Uniformization in Automata Theory

Arnaud Carayol
Laboratoire d'Informatique Gaspard Monge,
Université Paris-Est & CNRS
arnaud.carayol@univ-mlv.fr

Christof Löding
RWTH Aachen, Informatik 7, Aachen, Germany
loeding@informatik.rwth-aachen.de

December 20, 2013

Abstract

We survey some classical results on uniformizations of automaton definable relations by automaton definable functions. We consider the case of automatic relations over finite and infinite words and trees as well as rational relations over finite and infinite words. We also provide some new results concerning the uniformization of automatic and rational relations over finite words by subsequential transducers. We show that it is undecidable whether a given rational relation can be uniformized by a subsequential transducer and provide a decision procedure for the case of automatic relations.

1 Introduction

A uniformization of a (binary) relation is a function that selects for each element in the domain of the relation a unique image that is in relation with this element. In other words, a uniformization of $R \subseteq X \times Y$ is a function $f_R : X \to Y$ with the same domain as R and whose graph is a subset of R. The origin of uniformization problems comes from set theory, where the complexity of a class of definable relations is related with the complexity of uniformizations for these relations (see (Moschovakis, 1980) for results of this kind).

The aim of this paper is to give an overview of some uniformization results in the setting where the relations and functions are defined by finite automata. In analogy to the problems studied in set theory, the uniformization question for a given class of relations defined by some automaton model is whether each such relation has a uniformization in the same class. A motivation for studying these kinds of questions in computer science arises when the relation describes a specification relating inputs to allowed outputs, for example for a program or a circuit, as in Church's synthesis problem (Church, 1962). A uniformization of such a specification can be seen as a concrete implementation conforming to the specification because it selects for each input exactly one allowed output. In this view, the uniformization question then asks whether each specification from a given class can implemented within the same class.

These uniformization questions have already been studied in the early times of automata theory. The first class considered is that of word relations defined by finite automata (Elgot & Mezei, 1965) also called rational relations. In (Kobayashi, 1969, Theorem 3), it is first shown that any relation accepted by a finite automaton admits a uniformization also accepted by a finite automaton. Several alternative and simplified proofs were published (Eilenberg, 1974; Arnold & Latteux, 1979; Choffrut & Grigorieff, 1999). The proof of (Arnold & Latteux, 1979) builds on a decomposition theorem for rational functions from (Elgot & Mezei, 1965) and shows that uniformization can be realised by the composition of a sequential (i.e., input deterministic) transducer working from left to right followed by one working from right to left. (Choffrut & Grigorieff, 1999) contains the idea of the reduction to the length-preserving case that we use in this article as well as a generalisation to the infinite word case. The uniformization problem for relations on trees defined by finite automata was only considered more recently (Kuske & Weidner, 2011; Colcombet & Löding, 2007) but can be traced back to (Engelfriet, 1978).

The decision problem corresponding to the uniformization question is to decide whether a given relation has a uniformization. We consider this question in a setting where the relation comes from one class $\mathcal{C}$ and we are looking for a uniformization in another class $\mathcal{C}'$ (which is more restrictive than $\mathcal{C}$ in our setting). An early decidability results in this spirit was provided by Büchi and Landweber in (Büchi & Landweber, 1969) for relations of infinite words given by synchronous two-tape Büchi automata and to be uniformized by synchronous deterministic sequential transducers, that is, deterministic transducers that produce one output symbol for each input symbol. Modern presentations of these results can be found, e.g., in (Thomas, 2008) and (Löding, 2011).

This decidability result has been studied for variations where the transducer defining the uniformization is allowed to skip a bounded number of output symbols in order to obtain a bounded look-ahead (Hosch & Landweber, 1972). In (Holtmann, Kaiser, & Thomas, 2010) the condition is relaxed further by allowing an arbitrary finite number of skips. However, as shown in (Holtmann et al., 2010), this case can be reduced to a bounded number of skips, where the bound depends on the size of the automaton defining the specification.

We study a similar setting in the case of finite words. A standard model for deterministically computing functions over finite words are subsequential transducers. These are basically deterministic finite automata that output a finite word on each transition. We show that it is decidable whether an automatic relation can be uniformized by a subsequential transducer. The setting is similar to the one mentioned above for infinite words studied in (Holtmann et al., 2010). However, in the setting of finite words the number of required skips (where the transducer outputs the empty word) cannot be bounded because the input and the output might be of different length. This adds some new phenomena compared to the setting of (Holtmann et al., 2010).

Furthermore, we show that it is undecidable whether a rational relation can be uniformized by a subsequential transducer.

The paper is structured as follows. In Section 2 we consider uniformization automatic relations over finite and infinite words and trees. Section 3 is about uniformization of rational relations over finite and infinite words. In Section 4 we present some new results on uniformization of rational and automatic relations over finite words by subsequential transducers.

We assume the reader to be familiar with basic notions from automata theory and only provide some definitions for fixing the terminology.

2 Automatic Relations on Words and Trees

In this section, we consider the class of relations that are definable by synchronous finite automata, that is, automata that process the input and the output at the same time and at the same speed. Such relations are referred to as automatic (Khoussainov & Nerode, 1995; Blumensath & Grädel, 2000). Automatic relations can be defined over finite words, infinite words, finite trees, and infinite trees. For these four classes, we study the question whether a given automatic relation always has an automatic uniformization. We only consider the case of binary relations; uniformization questions for automatic relations of higher arity can be reduced to the binary case (which is not the case for rational relations, see Section 3). To simplify notation, we usually assume that the alphabets for the two components are the same, which is not a restriction.

2.1 Finite Words

We start by considering automatic relations over finite words. As usual, a finite word is finite sequence of letters over an alphabet Σ, where an alphabet is just a finite set of symbols. The set of all finite words over Σ is denoted by Σ^* and the empty word by ε. The length of a word w is denoted by $|w|$.

We use the standard model of finite automata on finite words. To fix the notation, a nondeterministic finite automaton (NFA) is of the form $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$, where Q is a finite set of states, Σ is the input alphabet, $q_0 \in Q$ is the initial state, $\Delta \subseteq Q \times \Sigma \times Q$ is the transition relation, and $F \subseteq Q$ is the set of final (or accepting) states. A run of $\mathcal{A}$ on $w \in \Sigma^*$ is a sequence $p_0, \ldots, p_n$ of states such that $(p_i, a_i, p_{i+1}) \in \Delta$ for all $i \in \{0, \ldots, n-1\}$, where $w = a_0 \cdots a_{n-1}$. We write $\mathcal{A} : p_0 \xrightarrow{w} p_n$ to indicate that there is a run of $\mathcal{A}$ on w from p_0 to p_n. An accepting run is a run that starts in q_0 and ends in a state from F. The set of words that labels an accepting run of $\mathcal{A}$ is denoted by $L(\mathcal{A})$ and is called the language accepted by $\mathcal{A}$. The class of languages that can be accepted by NFAs is called the class of regular languages. An NFA is deterministic (a DFA) if for each $q \in Q$ and $a \in \Sigma$ there is at most one $q' \in Q$ with $(q, a, q') \in \Delta$. In this case, we usually write the transition relation as a (partial) function $\delta : Q \times \Sigma \to Q$. We refer the reader who is not familiar with the basic results on regular languages and finite automata to (Hopcroft & Ullman, 1979).

One way to make finite automata process tuples of words (for defining a relation) is to use a product alphabet such that the automaton processes one letter from each word in the tuple in one step. This leads to the notion of automatic relations defined more formally below. To handle the case of words of different length, a dummy symbol $\Box$ is used to pad shorter words.

Formally, for $w_1, w_2 \in \Sigma^*$ we define

$$w_1 \otimes w_2 = \begin{bmatrix} a'_{11} \\ a'_{21} \end{bmatrix} \cdots \begin{bmatrix} a'_{1n} \\ a'_{2n} \end{bmatrix} \in (\Sigma^2_\Box)^*$$

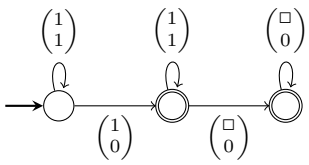

Figure 1: A finite automaton for the relation from Example 1.

where $\Sigma_\square = \Sigma \cup \{\square\}$, n is the maximal length of one of the words w_i, and a_{ij} is the jth letter of w_i if $j \leq |w_i|$ and $\square$ otherwise. A language $L \subseteq ((\Sigma \cup \{\square\})^2)^*$ defines a relation $R_L \subseteq (\Sigma^*)^2$ in the obvious way: $(w_1, w_2) \in R_L$ iff $w_1 \otimes w_2 \in L$. A relation $R \subseteq (\Sigma^*)^2 s$ is called automatic if $R = R_L$ for some regular language $L \subseteq ((\Sigma \cup \{\square\})^2)^*$. These definitions can easily be generalized to relations of higher arity, but as mentioned earlier, we restrict our considerations to the binary case.

Example 1 Consider the following relation of words over the alphabet $\{0, 1\}$:

$$R = \{(1^n, 1^m 0 1^k 0^*) \mid n = m + k + 1\}.$$

This relation is accepted by the DFA depicted in Figure 1 and hence, is an automatic relation (the initial state is marked by an incoming arrow and the accepting states by a double circle).

As mentioned above, we study in the section the question whether (binary) automatic relations have automatic uniformizations. A simple technique to define a uniformization is to fix a total well-ordering (a total ordering without infinite decreasing chains) on the elements of the domain (finite words in this case), and then to select for each word u the smallest word v such that u and v are in relation. One such well-ordering is the length-lexicographic ordering, which first orders words by their length and the words of the same length are ordered lexicographically according to some fixed order on the letters of the alphabet. We denote this ordering by $<_{\text{llex}}$. We observe that $<_{\text{llex}}$ is itself an automatic relation, and thus is a useful tool for uniformization.

For a relation $R \subseteq \Sigma^* \times \Sigma^*$ we define the length-lexicographic uniformization by

$$f_R(u) = \min_{<_{\text{llex}}} \{v \in \Sigma^* \mid (u, v) \in R\}$$

for all u in the domain of R. For the relation R from Example 1 we obtain $f_R(1^n) = 01^{n-1}$.

In general, one can always construct from the automaton for R a new automaton that checks for $u \otimes v$ whether $(u, v) \in R$ and at the same time verifies that there is no smaller word $v' <_{\text{llex}} v$ such that $(u, v') \in R$. This technique of selecting smallest representatives (according to some ordering) in a regular way goes back to Eilenberg's Cross-Section Theorem (Eilenberg, 1974). In the context of automatic relations, this technique is used with the ordering $<_{\text{llex}}$ in (Khoussainov & Nerode, 1995) to show that automatic equivalence relations have automatic cross-sections, which means that there is a regular set of representatives from the equivalence classes. The following theorem summarizes our

considerations. We do not attribute this theorem to a specific paper because its proof is rather simple and it can easily be derived in many ways from other results. An explicit statement of the result can be found in (Choffrut & Grigorieff, 1999).

THEOREM 2 *For an automatic relation R over finite words, the length-lexicographic uniformization of R is also automatic. In particular, each automatic relation has an automatic uniformization.*

2.2 Finite Trees

We now turn to the uniformization problem for automatic relations over finite trees. To define trees, we fix a ranked alphabet Σ, that is, each symbol in Σ has a rank (or arity), which is a natural number. A tree domain dom is a non-empty finite subset of $\mathbb{N}^*$ (the set of finite sequences of natural numbers) with the property that for each $x \in \mathbb{N}^*$ and $i \in \mathbb{N}$ with $x \cdot i \in dom$ we also have $x \in dom$ and $x \cdot j \in dom$ for each $j < i$. By using sequences of natural numbers, a tree domain is naturally equipped with a successor relation, namely $x \cdot i$ is a successor of x.

A (finite Σ-labeled) tree t is a mapping from a tree domain $dom(t)$ to the ranked alphabet Σ such that for each $x \in dom(t)$ the rank of $t(x)$ corresponds to the number of successors of x in $dom(t)$.

To define tree-automatic structures, we need a way to code tuples of finite trees, i.e., we need an operation $\otimes$ for finite trees similar to the one for words. For a tree $t : dom(t) \to \Sigma$ let $t^\Box : \mathbb{N}^* \to \Sigma_\Box$ be defined by $t^\Box(u) = t(u)$ if $u \in dom(t)$, and $t^\Box(u) = \Box$ otherwise. For finite Σ-labeled trees t_1, t_2, we define the $\Sigma^2_\Box$-labeled tree $t = t_1 \otimes t_2$ by $dom(t) = dom(t_1) \cup dom(t_2)$ and $t(u) = (t_1^\Box(u), t_2^\Box(u))$. When viewing words as unary trees, this definition corresponds to the operation $\otimes$ as defined for words. As in the case of words, a set T of finite $\Sigma^2_\Box$-labeled trees defines the relation R_T by $(t_1, t_2) \in R_T$ iff $t_1 \otimes t_2 \in T$.

We are not going to use any details on automata on finite trees. For the purpose of this paper, it is enough to know that there is a model of finite automata on finite trees that defines the class of regular tree languages whose closure and algorithmic properties are comparable to the class of regular word languages. We refer the reader to (Comon et al., 2007) or (Löding, 2012) for an introduction to automata on finite trees.

We call a (binary) relation R over the domain of finite Σ-labeled trees tree-automatic if $R = R_T$ for some regular language T of finite $\Sigma^2_\Box$-labeled trees.

A crucial difference to the setting of finite words is that there is no tree-automatic total well-ordering over the set of all finite Σ-labeled trees. This fact can be deduced from Theorem 6 presented in Section 2.4. Hence, the technique used for the uniformization of automatic relations cannot be extended to finite trees. However, a result from (Kuske & Weidner, 2011) (based on (Colcombet & Löding, 2007)) shows that tree-automatic equivalence relations have regular cross-sections (that is, given a tree-automatic equivalence relation $\sim$, it is possible to construct a tree automaton accepting a language that contains exactly one tree from each equivalence class of $\sim$). We can use this result to obtain a tree-automatic uniformization of an arbitrary (binary) tree-automatic relation R as follows. We define an equivalence relation $\sim_R$ over the set of finite $\Sigma^2_\Box$-labeled trees by $t_1 \otimes t_2 \sim_R t_3 \otimes t_4$ if $(t_1, t_2), (t_3, t_4) \in R$ and $t_1 = t_3$.

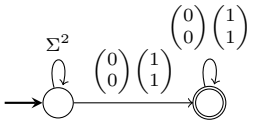

Figure 2: A Büchi automaton for the relation from Example 4.

Then $\sim_R$ is a tree-automatic equivalence relation and we obtain a regular tree language T of finite $\Sigma^2_\square$-labeled trees that contains exactly one representative from each equivalence class of $\sim_R$. The relation R_T defines a uniformization of R because for each possible first component t in the domain of R it contains exactly one pair (t, t').

The uniformization result for tree-automatic relations can also be deduced from (the proof of) a result in (Engelfriet, 1978) that shows that relations computed by nondeterministic top-down tree transducers can be uniformized by deterministic top-down tree transducers with regular look-ahead. The constructed transducer basically chooses at each node for a given input letter the least possible transition (according to some fixed ordering on the transitions) that admits a successful run on the remaining input. The regular look-ahead is used to check this property for the chosen transition. In our setting, the regular look-ahead could be simulated by nondeterministically guessing a transition and then verifying that all smaller transitions would not admit a successful computation on the remaining input.

THEOREM 3 *Every tree-automatic relation has a tree automatic uniformization.*

2.3 Infinite Words

An infinite word α over an alphabet Σ corresponds to a function $\alpha : \mathbb{N} \to \Sigma$, which naturally defines an infinite ordered sequence of letters. We denote the set of infinite words (also called ω-words) over Σ by Σ^ω.

Automatic relations over infinite words, called ω-automatic relations, are defined in the same way as for finite words using Büchi automata instead of standard finite automata (the definition is even simpler because for infinite words no padding is required). A Büchi automaton is given in the same way as an NFA. It accepts an infinite word if there is a run on this word that starts in the initial state and infinitely often visits an accepting state. For an introduction to the theory of automata on infinite words, we refer the reader to (Thomas, 1997).

EXAMPLE 4 Let Σ be some alphabet and consider the relation $R_\approx$ of ultimately equal words, that is,

$$R_\approx := \{(\alpha, \beta) \mid \exists u, v \in \Sigma^*, \gamma \in \Sigma^\omega : |u| = |v| \text{ and } \alpha = u\gamma \text{ and } \beta = v\gamma\}.$$

This is an ω-automatic equivalence relation that is accepted by the Büchi automaton depicted in Figure 2 for $\Sigma = \{0, 1\}$.

The techniques for uniformization of relations over finite words and trees presented above are both based on the regular cross-section property for (tree-)automatic equivalence relations. For infinite words, a corresponding result does not hold. As pointed out in (Kuske & Lohrey, 2006), the equivalence relation $R_{\approx}$ from Example 4 does not have the regular cross-section property (i.e., does admit a set of representatives that can be accepted by a Büchi automaton). This easily follows from the fact that each Büchi automaton that accepts infinitely many ω-words also accepts two different words that are ultimately equal.

However, uniformization of ω-automatic relations is still possible because selecting representatives from equivalence classes is a stronger requirement than selecting unique images for all elements in the domain of a relation (see also the reduction in Section 3.2). The uniformization result for ω-automatic relations was already obtained in (Siefkes, 1975). The technique that we present here is taken from (Choffrut & Grigorieff, 1999).

THEOREM 5 ((SIEFKES, 1975; CHOFFRUT & GRIGORIEFF, 1999)) *Every ω-automatic relation has an ω-automatic uniformization.*

Proof. The key idea for the construction is to take the accepting runs of an automaton for the relation as additional information for selecting an image. Given a Büchi automaton $\mathcal{A} = (Q, \Sigma^2, q_0, \Delta, F)$ recognizing $R \subseteq \Sigma^\omega \times \Sigma^\omega$, we consider an extended relation $R' \subseteq \Sigma^\omega \times \Sigma^\omega \times Q^\omega$ containing those tuples (α, β, ρ) such that $(\alpha, \beta) \in R$ and ρ is an accepting run of $\mathcal{A}$ on $\alpha \otimes \beta$.

Our aim is to define an ordering on the pairs (β, ρ) such that an automaton can select the smallest such pair for a given α. For ρ, we consider the unique sequence $i_1 < i_2 < \cdots$ of position at which ρ is in an accepting state, that is, with $\rho(i_j) \in F$.

This induces a factorization of the combined word $\beta \otimes \rho$ into finite segments $\beta_j \otimes \rho_j$ for $j \geq 1$ where each β_j is the segment of β from $i_{j-1} + 1$ to i_j and similarly for ρ (for the initial segment to be well defined, we set $i_0 = -1$). Note that the segments ρ_j all contain exactly one accepting state, namely at the last position.

Given another pair β' and ρ' with corresponding sequence $i'_1 < i'_2 < \cdots$, we obtain the factors $\beta'_j \otimes \rho'_j$. Now pick the first j such that $\beta_j \otimes \rho_j \neq \beta'_j \otimes \rho'_j$. We define $(\beta, \rho) < (\beta', \rho')$ if $(\beta_j, \rho_j) <_{\text{llex}} (\beta'_j, \rho'_j)$ for this j, where $<_{\text{llex}}$ refers to some fixed ordering on the set $\Sigma \times Q$.

One can verify that there is a Büchi automaton that accepts precisely those pairs $(\alpha, \beta) \in R$ such that there is ρ with $(\alpha, \beta, \rho) \in R'$ and for all $(\alpha, \beta', \rho') \in R'$ one has $(\beta, \rho) \leq (\beta', \rho')$.

It remains to verify that for each α in the domain of R such a minimal pair (β, ρ) exists. Such a minimal pair can be constructed for a given α by inductively defining a sequence of segments $\beta_j \otimes \rho_j$ as follows. The segment $\beta_j \otimes \rho_j$ is the $<_{\text{llex}}$-minimal segment with the property that ρ_j is in $(Q \setminus F)^* F$ and that the concatenation $\beta_0 \otimes \rho_0 \cdots \beta_j \otimes \rho_j$ can be extended to a sequence $\beta \otimes \rho$ such that $(\alpha, \beta, \rho) \in R'$. Taking the limit sequences $\beta_1 \beta_2 \cdots$ and $\rho_1 \rho_2 \cdots$ results in a pair with the desired properties. □

2.4 Infinite Trees

The theory of automata on infinite words can be generalized to automata on infinite trees. For simplicity, we only consider Σ-labeled complete binary trees, which are mappings $t : \{0,1\}^* \to \Sigma$. As for automata on finite trees, we do not detail the model here because it is not required for our considerations. We refer the reader to (Thomas, 1997) for the basics on this model.

The definitions for ω-tree-automatic structures are a straightforward generalization of the definitions for the other classes of automatic structures in this section. However, it is the only class of automatic structures that does not admit uniformization.

This result goes back to (Gurevich & Shelah, 1983) where it is shown that there is no choice function over the infinite binary tree that is definable in monadic second-order logic (MSO). In this setting, we view the (unlabeled) infinite binary tree as a structure $T_2 = (\{0,1\}^*, S_0, S_1)$ with universe $\{0,1\}^*$ and two successor relations S_0 and S_1 with the natural interpretation (S_0 corresponds to appending a 0 and S_1 to appending a 1).

As usual, monadic second-order logic is the extension of first-order logic by set quantifiers. An MSO definable choice function would be given by an MSO formula $\varphi(X, y)$ with one free set variable X and one free element variable y such that for each nonempty subset $U \subseteq \{0,1\}^*$ there is exactly one element $u \in U$ such that $T_2 \models \varphi[U, u]$.

THEOREM 6 ((GUREVICH & SHELAH, 1983)) *There is no MSO-definable choice function over the infinite binary tree.*

While the proof given in (Gurevich & Shelah, 1983) uses set-theoretic tools, more recently a new proof only based on automata-theoretic methods has been given in (Carayol & Löding, 2007; Carayol, Löding, Niwiński, & Walukiewicz, 2010). From this new proof, one can even derive a simple family of counter examples in the following sense. For $N \in \mathbb{N}$, define the set $U_N \subseteq \{0,1\}^*$ as

$$U_N := \{0,1\}^*(0^N 0^* 1)^N.$$

It turns out that this simple family of sets is sufficient to show that there is no MSO definable choice function over T_2.

THEOREM 7 ((CARAYOL & LÖDING, 2007; CARAYOL ET AL., 2010)) *For each MSO formula $\varphi(X, y)$ over the infinite binary tree, there exists N such that φ fails to choose a unique element from U_N.*

Using the tight connection between MSO and finite automata (see (Thomas, 1997)), one can rephrase the result from (Gurevich & Shelah, 1983) in automata theoretic terms. For this purpose, we define the relation $R_\in$ over infinite $\{0,1\}$-labeled trees as follows: $(t, t') \in R$ if

- there is exactly one $u \in \{0,1\}^*$ with $t'(u) = 1$, and
- $t(u) = 1$ for the unique $u \in \{0,1\}^*$ with $t'(u) = 1$.

Intuitively, the relation $R_\in$ corresponds to the element relation because the tree t represents a non-empty set U (via the 1-labeled nodes), t' corresponds to an

element u (via the unique 1-labeled node), and the last condition states that $u \in U$. An automaton defining a uniformization of $R_\in$ could be turned into an MSO formula defining a choice function, which is not possible according to Theorem 6.

COROLLARY 8 *There are ω-tree-automatic relations that do not have an ω-tree-automatic uniformization. In particular, the relation $R_\in$ does not have an ω-tree-automatic uniformization.*

3 Rational Word Relations

In this section, we consider the uniformization of relations on finite and infinite words accepted by asynchronous automata. These relation are called rational relations (as for finite words, they are the rational subsets of the product monoid). Following (Choffrut & Grigorieff, 1999), we will see that these problems can be reduced to the automatic case using a standard decomposition theorem (Eilenberg, 1974). The generalization of rational (word) relations to trees is unclear and several notions of asynchronous tree automata models have been proposed (see for instance (Comon et al., 2007, Chapter 6) as well as the discussion in (Raoult, 1997)). The uniformization problem for these various classes goes beyond the scope of this article.

3.1 Finite words

As in Section 2, we consider the case of binary relations where (most of the time) we assume the input and the output alphabet to be the same. The latter assumption is easily seen not to be a restriction. We comment on the case of higher arity at the end of this section.

In the case of finite words, a rational relation $R \subseteq (\Sigma^*)^2$ is accepted by a finite automaton which can read its two tapes (also called input and output tape in the binary setting) in an asynchronous fashion. Formally, such an automaton has its transitions labeled by elements of $(\Sigma \cup \{\varepsilon\})^2$. The language $L \subseteq ((\Sigma \cup \{\varepsilon\})^2)^*$ accepted by the automaton defines the relation $\{(\pi_1(w), \pi_2(w)) \mid w \in L\}$ where π_i is the morphism from $(\Sigma \cup \{\varepsilon\})^2$ to Σ^* corresponding to the projection over the i-th component.

EXAMPLE 9 Consider the rational relational over the alphabet $\{0, 1, 2\}$ taken from (Sakarovitch, 2009, Example 3.1,p. 687)

$$R = \{(0^m 1^n, 0^m 1) \mid m, n \geq 1\} \cup \{(0^m 1^n, 0^n 2) \mid m, n \geq 1\}.$$

This relation is accepted by the automaton of Figure 3, where we use the standard notation x/y for pairs of input $x \in \Sigma \cup \{\varepsilon\}$ and output $y \in \Sigma \cup \{\varepsilon\}$ processed by a transition.

From the definitions, it is clear that automatic relations are special cases of rational relations. Furthermore, a word morphism which replaces every letter by a given word is an example of rational function. More generally, rational substitutions in which a letter is replaced by a word chosen from a regular language also define rational relations. As an example for such a rational substitution

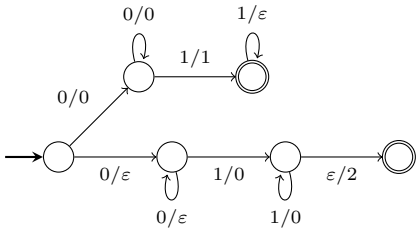

Figure 3: A transducer for the relation from Example 9.

over the alphabet $\{0,1\}$, consider the mapping given by $0 \mapsto 0^+$ and $1 \mapsto 0^*10^*$. This defines a relation that relates a non empty word w to any word obtained by inserting 0s at arbitrary positions in w, which is easily seen to be rational. In general, using automata for the languages in the images of the substitution one can easily build an asynchronous automaton for the corresponding relation.

Contrarily to the case of automatic relations, the length-lexicographic uniformization of a rational relation is not in general a rational function. Consider for instance the relation R of Example 9. By ordering the alphabet in the natural way (i.e., $0 < 1 < 2$), the length-lexicographic uniformization of R is the function S defined for all $m, n \geq 1$, by $S(0^m1^n) = 0^m1$ if $m \leq n$ and $S(0^m1^n) = 0^n2$ otherwise. The function S is not a rational function as for instance the inverse image by S of the regular set 0^+1 is the non-regular set $\{0^m1^n \mid m \leq n\}$.

Remark 10 In (Lombardy & Sakarovitch, 2010), it is shown that the length-lexicographic uniformization of a rational relation in $\Sigma^* \times \Gamma^*$ with $|\Sigma| = 1$ is a rational function.

Uniformization of rational relations is reduced to the automatic case using the following decomposition theorem which states that any rational relation whose domain does not contain the empty word can be expressed as the composition of an automatic relation followed by a rational substitution.

Theorem 11 ((Eilenberg, 1974)) *For any rational relation R over an alphabet Σ whose domain does not contain the empty word, there exists an alphabet Γ, a length-preserving automatic relation $S \subseteq \Sigma^* \times \Gamma^*$ and a rational substitution $\rho \subseteq \Gamma^* \times \Sigma^*$ with $dom(\rho) = \Gamma^*$ such that:*

$$R = S \circ \rho := \{(u, v) \in \Sigma^* \times \Sigma^* \mid \exists w \in \Gamma^* : (u, w) \in S \text{ and } (w, v) \in \rho\}.$$

Proof. Let R be a rational relation over Σ whose domain does not contain the empty word. Consider an automaton $A = (Q, (\Sigma \cup \{\varepsilon\})^2, q_0, \Delta, F)$ accepting R. For all states p and $q \in Q$, we let $A_{p,q}$ denote the automata $(Q, (\Sigma \cup \{\varepsilon\})^2, p, \Delta, \{q\})$ where p is the new initial state and q the unique final state.

Let Γ be the alphabet $Q \times \Sigma \times Q$. Consider the length-preserving automatic relation $S \subseteq \Sigma^* \times \Gamma^*$ associating to a word $a_1 \cdots a_n$ with $n \geq 1$ any word $(q_0, a_1, q_1) \cdots (q_{n-1}, a_n, q_n)$ such that q_n belongs to F. The rational substitution ρ associates to (p, a, q) the image of the letter a by the relation accepted by $A_{p,q}$. It can be shown that $R = S \circ \rho$.

To guarantee that $dom(\rho) = \Gamma^*$, it is enough to ensure that for all $a \in \Gamma$, the language $\rho(a)$ is non empty. If it is not the case, we restrict ρ to the alphabet $\Xi = \{a \in \Gamma \mid \rho(a) \neq \emptyset\}$ and we restrict the image of S to Ξ^*. □

REMARK 12 As a consequence of Theorem 11, we re-obtain the well-known result stating that all rational functions are unambiguous (i.e., accepted by an automaton having at most one accepting run for each pair of the relation). Indeed it shows that any rational function is the composition of length-preserving automatic function and of a morphism. A length-preserving automatic function being accepted by a DFA labeled by $\Sigma \times \Sigma$ is an unambiguous rational function. The unambiguity is easily shown to be preserved in the composition with a morphism.

In conjunction with the uniformization result for automatic relations, we obtain the uniformization theorem for rational relations.

THEOREM 13 *Rational relations can be uniformized by rational functions.*

Proof. Let R be a rational relation. It is enough to consider the case where the domain of R does not contain the empty word.[1] By Theorem 11, there exists an alphabet Γ such that R can be expressed as $S \circ \rho$ where $S \subseteq \Sigma^* \times \Gamma^*$ is a length-preserving automatic relation and a rational substitution ρ with $dom(\rho) = \Gamma^*$ (and hence $dom(R) = dom(S)$).

By Theorem 2, S admits a length-preserving automatic uniformization T. Furthermore the rational substitution ρ is uniformized by a morphism φ such that for all $a \in \Gamma$, $\varphi(a) \in \rho(a)$. We have $T \circ \varphi \subseteq R$ and as $dom(\varphi) = \Gamma^*$, $dom(T \circ \varphi) = dom(T) = dom(S) = dom(R)$. □

Contrary to the case of automatic relations, the uniformization theorem cannot be extended to arity greater than 2. Consider for instance the following rational relation in $\{a, b\}^* \times \{a\}^* \times \{a\}^*$

$$\{(a^n b^m, a^n, a) \mid n, m \geq 0\} \cup \{(a^n b^m, a^m, \varepsilon) \mid m, n \geq 0\}$$

In (Choffrut & Grigorieff, 1999, p. 7), it is shown by a pumping argument that this relation does admit any rational uniformization $f : \{a, b\}^* \times \{a\}^* \to \{a^*\}$. It is furthermore established that the class of rational relations in $\Gamma_1^* \times \cdots \times \Gamma_n^*$ for $n > 2$ enjoys the rational uniformization property if and only if all the Γ_i are unary alphabets. The converse implication follows from the fact that these relations which are essentially subsets of $\mathbb{N}^n$ are those relations definable in Presburger arithmetic. Their length-lexicographic uniformization is also definable by a Presburger formula and hence realizable by a rational relation.

Another notable difference with the automatic setting is that it is not known whether rational equivalence relations admit rational cross-sections (Johnson, 1986). Rational equivalence relations are more difficult to apprehend than their automatic counter-parts. For instance, it is undecidable whether a given rational relation is an equivalence relation (Johnson, 1986).

[1] The rational relation R can be decomposed into $R' \cup \{\varepsilon\} \times L$ where R' is a rational relation whose domain does not contain the empty word and L is a regular language.

3.2 Infinite words

Rational relations over infinite words are defined in the same way as for finite words using Büchi automata instead of standard finite automata. These relations are called ω-rational relations. For simplicity, we only consider rational relations whose domain only contains infinite words.

Using the same construction as in the finite word case, we can decompose an ω-rational relation into an ω-automatic relation followed by a rational substitution.

THEOREM 14 ((CHOFFRUT & GRIGORIEFF, 1999)) *For any ω-rational relation R over an alphabet Σ whose domain only contains infinite words, there exists an alphabet Γ, an ω-automatic relation $S \subseteq \Sigma^\omega \times \Gamma^\omega$ and a rational substitution $\rho \subseteq \Gamma^\omega \times \Sigma^\omega$ with $dom(\rho) = \Gamma^\omega$ such that $R = S \circ \rho$.*

As in the finite word case, the uniformization theorem for ω-rational relations follows from that of ω-automatic relations.

THEOREM 15 ((CHOFFRUT & GRIGORIEFF, 1999)) *ω-rational relations can be uniformized by ω-rational functions.*

4 Uniformization by Sequential Transducers

While in the previous section we considered the question whether relations from a given class have a uniformization within the same class, we now consider a setting in which the class of functions to choose the uniformization from is restricted.

As already mentioned, a uniformization of a relation can be viewed as a concrete implementation of a specification. The specification describes the admissible outputs for a given input, and the uniformization function selects one output for each input. In this section, we consider the setting where the output has to be constructed deterministically by a finite state device that reads the input letter by letter and can output finite words in each step. In the classical setting, this problem has been studied for infinite words, going back to a problem posed by Church (Church, 1962) that has been solved by Büchi and Landweber in (Büchi & Landweber, 1969) (see (Thomas, 2009) for a recent overview on this subject). For the setting of finite words, we use a standard transducer model, which basically can be seen as the subclass of asynchronous automata from Section 3 which are deterministic on their input.

A subsequential[2] transducer (ST) is of the form $\mathcal{T} = (S, \Sigma, \Gamma, s_0, \delta, F, f)$ where S is the finite set of states, Σ and Γ are the input and output alphabet, respectively, $s_0 \in S$ is the initial state, $\delta : S \times \Sigma \to S \times \Gamma^*$ is the transition function, $F \subseteq S$ is a set of final states, and $f : F \to \Gamma^*$ is the final output function. Such an ST behaves as a standard deterministic finite automaton but additionally produces a finite (possibly empty) output word in each transition. An input $u \in \Sigma^*$ is accepted if $s \in F$ for the state s reached after reading u. By $\mathcal{T}(u)$, we denote the output of $\mathcal{T}$ produced along the transitions while reading

[2]The prefix "sub" is added for transducers that can make a final output depending on the last state reached in a run, as opposed to sequential transducers that can only produce outputs on their transitions.

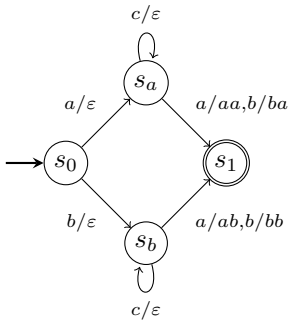

Figure 4: A subsequential transducer.

u (independent of whether u is accepted or not), and by $\mathcal{T}_f(u) := \mathcal{T}(u) \cdot f(s)$ the complete output including the final one at the last state s (if $s \notin F$, then $\mathcal{T}_f(u)$ is undefined). For a detailed introduction to this subject and functions definable by STs, we refer the reader to (Berstel, 1979).

Example 16 Figure 4 shows the transition graph of an ST $\mathcal{T}$ (where the notation $s \xrightarrow{a/u} s'$ denotes $\delta(s,a) = (s',u)$). We define the final output to be ε for the only final state s_1. Furthermore, we assume that the missing transitions lead to a rejecting sink state. The function defined by $\mathcal{T}$ has the domain $\{a,b\}c^*\{a,b\}$ with $\mathcal{T}(xc^*y) = \mathcal{T}_f(xc^*y) = yx$ for $x, y \in \{a,b\}$.

While it is decidable whether a rational function can be defined by an ST (see (Berstel, 1979, Theorem 6.2)), our first result shows uniformization of rational relations by STs is undecidable.

Theorem 17 *It is undecidable whether a given rational relation has a uniformization by a subsequential transducer.*

Proof. We sketch a reduction from the halting problem for Turing machines (TM). Given such a TM M, we describe a rational relation R_M. The interesting cases are those pairs (u,v) in which the first component is of the form

$$u = c_1\$c_2\$\cdots\$c_n\#^*X$$

where \$ and # are special symbols in the alphabet, each c_i is a configuration of M (coded as a word in a standard way), c_1 is the initial configuration of M on the empty tape, c_n is a halting configuration of M, and $X \in \{A,B\}$ is a letter that determines how the word in the second component has to look like. If u is of this form, we say that it codes a configuration sequence.

If u is does not code a configuration sequence, then every word v with $|v| = |u|$ is allowed in the second component. If u codes a configuration sequence and is ending in A, then $(u,v) \in R_M$ if, and only if, $u = v$. If u codes a configuration sequence and is ending in B, then $(u,v) \in R_M$ if, and only if, v is of the form

$$v = c_1'\$c_2'\$\cdots\$c_n'\#^*B$$

such that c_{i+1}' is not the successor configuration of c_i for some $i \in \{1,\ldots,n-1\}$.

First note that this defines a rational relation. An asynchronous automaton can guess at the beginning whether u codes a configuration sequence, and whether it ends in A or B. If it ends in A, then the automaton synchronously checks whether $u = v$, and if it ends in B, then the automaton guesses a c_i and asynchronously verifies that c'_{i+1} is not the successor configuration of c_i. To check this it advances to the next configuration in the output and compares $c'_{i+1}$and c_i.

We claim that R_M can be uniformized by an ST if, and only if, M does not halt. If M does not halt, then the ST that simply reproduces the input is a uniformization of R_M for the following reason: The only case to verify is the one where u codes a configuration sequence and ends in B. But since M does not halt, the configuration sequence in u must contain two configurations c_i and c_{i+1} such that c_{i+1} is not the successor configuration of c_i. Since $c'_{i+1} = c_{i+1}$, the condition is satisfied.

Now assume that M does halt and that there is an ST $\mathcal{T}$ that uniformizes R_M. Let k be the maximal length of an output string on the transitions of $\mathcal{T}$. Now consider an input u that codes the halting configuration sequence, followed by k times $\#$. Since the next input letter could be an A, $\mathcal{T}$ must have already reproduced the configuration sequence because it can produce at most k output letters on the last transition. But then the output does not satisfy the condition if the next input letter is B. □

The rational relation constructed in the reduction in the proof of Theorem 17 is not automatic, and in fact one can show that the problem becomes decidable when restricted to automatic relations.

THEOREM 18 *It is decidable whether a given automatic relation has a uniformization by a subsequential transducer.*

The proof of this result uses techniques similar to (Holtmann et al., 2010) where a similar problem on infinite words is studied: Given an automatic relation R of infinite words, decide if there is a sequential transducer such that for each input α in the domain of R, the computed output β is such that $(\alpha, \beta) \in R$.

At first glance, it might seem that this problem is more general because it is studied in the setting of infinite words, and finite words can be coded by infinite words using a dummy letter that is appended to the finite word. However, in the setting studied in (Holtmann et al., 2010), it is assumed that the sequential transducer produces an infinite output for each possible input. As a consequence, one can show that if there is a uniformization by a sequential transducer, then there is one of bounded delay, which means that the difference between the length of the processed input and the produced output is globally bounded.

In the setting of finite words that we study, this needs not to be true. Consider the following relation over the alphabet $\{a, b, c\}$, specified slightly informally using pairs of regular expressions, representing the Cartesian product of the respective languages:

$$(ac^*b, bc^*a) \cup (bc^*a, ac^*b) \cup (ac^*a, ac^*a) \cup (bc^*b, bc^*b) \ .$$

It is not difficult to verify that this is an automatic relation. It is uniformized by the ST from Example 16. This ST has arbitrarily long delays between input

and output, and this cannot be avoided because the first letter of the output depends on the last letter of the input.

However, the key insight for the decidability proof is that if such a long delay is necessary, then the connection between the remaining input and output cannot be very complex. This intuition of not being very complex is captured by the notion of recognizable relation. A relation $R \subseteq \Sigma^* \times \Gamma^*$ is called recognizable if it is of the form

$$R = \bigcup_{i=1}^{n} (U_i \times V_i)$$

for regular sets $U_i \subseteq \Sigma^*$ and $V_i \subseteq \Gamma^*$. From the definition, it is obvious that the above example is a recognizable relation. It is easy to verify that a relation R is recognizable if, and only if, there is a finite automaton that reads two words u and v sequentially (e.g., as $u\$v$ separated by a unique marker \$), and accepts if $(u, v) \in R$. It directly follows that a recognizable relation can be uniformized by an ST that first scans the entire input and then outputs some word in the matching set of output words (as the one in Figure 4 does).

Before making the link between large delays in the output and recognizable relations, we need some notations. For the remainder of this section, fix an automatic relation $R \subseteq \Sigma^* \times \Gamma^*$ recognized by a DFA $\mathcal{A} = (Q, (\Sigma \cup \{\Box\}) \times (\Gamma \cup \{\Box\}), q_0, \delta, F)$. For simplicity, we assume that $dom(R) = \Sigma^*$. Since $dom(R)$ is a regular language and an ST can test membership in regular languages, this assumption is not a restriction.

For $u \in \Sigma^*$ and $q \in Q$, let

$$R_q^u := \{(ux, v) \mid \mathcal{A} : q \xrightarrow{(ux)\otimes v} F\}$$

be the set of pairs that are accepted from q and where the input component starts with u. For R_q^ε we write R_q, and for $R_{q_0}^u$ we write R^u.

Our aim is to show that if R can be uniformized by an ST, then there is a bound on the delay between the input and the output, or the remaining relation can be uniformized by a recognizable one. This bound will be chosen such that an input word of this length contains some idempotent factor w.r.t. to some monoid structure that we define in the following.

For each input word u, we are interested in the types of behavior of $\mathcal{A}$ that can be induced by the input u together with some output (of same or smaller length). Hence, for each $v \in \Gamma^*$ with $|u| \geq |v|$, we consider the function (also called state transformation) $\tau_{u,v} : Q \to Q$ defined by $\tau_{u,v}(q) = p$ if $\mathcal{A} : q \xrightarrow{u\otimes v} p$.

The profile P_u of a word u contains the set of possible state transformations for the different types of words in the second component (of same length, shorter, or empty). More formally, $P_u = (P_{u,=}, P_{u,<}, P_{u,\varepsilon})$ with

- $P_{u,=} := \{\tau_{u,v} \mid v \in \Gamma^* \text{ and } |v| = |u|\}$,
- $P_{u,<} := \{\tau_{u,v} \mid v \in \Gamma^+ \text{ and } |v| < |u|\}$,
- $P_{u,\varepsilon} := \{\tau_{u,v} \mid v = \varepsilon\}$ (this set only contains one function but for consistency of notation, we prefer this definition).

It is not difficult to see that from the profiles of two words u and u' one can compute the profile of uu'. Hence, the set of profiles is naturally equipped with

a concatenation operation and a neutral element (the profile of the empty word), and the mapping that assigns the profile to a word u is a morphism from Σ^* to the profile monoid. A word u is called idempotent if $P_u = P_{uu}$, that is, if the corresponding monoid element is idempotent.

A consequence of Ramsey's Theorem (see (Diestel, 2000)) is that there is some $K \in \mathbb{N}$ such that all words $u \in \Sigma^*$ with $|u| \geq K$ contain an idempotent factor. With this in mind, the following lemma is a technical statement formalizing the intuition between long output delays and recognizable relations.

LEMMA 19 *Let $q \in Q$ and $u, v \in \Sigma^+$ with v idempotent. If R_q^{uv} is uniformized by an ST $\mathcal{T}$ such that $|\mathcal{T}(uv^n)| \leq |u|$ for all $n \in \mathbb{N}$, then R_q^{uv} can be uniformized by a recognizable relation.*

Proof. Consider an arbitrary word $w \in \Sigma^*$. We show that there is $x \in \Gamma^*$ whose length only depends on u and v such that $(uvw, x) \in R_q^{uv}$. For fixed u and v, there are only finitely many such words x and thus, R_q^{uv} can be uniformized by a recognizable relation (which can be shown using the technique from the proof of Proposition 20 below).

Since $|\mathcal{T}(uv^n)| \leq |u|$ for each n, the length of $\mathcal{T}(uv^n w)$ is independent of n for large n. We can thus choose n such that $|\mathcal{T}(uv^n w)| \leq |uv^n|$. Let $y = \mathcal{T}(uv^n w)$. We now consider the factorization $y = y'y''$ such that $|y'| = |u|$. Since v is idempotent, $P_v = P_{v^n}$, and thus there is some $z \in \Gamma^*$ with $|z| \leq |v|$ such that $\tau_{v,z} = \tau_{v^n,y''}$. Letting $x = y'z$, we obtain that the pair (uvw, x) induces the same state transformation on $\mathcal{A}$ as $(uv^n w, y)$ and hence $(uvw, x) \in R_q^{uv}$. □

Lemma 19 basically shows us that we can focus on the construction of STs in which the output delay is bounded. Once the output delay has to be larger than this bound, the uniformization task is either impossible or very simple (reduced to a recognizable relation). The following proposition shows that we can decide in which cases uniformization by a recognizable relation is possible.

PROPOSITION 20 *It is decidable whether an automatic relation can be uniformized by a recognizable relation.*

Proof. Let $R \subseteq \Sigma^* \times \Gamma^*$ be an automatic relation and let

$$V := \{v \in \Gamma^* \mid \exists u \in \Sigma^* : (u, v) \in R \text{ and } \forall v' \in \Gamma^* : (u, v') \in R \to v \leq_{\text{llex}} v'\}$$

be the set of words that are the length-lexicographically least output for some input. It is easy to verify that V is a regular set (an automaton for V can be constructed from an automaton for R using the closure properties of finite automata). We claim that R can be uniformized by a recognizable relation if, and only if, V is finite (and since V is regular, finiteness of V can be decided).

If V is finite, then for each $v \in V$, let U_v consist of those words $u \in \Sigma^*$ such that v is the length-lexicographically minimal word with $(u, v) \in R$. Then U_v is regular and $\bigcup_{v \in V} U_v \times \{v\}$ is a recognizable uniformization of R.

If $\bigcup_{i=1}^{n}(U_i \times \{v_i\})$ is a uniformization of R by a recognizable relation (in fact, a recognizable function), then consider the set

$$W := \{w \in \Gamma^* \mid w \leq_{\text{llex}} v_i \text{ for some } i \in \{1, \ldots, n\}\}.$$

Since $\leq_{\text{llex}}$ is a well-ordering, the set W is finite. Thus, V must also be finite because $V \subseteq W$. □

We are now ready to describe the decision procedure. Similar to (Holtmann et al., 2010), we consider a game between two players Input (In) and Output (Out). The game is played on a game graph such that In plays an input symbol and Out can react with a finite (possibly empty) sequence of output symbols. Player In wins the game if for the input sequence u that he has played so far, Out cannot extend her current output sequence such that the resulting pair is in R. Our goal is to construct the game graph such that an ST uniformizing R can be obtained from a winning strategy of Out.

The vertices of the game graph keep track of the current state of $\mathcal{A}$ on the combined part of the input and output, and possibly of the part of the input that is currently ahead. Making use of Lemma 19 (see the proof of Lemma 21 below), we can restrict the game to situations in which the input is ahead at most $2K$ steps (where K is such that words of length at least K contain an idempotent factor, see the description before Lemma 19). It turns out that we do not have to consider the case in which the output is ahead.

The game graph $G_{\mathcal{A}}^K$ consists of vertices for In and Out and a set of edges corresponding to the possible moves of the two players:

- $V_{\mathsf{In}} := \{(q, u) \in Q \times \Sigma^* \mid |u| \leq 2K)\}$ is the set of vertices of player In.
- $V_{\mathsf{Out}} := V_{\mathsf{In}} \times \{\mathsf{Out}\}$ is the set of vertices of player Out.
- From a vertex of In, the following moves are possible:
  - $(q, u) \xrightarrow{a} (q, ua, \mathsf{Out})$ if $|u| < 2K$ and $a \in \Sigma$
- From a vertex of Out, the following moves are possible:
  - $(q, u, \mathsf{Out}) \xrightarrow{b} (q', u', \mathsf{Out})$ for each $b \in \Gamma$ such that $u = au'$ for $a \in \Sigma$ and $q' = \delta(q, (a, b))$,
  - $(q, u, \mathsf{Out}) \xrightarrow{\varepsilon} (q, u)$,
- The initial vertex is (q_0, ε).

The winning condition should express the following property: at each point, In can extend the output such that the resulting pair of input and output is in R. For this purpose, we define a set B of bad vertices for player Out consisting of

1. all $(q, u) \in V_{\mathsf{Out}}$ such that $|u| < 2K$ and R_q^u is empty, and
2. all $(q, u) \in V_{\mathsf{Out}}$ such that $|u| = 2K$ and R_q^u cannot be uniformized by a recognizable relation.

Note that both conditions are decidable for a given vertex ((1) is emptiness of automatic relations and (2) is decidable by Proposition 20).

The objective of Out is to avoid the vertices in B. Games with an objective of this kind are usually referred to as safety games because the player has to stay within the safe region of the game graph.

A play is a maximal path in $G_{\mathcal{A}}^K$ starting in the initial vertex. Maximal means that the path is either infinite or it ends in a vertex without outgoing

edges (a vertex (q, u) with $|u| = 2K$). Out wins a play if no vertex from B occurs. A strategy for Out is a function that defines for finite sequences of moves that end in a vertex of Out the next move to be taken by Out. Such a strategy is winning if Out wins all plays in which she makes her moves according to the strategy.

The following lemma reduces our question to the existence of winning strategies in $G^K_{\mathcal{A}}$.

LEMMA 21 *The relation R can be uniformized by an ST if, and only if, Out has a winning strategy in $G^K_{\mathcal{A}}$.*

Proof. Assume that Out has a winning strategy in $G^K_{\mathcal{A}}$. Since $G^K_{\mathcal{A}}$ is a safety game, the player who has a winning strategy also has a positional one, which means that the next move chosen by the strategy only depends on the current vertex (see (Grädel, Thomas, & Wilke, 2002)). Such a strategy can be represented by a function $\sigma : V_{\mathsf{Out}} \to \Gamma \cup \{\varepsilon\}$ (because the moves of Out are deterministically labeled by letters in Γ or by ε).

We now describe how to construct the ST $\mathcal{T}$ that uniformizes R. Consider a pair (q, u) such that $|u| = 2K$ and R^u_q can be uniformized by a recognizable relation. For each such pair we choose an ST $\mathcal{T}^u_q$ that uniformizes R^u_q. Then $\mathcal{T}$ consists of the (disjoint) union of the transducers $\mathcal{T}^u_q$ and a part that uses V_{In} as states. The initial state of $\mathcal{T}^u_q$ is identified with $(q, u) \in V_{\mathsf{In}}$. The transitions of $\mathcal{T}$ for some $(q, u) \in V_{\mathsf{In}}$ with $|u| < 2K$ is defined as follows. Let $a \in \Sigma$. The strategy σ defines a unique finite sequence of moves of Out from (q, ua, Out). This sequence of moves corresponds to some finite word $w \in \Gamma^*$ and ends in a vertex (p, u'). We define $\delta_{\mathcal{T}}((q, u), a) := ((p, u'), w)$. Furthermore, we define the final output function f of $\mathcal{T}$ by $f(q, u) := v$ for some v such that $(u, v) \in R^u_q$, which exists since σ is a winning strategy and thus avoids all vertices in B. It is not difficult to verify that $\mathcal{T}$ indeed defines a uniformization of R.

For the other direction, assume that R is uniformized by some ST $\mathcal{T}$. A winning strategy for Out basically simulates $\mathcal{T}$ on the inputs played by In. However, it might happen that the output delay in $\mathcal{T}$ is larger than $2K$ or that for some input sequences the output sequence produced by $\mathcal{T}$ might be longer than the input sequence. These cases are not captured by the game graph.

To describe the strategy, we split the sequence of moves by In (simply referred to as input sequence) into blocks $u_i \in \Sigma^*$ of length K. So the current input sequence is always of the form $u_1 \cdots u_n u$ with $|u_i| = K$ and $|u| < K$. The strategy produces its output moves that are different from the ε-move in blocks of K. For the input $u_1 \cdots u_n u$, it will have produced output moves $v_1 \cdots v_{n-1}$ with $|v_i| = K$. This means that the corresponding vertex in the game graph is $(q_{n-1}, u_n u)$ with $q_{n-1} = \delta(q_0, (u_1 \cdots u_{n-1}) \otimes (v_1 \cdots v_{n-1}))$.

The output moves producing v_n are played once a vertex $(q_{n-1}, u_n u_{n+1}, \mathsf{Out})$ is reached (with $|u_n u_{n+1}| = 2K$). To define v_n, the ST $\mathcal{T}$ is not simulated on the original input sequence $u_1 \cdots u_n u_{n+1}$ but on a modification $u'_1 \cdots u'_n u'_{n+1}$ that is obtained by repeating some idempotent factors as follows: We let $u'_1 = u_1$. Now assume that u'_i is defined for all $1 \leq i \leq n$. Let $u_{n+1} = xyz$ with $y \neq \varepsilon$ idempotent. Then $u'_{n+1} = xy^m z$ for some m such that $|\mathcal{T}(u'_1 \cdots u'_{n+1})| \geq |u'_1 \cdots u'_n|$. If such an m does not exist, then Lemma 19 implies that $R^{u_n u_{n+1}}_{q_{n-1}}$ can be uniformized by a recognizable relation. Then Out can move to $(q_{n-1}, u_n u_{n+1})$ and wins.

Given this definition of the u'_i, we define v_n as follows: Let $v'_1 \cdots v'_n$ be the initial part of $\mathcal{T}(u'_1 \cdots u'_{n+1})$ such that $|v'_i| = |u'_i|$. Since u'_n is obtained from u_n by repeating an idempotent factor, the profiles of u_n and u'_n are the same and thus, there is some v_n such that $u_n \otimes v_n$ induces the same state transformation on $\mathcal{A}$ as $u'_n \otimes v'_n$. We pick such a v_n, and σ makes K moves from $(q_{n-1}, u_n u_{n+1}, \mathsf{Out})$ according to the letters in v_n, leading to some $(q_n, u_{n+1}, \mathsf{Out})$, and then takes the ε-move to (q_n, u_{n+1}).

To show that this defines a winning strategy for Out, it suffices to show that (q_n, u_{n+1}) is not in B. Consider $\mathcal{T}_f(u'_1 \cdots u'_{n+1})$, which is of the form $v'_1 \cdots v'_n v'$. Since $\mathcal{T}$ uniformizes R, we know that $(u'_1 \cdots u'_{n+1}, v'_1 \cdots v'_n v') \in R$. Because u'_{n+1} is obtained from u_{n+1} by repeating an idempotent factor, there is some v such that $u_{n+1} \otimes v$ induces the same state transformation on $\mathcal{A}$ as $u'_{n+1} \otimes v'$. In combination with the choice of the v_i, we obtain that $u'_1 \cdots u'_{n+1} \otimes v'_1 \cdots v'_n v'$ and $u_1 \cdots u_{n+1} \otimes v_1 \cdots v_n v$ induce the same state transformation in $\mathcal{A}$ and therefore, $(u_1 \cdots u_{n+1}, v_1 \cdots v_n v) \in R$ and $(u_{n+1}, v) \in R_{q_n}^{u_{n+1}}$. This shows that (q_n, u_{n+1}) is not in B. □

Theorem 18 follows from Lemma 21 and the fact that a winning strategy for Out can effectively be computed in $G_{\mathcal{A}}^K$ (see (Grädel et al., 2002)).

5 Conclusion

In this paper, we have given an overview of uniformization results for relations defined by various automaton models. Automatic relations can be uniformized by automatic functions for relations defined over finite words and trees, as well as for relations over infinite words. For infinite trees, the uniformization fails, for example for the element relation.

For rational relations, uniformization can be shown by a reduction to automatic relations using a composition theorem. This technique works for finite as well as infinite words.

Concerning the uniformization of relations over finite words by subsequential transducers, we have presented a decidability result for automatic relations and an undecidability result for rational relations. It turns out that compared to the case of automatic relations over infinite words, some new phenomena arise because of the possible length difference in the input and output.

One direction for future research is the uniformization of tree relations beyond automatic relations. For example, there is no canonical adaption of the notion of rational relations. However, there are various models of transducers for defining relations over finite trees (see (Comon et al., 2007) and (Raoult, 1997)), for which uniformization questions can be studied.

References

Arnold, A., & Latteux, M. (1979). A new proof of two theorems about rational transductions. *Theoretical Computer Science*, *8*, 261-263.

Berstel, J. (1979). *Transductions and context-free languages.* Stuttgart: Tebuner. (Electronic edition available via the homepage of the author)

Blumensath, A., & Grädel, E. (2000). Automatic structures. In *Proceedings of the 15th ieee symposium on logic in computer science, lics 2000* (pp. 51–62). IEEE Computer Society Press.

Büchi, J. R., & Landweber, L. H. (1969). Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, *138*, 295–311.

Carayol, A., & Löding, C. (2007). MSO on the infinite binary tree: Choice and order. In *Proceedings of the 16th annual conference of the european association for computer science logic, csl 2007* (Vol. 4646, pp. 161–176). Springer.

Carayol, A., Löding, C., Niwiński, D., & Walukiewicz, I. (2010). Choice functions and well-orderings over the infinite binary tree. *Central European Journal of Mathematics*, *8*(4), 662–682.

Choffrut, C., & Grigorieff, S. (1999). Uniformization of rational relations. In *Jewels are forever, contributions on theoretical computer science in honor of arto salomaa* (pp. 59–71). Springer.

Church, A. (1962). Logic, arithmetic and automata. In *Proceedings of the international congress of mathematicians* (pp. 23–35).

Colcombet, T., & Löding, C. (2007). Transforming structures by set interpretations. *Logical Methods in Computer Science*, *3*(2).

Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Löding, C., Lugiez, D., et al. (2007). *Tree Automata Techniques and Applications.* Available on `http://tata.gforge.inria.fr/`. (Last Release: October 12, 2007)

Diestel, R. (2000). *Graph theory* (Second ed.). Springer.

Eilenberg, S. (1974). *Automata, languages and machines* (Vol. A). Academic Press.

Elgot, C. C., & Mezei, J. E. (1965). On relations defined by generalized finite automata. *IBM J. Res. Dev.*, *9*(1), 47–68.

Engelfriet, J. (1978). On tree transducers for partial functions. *Inf. Process. Lett.*, *7*(4), 170–172.

Grädel, E., Thomas, W., & Wilke, T. (Eds.). (2002). *Automata, logics, and infinite games* (Vol. 2500). Springer.

Gurevich, Y., & Shelah, S. (1983). Rabin's uniformization problem. *J. Symb. Log.*, *48*(4), 1105–1119.

Holtmann, M., Kaiser, Ł., & Thomas, W. (2010). Degrees of lookahead in regular infinite games. In *Foundations of software science and computational structures* (Vol. 6014, pp. 252–266). Springer.

Hopcroft, J. E., & Ullman, J. D. (1979). *Introduction to automata theory, languages, and computation.* Addison Wesley.

Hosch, F. A., & Landweber, L. H. (1972). Finite delay solutions for sequential conditions. In *Icalp* (pp. 45–60).

Johnson, J. H. (1986). Rational equivalence relations. *Theoretical Computer Science*, *47*(3), 39-60.

Khoussainov, B., & Nerode, A. (1995). Automatic presentations of structures. In *Logical and computational complexity. selected papers. logic and computational complexity, international workshop lcc '94, indianapolis, indiana, usa, 13-16 october 1994* (Vol. 960, pp. 367–392). Springer.

Kobayashi, K. (1969). Classification of formal languages by functional binary transductions. *Information and Control*, *15*(1), 95-109.

Kuske, D., & Lohrey, M. (2006). First-order and counting theories of *omega-*

automatic structures. In *Foundations of software science and computation structures, 9th international conference, fossacs 2006, proceedings* (Vol. 3921, pp. 322–336). Springer.

Kuske, D., & Weidner, T. (2011). Size and computation of injective tree automatic presentations. In *Mathematical foundations of computer science 2011 - 36th international symposium, mfcs 2011, proceedings* (Vol. 6907, pp. 424–435). Springer.

Löding, C. (2011). Infinite games and automata theory. In K. R. Apt & E. Grädel (Eds.), *Lectures in game theory for computer scientists.* Cambridge University Press.

Löding, C. (2012). Basics on tree automata. In D. D'Souza & P. Shankar (Eds.), *Modern applications of automata theory.* World Scientific.

Lombardy, S., & Sakarovitch, J. (2010). Radix cross-sections for length morphisms. In *Latin 2010: Theoretical informatics, 9th latin american symposium, oaxaca, mexico, april 19-23, 2010. proceedings* (p. 184-195).

Moschovakis, Y. N. (1980). *Descriptive set theory* (Vol. 100). Amsterdam, New York, Oxford: North-Holland Publishing Company.

Raoult, J.-C. (1997). Rational tree relations. *Bulletin of the Belgian Mathematical Society*, *4*(1), 149–176.

Sakarovitch, J. (2009). *Elements of automata theory.* Cambridge University Press.

Siefkes, D. (1975). The recursive sets in certain monadic second order fragments of arithmetic. *Arch. für mat. Logik und Grundlagenforschung*, *17*, 71–80.

Thomas, W. (1997). Languages, automata, and logic. In G. Rozenberg & A. Salomaa (Eds.), *Handbook of Formal Language Theory* (Vol. III, pp. 389–455). Springer.

Thomas, W. (2008). Church's problem and a tour through automata theory. In *Pillars of computer science, essays dedicated to Boris (Boaz) Trakhtenbrot on the occasion of his 85th birthday* (Vol. 4800, pp. 635–655). Springer.

Thomas, W. (2009). Facets of synthesis: Revisiting Church's problem. In *Proceedings of the 12th international conference on foundations of software science and computational structures, fossacs 2009* (Vol. 5504, pp. 1–14). Springer.