Unambiguity in Automata Theory

Thomas Colcombet

Cnrs, Université Paris 7 - Paris Diderot
thomas.colcombet@liafa.univ-paris-diderot.fr

Abstract. Determinism of devices is a key aspect throughout all of computer science, simply because of considerations of efficiency of the implementation. One possible way (among others) to relax this notion is to consider unambiguous machines: non-deterministic machines that have at most one accepting run on each input.
In this paper, we will investigate the nature of unambiguity in automata theory, presenting the cases of standard finite words up to infinite trees, as well as data-words and tropical automata. Our goal is to show how this notion of unambiguity is so far not well understood, and how embarrassing open questions remain open.

1 Introduction

In many areas of computer science, the relationship between deterministic and non-deterministic devices is extensively studied. This is in particular the case in complexity theory, and also in automata theory. The notion of unambiguous devices, i.e., non-deterministic devices that have at most one accepting execution for each accepted input, is a natural intermediate class that is potentially more expressive (or succinct) than deterministic devices, while behaviorally easier to handle than general non-deterministic machines.

One specificity of this class is that it is a semantic one: a priori, nobody knows whether a given Turing machine is unambiguous or not. This is undecidable, and even providing a witness of unambiguity is not possible.

Even for weaker complexity classes, such as logspace, the status of the unambiguous machines is not settled. Indeed, unambiguous logspace (UL) is located somewhere between deterministic logspace (L) and non-deterministic logspace NL. Since L and NL are not known to be separated, the separation of UL with respect to either L or NL is also open. This class UL is also interesting, since it is known to contain planar reachability, while the main complete problem for NL is general reachability in a directed-graph. Interestingly, Allender and Reinhardt have shown that in non-uniform complexity classes (i.e., in the presence of advice), logspace and non-deterministic logspace coincide [35].

In the world of automata, the picture is better understood. As a first key difference, unambiguity becomes easily decidable, and furthermore, it is possible to compute and work with witnesses of unambiguity. Nevertheless, many questions related to unambiguity are embarrassingly open and surprisingly complicated.

In fact, the subject of unambiguity related to automata is so wide that it would require a much larger and ambitious presentation in itself. The reader

interested in pursuing these subjects further will find a lot of material in [15,36], and in surveys such as [17] for standard word automata.

Many subjects involving unambiguity cannot even be mentioned in this paper. This includes unambiguous non-finite state machines (such as pushdown automata, see e.g., [17,31] or less standard forms of automata such as constrained automaton in [6]) or unambiguous regular expressions. Even the theory of codes is in essence a study of unambiguity. There are also some intermediate forms of restricted ambiguity, such as m-ambiguity or polynomial ambiguity (when the number of accepting runs are bounded by m, or by a polynomial in the length of the input) [24], as well as restricted syntactic variants of unambiguous automata ([25] among others). The unambiguous polynomial closure of a family of languages has also been characterized [32]. Other algorithmic questions have also be addressed, such as the inference of automata [12]. Unambiguity is also a very important and a well studied subject in connection with transducers since unambiguous transducers are very close to functional ones (transducers which, instead of a relation, recognize a function) [39,2,43] . Unambiguity can also be considered in the analysis of rational subsets of monoids in the absolute [3]. Unambiguity can also be studied for extended notions of words, and in particular infinite words. Over infinite words of length ω, a very important notion of unambiguous automata is the one of prophetic automata [9] (a semantic version of determinism from right to left). Also, on infinite words of unrestricted langth, and even for more general classes of automata, compiling temporal logics yields unambiguous automata [13]. None of these topics will be addressed in this paper.

This paper does not intend to make any exhaustive survey of the large body of works related to unambiguity. It rather offers a tour, visiting several arbitrarily chosen topics, involving significantly different situations. This tour starts with standard non-deterministic word automata and their unambiguous subclasses. Several complexity arguments are elegant and worth knowing in this context, such as the use of communication complexity, and the counting principle for deciding universality in polynomial time. We continue with tropical automata. This specific kind of weighted automata computes functions from words to integers. We will see that some elementary problems are undecidable for such automata, and that unambiguous automata are a subclass that appears naturally and has good decidability properties. The description then proceeds with the infinite tree case, in which the story is completely different. There, unambiguity is related to the problem of existence of choice functions. We will finish with a study of register automata, where unambiguity turns out to be a very important subclass. Recent unpublished results obtained with Puppis and Skrcypczak sustain this idea.

In this paper, we will consider the case of finite-state automata (Section 2), of tropical automata (Section 3), of automata over infinite trees (Section 4) and of automata over data words (Section 5). In the first three situations, we will see that difficult questions remain open. The last case will report on recent unpublished work in which a constructive understanding of the notion of unambiguity yields new results.

2 Unambiguous word automata

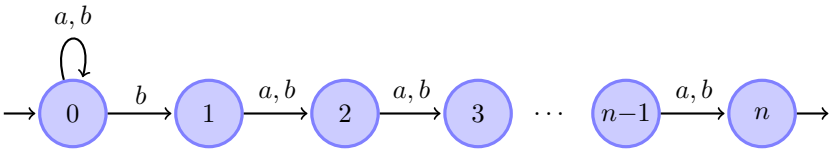
In this section, we consider unambiguous automata over finite words. Let us first briefly recall some standard notation.

In this section, we adopt the standard terminology concerning (finite-state) non-deterministic automata. A *non-deterministic automaton* $\mathcal{A}$ reading words over the *alphabet* A has a finite set of *states* Q, a set of *initial states* I, a set of *final states* F, and a *transition relation* $\Delta \subseteq Q \times A \times \Delta$. A *run* ρ over the input word $a_1 \cdots a_n$ of the automaton is a sequence of transitions of the form $(q_0, a_1, q_1)(q_1, a_2, q_2) \cdots (q_{n-1}, a_n, q_n) \in \Delta^*$. It is *accepting* if furthermore $q_0 \in I$ and $q_n \in F$. If an accepting run exists over an input word u, then u is *accepted*. The language *recognized* by the automaton is the set of accepted words. It is denoted $L(\mathcal{A})$. Languages recognized by an automaton are called *regular*. An automaton is *deterministic* if for all states p and all letters $a \in A$ there exists at most one state r such that (p, a, r) is a transition. An automaton is *unambiguous* if for all input words there exists at most one accepting run over it.

One of the first results we learn in an automata course is the inherent exponential blowup of determinization.

Theorem 1 ([34,27,28,29]). *Non-deterministic word automata of size n can be transformed into deterministic and complete automata of size at most 2^n for the same language. This bound is tight.*

The witness automaton for the lower bound is very natural: it checks that the n-th letter from the end is a b (over the alphabet a, b):



However, though this example is non-deterministic, if we reverse the orientation of its edges (yielding the *mirror automaton*), we obtain a deterministic automaton. This implies that it has at most one accepting run over each input: it is unambiguous. It happens also, as a consequence, that it is very easy to complement it: One adds a new state $\perp$ with self loops labelled a, b, and linked to 1 by an a transition. Then, it is sufficient to complement the set of initial states for complementing the accepted language.

This example shows that the class of unambiguous automata is potentially an interesting compromise between succinctness (these can be exponentially smaller than deterministic automata) and tractability of fundamental problems.

Theorem 2 ([22,23,24]). *Unambiguous automata can be exponentially more succinct than deterministic automata. Non-deterministic automata can be exponentially more succinct than unambiguous ones.*

Remark 1. Deciding if a non-deterministic word automaton $\mathcal{A}$ is unambiguous is doable in polynomial time. The principle is as follows: consider the product of the automaton $\mathcal{A}$ with itself. Over a given input, an accepting run of this new automaton can be seen as the pair of two accepting runs of $\mathcal{A}$ over this input. It is easy to slightly modify this automaton in such a way that it accepts an input if and only if the input is accepted by two distinct accepting runs of $\mathcal{A}$ (this can be achieved, e.g., by adding one extra bit to each state storing whether the two runs have differed so far). This new automaton has quadratic size in the original one. It accepts an input if and only if the original automaton is ambiguous. Thus, the non-deterministic automaton $\mathcal{A}$ is unambiguous if and only if the language recognized by this new automaton is empty. This can be tested in polynomial time (NL more precisely).

Using variations around these ideas, we can show that unambiguity is decidable for all the classes of automata considered in this paper (tropical automata, infinite tree automata as well as register automata).

There is at least one strong evidence that unambiguous automata are inherently simpler than general non-deterministic automata. This is the complexity of the equivalence, containment, and universality problems. In general, given two non-deterministic automata recognizing the languages K, L respectively, the problem of *equivalence* "$K = L$?", of *containment* "$K \subseteq L$?" and of *universality* "$K = A^*$?" are known to be PSPACE-complete. This is not the case for unambiguous automata, as shown in the following theorem.

Theorem 3 ([41,40]). *The problems of universality and equivalence of unambiguous automata as well as containment of a non-deterministic automaton in an unambiguous automaton are solvable in polynomial time.*

not shown in [41, 40]

neither here in the following…

We shall see in this section a complete proof of this result, which is a good excuse for introducing several important techniques.

Of course, knowing this complexity result, and since universality amounts to checking the emptiness of the complement, one might think that another proof of this result could be as follows: complement the unambiguous automaton with a polynomial blowup of states, and then test for emptiness in polynomial time. However, the question of whether unambiguous automata can be complemented with a polynomial blowup in the number of states is an open problem.

WRONG

Conjecture 1. It is possible to complement unambiguous automata of size n into unambiguous automata of size polynomial in n.

In fact, even whether we can complement an unambiguous automaton into a non-deterministic automaton of polynomial size is open. We lack techniques for addressing this question. In particular, how can we prove a lower bound on the size of an unambiguous automaton for a given language?

Communication complexity and the rank technique [37,23,24] There is a nice technique for proving lower bounds on the size of an unambiguous automaton for a language, based on communication complexity. Consider a language $L \subseteq A^*$.

Define the *communication relation* $\mathrm{Com}(L) \subseteq A^* \times A^*$ to be the set of ordered pairs (u, v) such that $uv \in L$. A subset of $A^* \times A^*$ is called a *rectangle* if it is of the form $M \times N$ for $M, N \subseteq A^*$. A *non-deterministic decomposition* of $R \subseteq A^* \times A^*$ is a finite union of rectangles, and its *complexity* is the number of rectangles involved in the union. An *unambiguous decomposition* is a non-deterministic decomposition into disjoint rectangles. The *non-deterministic complexity* of R (resp., *unambiguous complexity*) is the minimal complexity nd-comp(R) (resp., unamb-comp(R)) of a non-deterministic decomposition (resp., unambiguous decomposition) of R.

It is easy to show that a language L accepted by a non-deterministic automaton with n states is such that nd-comp$(\mathrm{Com}(L)) \leq n$. Indeed, define $L_{I,q}$ to be the language recognized by the automaton when the set of final states is set to $\{q\}$, and $L_{q,F}$ to be the language recognized by the automaton when the set of initial states is set to be $\{q\}$. Clearly, for all words u, v, $uv \in L$ if and only if there exists a state q such that $u \in L_{I,q}$ and $v \in L_{q,F}$. This means that $\mathrm{Com}(L) = \cup_{q \in Q} L_{I,q} \times L_{q,F}$. We have found a non-deterministic decomposition for $\mathrm{Com}(L)$ of complexity n.

Pushing further, a language L accepted by an unambiguous automaton of size n is such that unamb-comp$(\mathrm{Com}(L)) \leq n$. Indeed, consider the non-deterministic decomposition $\mathrm{Com}(L) = \cup_{q \in Q} L_{I,q} \times L_{q,F}$ as in the non-deterministic case, and assume it would be ambiguous. This would mean that there are two distinct states p, q such that $L_{I,p} \cap L_{I,q} \neq \emptyset$ and $L_{p,F} \cap L_{q,F} \neq \emptyset$. Let u be a word in the first intersection, and v be a word in the second. Then the word uv is accepted by two distinct runs: one that reaches state p after reading u, and the other that reaches state q at the same position. This contradicts the unambiguity assumption. Thus, $\cup_{q \in Q} L_{I,q} \times L_{q,F}$ is an unambiguous decomposition of complexity n.

Linear algebra offers an elegant way to bound the unambiguous complexity of a relation from below. Indeed, we can identify a relation $R \subseteq E \times F$ with its *characteristic matrix*: rows are indexed by E and columns by F, and the entry indexed by words x, y is 1 if $(x, y) \in R$ and 0 otherwise.

Lemma 1. $\mathrm{rank}(R) \leq$ unamb-comp(R).

Proof. A union of disjoint rectangles can be understood as the sum of the characteristic matrices representing them. Since the rank of a matrix that "contains only one rectangle" is 1 (or 0 if the rectangle is empty), and rank is subadditive, the rank of a matrix is smaller than its unambiguous complexity. □

From this we can derive an upper bound on non-universality witnesses.

Lemma 2 ([37]). *Any shortest witness of non-universality for an unambiguous automaton with n states has length at most n.*

Proof. Let $\mathcal{A}$ be an unambiguous automaton, and let $a_1 a_2 \cdots a_n \notin L(\mathcal{A})$ be the shortest witness of non-universality (if it exists). Consider the matrix N obtained

from $\mathrm{Com}(L(\mathcal{A}))$ by restricting the rows to $v_0 = \varepsilon, v_1 = a_1, v_2 = a_1a_2, \ldots, v_n = a_1 \cdots a_n$ and the columns to $w_0 = a_1 \cdots a_n, w_1 = a_2 \cdots a_n, \ldots, w_n = \varepsilon$.

Since $v_i w_i = u \notin L(\mathcal{A})$ for all $i = 0 \ldots n$, the diagonal of this matrix consists only of 0's. However, for all $0 \leq i < j \leq n$, we have $|v_i w_j| < n$. Hence the "upper right" part of the matrix consists solely of ones. We claim that this matrix has rank at least n. Indeed, that $(1) - N$ (where (1) is the matrix using with 1's on all its entries) is lower triangular with a diagonal of 1. Thus $(1) - N$ has rank $n + 1$. Since (1) has rank 1, we obtain that the rank of N is at least n by subadditivity.

It follows that an unambiguous automaton for $L(\mathcal{A})$, and thus in particular $\mathcal{A}$ has at least n states. □

Of course, from Lemma 2, one immediately gets a CoNP procedure for deciding whether an unambiguous automaton is universal. However, it is possible to do better, and prove Theorem 3.

Proof (of Theorem 3). For each letter of the alphabet a, consider the matrix $\eta(a) \in \mathbb{N}^{Q \times Q}$ that describes the transition relation of an unambiguous automaton $\mathcal{A}$: the entry p, q of $\eta(a)$ is 1 if there is a transition labelled a in $\mathcal{A}$ from state p to state q, and 0 otherwise. Let us extend η into a morphism from A^* to $\mathbb{N}^{Q \times Q}$ using the standard matrix multiplication: $\eta(\varepsilon) = \mathrm{Id}_Q$, and $\eta(ua) = \eta(u)\eta(a)$. It is easy to prove by induction that the entry p, q of $\eta(u)$ is the number of runs from state p to state q over the word u. Let also I, F be the characteristic vectors of the initial and final states of $\mathcal{A}$ respectively. Thus, ${}^tI\eta(u)F$ is the number of accepting runs of $\mathcal{A}$ over the word u $(\star)$.

Now let us count the number of accepted words up to length n. Define

$$B(n) = \sum_{u \in A^*,\ |u| \leq n} {}^tI\eta(u)F \ .$$

From $(\star)$, B_n is the number of accepting runs over words up to length n. Since furthermore the automaton $\mathcal{A}$ is unambiguous, B_n is also the number of accepted words up to length n.

Let us show that this quantity can be computed in time polynomial in n. Indeed, define for all $m = 0, \ldots, n$ the matrices:

$$E_m = \sum_{u \in A^*,\ |u| = m} \eta(u)\,, \qquad \text{and} \quad F_m = \sum_{u \in A^*,\ |u| \leq m} \eta(u)\,.$$

These are such that $F_0 = Id_Q$, $E_1 = \sum_{a \in A} \eta(a)$, and for all $m \geq 1$, $F_m = F_{m-1} + E_m$ and $E_{m+1} = E_m E_1$. These equations can be used to compute F_n in polynomial time (and the numbers in the matrices have a linear number of digits). Thus $B_{n+1} = {}^tIF_{n+1}F$ is computable in polynomial time.

To conclude, an algorithm that decides universality is as follows: compute B_n and check that it equals the number of words of length at most n (i.e., $(|A|^{n+1} - 1)/(|A| - 1)$). This procedure succeeds if an only if all words are accepted up to length n, which in turns holds (by Lemma 2) if and only if $\mathcal{A}$ is universal. □

What we have seen in this proof is that it is reasonable to conjecture that unambiguous automata are closed under complement with polynomial blowup. Indeed, this is consistent with (1) the complexity of universality, and (2) the size of witnesses of non-universality. In fact, we can also report on another related conjecture:

Conjecture 2. Given two regular languages K, L of empty intersection, there is an unambiguous automaton of polynomial size that recognizing U that separates them, i.e., such that $K \subseteq U$ and $U \cap L = \emptyset$.

In particular, this would imply that all regular languages L can be turned into an unambiguous automaton of size polynomial in the size of a non-deterministic automaton for L and a non-deterministic automaton for $L^{\complement}$.

We will indeed see later that the separation of classes of non-deterministic automata is often related to unambiguous automata (cf. Theorem 5 and 13).

3 Unambiguous tropical automata

We pursue our investigation of unambiguity in the world of automata theory with the more exotic context of tropical automata. Tropical automata belong to the wider class of weighted automata as introduced by Schützenberger [38]. We are interested here in min-plus and max-plus automata.

Min-plus and *max-plus automata* are non-deterministic automata that have their transitions labelled with integers (reals would not make a difference) called *weights*. Given an accepting run of such an automaton, its *weight* is the sum of the weights of the transitions seen along the run. The semantic of a min-plus automaton is to *recognize* the function:

$$\begin{aligned}\llbracket \mathcal{A} \rrbracket_{\min} \quad : \quad & A^* \to \mathbb{Z} \cup \{+\infty\} \\ & u \mapsto \begin{cases} +\infty & \text{if there are no accepting runs of } \mathcal{A} \text{ over } u, \\ \min\{\text{weight}(\rho) \mid \rho \text{ accepting run of } \mathcal{A} \text{ over } u\} & \text{otherwise.} \end{cases}\end{aligned}$$

Dually, the semantic of a max-plus automaton is to *recognize* the function:

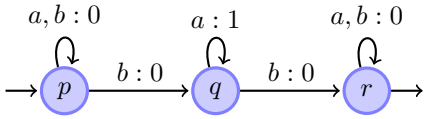
$$\begin{aligned}\llbracket \mathcal{A} \rrbracket_{\max} \quad : \quad & A^* \to \mathbb{Z} \cup \{-\infty\} \\ & u \mapsto \begin{cases} -\infty & \text{if there are no accepting runs of } \mathcal{A} \text{ over } u, \\ \max\{\text{weight}(\rho) \mid \rho \text{ accepting run of } \mathcal{A} \text{ over } u\} & \text{otherwise.} \end{cases}\end{aligned}$$

These two notions are formally dual in the following sense: Define $-\mathcal{A}$ to be the automaton $\mathcal{A}$ in which the weights of all transitions are the opposed weight, then $\llbracket -\mathcal{A} \rrbracket_{\min} = -\llbracket \mathcal{A} \rrbracket_{\max}$. There are several other ways to define the functions recognized by such automata, in particular using a matrix presentation. These automata appear in many applications. By *tropical automata* we refer indistinctly to either max-plus automata or min-plus automata.

A non-deterministic automaton $\mathcal{A}$ can be viewed as a min-plus automaton with all its transitions labelled with weight 0. In this case, it recognizes the

function $[\![\mathcal{A}]\!]_{\min}$ which maps a word to 0 if it belongs to $L(\mathcal{A})$, and to $+\infty$ otherwise. Symmetrically, a non-deterministic automaton can be viewed as max-plus automaton that recognizes the function which maps a word to 0 if it belongs to $L(\mathcal{A})$, and to $-\infty$ otherwise.

Example 1. The following tropical automaton uses its non-determinism for choosing a segment of consecutive a's surrounded by two b's, and computing its length.



If this automaton is a min-plus automaton, then it maps every word of the form $a^{n_0}ba^{n_1}b\cdots ba_k$ to $\min(n_1,\ldots,n_{k-1})$ if $k \geq 2$, and $+\infty$ otherwise. If this automaton is a max-plus automaton, then it maps every word of the form $a^{n_0}ba^{n_1}b\cdots ba_k$ to $\max(n_1,\ldots,n_{k-1})$ if $k \geq 2$, and $-\infty$ otherwise.

In the world of tropical automata, things are not as nice as for classical finite-state automata, in the sense that undecidability results occur immediately. The central result in this direction is the one of Krob.

Theorem 4 (Krob [21], and [1,14] for simple proofs). *Given a min-plus automaton recognizing a function f, it is undecidable whether $f \leq 0$. Given a max-plus automaton recognizing a function f, it is undecidable whether $f \geq 0$.*

In particular, this means that $f \leq g$ is undecidable for f, g recognized by tropical automata. In fact, this is not completely true: there is one case when this question is decidable, when f is recognized by a max-plus automaton and g by a min-plus automaton, while all other combinations are undecidable by the above theorem.

A tropical automaton is *unambiguous* if the underlying non-deterministic automaton is unambiguous. For instance, in the above example, the automaton is ambiguous. A more careful analysis would show that no unambiguous automata could recognize these functions.

The class of unambiguous tropical automata is interesting since in the definition of $[\![\cdot]\!]_{\min}$ and $[\![\cdot]\!]_{\max}$, the min and the max range over at most one accepting run. Hence, as long as we identify $+\infty$ and $-\infty$, $[\![\cdot]\!]_{\min}$ and $[\![\cdot]\!]_{\max}$ coincide. For this reason, we allow ourselves to simply mention *unambiguous tropical automata* without further mentioning whether these are min-plus or max-plus.

Theorem 5 ([26]). *Functions that are both recognized by min-plus automata and max-plus automata are recognized by unambiguous tropical automata.*

Very informally, if we interpret max-plus automata as a form of complement of min-plus automata, then we can see unambiguous tropical automata as automata that correspond to be both non-deterministic and of non-deterministic complement. We will see a similar phenomenon in the context of register automata in Section 5.

A natural question arises: can we decide whether an automaton is equivalent to an unambiguous one? Some first results were obtained in [20]. The best known result is the following:

Theorem 6 ([19]). *There is an algorithm which, given a polynomially ambiguous*[1] *tropical automaton decides whether there is an unambiguous tropical automaton recognizing the same function.*

Quite naturally, the most important question in this context is to lift the polynomial ambiguity assumption.

Question 1. Can we decide, given a tropical automaton whether it is equivalent to an unambiguous one?

4 Unambiguous infinite tree automata

Another situation where the notion of unambiguity is worth noticing is the context of infinite trees. To keep the presentation light, we expect the reader to know the notion of non-deterministic automaton over infinite trees. An introduction can, for instance, be found in [42].

Let us start by recalling some definitions. An *infinite tree* labelled by the alphabet A is a map from $\{0,1\}^*$ to A. The elements of $\{0,1\}^*$ are *nodes*. The node ε is the *root* of the infinite tree. Given a node u, $u0$ is its *left child* and $u1$ its *right child*. The transitive closure of the child relation is the *descendant relation*. A *branch* is a maximal set of nodes totally ordered under the descendant relation. In this section, *languages* are sets of infinite trees. An *infinite tree automaton* has a finite set of states Q, a set of *initial states* $I \subseteq Q$, and a set of *transitions* $\Delta \subseteq Q \times A \times Q \times Q$. A *run* of an automaton over an infinite tree t is an infinite tree ρ labelled by Q such that $(\rho(u), t(u), \rho(u0), \rho(u1)) \in \Delta$ for all nodes u. The run is *accepting* if $\rho(\varepsilon) \in I$, and for all branches B the set of states assumed on infinitely many nodes by ρ belongs to a given set $M \subseteq 2^Q$, called the *Muller acceptance condition*[2]. If there is an accepting run of the automaton over some input infinite tree, then the infinite tree is *accepted*. The set of infinite trees accepted is the language *recognized* by the automaton.

The central result concerning infinite tree automata is without any question Rabin's theorem stating that infinite tree automata have effectively the same expressive power as monadic second-order logic over infinite trees, and that, as a consequence, this monadic second-order logic is decidable over infinite trees. This logical aspect is certainly far beyond the topic of this paper, but the main lemma in the proof is very relevant:

Lemma 3 (Rabin main lemma [33]). *Infinite tree automata are effectively closed under complement.*

Once more, for this class of automata, the unambiguity notion is natural. An *unambiguous infinite tree automaton* is an infinite tree automaton such that for all input infinite trees, there is at most one accepting run. To start with,

[1] An automaton is *polynomially ambiguous* if the number of accepting runs over an input is bounded by a polynomial in the length of the input.

[2] Other choices are possible, but these distinctions do not make any difference here.

there are languages which are recognized by infinite tree automata, but by no unambiguous infinite tree automata, and there are languages that are recognized by unambiguous infinite tree automata, and by no deterministic automata[3]. However, the status of unambiguous automata is very different here than in simpler contexts. In particular, it is not clear whether all regular languages can be recognized by unambiguous automata. You just wrote that there are inherently ambiguous languages…

A first answer has been given by Niwiński and Walukiewicz:

Theorem 7 ([30]). *Consider the language "there is a node labelled by the letter a". If this language is recognized by an unambiguous infinite tree automaton, then there exists a regular choice function.*

Informally, a *choice function* is a language that implements the notion of "choice", i.e., given a non-empty set, it selects a unique element in it. One way to formalize this is as follows: consider the alphabet a, b, a^c (a stands for the set in which choice has to be performed, and a^c is the chosen node). A language C of a, b, a^c-labelled infinite trees is a *choice function* if:

- All infinite trees in C contain exactly one occurrence of the letter a^c,
- For all a, b-labelled infinite tree t containing at least one occurrence of the letter a, there exists one and only one a-labelled node x such that $t[x \leftarrow a^c]$ is accepted, where $t[x \leftarrow a^c]$ is the infinite tree t in which the label of the node x is changed into a^c.

A *regular choice function* is a choice function which is recognized by an infinite tree automaton. The existence of a regular choice function has been first studied in [16], where the non-existence of such function is established. However, there is a known unrecoverable hole in the proof. The result was established by Carayol and Löding using much simpler automata-theoretic arguments.

Theorem 8 ([7]). *There does not exist any regular choice function over infinite trees.*

These results were finally published together.

Theorem 9 ([8]). *The language of infinite trees "there is a node labelled by the letter a" is regular, but* intrinsically ambiguous*; i.e., there exists no unambiguous automaton for this language.*

As it is the case for tropical automata, deciding if a language can be recognized unambiguously is an open problem.

Question 2. Given a infinite tree automaton, can we decide whether its recognized language can be recognized by an unambiguous automaton?

[3] In the context of trees, two forms of determinism for automata are possible: *top-down determinism*, i.e., from root to leaves, and *bottom-up determinism*, i.e., from leaves to root. The former (considered here) is known to be strictly weaker than general automata, even over finite trees. The later does not make real sense over infinite trees, since there may be no leaves.

However, if we come back to simpler classes of models, namely finite words, infinite words of length ω or finite trees, it is very easy to have a regular choice function, and also to transform any automaton into an equivalent one that is unambiguous. Nevertheless, there is still an unclear situation. Call *tamed* (or scattered, or thin) an infinite tree that has countably many branches (the definition of an infinite tree needs to be slightly generalized for that, and has to allow leaves). This class is very important, and such infinite trees are significantly simpler than general ones (in particular automata are simpler). To some extend, tamed infinite trees can be understood as the joint extension of infinite words and finite trees.

Question 3. Can we separate unambiguous automata from general automata over tamed trees? Does there exist an automaton, unambiguous over tamed trees that recognizes the language "there is a node labelled by a"? Does there exist a regular choice function over tamed trees?

Let us conclude with another, intriguing, relation linking unambiguity over infinite trees and the existence of regular choice functions over tamed trees.

Theorem 10 ([4]). *Under the assumption that there are no regular choice functions over tamed trees, there is an algorithm which decides whether a regular language of infinite trees is bi-unambiguous*[4].

5 Unambiguous register automata

In this last section, we concentrate our attention to data languages. Once more the questions raised are of a slightly different nature. More positively, this is an instance of a situation where new results can be obtained thanks to a careful analysis of the nature of unambiguity. This section will mainly be a report on recent unpublished results obtained in collaboration with Gabriele Puppis and Michał Skrzypczak, in particular establishing conjectures raised in [10].

Originally, register automata were introduced by Kaminski and Francez [18] and were the subject of much attention. There are various ways to introduce this model, including the very interesting "atom approach" [5]. We adopt here a more model-theoretic presentation.

Let us fix ourselves an infinite set of *data values* $\mathbb{D}$. We are only allowed to compare such values using equalities, and as a consequence, the exact set $\mathbb{D}$ does not really matter. Depending on the context, data values can be the identifiers in a database, simply numbers, the agent in a concurrent system, and so on... *Data words* are words over $\mathbb{D}$, i.e., elements of $\mathbb{D}^*$. It is also often convenient to consider slightly richer data words which are elements of $(A \times \mathbb{D})^*$. This distinction has essentially no impact in what follows. Sets of data words are named *data languages*.

A (non-deterministic) *register automaton* has *states*, *initial states*, *final states* and *transitions* as a non-deterministic finite automaton, and furthermore:

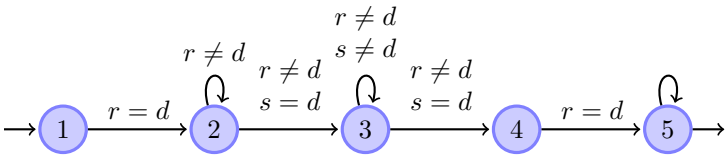
[4] A language of infinite trees is *bi-unambiguous* if it is accepted by a unambiguous infinite tree automaton as well as its complement.

– there is a finite set of *registers* $r, s, \ldots$, with values ranging in $\mathbb{D}$, and;
– transitions are equipped of *guards* that are boolean combinations of properties of the form
  - $r = s'$ for r, s registers, signifying that the value of the register r before the transition should be the same as the value of the register s after the transition,
  - $r = d$ for r a register, signifying that the value assumed by the registeŗ r before the transition is equal to the data read during the transition,
  - $d = s'$ is defined similarly.

This description should give a fairly good intuition of what is going on. A *run* of a register automaton is a sequence of *configurations* consisting of a state and a valuation of the registers, that respect the transitions and the guards. A run is *accepting* if it starts in an initial state and ends in a final state. It should be clear what an unambiguous register automaton is: an *unambiguous register automaton* is a register automaton such that on every input there is at most one accepting run.

Let us proceed with an example.

Example 2. Consider the following register automaton. We use two registers, r, s, and all transitions are assumed to preserve the values of these counters. Thus we only write on the transitions whether r and s should be equal or not-equal to the read data value.



Note first that this register automaton is non-deterministic: at the very beginning, we do not know the values of the registers (these have to be non-deterministically guessed in some sense). But even without this problem, it is hard to know when in state 2, whether the run should stay in state 2, or proceed with state 3.

In fact, at the same time that we describe the behavior of this automaton, we will see that it is unambiguous. Let us recall that in this register automaton, the values of the registers do not change along the run (we do not know these values *a priori*). While processing an input, this automaton has to take the as first transition the one from state 1 to state 2. Since this transition is guarded by $r = d$, it enforces the value of register r to be the first data value occurring in the input data word. Nox note that all transitions with both extremities among states $2, 3, 4$ have $r \neq d$ in their guard, and this enforces that the data value read to be different from the value of r. Note furthermore that the only transition that exits state 4 (and go to state 5) enforces $r = d$ in its guard. Hence, necessarily, the transition from 4 to 5 has to be taken the first time the value of r is seen again in the data word. This means that this position is unambiguously

determined. This means also that the moment the transition from 3 to 4 is used is also unambiguously determined (just one step before). Since furthermore the transition from 3 to 4 has guard $s = d$, the value of register s also has to be unambiguously determined. Overall, this means that, if there is an accepting run over some data word, then the values of r and s are uniquely determined: the value of r is the first data value in the input data word, and the value of s is the data value that occurs just before the second occurrence of the first data value. Once these values fixed, it is easy to see that this automaton is unambiguous, and the language it accepts can be described as follows:

$$\bigcup_{\substack{r,s \in \mathbb{D} \\ r \neq s}} r(\mathbb{D} \setminus \{r\})^* s(\mathbb{D} \setminus \{r,s\})^* sr\mathbb{D}^* \ .$$

It should also be clear that such a language cannot be determinized. Indeed, while reading an input word from left to right, it is not possible to know what the value of s should be as long as the second occurrence of the first data value is not met. Hence, a deterministic device should memorize all possible data values seen up to that moment. A similar argument prevents to determinize it from right to left.

When working with register automata, the undecidability is again close. The essential results are as follows.

Theorem 11 ([18]). *The languages recognized by register automata are effectively closed under union and intersection, and emptiness is decidable. The universality problem for register automata is undecidable.*

In [10], some conjectures were raised concerning the class of unambiguous register automata. These conjectures are now all established[5]. Let us briefly present these results.

Theorem 12 ([11]). *Unambiguous register automata are effectively closed under complement, and hence universality, containment and equivalence are decidable.*

However, in fact, using the same techniques, we obtain a separation result.

Theorem 13 ([11]). *Given two languages of data words K, L recognized by register automata of empty intersection, there exists a language of data words U recognized by an unambiguous register automaton that separates K and L, i.e., $K \subseteq U$ and $U \cap L = \emptyset$.*

In particular, a language of data words is recognized by an unambiguous register automaton if and only if both itself and its complement are recognized by register automata.

[5] Strictly speaking, Conjecture 6 is wrong, but has a corrected version.

6 Conclusion

In this paper, we have tried to present the notion of unambiguity following a rather non-standard path, in particular considering models that are usually not studied together. Along this presentation, we have seen several difficult open questions concerning unambiguous devices. These questions are natural, and show that unambiguity is still quite poorly understood. We have also seen several results that show that unambiguity arises sometimes from characterization reasons: (1) unambiguous tropical automata correspond to functions that are recognized by both min-plus and max-plus automata, and (2) unambiguous register automata correspond to languages that are both recognized as well as their complement by non-deterministic register automata. We believe that this characterization is more than a mere coincidence, and corresponds to the intrinsic nature of unambiguity.

Acknowledgment

I am really grateful to Jean-Éric Pin, Gabriele Puppis and Michał Skrypczak for their precious help and their discussions on the topic.

References

1. S. Almagor, U. Boker, and O. Kupferman. What's decidable about weighted automata? In *ATVA 2011*, Lecture Notes in Computer Science vol. 6996, pages 482–491. Springer, 2011.
2. Jean Berstel. *Transductions and context-free languages*, volume 38 of *Leitfäden der Angewandten Mathematik und Mechanik [Guides to Applied Mathematics and Mechanics]*. B. G. Teubner, Stuttgart, 1979.
3. Jean Berstel and Jacques Sakarovitch. Recent results in the theory of rational sets. In *Mathematical foundations of computer science, 1986 (Bratislava, 1986)*, volume 233 of *Lecture Notes in Comput. Sci.*, pages 15–28. Springer, Berlin, 1986.
4. Marcin Bilkowski and Michał Skrzypczak. Unambiguity and uniformization problems on infinite trees. In *CSL*, volume 23 of *LIPIcs*, pages 81–100, 2013.
5. Mikołaj Bojańczyk and Slawomir Lasota. Fraenkel-Mostowski sets with non-homogeneous atoms. In *RP*, volume 7550, pages 1–5, 2012.
6. Michaël Cadilhac, Alain Finkel, and Pierre McKenzie. Unambiguous constrained automata. *Int. J. Found. Comput. Sci.*, 24(7):1099–1116, 2013.
7. Arnaud Carayol and Christof Löding. MSO on the infinite binary tree: Choice and order. In *CSL*, pages 161–176, 2007.
8. Arnaud Carayol, Christof Löding, Damian Niwiński, and Igor Walukiewicz. Choice functions and well-orderings over the infinite binary tree. *Central European Journal of Mathematics*, 8(4):662–682, 2010.
9. Olivier Carton and Max Michel. Unambiguous Büchi automata. *Theoretical Computer Science*, 297(1-3):37–81, 2003.
10. Thomas Colcombet. Forms of determinism for automata. In Christoph Dürr and Thomas Wilke, editors, *STACS 2012: 29th International Symposium on Theoretical Aspects of Computer Science*, volume 14 of *LIPIcs*, pages 1–23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2012.

11. Thomas Colcombet, Gabriele Puppis, and Michal Skrypczak. Unambiguous register automata. Unpublished.
12. François Coste and Daniel Fredouille. Unambiguous automata inference by means of state-merging methods. In *Machine Learning: ECML 2003, 14th European Conference on Machine Learning, Cavtat-Dubrovnik, Croatia, September 22-26, 2003, Proceedings*, pages 60–71, 2003.
13. Julien Cristau. Automata and temporal logic over arbitrary linear time. *CoRR*, abs/1101.1731, 2011.
14. M. Droste and D. Kuske. Weighted Automata. To appear in Handbook AutoMathA, 2013.
15. Jonathan Goldstine, Martin Kappes, Chandra M. R. Kintala, Hing Leug, Andreas Malcher, and Detlef Wotschke. Descriptional complexity of machines with limited resources. *Journal of Universal Computer Science*, 8:193–234, 2002.
16. Yuri Gurevich and Saharon Shelah. Rabin's uniformization problem. *J. Symb. Log.*, 48(4):1105–1119, 1983.
17. Markus Holzer and Martin Kutrib. Descriptional complexity of (un)ambiguous finite state machines and pushdown automata. In *Reachability Problems, 4th International Workshop, RP 2010, Brno, Czech Republic, August 28-29, 2010. Proceedings*, pages 1–23, 2010.
18. Michael Kaminski and Nissim Francez. Finite-memory automata. *Theoretical Computer Science*, 134(2):329–363, 1994.
19. Daniel Kirsten and Sylvain Lombardy. Deciding unambiguity and sequentiality of polynomially ambiguous min-plus automata. In *26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009, February 26-28, 2009, Freiburg, Germany, Proceedings*, pages 589–600, 2009.
20. Ines Klimann, Sylvain Lombardy, Jean Mairesse, and Christophe Prieur. Deciding unambiguity and sequentiality from a finitely ambiguous max-plus automaton. *Theor. Comput. Sci.*, 327(3):349–373, 2004.
21. Daniel Krob. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. *Internat. J. Algebra Comput.*, 4(3):405–425, 1994.
22. Ernst L. Leiss. Succint representation of regular languages by boolean automata. *Theor. Comput. Sci.*, 13:323–330, 1981.
23. Hing Leung. Separating exponentially ambiguous finite automata from polynomially ambiguous finite automata. *SIAM J. Comput.*, 27(4):1073–1082, 1998.
24. Hing Leung. Descriptional complexity of NFA of different ambiguity. *Int. J. Found. Comput. Sci.*, 16(5):975–984, 2005.
25. Hing Leung. Structurally unambiguous finite automata. In *Implementation and Application of Automata, 11th International Conference, CIAA 2006, Taipei, Taiwan, August 21-23, 2006, Proceedings*, pages 198–207, 2006.
26. Sylvain Lombardy and Jean Mairesse. Series which are both max-plus and min-plus are unambiguous. *RAIRO - Theor. Inf. Appl.*, 40(1):1–14, 2006.
27. Oleg B. Lupanov. A comparison of two types of finite sources. *Problemy Kybernetiki*, 9:321–326, 1963.
28. Albert R. Meyer and Michael J. Fischer. Economy of description by automata, grammars, and formal systems. In *Symposium on Switching and Automata Theory*, pages 188–191. IEEE, 1971.
29. F. R. Moore. On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic, and two-way finite automata. *IEEE Trans. Comput.*, 20(10):1211–1214, 1971.
30. Damian Niwiński and Igor Walukiewicz. Ambiguity problem for automata on infinite trees. unpublished, 1996.

31. Alexander Okhotin and Kai Salomaa. Descriptional complexity of unambiguous input-driven pushdown automata. *Theor. Comput. Sci.*, 566:1–11, 2015.
32. Jean-Éric Pin and Pascal Weil. Polynomial closure and unambiguous product. *Theory Comput. Syst.*, 30(4):383–422, 1997.
33. Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. Amer. Math. Soc.*, 141:1–35, 1969.
34. Michael O. Rabin and Dana Scott. Finite automata and their decision problems. *IBM J. Res. and Develop.*, 3:114–125, April 1959.
35. Klaus Reinhardt and Eric Allender. Making nondeterminism unambiguous. *SIAM J. Comput.*, 29(4):1118–1131, 2000.
36. Jacques Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.
37. Erik Meineche Schmidt. *Succinctness of descriptions of Context-Free, Regular and Finite Languages*. PhD thesis, Cornell University, 1977.
38. Marcel-Paul Schützenberger. On the definition of a family of automata. *Information and Control*, 4:245–270, 1961.
39. Marcel-Paul Schützenberger. Sur les relations fonctionnelles. In *Automata Theory and Formal Languages*, volume 33 of *LNCS*, pages 209–213, 1975.
40. Helmut Seidl. Deciding equivalence of finite tree automata. *SIAM J. Comput.*, 19(3):424–437, 1990.
41. Richard Edwin Stearns and Harry B. Hunt III. On the equivalence and containment problems for unambiguous regular expressions, grammars, and automata. In *22nd Annual Symposium on Foundations of Computer Science, Nashville, Tennessee, USA, 28-30 October 1981*, pages 74–81, 1981.
42. Wolfgang Thomas. Languages, automata and logic. In Gregorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, volume 3, chapter 7, pages 389–455. 1997.
43. Andreas Weber. Decomposing a k-valued transducer into k unambiguous ones. *ITA*, 30(5):379–413, 1996.