arXiv:2309.07806v5 [cs.FL] 24 Jun 2025

FEASIBILITY OF LEARNING WEIGHTED AUTOMATA ON A SEMIRING

LAURE DAVIAUD [a] AND MARIANNE JOHNSON [b]

[a] University of East Anglia, UK

[b] University of Manchester, UK

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 to learn various extensions of automata. For weighted automata, algorithms for restricted cases have been developed in the literature, but so far there was no global approach or understanding how these algorithms could apply (or not) in the general case. In this paper we chart the boundaries of the Angluin approach. We use a class of *hypothesis automata* which are constructed, in Angluin's style, by using membership and equivalence queries and solving certain finite systems of linear equations over the semiring, and we show the theoretical limitations of this approach. We classify functions with respect to how 'guessable' they are, corresponding to the existence of hypothesis automata computing a given function, and how such an hypothesis automaton can be found. Of course, from an algorithmic standpoint, knowing that a solution (hypothesis automaton) exists need not translate into an effective algorithm to find one. We relate our work to the existing literature with a discussion of some known properties ensuring algorithmic solutions, illustrating the ideas over several familiar semirings (including the natural numbers).

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 (finitely many) input-output pairs, 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 using this new input-output pair together with further observations, you construct a new model to test. This is active learning.

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

© LEARNING WEIGHTED AUTOMATA
Creative Commons

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 [26, 23, 35]. In particular, the L^*-algorithm has been used in [43] 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 [33, 19, 20]. They find applications in image compression [24, 16], language and speech processing [29, 30], bioinformatics [1], formal verification [14, 3], analysis of on-line algorithms [2] and probabilistic systems [39]. Notably, they have also been used recently to model recurrent neural networks [32, 31, 44, 42].

Extending automata learning: Active learning has been extended to weighted automata but only for restricted classes: it has first been studied for fields [9, 7], and then principal ideal domains [36]. The Angluin framework has also been generalised to other extensions of automata and transducers [40, 41, 12, 10, 6, 5, 11, 18, 28, 38]. In [15] (see also [37] [8] and [34] for earlier works), 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 [36], van Heerdt, Kupke, Rot and Silva set a general framework for learning weighted automata, present a general learning algorithm, and focus on specifying two conditions on the semirings (ascending chain condition and 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. More recently, a polynomial-time algorithm within this general framework has been developed in [13] for the principal ideal domain $\mathbb{Z}$. Section 5 and Theorem 17 of [36] also present a specific example of an automaton (as well as a more general class) over the non-negative integers for which their algorithm does not terminate. In the present paper, we develop a general framework that gives a deeper understanding of the reasons for that.

Contributions: Our approach is somewhat different to what has been done so far: we look globally at all functions computed by finite state weighted automata and investigate, for each of them, whether they *can potentially be learned* by an Angluin-style algorithm, i.e. where the automata that are guessed (hypothesis automata, see Definition 2.10) are constructed in a prescribed way using finitely many membership queries (to be explained in detail below). Working over an arbitrary semiring, we define levels of guessability and a hierarchy of functions with respect to these levels: from the ones for which no such guess will be correct to the ones where there is a simple theoretical strategy to construct a correct guess. Our main result in this regard (Theorem 3.6) is a syntactic condition on the functions that have the potential to be learnt. For such a function (weakly guessable functions, see Definition 3.1) there must exist a finite state automaton of a prescribed type computing it: we call these automata *literal* (Definition 3.2) and they are closely related to residual automata. In cases already widely studied in the literature (e.g. where the weights are drawn from fields or integral domains), it turns out that *every* function computed by a finite state automaton can be computed by a literal automaton. Thus, in the existing literature, focus is (reasonably) on algorithmic issues. In this paper, we focus on picturing the landscape for active learning in arbitrary semirings, and showing the limitations; for instance over the non-negative integers, it is easily seen that the example of van Heerdt, Kupke, Rot and Silva mentioned in the previous paragraph cannot be computed by a literal automaton.

Within this framework, we then consider some properties of semirings that cause certain parts of our general hierarchy to collapse, and consider several specific well-studied examples of semirings (including the non-negative integers) in this context and provide a more detailed picture of the hierarchy for these (Section 4). Note that the learning process is based on equivalence queries, and one might ask whether the undecidability of equivalence for certain semirings [25] 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.

The difficulty of this work was in setting a new framework and the (we believe) right notions for it. Once the correct definitions are in place, the results and proofs follow nicely.

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 which motivate the definitions and results in the rest of the paper. In Section 3, we explain what we mean by levels of guessability and define classes of functions depending on them. We give several characterisations for these classes of functions, and relate our work back to the existing literature (specifically, [36]), which focuses on algorithmic issues in specific settings. In Section 4, we picture the hierarchy based on the previous definitions, prove its strictness and give conditions which cause parts of the hierarchy to collapse. Throughout we raise a number of questions and indicate directions for future progress, summarising these in Section 5.

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.

We begin by recording some preliminary definitions and notation. A finite alphabet Σ is a finite set of symbols, called letters. A finite word is a finite sequence of letters and ε denotes the empty word. We write Σ^* to denote the set of all words over the alphabet Σ, including 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 $(M, *, e)$ is a set M equipped with a binary operation $*$ that is associative and has a neutral element e. For example, the set Σ^* together with the binary operation of concatenation of words and the empty word ε is a monoid; it is the free monoid on the set Σ. 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.

Recall that for a set Y, we write S^Y to denote the set of all functions from Y to S. For non-empty (but possibly infinite) countable sets I and J we say that $A \in S^{I\times J}$ is a matrix, with rows indexed by the set I and columns indexed by the set J. For $A \in S^{I\times J}$ and $B \in S^{J\times K}$ where J is finite we define the product AB in the usual way via $(AB)_{i,k} = \bigoplus_{j\in J}(A_{i,j} \otimes B_{j,k})$. Analogously, it will sometimes also be convenient to view elements of S^J as (row or column) 'vectors' with entries indexed by J; for $\underline{x} \in S^J$ and $j \in J$ we write $\underline{x}_j$ to denote the value computed by the function $\underline{x}$ on input j, and if J is finite we define the product $A\underline{x}$ where $A \in S^{I\times J}$ in the obvious way. It is also straightforward to check that this multiplication is associative, that is, for all $A \in S^{I\times J}$, $B \in S^{J\times K}$, $C \in S^{K\times L}$, where J and L are both finite, we have $A(BC) = (AB)C$.

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 $\underline{\alpha} \in S^Q$ (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 $\underline{\eta} \in S^Q$ (which we view as a column vector). The automaton $\mathcal{A}$ computes a function $f_{\mathcal{A}} : \Sigma^* \to S$ via $f_{\mathcal{A}}(w_1 \cdots w_k) = \underline{\alpha}M(w_1)\cdots M(w_k)\underline{\eta}$. 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 (labelled, directed) graph with set of vertices Q, such that (i) for each state $p \in Q$ there is an initial weight $\underline{\alpha}_p$ (represented by an incoming arrow at state p with label $\underline{\alpha}_p$) and a final weight $\underline{\eta}_p$ (represented by an outgoing arrow at state p with label $\underline{\eta}_p$); and (ii) for each pair of states $p, q \in Q$ and each $a \in \Sigma$ there is a transition (labelled edge) $p \xrightarrow{a\,:\,m} q$ where $m = M(a)_{p,q}$ is called the weight of the transition (and if $m = 0_S$, then we often omit this edge from the graph). A run from a state p to a state q is a (directed) path in the graph and the weight of a run from p to q is the product of $\underline{\alpha}_p$ by the product of the weights on the transitions (taken in the order traversed) followed by $\underline{\eta}_q$. The value computed on a word w is the sum of all weights of runs labelled by w. A state p such that $\underline{\alpha}_p$ (resp. $\underline{\eta}_p$) is not 0_S is referred to as an initial (resp. final) state. We use both the matrix and graph vocabulary in this paper, as convenient.

Notations. Throughout the paper, Σ denotes a finite alphabet and S a semiring (which will not be assumed to be commutative unless explicitly stated). The lower case symbols a, b, c will be exclusively used to denote letters of an alphabet, whilst the symbols u, v, w will be exclusively used to denote words over the alphabet Σ. We use calligraphic upper-case symbols such as $\mathcal{A}, \mathcal{B}, \mathcal{H}$ to denote weighted automata, whilst the lower-case symbols f, g, h are always used to denote functions from Σ^* to S. Greek letters such as $\alpha, \beta, \lambda, \mu, \rho, \eta$ will be used to denote elements of the semiring S, upper-case symbols $A, B, \ldots, F$ will be used to denote (possibly infinite) matrices over S, and underlined symbols (e.g. $\underline{x}$ or $\underline{\lambda}$) will denote (possibly infinite) vectors over S; in both cases the index sets used vary but will be made explicit. Upper-case symbols $I, J, \ldots, Z$ will typically be used for sets; frequently these will be (countable) subsets of S^{Σ^*} or Σ^*, and moreover such subsets will frequently be used as indexing sets.

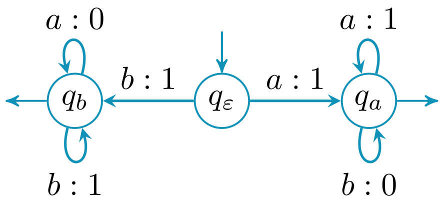
Specific semirings and examples: Besides giving a general framework valid for any semiring, we consider several specific well-studied semirings that arise frequently in application areas. We will consider the semiring $(\mathbb{R}, +, \times, 0, 1)$ and its restriction to non-negative reals $\mathbb{R}_{\geq 0}$, 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 (playing the role of addition) and concatenation ($XY = \{xy \mid x \in X, y \in Y\}$, playing the role of multiplication) and neutral elements $\emptyset$ and $\{\varepsilon\}$. This semiring will be denoted $\mathcal{P}_{\text{fin}}(\Sigma^*)$.

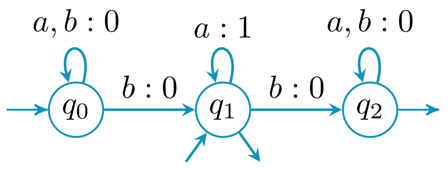
Example 2.2. We consider $\Sigma = \{a, b\}$. Figure 1 depicts several weighted automata where the initial (resp. final) states with weight 1_S are indicated with an ingoing (resp. outgoing) arrow, the other states have initial or final weights 0_S. All six examples will be used in Section 4 and the 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 word 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. It is worth mentioning here that several of the examples above could be viewed as examples over smaller semirings (e.g. Figure 1b could be considered over the semiring $\mathbb{N}_{\max}$). However, as we shall see below, changing the semiring over which one views a given example necessarily changes the space of hypothesis automata that one can construct using membership and equivalence queries.

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, S^Y is a left S-module via the following action: for each element $\underline{x}$ of S^Y and λ in S, $\lambda \otimes \underline{x}$ denotes the element of S^Y computed from $\underline{x}$ by multiplying every component of $\underline{x}$ by λ on the left. Given a subset $X \subseteq S^Y$, a left-linear combination over X is one of the form $\bigoplus_{\underline{x} \in X} \lambda_{\underline{x}} \otimes \underline{x}$ for some $\lambda_{\underline{x}}$ in S, with only finitely many $\lambda_{\underline{x}}$ 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. (Dually, one can view S^Y as a right S-module, and make the corresponding definitions of right-linear combination, right-semimodule generated by X etc.)

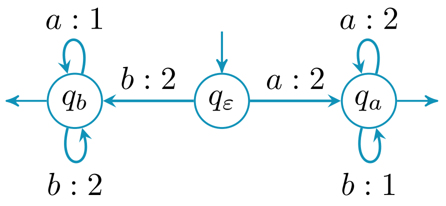
Mirror: Finally, we will use the mirror operation on functions $\Sigma^* \to S$. The mirror of a word $w = w_1 w_2 \cdots w_n$ where for all $i = 1, \ldots, n$, $w_i \in \Sigma$ is the word $\bar{w} = w_n \cdots w_2 w_1$. For $f : \Sigma^* \to S$, the mirror of f, denoted by $\bar{f} : \Sigma^* \to S$ is defined as $\bar{f}(w) = f(\bar{w})$. Given a weighted automaton $\mathcal{A}$, the mirror of $\mathcal{A}$ is obtained by reversing each of its transitions and swapping the initial-state vector and the final-state vector. For example, the automaton in Figure 1B is equal to its mirror. 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



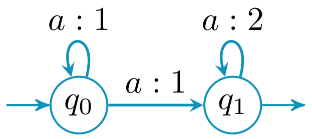
(A) Over $\mathbb{N}_{\max}$: computes $|w|_x$ if w starts with letter $x \in \{a,b\}$ and $-\infty$ on ε.



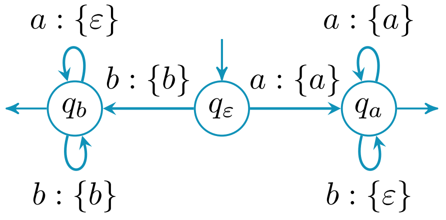
(B) Over $\mathbb{R}_{\max}$: computes the length of the longest block of consecutive a's in w.



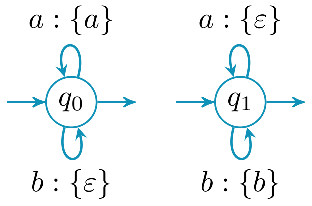
(C) Over $\mathbb{N}$: computes $2^{|w|_x}$ if w starts with letter $x \in \{a,b\}$ and 0 if $w = \varepsilon$.



(D) Over $\mathbb{R}_{\geq 0}$: computes $2^n - 1$ if $w = a^n$ for some $n \geq 1$, and 0 otherwise.



(E) Over $\mathcal{P}_{\text{fin}}(\Sigma^*)$: computes $\{x^{|w|_x}\}$ if w starts with letter $x \in \{a,b\}$ and $\emptyset$ if $w = \varepsilon$.



(F) Over $\mathcal{P}_{\text{fin}}(\Sigma^*)$: computes $\{a^{|w|_a}, b^{|w|_b}\}$

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

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).

2.2. Learning with membership and equivalence queries.

We are investigating one type of learning done by membership and equivalence queries [4, 9, 7] where the learner makes guesses by solving systems of left-linear (or right-linear) equations over S. Specifically, suppose that there is an oracle who can answer two types of questions about a function $f : \Sigma^* \to S$:

- **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" (sometimes also called "output 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 explain below. In certain circumstances (i.e. by placing conditions on the weighted automaton and/or the semiring) it is already known that one can efficiently learn a weighted automaton computing the function f using finitely membership and equivalence queries. Indeed, the underlying ideas introduced in the remainder of this section have been introduced and studied in the literature; we give proofs of some known results for completeness, putting these into our general framework. Our presentation differs slightly and leads to questions that are answered in Sections 3 and 4 and represent our main contribution.

Hankel matrix: As before, we use the notation $\underline{x}$ for elements of S^{Σ^*}, and for a word $w \in \Sigma^*$ we write $\underline{x}_w$ to denote the w-th component of $\underline{x}$. The restriction of $\underline{x}$ to a subset $Z \subseteq \Sigma^*$ will be denoted by $\underline{x}_Z$. We shall also make use of the following right action of Σ^* on S^{Σ^*}: given $\underline{x} \in S^{\Sigma^*}$ and $u \in \Sigma^*$, we denote by $\underline{x} \cdot u$ the element of S^{Σ^*} defined as the shift of $\underline{x}$ by u, that is, $(\underline{x} \cdot u)_w = \underline{x}_{uw}$ for all words w. The right action of Σ on S^{Σ^*} commutes with the left action of S on S^{Σ^*} in the sense that for all $\lambda \in S$, $\underline{x} \in S^{\Sigma^*}$ and $u \in \Sigma^*$ we have $(\lambda \cdot \underline{x}) \cdot u = \lambda \cdot (\underline{x} \cdot u)$. Given a function $f : \Sigma^* \to S$, the infinite matrix F whose rows and columns are indexed by Σ^* and with entries $F_{w,w'} = f(ww')$ is called the Hankel matrix of f. It will also be convenient to have a notation for elements of S^{Σ^*} that arise as rows of the Hankel matrix. If $v \in \Sigma^*$ is a word and f and F are clear from context, we shall write $\langle v \rangle$ to denote the infinite row indexed by v in F, viewed as an element of S^{Σ^*}. Note that if v is a word, f a function and F its Hankel matrix, then the right action of Σ^* on $\langle v \rangle$ corresponds to the obvious right action of Σ^* on v, that is: $(\langle v \rangle \cdot u)_w = (\langle v \rangle)_{uw} = F_{v,uw} = f(vuw) = F_{vu,w} = \langle vu \rangle_w$.

Example 2.3. Consider the function f 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 $f : \Sigma^* \to S$. A subset $X \subseteq S^{\Sigma^*}$ is said to be: (i) left-closed if for all $\underline{x}$ in X and $a \in \Sigma$, $\underline{x} \cdot a$ is left-generated by X; (ii) left-closed on Z if for all $\underline{x}$ in X and $a \in \Sigma$, $(\underline{x} \cdot a)_Z$ 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 language, we say that a set of words[1] Q is left-closed (resp. left-closed on Z, row-generating for f) if the set $\underline{Q} = \{\langle q \rangle : q \in Q\}$ has this property.

The following simple observations are key in what follows.

Lemma 2.4. *Let $X \subseteq S^{\Sigma^*}$.*

(1) *If X is left-closed and left-generates the row $\langle \varepsilon \rangle$ of the Hankel matrix of f then X is row-generating for f.*
(2) *If X is a subset of the rows of the Hankel matrix and is a row-generating set for f, then X is left-closed (and left-generates the row $\langle \varepsilon \rangle$).*

[1]Using 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.

Proof. (1) Since X left generates $\langle\varepsilon\rangle$ we may write $\langle\varepsilon\rangle = \bigoplus_{\underline{x}\in X} \lambda_{\underline{x}} \otimes \underline{x}$, where only finitely many of the coefficients $\lambda_{\underline{x}}$ are non-zero. Then for all words $u, v \in \Sigma^*$ we have

$$\langle u\rangle_v = \langle\varepsilon\rangle_{uv} = \bigoplus_{\underline{x}\in X} \lambda_{\underline{x}} \otimes \underline{x}_{uv} = \bigoplus_{\underline{x}\in X} \lambda_{\underline{x}} \otimes (\underline{x}\cdot u)_v.$$

Then, using the fact that X is left-closed, an easy induction on the length of u then gives that $\underline{x}\cdot u$ is left-generated by X, in turn showing that each row $\langle u\rangle$ of the Hankel matrix is a left-linear combination of elements of X.
(2) That X left-generates $\langle\varepsilon\rangle$ is completely clear from definition of row generating set. Since X is a subset of the rows of the Hankel matrix of f, each element of X may be written in the form $\langle w\rangle$ for some $w \in \Sigma^*$, in which case $wa \in \Sigma^*$ for all $a \in \Sigma$, and hence $\langle w\rangle \cdot a = \langle wa\rangle$ is left-generated by X. □

Example 2.5. 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 left-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 left-closed on $\{\varepsilon\}$, since $\langle b\rangle_\varepsilon = \langle ab\rangle_\varepsilon = 1 = 0 + 1 = 0 \otimes \langle a\rangle_\varepsilon$ and $\langle aa\rangle_\varepsilon = 2 = 1 + 1 = 1 \otimes \langle a\rangle_\varepsilon$. The set $\{\langle\varepsilon\rangle, \langle a\rangle, \langle b\rangle\}$ is left-closed and also row-generating since for $x \in \{a, b\}$, $\langle xw\rangle = |w|_x \otimes \langle x\rangle$.

It turns out that if one can find a finite subset $X \subseteq S^{\Sigma^*}$ that is left-closed and such that the row $\langle\varepsilon\rangle$ is left-generated by X, then by solving the corresponding system of S-linear equations involving the (infinite) vectors in $X \cup \{\langle\varepsilon\rangle\}$ one can construct an automaton computing the correct function. We call this a Hankel automaton and describe its construction below. However, in practice, using finitely many membership queries only 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 subset X 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: We first set up some notation to describe the construction of hypothesis automaton.

Notation 2.6. Given a function f (and in particular the row indexed by ε in its Hankel matrix), a finite non-empty subset Q of vectors of S^{Σ^*} and a non-empty subset T of Σ^*, we define the left solution set $\Lambda_{Q,T}$ to be the set of vectors $\underline{\lambda} \in S^{Q\cup(Q\times\Sigma\times Q)}$ satisfying:

- $\langle\varepsilon\rangle_T = \bigoplus_{\underline{q}\in Q} \underline{\lambda}_{\underline{q}} \otimes \underline{q}_T$, and
- for all $\underline{q} \in Q$ and all $a \in \Sigma$, $(\underline{q}\cdot a)_T = \bigoplus_{\underline{p}\in Q} \underline{\lambda}_{\underline{q},a,\underline{p}} \otimes \underline{p}_T$.

If $T = \Sigma^*$, we will simply write Λ_Q instead of Λ_{Q,Σ^*}. Note that the left solution sets $\Lambda_{Q,T}$ and Λ_Q depend on the function f (and hence also the semiring S), although we have chosen to suppress this dependence in our notation for readability. If W is a finite subset of Σ^* we will also, by a slight abuse of notation, write Λ_W and $\Lambda_{W,T}$ to mean Λ_Q and $\Lambda_{Q,T}$ where $Q = \{\langle w\rangle : w \in W\}$.

Remark 2.7. An element $\underline{\lambda} \in \Lambda_{Q,T}$ represents a collection of coefficients witnessing that $\langle\varepsilon\rangle_T$ and each of the $(\underline{q}\cdot a)_T$ (for all $\underline{q} \in Q$ and all $a \in \Sigma$) is a left-linear combinations of the $\underline{q}_T$ for

q in Q. In other words, $\Lambda_{Q,T}$ represents the solution space to the system of S-linear equations outlined in Notation 2.6. The key idea here is that the set $\Lambda_{Q,T}$ is non-empty if and only if a solution to this system exists; in the language introduced above, if and only if Q is left-closed on T and $\langle\varepsilon\rangle_T$ is left-generated by Q restricted to T.

Remark 2.8. Let $T \subseteq \Sigma^*$, and let $Q \subseteq S^{\Sigma^*}$ be a finite set. If $T \subseteq T' \subseteq \Sigma^*$, then it is clear from the above equations that $\Lambda_{Q,T} \supseteq \Lambda_{Q,T'} \supseteq \Lambda_Q$. However, if Q' is a finite set contained in or contained by Q, then there is no containment between $\Lambda_{Q,T}$ (which lies in $S^{Q\cup(Q\times\Sigma\times Q)}$) and $\Lambda_{Q',T}$ (which lies in $S^{Q'\cup(Q'\times\Sigma\times Q')}$).

Lemma 2.9. *Let $f : \Sigma^* \to S$ be a function, $T \subseteq \Sigma^*$, and suppose that $Q' \subseteq Q$ are finite sets of rows of the Hankel matrix of f. If $\Lambda_{Q'} \neq \emptyset$ then $\Lambda_{Q,T} \neq \emptyset$.*

Proof. Since each element of Q is a row of the Hankel matrix, there exists a finite set of words $W \subseteq \Sigma^*$ such that $Q = \{\langle w\rangle : w \in W\}$. Since $\Lambda_{Q'} \neq \emptyset$, we have that Q' is left-closed and left-generates the row $\langle\varepsilon\rangle$, and so Q' is row-generating for f, by Lemma 2.4. In particular, $\langle\varepsilon\rangle$ and each row $\langle wa\rangle$ where $w \in W$ and $a \in \Sigma$ can be expressed as a left linear combination of the elements of Q', and hence also of Q. Since $\langle wa\rangle = \langle w\rangle \cdot a$, it then follows easily that the system of equations given in Notation 2.6 has a solution. $\square$

Definition 2.10. Let $f : \Sigma^* \to S$ be a function, and suppose that Q is a finite subset of S^{Σ^*} and T a subset of Σ^* such that $\Lambda_{Q,T} \neq \emptyset$. Then for each $\underline{\lambda} \in \Lambda_{Q,T}$, we define a finite weighted automaton $\mathcal{H}_{Q,T,\underline{\lambda}}$ with:

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

If Q is a finite set of rows of the Hankel matrix of f and $T = \Sigma^*$, we say $\mathcal{H}_{Q,T,\underline{\lambda}}$ is a *Hankel automaton* and write simply $\mathcal{H}_{Q,\underline{\lambda}}$ to reduce notation. If Q is a finite set of rows of the Hankel matrix and T is finite, we say that $\mathcal{H}_{Q,T,\underline{\lambda}}$ is an *hypothesis automaton*; in this case we may view Q as a finite set of words by identifying with an appropriate index set (note: there need not be a unique set of words).

Example 2.11. Consider the function f over $\mathbb{N}_{\max}$ computed by our running example from Figure 1a. The set $\{\langle\varepsilon\rangle, \langle a\rangle, \langle b\rangle\}$ containing three rows of the Hankel matrix clearly left-generates $\langle\varepsilon\rangle$ and, as shown previously, is left-closed. Let q_ε, q_a and q_b denote these rows (omitting the underlining of these elements for ease of reading). Then considering the coefficients $\underline{\lambda}_{q_\varepsilon} = 0$, $\underline{\lambda}_{q_\varepsilon,a,q_a} = \underline{\lambda}_{q_\varepsilon,b,q_b} = \underline{\lambda}_{q_a,b,q_a} = \underline{\lambda}_{q_b,a,q_b} = 0$, $\underline{\lambda}_{q_a,a,q_a} = \underline{\lambda}_{q_b,b,q_b} = 1$ and all non-specified coefficients $-\infty$, the corresponding Hankel automaton would be the automaton with three states 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 left-closed on T. By taking $\underline{\lambda}_{q_\varepsilon} = 0$, $\underline{\lambda}_{q_\varepsilon,a,q_a} = \underline{\lambda}_{q_\varepsilon,b,q_a} = \underline{\lambda}_{q_a,b,q_a} = 0$, $\underline{\lambda}_{q_a,a,q_a} = 1$ 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 left-closed on T'.

Fliess's theorem [21] states that a function is computed by a weighted automaton over the field of reals if and only if its Hankel matrix has finite rank, and moreover, this rank is exactly the minimal number of states of a weighted automaton computing the function. There are many equivalent definitions of the rank of a matrix with entries over a field whose analogues cease to coincide over an arbitrary semirings, however, the proof of Fliess' Theorem can be easily adapted to obtain the following statement, and since the details are instructive, we include a short proof.

Theorem 2.12. *Let $f : \Sigma^* \to S$ be a function. The following are equivalent:*

(1) *there exists a finite subset $Q \subseteq S^{\Sigma^*}$ that is both left-closed and left-generates the row indexed by ε in the Hankel matrix of f (i.e. $\Lambda_Q \neq \emptyset$ for some finite subset $Q \subseteq S^{\Sigma^*}$);*
(2) *f is computed by a finite-state weighted automaton $\mathcal{A}$ over S.*

Specifically, if $\Lambda_Q \neq \emptyset$, then $\mathcal{H}_{Q,\underline{\lambda}}$ computes f for all $\underline{\lambda} \in \Lambda_Q \neq \emptyset$, whilst if f is computed by the finite-state weighted automaton $\mathcal{A}$ over S with state set Q, then $\Lambda_{\underline{Q}} \neq \emptyset$ for $\underline{Q} = \{\underline{q} : q \in Q\} \subseteq S^{\Sigma^*}$ where for each $w \in \Sigma^*$ we define $\underline{q}_w$ to be the value on input w computed by the automaton $\mathcal{A}_q$ obtained from $\mathcal{A}$ by making q the unique initial state with weight 1_S and all other weights are the same.

Remark 2.13. Before presenting the proof, we alert the reader to the fact that in Theorem 2.12 the set Q is not assumed to be a set of rows of the Hankel matrix of f, and as we shall see below, in general, a function computed by a finite-state weighted automaton over an arbitrary semiring may or may not satisfy $\Lambda_Q \neq \emptyset$ for some finite subset Q of the rows of its Hankel matrix. However, in view of Notation 2.6, Remark 2.7 and Definition 2.10, **if there exists** a finite set Q of rows of the Hankel matrix such that Λ_Q is non-empty, then for any finite set $T \subseteq \Sigma^*$ and any $\underline{\lambda} \in \Lambda_Q$ we have $\underline{\lambda} \in \Lambda_Q \subseteq \Lambda_{Q,T}$, so that (by construction) the hypothesis automaton $\mathcal{H}_{Q,T,\underline{\lambda}}$ constructed from the finite (Q,T) submatrix of the Hankel matrix of f, will be equal to the Hankel automaton $\mathcal{H}_{Q,\underline{\lambda}}$ and, so by Theorem 2.12, will compute f. The success of Angluin-type algorithms in the existing literature is therefore typically based on the ability to find (using finitely many membership and equivalence queries): a finite set of words Q such that Λ_Q is non-empty, together with a finite set of words T such that $\Lambda_{Q,T} = \Lambda_Q$ in the given setting.

Proof. Suppose that $Q \subseteq S^{\Sigma^*}$ is a finite subset such that $\Lambda_Q \neq \emptyset$ (or equivalently by Remark 2.7, Q is left-closed and left-generates $\langle\varepsilon\rangle$) and let $\underline{\lambda} \in \Lambda_Q$. We shall show that $\mathcal{H}_{Q,\underline{\lambda}}$ computes f. To begin with, for each $\underline{q}$ in Q, let $\mathcal{H}_{\underline{q}}$ be the automaton obtained from $\mathcal{H}_{Q,\underline{\lambda}}$ by imposing the condition that $\underline{q}$ is the unique initial state with weight 1_S, and keeping all other weights the same. We claim that for all w in Σ^*, $\underline{q}_w$ is equal to the weight computed by $\mathcal{H}_{\underline{q}}$ on input w. The proof is by induction on the length of w. If $w = \varepsilon$, then $\underline{q}_\varepsilon$ is (by construction) equal to the final weight of $\underline{q}$ in $\mathcal{H}_{Q,\underline{\lambda}}$ which proves the property. For a in Σ and w in Σ^*, $\underline{q}_{aw} = (\underline{q} \cdot a)_w = \bigoplus_{\underline{p} \in Q} \underline{\lambda}_{\underline{q},a,\underline{p}} \otimes \underline{p}_w$ since $\underline{\lambda} \in \Lambda_Q$. By induction hypothesis, $\underline{p}_w$ is the weight computed by $\mathcal{H}_{\underline{p}}$ on input w, and moreover $\underline{\lambda}_{\underline{q},a,\underline{p}}$ is (by construction) the weight of the transition from state $\underline{q}$ to state $\underline{p}$ labelled by a. So $\underline{q}_{aw}$ is indeed the weight computed by $\mathcal{H}_{\underline{q}}$ on input aw. 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 $\underline{\lambda} \in \Lambda_Q$ we have $f(w) = \langle\varepsilon\rangle_w = \bigoplus_{\underline{q} \in Q} \underline{\lambda}_{\underline{q}} \otimes \underline{q}_w$; by our previous observation (together with the fact that $\underline{\lambda}_Q$ is the initial state vector of $\mathcal{H}_{Q,\underline{\lambda}}$), this is then exactly the value computed by $\mathcal{H}_{Q,\lambda}$ on input w.

Now suppose that $f : \Sigma^* \to S$ 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. For each $q \in Q$, define $\underline{q} \in S^{\Sigma^*}$ where for all $u \in \Sigma^*$, $\underline{q}_u$ is equal to the value computed by ${}_q\mathcal{A}$ on input u. Let

$\underline{Q} = \{\underline{q} \mid q \in Q\}$. First $\underline{Q}$ is left-closed since, by definition, for all words $u \in \Sigma^*$, all states $q \in Q$ and all $a \in \Sigma$ we have

$$(\underline{q} \cdot a)_u = \underline{q}_{au} = \bigoplus_{p \in Q} (\nu_p \otimes \underline{p}_u)$$

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 as before 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, $\langle w \rangle = \bigoplus_{q \in Q} \mu_q(w) \otimes \underline{q}$, showing that $\underline{Q}$ is row-generating for f and hence in particular $\langle \varepsilon \rangle$ is left-generated by $\underline{Q}$. □

Remark 2.14. In this section we have taken an approach based on "rows", describing how to construct hypothesis automata utilising finitely many partial rows of the Hankel matrix. Clearly, a dual construction can be made working with the transpose of the Hankel matrix, that is, using partial columns and corresponding notions of right-linear combinations, right semimodules, right-generating sets, right-closed sets, column-generating sets, etc. In the text that follows we shall refer to the analogues of Hankel automata and hypothesis automata constructed by this dual method as co-Hankel automata and co-hypothesis automata. For the readers convenience we outline the details of these constructions and summarise the dual definition to those presented in the main text in Appendix A.

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

Corollary 2.15. *Let S be a semiring and $f : \Sigma^* \to S$. 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 left-subsemimodule of S^{Σ^*} generated by a finite left-closed set;*
(3) *the columns of the Hankel matrix of f lie in a right-subsemimodule of S^{Σ^*} generated by a finite right-closed set.*

Remark 2.16. Notice that if S is a field (as in [21]) then the second condition of the previous result implies 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, over an arbitrary semiring, 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 algorithm (using linear algebra) to find, by membership and equivalence queries, finite sets of **words** Q and T such that $\Lambda_{Q,T}$ is non-empty together with an element $\underline{\lambda} \in \Lambda_{Q,T}$ such that the hypothesis automaton $\mathcal{H}_{Q,T,\underline{\lambda}}$ computes the target function (see for example [9]). For an arbitrary (but fixed) semiring S, the following questions arise:

- For which functions $f : \Sigma^* \to S$ does there *exist* an hypothesis automaton $\mathcal{H}_{Q,T,\underline{\lambda}}$ computing f? We will call the class of functions which can be computed by *at least one* hypothesis automaton (which recall is built using a finite portion of the Hankel matrix) weakly guessable. Clearly, if f is not weakly guessable then any learning algorithm based around making hypotheses of the kind under discussion (constructed from certain left linear combinations of partial rows of the Hankel matrix) will fail to learn f.
- For weakly guessable functions over S, 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 with $\Lambda_Q \neq \emptyset$, it may be that there is no finite set T such that $\Lambda_{Q,T} = \Lambda_Q$, and therefore no obvious general method to discover a "correct" $\underline{\lambda} \in \Lambda_Q$ in finite time. We say that f is guessable if there exist finite sets Q and T such that $\Lambda_{Q,T} = \Lambda_Q \neq \emptyset$ and so *every* hypothesis automaton $\mathcal{H}_{Q,T,\lambda}$ with $\lambda \in \Lambda_{Q,T}$ computes f. Since Σ^* is countable, so too is the set of all pairs of *finite* subsets of Σ^*; thus if armed with an enumeration of all such pairs together with the ability to compute an element of every left solution set $\Lambda_{Q,T}$ (or else determine that the set is empty), it is clear that there is a naive algorithm (working through the list of all pairs of subsets, and performing an equivalence query where possible) that is guaranteed to succeed on the class of guessable functions (since $\Lambda_{Q,T} = \Lambda_Q \neq \emptyset$ holds for at least one pair).
- In order to find a correct hypothesis automaton, some of the existing approaches and algorithms typically alternate between 'incrementation' of the sets of words Q and T adding elements to Q in a prescribed manner when the left-solution set $\Lambda_{Q,T}$ is found to be empty, and adding elements to T in a prescribed manner when a counter example is given, eventually converging on a pair for which $\Lambda_{Q,T} = \Lambda_Q \neq \emptyset$. A potential problem here is that, in general, one does not know for a given set Q' whether it is worth persevering by continuing to add words to T or not (in other words, even though there exists a finite set Q such that $\Lambda_{Q,T} = \Lambda_Q \neq \emptyset$ for some finite set T, it may be the case that for the current choice Q' we have $\Lambda_{Q',T} \supsetneq \Lambda_{Q'}$ for all finite sets T). With this in mind, we say that a guessable function f is strongly guessable if for all finite sets of words Q there exists a finite set of words T, such that $\Lambda_{Q,T} = \Lambda_Q$. For strongly guessable functions, the general problem therefore reduces to the question of how to find a set Q with the property that $\Lambda_Q \neq \emptyset$.
- Finally, given a partial Hankel matrix one could ask the corresponding questions concerning existence of co-hypothesis automata constructed from right-linear combinations of partial columns of the Hankel matrix (see Appendix A for details), leading to the dual notion of weakly co-guessable, co-guessable, and strongly co-guessable functions.

In Section 3, we define the classes of weakly guessable, guessable and strongly guessable functions and give several equivalent characterisations for them in Theorem 3.6 and 3.13. In Section 4, we give a general hierarchy for these classes with some specific examples, and give conditions for this hierarchy to collapse. This allows us to have a clearer picture of what happens for several well-studied examples of semirings arising in application areas, where limitations of this approach may be of interest. This opens many questions for further research that are stated in the concluding Section 5.

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 $f : \Sigma^* \to S$ is said to be weakly guessable if there exists a finite set Q of rows of the Hankel matrix of f such that Λ_Q is non-empty.

Observe that by Theorem 2.12, this implies that for all $\underline{\lambda} \in \Lambda_Q$ the Hankel automaton $\mathcal{H}_{Q,\underline{\lambda}}$ computes f and hence for any non-empty set of words T, there exists an hypothesis automaton $\mathcal{H}_{Q,T,\underline{\lambda}}$ computing f since (by Remark 2.8) $\Lambda_Q \subseteq \Lambda_{Q,T}$. 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 (literal automaton) turns out to provide a syntactic characterisation of these functions in terms of automata, and links to residual automata. A literal automaton is a residual automaton, as defined for example for non-deterministic or nominal automata [22, 27] with extra conditions to deal with the weights. It is very close to the reduced residual automata in the stochastic settings defined in [17], with a slight difference in the way the weights are handled - the reason being that, while non-zero probabilities have an inverse for the multiplication, this is not systematically the case in general semirings. We use the terminology literal automata to avoid any confusion with [17], our choice of terminology reflecting the fact that each state of a literal automaton can be labelled by a word with the property that any run from the initial state labelled by that word terminates in the given state.

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 $\sigma : Q \to W$ 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.

Remark 3.3. In a literal automaton $\mathcal{A}$ with set of states Q, each state q can be labelled by a word $\sigma(q)$ which when taken as input terminates in state q (via a unique run, which has weight 1_S), and so in this case the set $\underline{Q} = \{\underline{q} \mid q \text{ is a state of } \mathcal{A}\}$ defined in Theorem 2.12 is easily seen to be precisely the set of rows of the Hankel matrix of $\mathcal{A}$ indexed by the words chosen to label the states, that is, $\underline{Q} = \{\langle\sigma(q)\rangle : q \in Q\}$.

Finally, we link the functions computed by literal automata to a weaker variant of a property given in [36], demonstrating that this corresponds exactly to the weakly guessable functions.

Definition 3.4. Given a function $f : \Sigma^* \to S$ and F its Hankel matrix, f satisfies the weak ascending chain condition if for all chains of left-semimodules $X_0 \subseteq X_1 \subseteq X_2 \subseteq \dots$ such that

- each X_i is generated by a finite set of rows of F;
- each row of F belongs to some X_i

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

Remark 3.5. The terminology "weak" in the previous definition reflects the fact that we only consider *certain chains* of left-semimodules. A left-semimodule is said to be Noetherian if *all* chains of left-subsemimodules satisfy the ascending chain condition. Thus, if every left-semimodule generated by a finite set of rows of F is Noetherian, then f satisfies the weak ascending chain condition above, but not conversely (in general, a submodule of a finitely generated semimodule need not be finitely generated, nor need a subsemimodule of a semimodule generated by rows of the Hankel matrix also be generated by rows of the Hankel matrix).

Theorem 3.6. *Given a function $f : \Sigma^* \to S$, the following assertions are equivalent:*
(1) *f is weakly guessable;*
(2) *there exists a literal automaton computing f;*
(3) *there exists a finite set of rows of the Hankel matrix of f that is row-generating for f;*
(4) *f satisfies the weak ascending chain condition.*

Remark 3.7. Before presenting the proof, it is instructive to compare Theorem 3.6 (characterising weakly guessable functions) with Theorem 2.12 (characterising functions computed by a finite weighted automata). Notice that point (3) of Theorem 3.6 states that there is a finite subset of the *rows of the Hankel matrix* that left-generates every row of the Hankel matrix, whilst Theorem 2.12, states that the rows of the Hankel matrix can be generated by finitely many elements of S^{Σ^*} (which need not themselves be rows of the Hankel matrix; see also Remark 2.13).

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. Let W be a finite set of words with $|W| = |Q|$ such that Q is the set of rows of the Hankel matrix indexed by W and let W' denote the set of all prefixes of words in W, with Q' the corresponding set of rows of the Hankel matrix, so that $Q \subseteq Q' = \{\langle q\rangle \mid q \in W'\}$. Since $\Lambda_Q \neq \emptyset$, we have seen (in Lemma 2.9) that $\Lambda_{Q'} \neq \emptyset$; we aim to show that there exists $\underline{\mu} \in \Lambda_{Q'}$ such that the corresponding Hankel automaton $\mathcal{H}_{Q',\underline{\mu}}$ (which by Theorem 2.12 computes f) is literal. To this end, we let $\underline{\lambda} \in \Lambda_Q$ so that (by Theorem 2.12) $\mathcal{H}_{Q,\underline{\lambda}}$ computes f and use the values in $\underline{\lambda}$ to define $\underline{\mu} \in S^{Q' \cup (Q' \times \Sigma \times Q')}$ as follows:

- $\underline{\mu}_{\langle q\rangle} = 1_S$ if and only if $\langle q\rangle = \langle\varepsilon\rangle$ and $\underline{\mu}_{\langle q\rangle} = 0_S$ for all other $\langle q\rangle \in Q'$,
- if $q \in W'$, $a \in \Sigma$ and $qa \in W'$, then $\underline{\mu}_{\langle q\rangle,a,\langle qa\rangle} = 1_S$, and $\underline{\mu}_{\langle q\rangle,a,\langle p\rangle} = 0_S$ for all other $\langle p\rangle \in Q'$;
- if $q \in W$, $a \in \Sigma$ and qa is not in W', then $\underline{\mu}_{\langle q\rangle,a,\langle p\rangle} = \underline{\lambda}_{\langle q\rangle,a,\langle p\rangle}$ for all $p \in W$ and $\underline{\mu}_{\langle q\rangle,a,\langle p\rangle} = 0_S$ for all $\langle p\rangle \in Q' \setminus Q$;
- if $q \in W' \setminus W$, a in Σ and qa is not in W', then for each $\langle p\rangle \in Q'$ we define $\underline{\mu}_{\langle q\rangle,a,\langle p\rangle}$ by induction on the length of q, as follows:
  If $q = \varepsilon$, then since $\underline{\lambda} \in \Lambda_Q$ for all words w we have $\langle\varepsilon\rangle_w = \bigoplus_{\langle r\rangle\in Q} \underline{\lambda}_{\langle r\rangle} \otimes \langle r\rangle_w$ and so
  $$\langle a\rangle_w = \langle\varepsilon\rangle_{aw} = \bigoplus_{\langle r\rangle\in Q} \underline{\lambda}_{\langle r\rangle} \otimes \langle r\rangle_{aw} = \bigoplus_{\langle r\rangle\in Q} \underline{\lambda}_{\langle r\rangle} \otimes \langle ra\rangle_w = \bigoplus_{\langle r\rangle,\langle p\rangle\in Q} \underline{\lambda}_{\langle r\rangle} \otimes \underline{\lambda}_{\langle r\rangle,a,\langle p\rangle} \otimes \langle p\rangle_w,$$
  using the fact that $\langle ra\rangle = \langle r\rangle \cdot a$. In this case we set: $\underline{\mu}_{\langle\varepsilon\rangle,a,\langle p\rangle} = \bigoplus_{\langle r\rangle\in Q} \underline{\lambda}_{\langle r\rangle} \otimes \underline{\lambda}_{\langle r\rangle,a,\langle p\rangle}$ for all $\langle p\rangle \in Q$ and $\mu_{\langle\varepsilon\rangle,a,\langle p\rangle} = 0_S$ for all $\langle p\rangle \in Q' \setminus Q$.
  If $q = ra$ and $\underline{\mu}_{\langle r\rangle,a,\langle p\rangle}$ has already been defined for all $\langle p\rangle \in Q'$ in such a way that expresses $\langle q\rangle = \langle ra\rangle = \langle r\rangle \cdot a$ as a left-linear combination of the rows Q (i.e. such that $\underline{\mu}_{\langle r\rangle,a,\langle p\rangle} = 0_S$ if $\langle p\rangle$ is not in Q), then for all words w and all $b \in \Sigma$ we have
  $$\begin{aligned}(\langle q\rangle \cdot b)_w = \langle qb\rangle_w &= \langle rab\rangle_w = \langle ra\rangle_{bw} = \bigoplus_{\langle p\rangle\in Q} \underline{\mu}_{\langle r\rangle,a,\langle p\rangle} \otimes \langle p\rangle_{bw}\\ &= \bigoplus_{\langle p\rangle\in Q} \underline{\mu}_{\langle r\rangle,a,\langle p\rangle} \otimes \langle pb\rangle_w = \bigoplus_{\langle p\rangle,\langle s\rangle\in Q} \underline{\mu}_{\langle r\rangle,a,\langle p\rangle} \otimes \underline{\lambda}_{\langle p\rangle,b,\langle s\rangle} \otimes \langle s\rangle_w.\end{aligned}$$
  In this case we set $\underline{\mu}_{\langle ra\rangle,b,\langle s\rangle} = \bigoplus_{\langle p\rangle\in Q} \underline{\mu}_{\langle r\rangle,a,\langle p\rangle} \otimes \underline{\lambda}_{\langle p\rangle,b,\langle s\rangle}$ for all $\langle s\rangle \in Q$ and $\mu_{\langle ra\rangle,b,\langle s\rangle} = 0_S$ for all $\langle s\rangle \in Q' \setminus Q$.

By construction we then have that $\underline{\mu}$ belongs to $\Lambda_{Q'}$ and hence by Lemma 2.12, $\mathcal{H}_{Q',\underline{\mu}}$ computes f. The automaton $\mathcal{H}_{Q',\underline{\mu}}$ 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 2.12, the set $\underline{Q} = \{\underline{q} \mid q \in Q\}$, where $\underline{q}_w$ is defined as the value computed by the automaton $\mathcal{A}_q$ on input w, is row-generating for f and, as observed in Remark 3.3, there is a finite set of words $\{\sigma(q) \mid q \in Q\}$ such that $\underline{Q} = \{\langle\sigma(q)\rangle \mid q \in Q\}$.

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

(4. implies 1.) For all positive integers i, let X_i be the left-semimodule generated by the set Q_i containing all rows of F 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, $X_m = X_n$. Necessarily the set Q_n is row-generating and hence (by Lemma 2.4) left-closed, and left-generates $\langle\varepsilon\rangle$, so there exists some $\lambda \in \Lambda_Q$, giving $\Lambda_{Q_n} \neq \emptyset$ which demonstrates that f is weakly guessable and concludes the proof. □

Remark 3.8. As noted in Remark 2.14, one can also work with the transpose of the Hankel matrix, leading to obvious left-right dual notions which we refer to as weakly co-guessable, column generating sets, and weak co-ascending chain condition: see Appendix A for full details. We say that an automaton is co-literal if its mirror is literal. With these definitions in place, one can prove the following entirely analogous result.

Theorem 3.9. *Given a function $f : \Sigma^* \to S$, 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 set of columns of the Hankel matrix of f that is column-generating for f;*
(4) *f satisfies the weak co-ascending chain condition.*

Remark 3.10. The semirings where *all* functions computed by finite state weighted automata satisfy the weak ascending and co-ascending chain conditions are therefore the semirings for which all functions computed by finite state weighted automata are both weakly guessable and weakly co-guessable (by Theorem 3.6 and its dual Theorem 3.9). 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. [36]). More generally, if R is any left-Noetherian ring, then all finitely generated left R-modules are Noetherian, and hence by Remark 3.5, *all* functions computed by finite state weighted automata over R are weakly guessable. This covers the case of finite rings, fields, principal ideal domains, and division rings.

3.2. **Guessable and strongly guessable functions.**

The definition of weakly guessable functions ensures the existence of some finite sets $Q, T \subseteq \Sigma^*$ and coefficients $\underline{\lambda} \in \Lambda_{Q,T}$ (writing here and from now on simply Q in place of the set of rows of the Hankel matrix indexed by the words in Q, as discussed in Notation 2.6) 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. In the following definition, the reader should recall from Remark 2.7 that the sets $\Lambda_{Q,T}$ depend upon the function f (and hence also the semiring S).

Definition 3.11. A function $f : \Sigma^* \to S$ 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 a guessable function f is strongly guessable if for all finite sets $Q \subseteq \Sigma^*$, there exists a finite set $T \subseteq \Sigma^*$ such that $\Lambda_{Q,T} = \Lambda_Q$.

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 2.12, 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 [36] to characterise guessable functions.

Definition 3.12. Let Q be a finite subset of Σ^*. A total Q-chain is an infinite descending chain of left-solution sets $\Lambda_{Q,T_0} \supsetneq \Lambda_{Q,T_1} \supsetneq \Lambda_{Q,T_2} \supsetneq \cdots$ with $T_i \subseteq T_{i+1}$ for all $i \geq 0$ and such that each word in Σ^* is contained in some T_i. We say that a function $f : \Sigma^* \to S$ is row-bound if there exists a finite set of words Q such that $\Lambda_Q \neq \emptyset$ and there is no total $Q-$chain. We say that $f : \Sigma^* \to S$ is strongly row-bound if for *every* finite set of words Q, there is no total Q-chain.

The above definition is related to the progress measure defined in [36, Definition 11]; we will discuss this in the next subsection.

Proposition 3.13. *Let $f : \Sigma^* \to S$. Then f is guessable if and only if f is row-bound. Moreover, f is strongly guessable if and only if f is strongly row-bound and weakly guessable.*

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 sequence of subsets of Σ^* with $T_i \subseteq T_{i+1}$ for all i and such that $\bigcup_i T_i = \Sigma^*$. 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 total Q-chains. 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 (and weakly guessable).

Now, let Q be a finite set of words such that there is no total Q-chain. Then taking T_i to be the finite set of words of length at most i for each positive integer i, we have that $T_i \subseteq T_{i+1}$ and $\bigcup_i T_i = \Sigma^*$ 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 $\Lambda_{Q,T_n} = \Lambda_Q$. Thus if f is row-bound, there is such a Q such that $\Lambda_Q \neq \emptyset$ and we conclude that f is guessable. Additionally, if f is strongly row-bound and weakly guessable then the property holds for all Q and at least one of them is such that $\Lambda_Q \neq \emptyset$, which makes f strongly guessable.

□

Remark 3.14. As in Remarks 2.14 and 3.8, one can also work with the transpose of the Hankel matrix, leading to obvious left-right dual notions of co-guessable, strongly co-guessable:

see Appendix A for full details. With these definitions, one can prove the following entirely analogous result.

Proposition 3.15. *Let $f : \Sigma^* \to S$. Then f is (strongly) co-guessable if and only if f is (strongly) column-bound and weakly co-guessable.*

3.3. **Algorithmic strategies.**

Game strategies. We now explain what the notions of (weakly/strongly) guessable mean in terms of strategies for the learner to discover an automaton correctly computing the target function via a sequence of Angluin-style queries.

The hierarchy we propose is (in general) orthogonal to any complexity classes - in particular there is no guarantee that a strongly guessable function can be learnt in polynomial time. See Remark 3.21 at the end of the section.

Remark 3.16. Provided the semiring S is countable, a trivial algorithm would be to enumerate all the automata and make an equivalence query on each of them in turn. One would of course like to improve upon this exponential time strategy. Even though there is no guarantee in terms of time complexity of the Angluin-style process we describe next, note that there are two main differences: (1) the learning process still applies when the semiring is uncountable, (2) it will only explore, in a prescribed manner, a subset of automata i.e. the class of hypothesis automata.

Consider a two player game between the teacher and the learner, in which the learner's objective is to find an automaton computing a target function. The rules are as follows. At every round, the learner is allowed to:
- choose a pair of finite subsets $Q, T \subseteq \Sigma^*$;
- ask the teacher to provide an hypothesis automaton constructed from Q and T.

The teacher must:
- declare if there are no possible hypothesis automaton for a given pair Q, T (that is, if $\Lambda_{Q,T} = \emptyset$) or provide a hypothesis automaton $\mathcal{H}_{Q,T,\underline{\lambda}}$ for this pair (by selecting $\underline{\lambda} \in \Lambda_{Q,T}$);
- declare whether each provided automaton computes the target function and if not, provide a counter-example.

The game stops when the teacher declares an automaton that computes the target function. We view the teacher as an oracle, able to determine an element of each of the sets $\Lambda_{Q,T}$, Λ_Q and $\Lambda_{Q,T} \setminus \Lambda_Q$ (for all finite subsets of words Q, T), or else determine that the set is empty. In this way, all computation is deferred to the teacher. The teacher must always answer truthfully, but may play as an ally (meaning whenever (Q,T) is queried with $\Lambda_Q \neq \emptyset$, the teacher will choose to reply with a correct automaton $\mathcal{H}_{Q,T,\underline{\lambda}}$ where $\underline{\lambda} \in \Lambda_Q$) or as an adversary (whenever (Q,T) is queried with $\Lambda_{Q,T} \neq \Lambda_Q$, the teacher will choose to reply with an incorrect automaton $\mathcal{H}_{Q,T,\underline{\lambda}}$ where $\underline{\lambda} \in \Lambda_{Q,T} \setminus \Lambda_Q$).

One can record the history of a game up to any given point as the finite (possibly empty) sequence of pairs (Q,T) queried so far, together with the replies from the teacher (i.e. the hypothesis automata and counter-examples or the fact that $\Lambda_{Q,T} = \emptyset$). A strategy for the learner is a function mapping each history h where the last pair in the sequence does not signify the end of the game to a new pair $(Q_h, T_h) \in \Sigma^* \times \Sigma^*$. Additionally, the strategy is said to be incremental if for all histories h, $Q_h \supseteq Q$ and $T_h \supseteq T$ holds for all pairs (Q,T) already queried in h. A strategy is winning for the learner, if, for all possible plays following that strategy, the teacher eventually declares that a correct automaton is found.

Proposition 3.17. *A function is weakly guessable if and only if the learner has a winning strategy in the game where the teacher plays as an ally.*

Proof. For a weakly guessable function there exists a non-empty finite subset $Q \subseteq \Sigma^*$ such that $\Lambda_Q \neq \emptyset$. Since Σ^* is countable, so is the set of non-empty finite subsets of Σ^*. Let $(Q_i)_{i\in\mathbb{N}}$ be an enumeration of the finite non-empty subsets of Σ^* and consider the following strategy for the learner: if h is empty, query $(Q_0, \{\varepsilon\})$; if the last query of h is $(Q_i, \{\varepsilon\})$ and does not halt the game then query $(Q_{i+1}, \{\varepsilon\})$ (no matter the automata and counter-examples given by the teacher). The learner will eventually query $(Q, \{\varepsilon\})$ and since $\Lambda_Q \subseteq \Lambda_{Q,\{\varepsilon\}}$ the teacher (playing as an ally) will choose to declare a correct hypothesis automaton. Conversely, if the learner has a winning strategy, then clearly there exist finite sets $Q, T \subseteq \Sigma^*$ and $\underline{\lambda} \in \Lambda_{Q,T}$ such that $\mathcal{H}_{Q,T,\underline{\lambda}}$ computes the target function and so by definition the function is weakly guessable. □

Proposition 3.18. *A function is guessable if and only if the learner has a winning strategy in the game where the teacher plays as an adversary.*

Proof. For a guessable function there exists finite subsets $Q, T \subseteq \Sigma^*$ such that $\Lambda_{Q,T} = \Lambda_Q \neq \emptyset$. Since Σ^* is countable, so is the set of pairs of non-empty finite subsets of Σ^*; let $(Q_i, T_i)_{i\in\mathbb{N}}$ be an enumeration of all such pairs and consider the following strategy for the learner: if the history is empty, query (Q_0, T_0); if the last query of h is (Q_i, T_i) and does not halt the game then query (Q_{i+1}, T_{i+1}) (no matter the automata and counter-examples given by the teacher). The learner will eventually query (Q, T) and since $\Lambda_Q = \Lambda_{Q,T}$, the teacher will be forced to declare a correct hypothesis automaton.

Conversely, if the learner has a winning strategy in the game where the teacher plays as an adversary, then clearly there exist finite sets $Q, T \subseteq \Sigma^*$ and $\underline{\lambda} \in \Lambda_{Q,T}$ such that $\mathcal{H}_{Q,T,\underline{\lambda}}$ computes the target function and $\Lambda_{Q,T} \setminus \Lambda_Q = \emptyset$ (otherwise the teacher can always choose to provide an incorrect automaton). So by definition the function is guessable. □

Proposition 3.19. *If a function is strongly guessable, then the learner has a winning strategy that is incremental in the game where the teacher plays as an adversary.*

Proof. Let $Q_i = T_i$ be the set of all words of length at most i, and consider the following strategy for the learner: if the history is empty, query (Q_0, T_0); if (Q_i, T_j) is the last pair queried in h and the teacher declared that Λ_{Q_i,T_j} is non-empty, but did not give a correct automaton, then query (Q_i, T_{j+1}); if (Q_i, T_j) is the last pair queried in h and Λ_{Q_i,T_j} is empty then query (Q_{i+1}, T_j); otherwise if (Q_i, T_j) is the last pair queried in h and the teacher declared a correct automaton, the game halts.

For any given target function this process is completely deterministic, but possibly infinite: since the teacher plays as an adversary (and so never supplies a correct automaton if there is an incorrect choice), the sequence of pairs queried by the learner for this function will be the same regardless of different choices of automata or counterexample supplied by the teacher in response to each query (only the question of their existence is relevant to proceedings). Our aim is to show that whenever the target function is strongly guessable, this process terminates.

Since f is weakly guessable and for all non-negative integers i, $Q_i \subseteq Q_{i+1}$ then there exists a non-negative integer I such that for all $i < I$, Λ_{Q_i} is empty and Λ_{Q_I} is non-empty. Hence by strong guessability, for all $i < I$, there exists T such that $\Lambda_{Q_i,T}$ is empty and there exists T' such that $\Lambda_{Q_I,T'} = \Lambda_{Q_I}$ is non-empty. By Remark 2.8 and Lemma 2.9, it follows that for all $i < I$, there exists ℓ such that for all $j \geq \ell$, Λ_{Q_i,T_j} is empty and J such that for all $j \geq J$, $\Lambda_{Q_I,T_j} = \Lambda_{Q_I}$ is non-empty (and none of the $\Lambda_{Q_I,T}$ is empty for any T). Hence, the process terminates, and the strategy above is winning for the learner. □

Remark 3.20. We leave open the question whether the converse of Proposition 3.19 holds. This is related to the question of the existence of functions that are guessable but not strongly guessable. This question remains open (see Section 4.2 and Conclusion).

Remark 3.21. The strategies mentioned above do not guarantee any polynomial time complexity. For strongly guessable functions, a polynomial time strategy would require the learner to make more well-informed choices of Q and T (by making use of the counter-examples) so that the size of Q and T remain polynomial with respect to the size of the automaton to learn and counter-examples given. For guessable and weakly guessable functions, in addition to the constraints on the size of Q and T, the learner needs to be able to find a correct $\underline{\lambda}$ in polynomial time; this may be possible over certain semirings.

Relationship to the algorithmic approach of [36]**.** In [36], a general learning algorithm is given (subject to the existence of a 'closedness strategy') and three conditions (solvability, ascending chain condition, and progress measure) are requested to ensure termination and correctness of the algorithm *for all functions computed by weighted automata over a fixed semiring.* We give an overview of this algorithm using our terminology in Figure 2, and discuss below some weaker conditions that are easily seen to be sufficient for the proof of termination and correctness *on a given target function*, and how these relate to our definitions of guessability.

```
Learning Algorithm
Initialisation: Set $Q = T = \{\varepsilon\}$.
Closure: While $Q$ is not left-closed on $T$, do:
    Find $q \in Q$, $a \in \Sigma$ such that $\langle qa \rangle_T$ is not a left-linear combination of the rows in $Q$.
    Set $Q := Q \cup \{qa\}$.
Test: Find $\lambda$ in $\Lambda_{Q,T}$ and perform an equivalence query for $\mathcal{H}_{Q,T,\lambda}$.
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 2. Outline of learning algorithm from [36], subject to the existence of a closedness strategy.

First, note that this algorithm requires access to a procedure (referred to as a 'closedness strategy' in [36]) that can (a) determine whether Q is left-closed on T (or in other words, whether $\Lambda_{Q,T} \neq \emptyset$); (b) if $\Lambda_{Q,T} = \emptyset$, can find $q \in Q$ and $a \in \Sigma$ such that the corresponding equation has no solution; and (c) if $\Lambda_{Q,T} \neq \emptyset$, can find an element $\underline{\lambda}$ of $\Lambda_{Q,T}$.

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 there is a closedness strategy for all functions over this semiring. We note that in [36] 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 [36] the authors provide a sufficient condition for a principal ideal domain to be solvable (specifically: all ring operations are computable, and each element can be effectively factorised into irreducibles).

Remark 3.22. Turning to other examples, we note over $\mathbb{Z}_{\max}$ there is a straightforward (and well-known) method to determine whether a system of the form $A \otimes \underline{x} = \underline{b}$ has a solution. Indeed, if $\underline{x}$ is a solution, we must in particular have that $A_{i,j} + \underline{x}_j \leq \underline{b}_i$ for all i, j. If $A_{i,j} = -\infty$, then this inequality holds automatically, otherwise, we require that $\underline{x}_j \leq \underline{b}_i - A_{i,j}$. 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 $\underline{x}_j = \min\{\underline{b}_i - A_{i,j} : A_{i,j} \neq -\infty\}$, and set all remaining $\underline{x}_j$ 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 system of inequalities $A \otimes \underline{x} \leq \underline{b}$. Furthermore, it can also be seen that either this solution is also a solution to the equation $A \otimes \underline{x} = \underline{b}$, or else no solution to the equation $A \otimes \underline{x} = \underline{b}$ exists. (Indeed, if the solution to the inequality that we have constructed does *not* satisfy the equation, then we must have that for some i the following strict inequality holds: $\max(A_{i,j} + \underline{x}_j) < \underline{b}_i$, meaning that either all $A_{i,j}$ are equal to $-\infty$ and so there is no hope to satisfy the corresponding equation, or else in order to attain the equality we require that one of the $\underline{x}_j$ for j with $A_{i,j} \neq -\infty$ must take a larger value than set above, which by construction is not possible.) Thus $\mathbb{Z}_{\max}$ (and likewise $\mathbb{N}_{\max}$) is solvable.

Remark 3.23. In [36], an example showing that the algorithm in Figure 2 does not always terminate is given. This example corresponds to the automaton given in Figure 1d, that (as we shall see in the next section) is not weakly guessable. Since the algorithm can only guess hypothesis automata, it is clear that the algorithm will not terminate on any function that is not weakly guessable. In fact, termination cannot be ensured if the target function is not guessable. Indeed, if the function is not guessable, then (by definition) for each finite set of words Q with $\Lambda_Q \neq \emptyset$ one has that $\Lambda_Q \neq \Lambda_{Q,T}$ for all finite sets of words T. This in turn means that there is no guarantee that the algorithm will terminate, since it could be that in *every test step* the element $\underline{\lambda}$ provided by the closedness strategy generates an incorrect hypothesis automaton.

For the remainder of this section, let us assume that f is a function for which a closure strategy exists. The following condition ensures that the closure step terminates in finite time for each fixed T and that it cannot be taken infinitely many times for different T's.

Definition 3.24. A function $f : \Sigma^* \to S$ with Hankel matrix F satisfies the ascending chain condition on rows if for all sequences of left-semimodules $(X_i)_{i\in\mathbb{N}}$ generated by finite sets of rows of F, such that $X_0 \subseteq X_1 \subseteq X_2 \subseteq \cdots$ there is n such that for all $m \geq n$, $X_m = X_n$.

Finally, we note that the following condition (which mimics the definition of progress measure given in [36], rewritten in the language of the present paper and extracting the key property) ensures that one cannot go through the Test step infinitely many times without changing the set Q in the closure step.

Definition 3.25. Let f be a function computed by a weighted automaton over S and let

$$\mathcal{L} := \{(Q,T) : Q, T \text{ are finite sets of words and } Q \text{ is left-closed on } T\}.$$

We say that f has a weak progress measure if there is a totally ordered set $(\mathbb{K}, \prec)$ with no infinite descending chain and a function $\mu : \mathcal{L} \to \mathbb{K}$ with the property that if $T \subseteq T'$ and $\Lambda_{Q,T} \supsetneq \Lambda_{Q,T'}$ we have $\mu(Q,T) \succ \mu(Q,T')$.

We summarise the observations above in the following proposition, which provides a slight generalisation of statement of the main result of [36]; the proof derives in a similar manner to results presented in [36].

Proposition 3.26. [36, Theorem 14] *Suppose that a closure strategy exists for f. If f satisfies the ascending chain condition on rows and has a weak progress measure, then the learning algorithm from Figure 2 terminates and returns an automaton computing f.*

Remark 3.27. Finally, we note the relationship between our notions of guessability and the termination conditions stated above.

- The ascending chain condition on rows is weaker than the ascending chain condition considered in [36] (which considers *all* chains of submodules, including those which may not be generated by finitely many *rows* of the matrix) but stronger than the weak ascending chain condition from Definition 3.4 (which considers only *certain* chains of submodules generated by finitely many rows of the matrix). Thus a function satisfying the termination conditions above is certainly (by Theorem 3.6) weakly guessable. Weak guessability alone is not sufficient to ensure termination of algorithm from Figure 2 (as we have seen, it is precisely the condition required in order for a correct hypothesis automaton to exist).
- The existence of a progress measure (in the sense of [36]) for a Hankel matrix implies that the corresponding function is strongly row-bound. However, the converse need not be true. A progress measure implies a uniform bound, for all Q, on the number of elements in a total Q-chain, 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.
- Each progress measure gives rise to a weak progress measure, and moreover, if a function has a weak progress measure then it is strongly row-bound or equivalently (by Proposition 3.13) strongly guessable. Thus the proposition above guarantees that the learning algorithm in Figure 2 terminates for a class of strongly guessable functions. Strong guessability alone is not sufficient to ensure termination of algorithm from Figure 2. However, it follows from the proof of Proposition 3.19 that there is an algorithm with a different (incremental) update strategy for the sets Q and T that will terminate with success on any strongly guessable function for which a closure strategy exists.

4. Hierarchy of guessable functions

In the previous section, we have defined the classes of strongly guessable, guessable, and weakly guessable functions, and in Appendix A the dual notions of strongly co-guessable, co-guessable, and weakly co-guessable functions.

4.1. The general hierarchy.

First, we show that for any semiring: if a function is both weakly and co-weakly guessable then it is also both strongly and co-strongly guessable, so that the general hierarchy is as depicted in Figure 3.

Lemma 4.1. *Let S be a semiring and let $f : \Sigma^* \to S$ 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, we have $\Lambda_Q = \Lambda_{Q,T}$.

Thus let Q be arbitrary. Since $\Lambda_Q \subseteq \Lambda_{Q,T}$ is always true, if $\Lambda_{Q,T} = \emptyset$ then the equality holds. Otherwise, let $\underline{\lambda} \in \Lambda_{Q,T}$. Then for all $q, p \in Q$, all $a \in \Sigma$ and all $t \in T$ we have: $F_{qa,t} = \bigoplus_{p \in Q} \underline{\lambda}_{q,a,p} \otimes F_{p,t}$. By our assumption on T (recalling that we consider *right* linear combinations in the definition of weakly co-guessable) we know that for all words w and

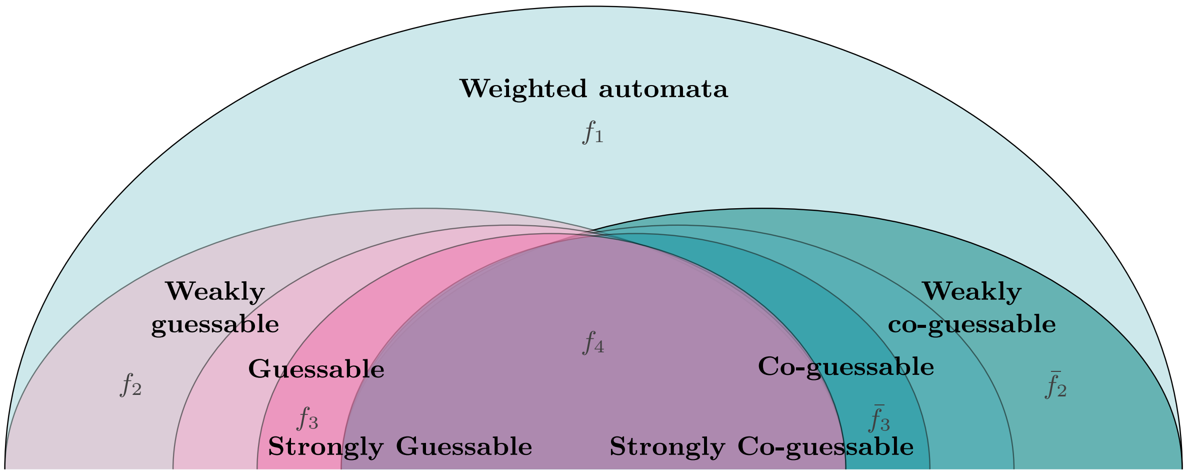


FIGURE 3. A general hierarchy for levels of guessability

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

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

In other words, $\langle qa \rangle = \bigoplus_{p \in Q} \underline{\lambda}_{q,a,p} \otimes \langle p \rangle$. A similar argument shows that $\langle \varepsilon \rangle = \bigoplus_{q \in Q} \underline{\lambda}_q \otimes \langle q \rangle$, hence giving $\underline{\lambda} \in \Lambda_Q$, as required. This demonstrates that f is strongly guessable. A dual argument (interchanging the rows/columns, and left/right combinations) demonstrates that f is strongly co-guessable. □

Remark 4.2. The proof of Lemma 4.1 actually shows that any function that is both weakly guessable and weakly co-guessable has the following stronger property. Say that a guessable function $f : \Sigma^* \to S$ is uniformly strongly guessable if there exists a finite set $T \subseteq \Sigma^*$ such that for all finite sets $Q \subseteq \Sigma^*$ one has $\Lambda_{Q,T} = \Lambda_Q$. Note the inversion of quantifiers as compared to the definition of strongly guessable; here the set T has to be the same for all the sets Q. Dually one can define uniformly strongly co-guessable. The previous proof shows that the intersection of the class of weakly guessable and weakly co-guessable functions coincides with the intersection of the class of uniformly strongly guessable and uniformly strongly co-guessable functions. We leave open the question of whether this intersection is precisely characterised by uniform strong guessability. However, we note that it can be shown that the class of uniformly strongly guessable functions is strictly contained in the class of strongly guessable functions (as an example, the function f_3 discussed below is not uniformly strongly guessable).

4.2. **Examples for the strict hierarchy.** We now give examples to demonstrate that certain inclusions in the hierarchy can be strict. The examples are drawn from various well-studied semirings discussed in Section 2, but one could obtain a unified semiring separating these classes by considering their product with component-wise operations. The full proofs of the membership of the following functions to certain parts of the hierarchy are given in Appendix B. We give here a quick overview making frequent use of the following lemma:

Lemma 4.3. *If S is a commutative semiring then a function is weakly guessable (respectively, guessable, strongly guessable) if and only if its mirror is weakly co-guessable (respectively, co-guessable, strongly co-guessable).*

Proof. Let $f : \Sigma^* \to S$, and consider its mirror $\overline{f} : \Sigma^* \to S$. Let us write F to denote the Hankel matrix of f and $\overline{F}$ to denote the Hankel matrix of $\overline{f}$, and for a set of words $W \subseteq \Sigma^*$ let us also write $\overline{W} = \{\overline{w} : w \in W\}$. By definition, for all words $u, v \in \Sigma^*$ we have

$$\overline{F}_{u,v} = \overline{f}(uv) = f(\overline{uv}) = f(\overline{v}\,\overline{u}) = F_{\overline{v},\overline{u}}.$$

From this one sees that for any pair of finite subsets $Q, T \subseteq \Sigma^*$, the equations given in Notation 2.6 for the function f (encoding certain left-linear combinations of the rows of F indexed by Q restricted to columns in T) are, due to commutativity of multiplication in S, equivalent to the equations encoding (right) linear combinations of the columns of $\overline{F}$ indexed by $\overline{Q}$ restricted to rows in $\overline{T}$, as obtained from A.1. Since weakly guessable, guessable and strongly guessable and their duals are described in terms of properties of these solution sets (considered for all finite sets Q, T), the result follows. □

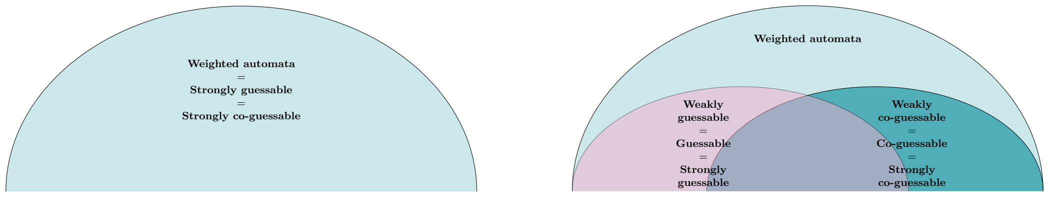
For some semirings S, there are functions $f : \Sigma^* \to S$ that are neither weakly guessable, nor weakly co-guessable. We show that this is the case for f_1 computed by the automaton given in Figure 1b, whether viewed over $\mathbb{N}_{\max}$, $\mathbb{Z}_{\max}$ or $\mathbb{R}_{\max}$, f_1' computed 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 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 (by Lemma 4.3). Finally, we 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: for commutative semirings, any function computed by an automaton that is both literal and co-literal lies in this intersection. A trivial example of such function would be the constant function f_4 mapping all words to 1_S.

4.3. **Collapse in some familiar semirings.**

In many well-studied semirings, we note that the hierarchy collapses further, due to the fact that certain ascending and descending chain conditions are satisfied. It is clear from Theorem 3.6, Theorem 3.9, Proposition 3.13, Proposition 3.15 and Lemma 4.3 that the following two properties characterise (A) when all functions computed by a finite state weighted automaton over S are strongly guessable and strongly co-guessable (i.e. everything collapses, as in Figure 4a) and (B) when all weakly guessable functions are strongly guessable and all weakly co-guessable functions are strongly co-guessable (as in Figure 4b):

- Property A: all functions computed by finite state weighted automata over S satisfy the weak ascending and weak co-ascending chain condition;

- Property B: all functions computed by finite state weighted automata over S which satisfy the weak ascending chain condition are strongly row-bound and all functions computed by finite state weighted automata over S which satisfy the weak co-ascending chain condition are strongly column-bound.



(A) Property A (B) Property B

FIGURE 4. Hierarchies for semirings satisfying properties A and B.

Note that Property A holds in the semirings $\mathbb{Z}$, $\mathbb{R}$ and $\mathbb{B}$ (as special cases of principal ideal domains, fields and finite semirings respectively) for example. Further, the semirings $\mathbb{N}_{\max}$, $\mathbb{N}$ and $\mathcal{P}_{\text{fin}}(\Sigma^*)$ satisfy Property B:

Lemma 4.4. *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 dual. 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 $Q \subseteq \Sigma^*$ be a finite set. We aim to show that there is no total Q-chain. First, note that, for some fixed T, if no row indexed by Q and restricted to T is fully 0_S, then in each of three semirings considered, $\Lambda_{Q,T}$ is finite (possibly empty) from which it immediately follows that there is no total Q-chain involving $\Lambda_{Q,T}$. Let $Z_T = \{q \in Q \mid \langle q \rangle_T \text{ is a } 0_S \text{ row}\}$, $Q' = Q \setminus Z_T$ and $N_T \subseteq S^{Q' \cup (Q \times \Sigma \times Q')}$ be the set obtained from $\Lambda_{Q,T} \subseteq S^{Q \cup (Q \times \Sigma \times Q)}$ by restricting the entries $\underline{\lambda}_p$ and $\underline{\lambda}_{q,a,p}$ to only permit $p \in Q'$. Clearly Z_T is finite and its size is bounded above by $|Q|$. Notice that N_T is also a finite set, since each $\underline{\lambda}_p, \underline{\lambda}_{q,a,p}$ where $p \in Q'$ can only take a finite number of different values. Moreover, it is clear from the definitions that $T \subseteq T'$ forces $Z_T \supseteq Z_{T'}$ (i.e. any row of the Hankel matrix that is zero on all columns indexed by T' is also zero on all columns indexed by T), $\Lambda_{Q,T} \supseteq \Lambda_{Q,T'}$ (by Remark 2.8) and $N_T \supseteq N_{T'}$ (since these sets are obtained from the previous by restricting attention to coefficients of rows indexed by Q that are not identically zero on T and T' respectively). Moreover, notice that if $Z_T = Z_{T'}$ (the same subset of elements of Q are zero when restricted to T or T') and $N_T = N_{T'}$ then we must have $\Lambda_{Q,T} = \Lambda_{Q,T'}$. Now, suppose that $T \subsetneq T'$ with $\Lambda_{Q,T} \supsetneq \Lambda_{Q,T'}$. It is clear from the above 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. □

We remark that in semirings where the concepts of weakly guessable and strongly guessable (resp. weakly co-guessable are strongly co-guessable) coincide (as is the case for fields and finite semirings, for example), this is inherited by subsemirings:

Lemma 4.5. *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. In particular, if S is a subsemiring of a field (e.g. an integral domain), then every weakly guessable function over S is strongly guessable.*

Proof. Suppose that $f : \Sigma^* \to S'$ 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 over S. Suppose for contradiction that f is not strongly guessable over S'. Then there exists a finite set Q such that for all finite sets T there exists $\underline{\lambda}_T \in \Lambda_{Q,T} \setminus \Lambda_Q$, where these solution sets are considered over S'. However, noting that the coefficients $\underline{\lambda}_T$ lie in S, this directly contradicts that f is strongly guessable over S. The final statement follows from Remark 3.10 together with the fact that any integral domain embeds into a field (its field of fractions). □

Remark 4.6. Suppose S' is a subsemiring of S and $f : \Sigma^* \to S'$. Then we may also regard f as a function with co-domain S, and as observed in the previous proof, if $f : \Sigma^* \to S'$ is weakly guessable then f is also weakly guessable when regarded as a function with co-domain S. However, if f is guessable (respectively, strongly guessable) when considered as function with co-domain S', it need not be guessable (respectively, strongly guessable) when considered as a function with co-domain S. For example, the function computed by the automaton depicted in Figure 1b is strongly guessable when the weights (and hence the co-domain of the function) are considered to lie in the semiring $S' = \mathbb{N}_{\max}$, but it is *not* guessable when the weights are considered to lie in $S = \mathbb{Z}_{\max}$ (see the functions denoted f_2 and f_3 in Appendix B for details).

We now apply the results and discussion of this section to determine whether Property A or B holds in the semirings used in our examples. The table in Figure 5 demonstrates that there are interesting and important semirings where Property B holds (i.e. all weakly (co-)guessable functions are strongly (co-)guessable) whilst Property A does not hold (i.e. there exist functions that are not weakly guessable). Since over such semirings the property of being *strongly* guessable is characterised in terms of automata (by Theorem 3.6), it is possible for the 'teacher' to select a strongly guessable target function to begin with, and in light of the discussion in Remark 3.27, there is then an algorithm (similar to that of [36], but using a different update strategy) that terminates with success for all such target functions with a closure strategy.

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

FIGURE 5. Summary of collapse of classes in familiar semirings.

5. 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 states (not from rows of the Hankel matrix) by strengthening the queries. 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 example, the algorithm of [36] can fail for functions that are not guessable (i.e. if for some finite set Q at every Test step the counter example provided by the teacher allows for the possibility that the closure strategy produces a solution that is not in Λ_Q). For guessable functions, a naive way to construct an algorithm that terminates with success is to use the update strategy suggested by the proof of Proposition 3.18. 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 is a class for which we neither have an example of non-emptiness or an argument for collapse: the functions that are guessable but not strongly guessable (and its dual). Finally, it would be interesting to study the notion of uniform strong guessability introduced in Remark 4.2 and in particular, try to characterise the class of functions lying in the intersection of the six classes we have introduced. It would also be of interest to understand how termination of the algorithm from [36] relates to strong guessability.

Acknowledgment

This work has been supported by the EPSRC grant EP/T018313/1 and a Turing-Manchester Exchange Fellowship, The Alan Turing Institute. We also thank Nathanaël Fijalkow for useful discussions on the topic, and the anonymous reviewers for their deep and useful comments.

References

[1] Cyril Allauzen, Mehryar Mohri, and Ameet 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] Simone Barlocco, Clemens Kupke, and Jurriaan Rot. Coalgebra learning via duality. In Mikołaj Bojańczyk and Alex Simpson, editors, *Foundations of Software Science and Computation Structures - 22nd International Conference, FOSSACS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings*, volume 11425 of *Lecture Notes in Computer Science*, pages 62–79. Springer, 2019.

[9] Francesco Bergadano and Stefano Varricchio. Learning behaviors of automata from multiplicity and equivalence queries. *SIAM J. Comput.*, 25(6):1268–1280, 1996.

[10] 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.

[11] 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.

[12] 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.

[13] Alex Buna-Marginean, Vincent Cheval, Mahsa Shirmohammadi, and James Worrell. On learning polynomial recursive programs. In Michael Hicks, editor, *Proceedings of the ACM on Programming Languages*, volume 8, pages 1001 – 1027. Association for Computing Machinery, New York, 2024.

[14] Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Quantitative languages. *ACM Trans. Comput. Logic*, 11(4), jul 2010.

[15] 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.

[16] Karel Culik and Jarkko Kari. *Digital Images and Formal Languages*, pages 599–616. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997.

[17] François Denis and Yann Esposito. On rational stochastic languages. *Fundam. Informaticae*, 86(1-2):41–77, 2008.

[18] 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.

[19] Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of Weighted Automata*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[20] 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.

[21] Michel Fliess. Matrices de hankel. *Journal de Mathématiques Pures et Appliquées*, 1974.

[22] Pierre Ganty, Elena Gutiérrez, and Pedro Valero. A quasiorder-based perspective on residual automata. In Javier Esparza and Daniel Král', editors, *45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020, August 24-28, 2020, Prague, Czech Republic*, volume 170 of *LIPIcs*, pages 40:1–40:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[23] 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.

[24] Karel Culík II and Jarkko Kari. Image compression using weighted finite automata. *Comput. Graph.*, 17(3):305–313, 1993.

[25] 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.

[26] 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.

[27] Joshua Moerman and Matteo Sammartino. Residuality and learning for nondeterministic nominal automata. *Log. Methods Comput. Sci.*, 18(1), 2022.

[28] 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.

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

[30] Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted automata in text and speech processing. *CoRR*, abs/cs/0503077, 2005.

[31] Takamasa Okudono, Masaki Waga, Taro Sekiyama, and Ichiro Hasuo. Weighted automata extraction from recurrent neural networks via regression on state spaces. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 5306–5314. AAAI Press, 2020.

[32] Guillaume Rabusseau, Tianyu Li, and Doina Precup. Connecting weighted automata and recurrent neural networks through spectral learning. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, volume 89 of *Proceedings of Machine Learning Research*, pages 1630–1639. PMLR, 2019.

[33] Marcel-Paul Schützenberger. On the definition of a family of automata. *Information and Control*, 4(2):245–270, 1961.

[34] Henning Urbat and Lutz Schröder. Automata learning: An algebraic approach. In *LICS '20: Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 900–914. Association for Computing Machinery, New York, 2020.

[35] Frits W. Vaandrager. Model learning. *Commun. ACM*, 60(2):86–95, 2017.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5244–5253. PMLR, 2018.

[44] Xiyue Zhang, Xiaoning Du, Xiaofei Xie, Lei Ma, Yang Liu, and Meng Sun. Decision-guided weighted automata extraction from recurrent neural networks. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 11699–11707. AAAI Press, 2021.

Appendix A. Dual approach using columns

In this section, we present (for clarity) the natural dual definitions to those provided in Section 2 and Section 3 working with (partial) columns instead of (partial) rows and consider right multiplication instead of left multiplication. As a result, the role of final and initial states is swapped in our constructions. S still denotes a semiring and Σ an alphabet.

Right semimodules, closed sets and right generating sets: Given a set Y, S^Y is a *right* S-module via the following action: for each element $\underline{x} \in S^Y$ and λ in S, $\underline{x} \otimes \lambda$ denotes the element of S^Y computed from $\underline{x}$ by multiplying every component of $\underline{x}$ by λ on the *right*. Given a subset $X \subseteq S^Y$, a right-linear combination over X is one of the form $\bigoplus_{\underline{x} \in X}(\underline{x} \otimes \lambda_x)$ for some $\lambda_{\underline{x}} \in S$, with only finitely many $\lambda_{\underline{x}}$ 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.

As before, we will use the notation $\underline{x}$ for elements of S^{Σ^*} and $\underline{x}_Z$ for the restriction of $\underline{x}$ to entries indexed by a subset $Z \subseteq \Sigma^*$, remembering that the action of S is now on the *right*. We also define a *left* action of Σ^* on S^{Σ^*} as follows: given an element $\underline{x} \in S^{\Sigma^*}$, and $u \in \Sigma^*$, we denote by $u \cdot \underline{x}$ the element of S^{Σ^*} defined by $(u \cdot \underline{x})_w = \underline{x}_{wu}$ for all words w. It will also be convenient to have a notation for elements of S^{Σ^*} that arise as columns of a given Hankel matrix F. If v is a word, we shall write $[v]$ to denote the infinite column indexed by v in F, viewed as an element of S^{Σ^*}. Note that the left action of Σ^* on $[v]$ corresponds to the obvious left action of Σ^* on v, that is: $(u \cdot [v])_w = [v]_{wu} = F_{wu,v} = f(wuv) = F_{w,uv} = [uv]_w$.

Let $Z \subseteq \Sigma^*$ and $f : \Sigma^* \to S$. A subset $X \subseteq S^{\Sigma^*}$ is said to be: (i) right-closed if for all $\underline{x}$ in X and $a \in \Sigma$, $a \cdot \underline{x}$ is right-generated by X; (ii) right-closed on Z if for all $\underline{x}$ in X and $a \in \Sigma$, $(a \cdot \underline{x})_Z$ 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 language, we say that a set of words T is right-closed (resp. right-closed on Z, column-generating for f) if the set $\underline{T} = \{[t] : t \in T\}$ has this property.

Co-Hankel automata and co-hypothesis automata:

Notation A.1. Given a function $f : \Sigma^* \to S$, a finite non-empty subset T of S^{Σ^*} and a non-empty subset Q of Σ^*, we define the right-solution set $\Gamma_{Q,T}$ to be set of elements $\underline{\gamma} \in S^{T \cup (T \times \Sigma \times T)}$ satisfying:

- $[\varepsilon]_Q = \bigoplus_{\underline{t} \in T} \underline{t}_Q \otimes \gamma_{\underline{t}}$, and
- for all $\underline{t} \in T$ and all $a \in \Sigma$, $(a \cdot \underline{t})_Q = \bigoplus_{\underline{s} \in T} \underline{s}_Q \otimes \underline{\gamma}_{\underline{s},a,\underline{t}}$.

If $Q = \Sigma^*$, we will simply write Γ_T instead of $\Gamma_{Q,T}$. Moreover, if $R \subseteq \Sigma^*$, then to reduce notation we will also frequently write simply $\Gamma_{Q,R}$ and Γ_R instead of $\Gamma_{Q,T}$ and Γ_T where $T = \{[t] : t \in R\}$.

It is clear from the definitions above that $\Gamma_{Q,T}$ is non-empty if and only if $[\varepsilon]_Q$ is right-generated by T restricted to Q and T is right-closed on Q. Also, if $Q' \subseteq Q$, then $\Gamma_{Q',T} \supseteq \Gamma_{Q,T}$. However, if T' is a finite set contained in or contained by T, there is no containment between $\Gamma_{Q,T}$ (which lies in $S^{T \cup (T \times \Sigma \times T)}$) and $\Gamma_{Q,T'}$ (which lies in $S^{T' \cup (T' \times \Sigma \times T')}$).

Definition A.2. Let $f : \Sigma^* \to S$ be a function and suppose that T is a finite subset of S^{Σ^*} and Q is a subset of Σ^* such that $\Gamma_{Q,T} \neq \emptyset$. Then for each $\gamma \in \Gamma_{Q,T}$, we define a finite weighted automaton $\mathcal{H}^{co}_{Q,T,\underline{\gamma}}$ with:

- finite set of states T;
- initial-state vector $(\underline{t}_\varepsilon)_{\underline{t}\in T}$;
- final-state vector $\underline{\gamma}_T$; and
- for all $\underline{s}, \underline{t} \in T$ and all $a \in \Sigma$, a transition from state $\underline{s}$ to state $\underline{t}$ labelled by the letter a with weight $\underline{\gamma}_{\underline{s},a,\underline{t}}$.

If $Q = \Sigma^*$, we say $\mathcal{H}^{co}_{Q,T,\underline{\gamma}}$ is a *co-Hankel automaton* and write simply $\mathcal{H}^{co}_{T,\underline{\gamma}}$ to reduce notation. If T is a finite set of columns of the Hankel matrix and Q is finite, we say that $\mathcal{H}^{co}_{Q,T,\underline{\gamma}}$ is a *co-hypothesis automaton*; in this case we may view T as a finite set of words by identifying with an appropriate index set note: there need not be a unique set of words.

Theorem A.3. *Let $f : \Sigma^* \to S$ be a function. The following are equivalent:*

(1) *there exists a finite subset $T \subseteq S^{\Sigma^*}$ that is both right-closed and right-generates the column indexed by ε in the Hankel matrix of f (i.e. $\Gamma_T \neq \emptyset$ for some finite subset $T \subseteq S^{\Sigma^*}$);*
(2) *f is computed by a finite-state weighted automaton $\mathcal{A}$ over S.*

Specifically, if $\Gamma_T \neq \emptyset$, then $\mathcal{H}^{\mathrm{co}}_{T,\underline{\gamma}}$ computes f for all $\underline{\gamma} \in \Gamma_T$, whilst if f is computed by the finite-state weighted automaton $\mathcal{A}$ over S with state set T, then $\Gamma_{\underline{T}} \neq \emptyset$ for $\underline{T} = \{\underline{t} : t \in T\} \subseteq S^{\Sigma^*}$ where for each $w \in \Sigma^*$ we define $\underline{t}_w$ to be the value on input w computed by the automaton $\mathcal{A}_t$ obtained from $\mathcal{A}$ by making t the unique final state with weight 1_S and all other weights are the same).

(Weakly/strongly) co-guessable functions and chain conditions: Here we record the definitions used in the statement

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

Definition A.5. A function $f : \Sigma^* \to S$ 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 a co-guessable function f is strongly co-guessable if for all finite sets $T \subseteq \Sigma^*$, there exists a finite set $Q \subseteq \Sigma^*$ such that $\Gamma_{Q,T} = \Gamma_T$.

Definition A.6. Given a function $f : \Sigma^* \to S$ and F its Hankel matrix, f satisfies the weak co-ascending chain condition if for all chains of right-semimodules $X_0 \subseteq X_1 \subseteq X_2 \subseteq \dots$ such that

- each X_i is generated by a finite set of columns of F;
- each column of F belongs to some X_i

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

Definition A.7. Let T be a finite subset of Σ^*. A total T-chain is an infinite descending chain of right-solution sets $\Gamma_{Q_0,T} \supsetneq \Gamma_{Q_1,T} \supsetneq \Gamma_{Q_2,T} \supsetneq \cdots$ with $Q_i \subseteq Q_{i+1}$ for all $i \geq 0$ and such that each word in Σ^* is contained in some Q_i. We say that a function $f : \Sigma^* \to S$ is column-bound if there exists a finite set of words T such that $\Gamma_T \neq \emptyset$ and there is no total $T-$chain. We say that $f : \Sigma^* \to S$ is strongly column-bound if for *every* finite set of words T, there is no total T-chain.

Appendix B. Examples

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. Thus $f_1(w)$ is equal to the length of the longest block of consecutives a's in w. 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 Theorem 3.6 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. Thus $f_1'(w)$ is equal $2^n - 1$ if $w = a^n$ for some $n > 0$ and 0 otherwise. 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 and restricting attention to row w, 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 $f_1'(w_i a) = 0 = f'(w_i)$. 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. Thus $f_1''(w) = \{a^{|w|_a}, b^{|w|_b}\}$. 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 possibly 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. Thus $f_2(w) = |w|_a$ if w starts with an a, $f_2(w) = |w|_b$ if w starts with an b, and $f_2(\varepsilon) = -\infty$. Recalling that multiplication over this semiring is given by usual addition, if w starts with a we have $\langle w \rangle = (|w|_a - 1) \otimes \langle a \rangle$. Likewise, if w starts with b we have $\langle w \rangle = (|w|_b - 1) \otimes \langle b \rangle$. It follows from these simple observations that every row of the Hankel matrix is a scalar multiple of one of the rows $\langle \varepsilon \rangle, \langle a \rangle$ or $\langle b \rangle$. Moreover, it is not too difficult to see that none of these three can be expressed as a linear combination of the other two: for example the fact that $f_2(\varepsilon) = -\infty$ and $f_2(u) \neq -\infty$ for all non-empty words u quickly demonstrates that $\langle \varepsilon \rangle$ cannot be generated by any combination of the other rows.

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 $W = \{w_1, \ldots, w_k\}$ 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 $\Lambda_{Q,T} = \Lambda_Q \neq \emptyset$, and hence for all $\underline{\lambda}$ in $\Lambda_{Q,T}$, we have that $\mathcal{H}_{Q,T,\lambda}$ computes f_2. By the observations at the start of this subsection, we must have that Q contains ε and at least one word, say q, beginning with a and at least one word, say q', beginning with b. Without loss of generality, we may assume that for a fixed constant N (to be chosen below) we have that the length of the longest word in T is greater than N (since for all $T' \supseteq T$ we also have $\Lambda_{Q,T} = \Lambda_{Q,T'} = \Lambda_Q \neq \emptyset$). Now let n be equal to the maximal sum of the length a word in Q and the length of a word in T (which by the above observation can be made arbitrarily large, by modifying T if necessary). For this choice of n and for all $t \in T$ we have

$$\begin{aligned}
\langle qa \rangle_t &= f_2(qat) = |qat|_a = |qt|_a + 1 \geq 0 \geq |q't|_b - n \Rightarrow \langle qa \rangle_T = (1 \otimes \langle q \rangle_T) \oplus (-n \otimes \langle q' \rangle_T) \\
\langle qb \rangle_t &= f_2(qbt) = |qbt|_a = |qt|_a + 0 \geq 0 \geq |q't|_b - n \Rightarrow \langle qb \rangle_T = (0 \otimes \langle q \rangle_T) \oplus, (-n \otimes \langle q' \rangle_T) \\
\langle q'a \rangle_t &= f_2(q'at) = |q'at|_b = |qt|_b + 0 \geq 0 \geq |qt|_a - n \Rightarrow \langle q'a \rangle_T = (-n \otimes \langle q \rangle_T) \oplus 0\langle q' \rangle_T \\
\langle q'b \rangle_t &= f_2(q'bt) = |q'bt|_b = |q't|_b + 1 \geq 0 \geq |qt|_a - n \Rightarrow \langle q'b \rangle_T = (-n \otimes \langle q \rangle_T) \oplus (1 \otimes \langle q' \rangle_T).
\end{aligned}$$

Now let $\mathcal{H}_{\mathcal{Q},\mathcal{T},\underline{\lambda}}$ be the hypothesis automaton constructed from an element $\underline{\lambda} \in \Lambda_{Q,T} = \Lambda_Q$ using the linear combinations above (containing the *negative* coefficients $-n$) and let $\mathcal{H}_q$ denote the automaton obtained from $\mathcal{H}_{Q,T,\underline{\lambda}}$ by making $\langle q \rangle$ the unique initial state. Since

$\underline{\lambda} \in \Lambda_Q$, it follows from the first part of the proof of Theorem 2.12 that for all words w, and all $q \in Q$, $\langle q \rangle_w$ is equal to the value computed by $\mathcal{H}_q$ on input w. Thus in particular we have that $f_2(qa^n bb^{2n})$ is equal to the value computed by $\mathcal{H}_q$ on input $a^n bb^{2n}$. Consider the run composed of: the transition from q to q labelled by a for each instance of a in the factor a^n, the transition from q to q' labelled by b for the first factor of b, and the transition from q' to q' labelled by b for each instance of b in the remaining factor b^{2n}. Since the final weight at state q' is (by definition) $\langle q' \rangle_\varepsilon = f_2(q') = |q'|_b$, it then follows from the equations above (recalling that $\otimes$ denotes usual addition in the semirings under consideration) that this run has weight $n + (-n) + n + |q'|_b = 2n + |q'|_b$ giving $f_2(qa^n bb^{2n}) > 2n$ (recalling that the addition of this semiring is maximisation). On the other hand, since q starts with the letter a, we have $f_2(qa^n bb^{2n}) = |q|_a + n$ (from the definition of this function). But now taking $N = |q|_a$ we have $f_2(qa^n bb^{2n}) > 2n > n + N = |q|_a + n = f_2(qa^n bb^{2n})$, giving a contradiction.

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 from Lemma 4.3 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 4.4, 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 Lemma 4.5. 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. Thus $f_3(w) = |w|_a$ if w starts with an a, $f_3(w) = |w|_b$ if w starts with an b, and $f_3(\varepsilon) = -\infty$, and (by the same argument as for f_2) we find that f_3 is computed by a literal automaton and weakly guessable; in this case however Lemma 4.4 applies to give that f_3 is also strongly guessable (and note that the proof that f_2 is *not* guessable made use of *negative* coefficents that we do not have access to here!)) 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 $W = \{w_1, \ldots, w_k\}$ 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. Thus $f'_3(w) = 2^{|w|_a}$ if w starts with an a, $f_3(w) = 2^{|w|_b}$ if w starts with an b, and $f_3(\varepsilon) = 0$. 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 $W = \{w_1, \ldots, w_k\}$ 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. Thus $f_3''(w) = \{x^{|w|_x}\}$ if w starts with an $x \in \{a, b\}$, and $f_3''(\varepsilon) = \emptyset$. 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 $W = \{w_1, \ldots, w_k\}$ 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. That is, the function computed, which we denote by f_5, is defined by $f_5(w) = 2^{|w|_a}$ if w begins with a, $f_5(w) = 1$ if w begins with b and $f_5(\varepsilon) = 0$. In a similar manner to the previous cases, it is easy to see that f_5 is computed by a literal automaton and hence is weakly guessable. 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 $W = \{w_1, \ldots, w_k\}$ 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 (by Lemma 4.3) $\bar{f}_3$ and $\bar{f}_3'$ are strongly co-guessable but not weakly guessable .

This work is licensed under the Creative Commons Attribution License. To view a copy of this license, visit `https://creativecommons.org/licenses/by/4.0/` or send a letter to Creative Commons, 171 Second St, Suite 300, San Francisco, CA 94105, USA, or Eisenacher Strasse 2, 10777