the equation $A \otimes x = b$, or else no solution to the equation $A \otimes x = b$ exists. (Indeed, if the solution to the inequality we constructed does *not* satisfy the equation, then we must have that for some j the strict inequality holds: $\max(A_{i,j} + x_j) < b_i$, meaning that either all $A_{i,j}$ are equal to $-\infty$ and so there is no hope to satisfy the equation $\max(A_{i,j} + x_j) < b_i$, or else in order to attain $A_{i,j} + x_j = b_i$ we require that one of the x_j with $A_{i,j}$ must take a larger value, which as we have seen, is not possible.) Thus \mathbb{Z}_{\max} (and likewise \mathbb{N}_{\max}) is solvable.

The following strong ascending chain condition ensures that the closure step always terminates in finite time for a fixed T and that it cannot be taken infinitely many times for different T ’s.

Definition 5.2. A function $f : \Sigma^* \rightarrow S$ with Hankel matrix F satisfies the strong ascending chain condition if for all sequences of left-semimodules $(X_i)_{i \in \mathbb{N}}$ generated by finite sets of rows of F , such that $X_0 \subseteq X_1 \subseteq X_2 \subseteq \dots$ there is n such that for all $m \geq n$, $X_m = X_n$.

Finally, the progress measure ensures that one cannot go through the Test step infinitely many times without changing the set Q in the closure step.

Definition 5.3. Let f be a function computed by a weighted automaton over S , let $\mathcal{L} := \{(Q, T) : Q, T \text{ are finite sets of words and } Q \text{ is row-closed on } T\}$. We say that f has a progress measure if there is a totally ordered set (\mathbb{K}, \succ) with no infinite decreasing chain and a function $\mu : \mathcal{L} \rightarrow \mathbb{K}$ with the property that whenever $\Lambda_{Q,T} \supsetneq \Lambda_{Q',T'}$ we have $\mu(Q, T) \succ \mu(Q', T')$.

All these conditions ensure termination and correctness of the algorithm. The following proposition derives from the results presented in [28].

Proposition 5.4. If a function f is solvable, satisfies the strong ascending chain condition and has a progress measure, then the learning algorithm from Figure 5 terminates and returns an automaton computing f .

6. CONCLUSION AND OPEN QUESTIONS

We have pictured a landscape for learning weighted automata on any semiring and shown that an approach à la Angluin can only work in general on a restricted class of functions. This paper opens many questions, both general and specific. First, for functions that are not weakly guessable, one can ask whether there is still a way to find appropriate rows (not from the Hankel matrix) by strengthening the queries to the oracle such as ones attained as limit points. Next, for weakly guessable functions, one can ask whether there is a way to design strategies to update Q and T (other than closure and adding suffixes of counter-examples to T) that would ensure to find a suitable hypothesis automaton. In particular, is there a way to choose “cleverly” the coefficients of the linear combinations under consideration? For guessable functions, a naive way to ensure

termination is to make sure to enumerate all the words in T . But is there a better way? And if so, it would be interesting to see if the class of functions learnable by a polynomial time algorithm can be characterised. Regarding the general hierarchy of functions, there are several classes for which we neither have an example of non-emptiness or an argument for collapse, in the non-commutative case. We have also left several open questions for the standard semirings \mathbb{Z}_{\max} and \mathbb{R}_{\max} in Figure 4.

ACKNOWLEDGMENT

This work has been supported by the EPSRC grant EP/T018313/1 and a Turing-Manchester Exchange Fellowship, The Alan Turing Institute.

REFERENCES

- [1] Cyril Allauzen, Mehryar Mohri, and Amee Talwalkar. Sequence kernels for predicting protein essentiality. In William W. Cohen, Andrew McCallum, and Sam T. Roweis, editors, *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, volume 307 of *ACM International Conference Proceeding Series*, pages 9–16. ACM, 2008.
- [2] Benjamin Aminof, Orna Kupferman, and Robby Lampert. Reasoning about online algorithms with weighted automata. *ACM Trans. Algorithms*, 6(2), apr 2010.
- [3] Benjamin Aminof, Orna Kupferman, and Robby Lampert. Formal analysis of online algorithms. In Tevfik Bultan and Pao-Ann Hsiung, editors, *Automated Technology for Verification and Analysis, 9th International Symposium, ATVA 2011, Taipei, Taiwan, October 11-14, 2011. Proceedings*, volume 6996 of *Lecture Notes in Computer Science*, pages 213–227. Springer, 2011.
- [4] Dana Angluin. Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75(2):87–106, 1987.
- [5] Dana Angluin, Sarah Eisenstat, and Dana Fisman. Learning regular languages via alternating automata. In Qiang Yang and Michael J. Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 3308–3314. AAAI Press, 2015.
- [6] Dana Angluin and Dana Fisman. Learning regular omega languages. In Peter Auer, Alexander Clark, Thomas Zeugmann, and Sandra Zilles, editors, *Algorithmic Learning Theory - 25th International Conference, ALT 2014, Bled, Slovenia, October 8-10, 2014. Proceedings*, volume 8776 of *Lecture Notes in Computer Science*, pages 125–139. Springer, 2014.
- [7] Borja Balle and Mehryar Mohri. [Learning weighted automata](#). In Andreas Maletti, editor, *Algebraic Informatics - 6th International Conference, CAI 2015, Stuttgart, Germany, September 1-4, 2015. Proceedings*, volume 9270 of *Lecture Notes in Computer Science*, pages 1–21. Springer, 2015.
- [8] Francesco Bergadano and Stefano Varricchio. [Learning behaviors of automata from multiplicity and equivalence queries](#). *SIAM J. Comput.*, 25(6):1268–1280, 1996.
- [9] Adrien Boiret, Aurélien Lemay, and Joachim Niehren. Learning rational functions. In Hsu-Chun Yen and Oscar H. Ibarra, editors, *Developments in Language Theory - 16th International Conference, DLT 2012, Taipei, Taiwan, August 14-17, 2012. Proceedings*, volume 7410 of *Lecture Notes in Computer Science*, pages 273–283. Springer, 2012.
- [10] Adrien Boiret, Aurélien Lemay, and Joachim Niehren. Learning top-down tree transducers with regular domain inspection. In Sicco Verwer, Menno van Zaanen, and Rick Smetsers, editors, *Proceedings of the 13th International Conference on Grammatical Inference, ICGI 2016, Delft, The Netherlands, October 5-7, 2016*, volume 57 of *JMLR Workshop and Conference Proceedings*, pages 54–65. JMLR.org, 2016.
- [11] Benedikt Bollig, Peter Habermehl, Carsten Kern, and Martin Leucker. Angluin-style learning of NFA. In Craig Boutilier, editor, *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 1004–1009, 2009.
- [12] Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Quantitative languages. *ACM Trans. Comput. Logic*, 11(4), jul 2010.
- [13] Thomas Colcombet, Daniela Petrisan, and Riccardo Stabile. Learning automata and transducers: A categorical approach. In Christel Baier and Jean Goubault-Larrecq, editors, *29th EACSL Annual Conference on Computer Science Logic, CSL 2021, January 25-28, 2021, Ljubljana, Slovenia (Virtual Conference)*, volume 183 of *LIPICs*, pages 15:1–15:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
- [14] Karel Culik and Jarkko Kari. *Digital Images and Formal Languages*, pages 599–616. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997.
- [15] Samuel Drews and Loris D’Antoni. Learning symbolic automata. In Axel Legay and Tiziana Margaria, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Part I*, volume 10205 of *Lecture Notes in Computer Science*, pages 173–189, 2017.
- [16] Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of Weighted Automata*. Springer Publishing Company, Incorporated, 1st edition, 2009.
- [17] Manfred Droste and Dietrich Kuske. Weighted automata. In Jean-Éric Pin, editor, *Handbook of Automata Theory*, pages 113–150. European Mathematical Society Publishing House, Zürich, Switzerland, 2021.
- [18] Michel Fliess. Matrices de hankel. *Journal de Mathématiques Pures et Appliquées*, 1974.

- [19] Falk Howar and Bernhard Steffen. **Active automata learning in practice - an annotated bibliography of the years 2011 to 2016**. In Amel Bennaceur, Reiner Hähnle, and Karl Meinke, editors, *Machine Learning for Dynamic Software Analysis: Potentials and Limits - International Dagstuhl Seminar 16172, Dagstuhl Castle, Germany, April 24-27, 2016, Revised Papers*, volume 11026 of *Lecture Notes in Computer Science*, pages 123–148. Springer, 2018.
- [20] Karel Culík II and Jarkko Kari. Image compression using weighted finite automata. *Comput. Graph.*, 17(3):305–313, 1993.
- [21] Daniel Krob. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. In Werner Kuich, editor, *Automata, Languages and Programming, 19th International Colloquium, ICALP92, Vienna, Austria, July 13-17, 1992, Proceedings*, volume 623 of *Lecture Notes in Computer Science*, pages 101–112. Springer, 1992.
- [22] Martin Leucker. **Learning meets verification**. In Frank S. de Boer, Marcello M. Bonsangue, Susanne Graf, and Willem P. de Roever, editors, *Formal Methods for Components and Objects, 5th International Symposium, FMCO 2006, Amsterdam, The Netherlands, November 7-10, 2006, Revised Lectures*, volume 4709 of *Lecture Notes in Computer Science*, pages 127–151. Springer, 2006.
- [23] Joshua Moerman, Matteo Sammartino, Alexandra Silva, Bartek Klin, and Michal Szynwelski. Learning nominal automata. In Giuseppe Castagna and Andrew D. Gordon, editors, *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017*, pages 613–625. ACM, 2017.
- [24] Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.
- [25] Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted automata in text and speech processing. *CoRR*, abs/cs/0503077, 2005.
- [26] Marcel-Paul Schützenberger. On the definition of a family of automata. *Information and Control*, 4(2):245–270, 1961.
- [27] Frits W. Vaandrager. **Model learning**. *Commun. ACM*, 60(2):86–95, 2017.
- [28] Gerco van Heerdt, Clemens Kupke, Jurriaan Rot, and Alexandra Silva. Learning weighted automata over principal ideal domains. In Jean Goubault-Larrecq and Barbara König, editors, *Foundations of Software Science and Computation Structures - 23rd International Conference, FOSSACS 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25-30, 2020, Proceedings*, volume 12077 of *Lecture Notes in Computer Science*, pages 602–621. Springer, 2020.
- [29] Gerco van Heerdt, Matteo Sammartino, and Alexandra Silva. CALF: Categorical Automata Learning Framework. In Valentin Goranko and Mads Dam, editors, *26th EACSL Annual Conference on Computer Science Logic (CSL 2017)*, volume 82 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 29:1–29:24, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- [30] Gerco van Heerdt, Matteo Sammartino, and Alexandra Silva. Learning automata with side-effects. In Daniela Petrisan and Jurriaan Rot, editors, *Coalgebraic Methods in Computer Science - 15th IFIP WG 1.3 International Workshop, CMCS 2020, Colocated with ETAPS 2020, Dublin, Ireland, April 25-26, 2020, Proceedings*, volume 12094 of *Lecture Notes in Computer Science*, pages 68–89. Springer, 2020.
- [31] Moshe Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, pages 327–338. IEEE Computer Society, 1985.
- [32] Juan Miguel Vilar. Query learning of subsequential transducers. In Laurent Miclet and Colin de la Higuera, editors, *Grammatical Inference: Learning Syntax from Sentences, 3rd International Colloquium, ICGI-96, Montpellier, France, September 25-27, 1996, Proceedings*, volume 1147 of *Lecture Notes in Computer Science*, pages 72–83. Springer, 1996.
- [33] Juan Miguel Vilar. Improve the learning of subsequential transducers by using alignments and dictionaries. In Arlindo L. Oliveira, editor, *Grammatical Inference: Algorithms and Applications, 5th International Colloquium, ICGI 2000, Lisbon, Portugal, September 11-13, 2000, Proceedings*, volume 1891 of *Lecture Notes in Computer Science*, pages 298–311. Springer, 2000.
- [34] Zeming Wei, Xiyue Zhang, and Meng Sun. Extracting weighted finite automata from recurrent neural networks for natural languages. In Adrián Riesco and Min Zhang, editors, *Formal Methods and Software Engineering - 23rd International Conference on Formal Engineering Methods, ICFEM 2022, Madrid, Spain, October 24-27, 2022, Proceedings*, volume 13478 of *Lecture Notes in Computer Science*, pages 370–385. Springer, 2022.
- [35] Gail Weiss, Yoav Goldberg, and Eran Yahav. Extracting automata from recurrent neural networks using queries and counterexamples. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5244–5253. PMLR, 2018.

APPENDIX A. SWAPPING ROWS AND COLUMNS

In this section, we present Section 2 and Section 3 swapping rows and columns, left and right and initial and final states. S still denotes a semiring and Σ an alphabet.

A.1. Preliminaries. Given a set Y , an element x of S^Y and λ in S , $x \otimes \lambda$ denotes the element of S^Y computed from x by multiplying every component of x by λ on the right. Given a set $X = \{x_1, x_2, \dots\}$ of elements of S^Y , a right-linear combination over X is one of the form $(x_1 \otimes \lambda_1) \oplus (x_2 \otimes \lambda_2) \oplus \dots$ for some $\lambda_1, \lambda_2, \dots$ in S , such that only a finite number are different from 0_S . The right-semimodule generated by X is defined as all the elements of S^Y that can be written as a right-linear combination over X . An element of the right-semimodule generated by X will simply be said to be right-generated by X .

A.2. Hankel matrices and co-hypothesis automaton. We will use the notation $[x]$ for elements of S^{Σ^*} viewed as infinite columns; thus for a word $w \in \Sigma^*$ we write $[x]_w$ to denote the w -th component of $[x]$. The restriction of an infinite column to entries indexed by a subset $Z \subseteq \Sigma^*$ will be denoted by $[x]_Z$. Given an infinite column $[x] \in S^{\Sigma^*}$, and a letter a of Σ , we denote by $[ax]$ the infinite column defined by $[ax]_w = [x]_{wa}$ for all words w . If x is a word and f and F are clear from context, we shall write $[x]$ to denote the infinite column indexed by x in F . Note that if x is a word, f a function and F its Hankel matrix, this definition is consistent: $[ax]_u = F_{u,ax} = f(ua) = F_{ua,x} = [x]_{ua}$. For a set of columns X and $[x]$ in X , we will sometimes simplify notation by writing $x \in X$.

Closed and generating sets. Let $Z \subseteq \Sigma^*$ and $f : \Sigma^* \rightarrow S$. A subset $X \subseteq S^{\Sigma^*}$ is said to be: (i) column-closed if for all $[x]$ in X and $a \in \Sigma$, $[ax]$ is right-generated by X ; (ii) column-closed on Z if for all $[x]$ in X and $a \in \Sigma$, $[ax]_Z$ is right-generated by the elements of X restricted to Z ; (iii) column-generating for f if all the columns of the Hankel matrix of f are right-generated by X . By an abuse of notation, we say that a set of words Q is column-closed (resp. column-closed on Z , column-generating for f) if the set $X = \{[q] : q \in Q\}$ has this property.

Co-Hankel automata and co-hypothesis automata. Given a function $f : \Sigma^* \rightarrow S$, a finite subset T of S^{Σ^*} and a subset Q of Σ^* , we define $\Gamma_{Q,T} \subseteq S^T \times S^{T \times \Sigma \times T}$ to be the set of tuples $\lambda = (\lambda_q^\varepsilon, \lambda_{q,a,p})_{q \in T, a \in \Sigma, p \in T}$ such that:

- $[\varepsilon]_Q = \bigoplus_{q \in T} [q]_Q \otimes \lambda_q^\varepsilon$, and
- for all $q \in T$ and all $a \in \Sigma$, $[aq]_Q = \bigoplus_{p \in T} [p]_Q \otimes \lambda_{q,a,p}$.

Note that $\Gamma_{Q,T}$ is non-empty if and only if T is column-closed on Q and $[\varepsilon]_Q$ is right-generated by T restricted to Q . Note also that if $Q \subseteq Q'$, then $\Gamma_{Q,T} \supseteq \Gamma_{Q',T}$. If $Q = \Sigma^*$, we simply write Γ_T .

Definition A.1. Given a finite subset T of S^{Σ^*} and a subset Q of Σ^* , for each $\lambda \in \Gamma_{Q,T}$, we define a finite weighted automaton $\mathcal{H}_{Q,T,\lambda}^{\text{co}}$ with:

- finite set of states T ;
- initial-state vector $([q]_\varepsilon)_{q \in T}$;
- final-state vector $(\lambda_q^\varepsilon)_{q \in T}$; and
- for all $q, p \in T$ and all $a \in \Sigma$, a transition from p to q labelled by a with weight $\lambda_{q,a,p}$.

If $Q = \Sigma^*$, we say $\mathcal{H}_{Q,T,\lambda}^{\text{co}}$ is a co-Hankel automaton and write simply $\mathcal{H}_{T,\lambda}$ to reduce notation. If T is a finite set of columns of the Hankel matrix and Q is finite, we say that $\mathcal{H}_{Q,T,\lambda}^{\text{co}}$ is a co-hypothesis automaton, and view T as a finite set of words by identifying with an appropriate index set.

Lemma A.2. Given a function f and a finite subset T of S^{Σ^*} , for all λ in Γ_T , the automaton $\mathcal{H}_{T,\lambda}^{\text{co}}$ computes f .

Theorem A.3. Let $f : \Sigma^* \rightarrow S$ be a function. There exists a finite set of columns $T \subseteq S^{\Sigma^*}$ that is column-closed and right-generates $\langle \varepsilon \rangle$ if and only if f is computed by a finite weighted automaton \mathcal{A} over S . If the first condition holds, then $\mathcal{H}_{T,\lambda}^{\text{co}}$ computes f for all $\lambda \in \Gamma_T \neq \emptyset$, whilst if the second condition holds, the set of all $[q]$ where q is a state of \mathcal{A} and $[q]_w$ is the weight of w to q in \mathcal{A} is column-closed and right-generates every column of the Hankel matrix of f .

A.3. Weakly co-guessable functions.

Definition A.4. A function $f : \Sigma^* \rightarrow S$ is said to be weakly co-guessable if there exist a finite set T of columns of the Hankel matrix of f such that Γ_T is non empty.

Definition A.5. An automaton is said to be co-literal if its mirror is literal.

Definition A.6. Given a function $f : \Sigma^* \rightarrow S$ and F its Hankel matrix, f satisfies the weak co-ascending chain condition if for all sequences of right-semimodules $(X_i)_{i \in \mathbb{N}}$ generated by finite sets of columns of F , such that all columns of F belongs to $\cup_{i \in \mathbb{N}} X_i$ and

$$X_0 \subseteq X_1 \subseteq X_2 \subseteq \dots$$

there is n such that for all $m \geq n$, $X_m = X_n$.

Proposition A.7. Given a function $f : \Sigma^* \rightarrow S$, the following assertions are equivalent:

- (1) f is weakly co-guessable;
- (2) there exists a co-literal automaton computing f ;
- (3) there exists a finite subset of Σ^* that is column-generating for f ;
- (4) f satisfies the weak co-ascending chain condition.

Lemma A.8. If S is a commutative semiring then a function is weakly guessable if and only if its mirror is weakly co-guessable.

A.4. Co-guessable and strongly co-guessable functions.

Definition A.9. A function $f : \Sigma^* \rightarrow S$ is co-guessable if there exist finite subsets $Q, T \subseteq \Sigma^*$ such that $\Gamma_{Q,T} = \Gamma_T \neq \emptyset$. We say that the pair (Q, T) witnesses that f is co-guessable. Furthermore, we say that f is strongly co-guessable if for all finite sets $T \subseteq \Sigma^*$ with $\Gamma_T \neq \emptyset$ there exists a finite set $Q \subseteq \Sigma^*$ such that (Q, T) witnesses that f is co-guessable.

Definition A.10. We say that a function $f : \Sigma^* \rightarrow S$ is column-bound if there exists a finite set of words T such that $\Gamma_T \neq \emptyset$ and there is no infinite descending chain of the form $\Gamma_{Q_0,T} \supsetneq \Gamma_{Q_1,T} \supsetneq \Gamma_{Q_2,T} \supsetneq \dots$ where $(Q_i)_{i \geq 0}$ is a total sequence of words. We say that $f : \Sigma^* \rightarrow S$ is strongly column-bound if for every finite set of words T such that $\Gamma_T \neq \emptyset$ there is no infinite descending chain of the form $\Gamma_{Q_0,T} \supsetneq \Gamma_{Q_1,T} \supsetneq \Gamma_{Q_2,T} \supsetneq \dots$ where $(Q_i)_{i \geq 0}$ is a total sequence of words.

Proposition A.11. Let $f : \Sigma^* \rightarrow S$ be a weakly co-guessable function. Then f is (strongly) co-guessable if and only if f is (strongly) column-bound.

Lemma A.12. For a commutative semiring, a function is (strongly) guessable if and only if its mirror is (strongly) co-guessable.