Internship Report - Church Synthesis and Uniformization of Relations on Words

Léo Exibard[1]

1 **Département d'Informatique**
E.N.S de Lyon
France

This work was conducted under the supervision of Emmanuel Filiot: Département d'Informatique, Université Libre de Bruxelles (U.L.B.), Belgium.

Contents

1 Introduction

1.1 Context and motivations

Transducers Automata are finite state machines that are designed to recognize languages. Similarly, we can define transducers (abbreviated NFT, for nondeterministic finite transducers), which can be understood as automata with outputs: at each transition, they read a letter or a word, and they output another word. These devices recognize relations, between the input words and the output words.

Transducers can also model interactive systems: the environment inputs words, the system outputs other words, and the sequence of events is valid if the word that results from the interaction is in a given language.

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Verification *Computer aided verification* is a branch of theoretical computer science whose contributions are mathematical formalisms for modelling and reasoning on hardware and software systems. In the automata-based approach to verification [24], the system to design is modeled at a high level of abstraction as an automaton that defines the set of its possible behaviours. All its behaviours should respect a set of correctness properties, classically expressed within a temporal logic [18]. The *model-checking* technics [7, 20, 8] are based on these foundations, and have seen substantial developments in the last two decades. However, the theory of model-checking has mainly focused so far on reactive systems.

System synthesis goes a step further than model-checking. Its aim is to automatically generate a program that satisfies a given specification. The underlying goal is to improve program reliability and optimize design constraints, like time and human errors, by getting rid of the low-level programming task, replacing it with the design of high-level specifications. The challenge of automatic synthesis, long ago introduced by *Church* [6], is difficult to realize for general-purpose programming languages. However in recent years, there has been a renewed interest in feasible methods for the synthesis of application-specific programs, which have been, for instance, successfully applied to reactive systems [17, 16, 10, 12, 21].

Since the actions of the environment in which the system is executed are uncontrollable, the synthesis is classically formalised as a game theoretic problem, where the environment is seen as an adversary [23]. It was first formalised by Church [6]: two players (the environment and the system) alternate in choosing a bit in $\{0, 1\}$. The game lasts for an infinite number of rounds and the interaction generates an infinite word s over $\{0, 1\}$. The winning condition for the system is given by a language R of infinite words. The system wins the game if $s \in R$. The goal of the system is therefore to find a strategy P such that whatever the environment does, all the possible executions of the game, where the system plays according to the strategy P, belongs to the winning language R. The set R represents the set of *good* executions, that the system must try to ensure. It is usually modelled as a set of *requirements* defined in some logic (MSO, temporal logic, ...). A strategy P is a function that, given the whole history of the play (i.e. a finite word over $\{0, 1\}$) tells the system whether to output 0 or 1. Such a strategy may not exist, in which case we say that requirements R are not realizable. Otherwise, the synthesis problem asks to generate a finite representation of a winning strategy P (a program). When the requirements are defined in MSO, the synthesis problem is decidable [5] and moreover, if the requirements are realizable, they are realizable by simple finite state programs (Moore machines). This famous result by Büchi and Landweber was later on refined by Pnueli and Rosner, who proves that when the requirements are given in linear time temporal logic (LTL), the problem is 2ExpTime-complete. The basic framework of Church and synthesis has also motivated a very active domain of research: the theory of games played on graphs. In this setting, the game arena is a graph whose vertices are intended to model the states of the system and the edges the actions of both the system and the environment.

Limitations So far, the theory has mainly focused on reactive systems, modeled as simple finite state transducers that alternatively receive a bit and produce one single bit (Moore machines). This abstraction is too restrictive to model more complex systems, such as string, list or stream processing programs. Strings and lists can be viewed as words, and therefore we will refer to these programs as word processing programs. In a word processing program, one may want to delete some parts of the word, or swap two subwords, or duplicate some information. Such systems are *not reactive*, in the sense that the system does not have to produce the output in a synchronous manner.

Unified framework for synthesis Extensions of the classical setting of synthesis to quantitative, multi-components and probabilistic systems are now actively studied. Still, the focus is on reactive systems. To overcome this limitation, another way to generalise Church's setting is to relax the notions of strategies and requirements. We propose the following generalisation. Given two classes of (finite or infinite) word transformations $\mathcal{R}$ and $\mathcal{F}$ over some alphabet Σ, such that $\mathcal{F}$ are *functions*, the problem $\mathsf{Church}(\mathcal{R}, \mathcal{F})$ asks, given a transformation $R \in \mathcal{R}$, if there exists a transformation $f \in \mathcal{F}$ such that (i) R and f have the same domain, and (ii) $f \subseteq R$. The transformation R is called the *requirements*, and f, which must be a function, is called the *program*. Condition (i) ensures that any input word produced by the (uncontrollable) environment can be processed by f. Condition (ii) ensures that f is correct with respect to the requirements. This novel generalisation establishes a bridge between the synthesis and language theory communities, by providing a unified framework for synthesis, under which fall some important particular cases.

Known instances For example, the problem $\mathsf{Church}(\mathsf{Reg}^2, \mathsf{Moore})$, where here Reg^2 stands for any requirement R defined as a regular language of infinite words over the product alphabet Σ^2 (and therefore a particular kind of transformations), and Moore stands for any program represented as a Moore machine, is decidable when R is defined in MSO, by Büchi-Landweber's theorem [5]. The particular instance $\mathsf{Church}(\mathsf{LTL}, \mathsf{Moore})$ has been studied by Pnueli and Rosner [19]. The instance $\mathsf{Church}(\mathsf{NFT}, \mathsf{FFT})$, where FFT are functional NFT, has always a positive answer and is known as the uniformisation of rational relations in transducer theory [3]. As another instance of the problem, this uniformisation theorem has been extended to two-way transducers [9]. Finally, the problem $\mathsf{Church}(\mathsf{Reg}^2, \mathsf{DelayStrat})$, where $\mathsf{DelayStrat}$ are strategies that can delay their actions, has been considered in [15].

Targeted instances The final goal of this work is to examine $\mathsf{Church}(\mathrm{NFT}, \mathrm{DFT})$: can a nondeterministic transducer be uniformized by a deterministic one?

This may also shed light on $\mathsf{Church}(\mathrm{MSOT}, \mathrm{DFT})$: can a relation defined in monadic second order logic be uniformized by a deterministic transducer?

Solved subcases We were able to decide $\mathsf{Church}_k(\mathrm{NFT}, \mathrm{DFT})$, which is a particular case where we strenghtened the notion of equivalence: the uniformizer cannot wait for too long before outputting its answer. But the main result of this work is the decision of $\mathsf{Church}(\cup\mathrm{DFT}, \mathrm{DFT})$: we characterized the union of deterministic transducers which are uniformizable, by a decidable property.

1.2 Preliminaries

1.2.1 Words and languages

We first remind some notations over words and languages, then extend the classical framework to improve the readability of our proofs. In this paper, A, B and Σ are finite alphabets, and ε is the empty word, whatever the alphabet.

▶ **Definition 1.1** (Mismatch)**.** Let $u, v \in \Sigma^*$. We say that u and v *mismatch* if $\exists i \in \{0, \dots, \min(|u|, |v|), u[i] \neq v[i]$

▶ **Definition 1.2** (Prefixes)**.** Let $u, v \in \Sigma^*$. We say that u is prefix to v, and we denote $u \preccurlyeq v$, if $\exists v' \in \Sigma^*$ such that $v = uv'$.

▶ **Definition 1.3** (Longest common prefix)**.** For $u, v \in \Sigma^*$, we denote by $u \wedge v$ the longest common prefix of u and v, ie the $\delta \in \Sigma^*$ such that $\delta \preccurlyeq u$, $\delta \preccurlyeq v$ and $\forall a \in \Sigma, \delta a \not\preccurlyeq u$ or $\delta a \not\preccurlyeq v$. We can notice that $\wedge$ is commutative and associative. This allows us to extend this notation to finite sets of words: for all $v_1, \cdots, v_n \in \Sigma^*$, we denote $\bigwedge_{i=1}^{n} v_i = v_1 \wedge \cdots \wedge v_n$.

Now, we extend the free monoid associated with an alphabet to a group:

▶ **Definition 1.4** (Group associated with an alphabet)**.** Let Σ be a finite alphabet. For all $a \in \Sigma$, we assume that there exists a^{-1} such that $aa^{-1} = a^{-1}a = \varepsilon$: a^{-1} is the *inverse* of a. We will always assume that $\forall a \in \Sigma, a^{-1} \notin \Sigma$. The set of the inverses of the elements of Σ is denoted by Σ^{-1}.

The group associated with Σ is then $\Sigma^G = (\Sigma + \Sigma^{-1})^*$, modulo the congruence given by the identities $aa^{-1} = a^{-1}a = \varepsilon$. For example, $aa^{-1}b \equiv b \equiv bc^{-1}c$.

Formally, we say that $u \equiv v$ if v can be obtained from u by using the following rules, defined for all $a \in \Sigma \cup \Sigma^{-1}$, finitely many times: $aa^{-1} \leftrightarrow a^{-1}a$, $a^{-1}a \leftrightarrow \varepsilon$, $a\varepsilon \leftrightarrow \varepsilon a$, $\varepsilon a \leftrightarrow a$.

Then, we define $\Sigma^G = (\Sigma + \Sigma^{-1})^*_{/\equiv}$. Now, every word $u = u_1 \cdots u_n \in \Sigma^G$ has an inverse: $u^{-1} = u_n^{-1} \cdots u_1^{-1} \in \Sigma^G$.

We also define a length over this group:

▶ **Definition 1.5** (Length over Σ^G)**.** We already have a length defined over $(\Sigma + \Sigma^{-1})^*$, which consists in discounting the letters: $\forall u = u_1 \cdots u_n \in (\Sigma + \Sigma^{-1})^*, |u|_{(\Sigma+\Sigma^{-1})^*} = n$. We extend this to:

$$|\cdot|_{\Sigma^G} : \left\{ \begin{array}{rcl} \Sigma^G & \to & \mathbb{N} \\ \tilde{u} & \mapsto & \min\limits_{v \equiv \tilde{u}} |v|_{(\Sigma+\Sigma^{-1})^*} = \min\{n \in \mathbb{N} \mid \exists u_1, \cdots, u_n \in \Sigma \cup \Sigma^{-1}, \tilde{u} \equiv u_1 \cdots u_n\} \end{array} \right\}$$

In the following, if $u \in (\Sigma + \Sigma^{-1})^*$, $|u|$ will denote $|\tilde{u}|_{\Sigma^G}$. We can notice that if $u \in \Sigma^*$, $|u|_{\Sigma^*} = |\tilde{u}|_{\Sigma^G}$.

We finally define a distance over Σ^*.

▶ **Definition 1.6** (Distance over Σ^*)**.** For $u, v \in \Sigma^*$, we define $d(u, v) = |u| + |v| - 2|u \wedge v|$

▶ Remark. This distance can also be defined as $\forall u, v \in \Sigma^*, d(u, v) = |v^{-1}u|_{\Sigma^G} = |u^{-1}v|_{\Sigma^G}$

1.2.2 Transducers

Here, we follow the introduction given in [2].

▶ **Definition 1.7** (Transducer)**.** Let A, B be two finite alphabets. A transducer (we will also write NFT, which stands for nondeterministic finite transducer) $\mathcal{T}$ over $A^* \times B^*$ is a quadruple $\mathcal{T} = (Q, E, I, F)$ where:

- Q is the set of states
- $E \subseteq Q \times A^* \times B^* \times Q$ are the transitions
- I is the set of initial states
- F is the set of final states

A transition (p, u, v, q) is also denoted as $p \xrightarrow{u|v} q$. The word u is the *input*, and the word v the *output*. Similarly to automata, we define a path labeled by input u and output v as: $p_0 \xrightarrow{u_0|v_0} p_1 \xrightarrow{u_1|v_1} \cdots \xrightarrow{u_n|v_n} p_{n+1}$, also denoted $p_0 \xrightarrow[\mathcal{T}]{u|v} p_{n+1}$. A path is *successful* if p_0 is an initial state and p_{n+1} a final state. We will also use the term *run*.

For $p \in Q$, we denote by $\mathcal{T}_p = (Q, E, \{p\}, F)$ the transducer $\mathcal{T}$ in which we start from p.

▶ **Definition 1.8** (Associated relation)**.** The relation associated with $\mathcal{T}$ is:
$R_{\mathcal{T}} = \{(u,v) \in A^* \times B^* \mid u|v \text{ is the label of a successful path}\}$.

If $R_{\mathcal{T}}$ is a function, we say that $\mathcal{T}$ is a *functional transducer* (abbr. FFT). We define $\mathrm{dom}(\mathcal{T}) = \pi_1(R_{\mathcal{T}})$, and $\mathrm{Im}(\mathcal{T}) = \pi_2(R_{\mathcal{T}})$ where π_1 and π_2 are the projections respectively on the first and on the second component. For $u \in \mathrm{dom}(\mathcal{T})$, we denote $\mathcal{T}(u) = \{v \in B^* \mid (u,v) \in R_{\mathcal{T}}\}$. For readability, when $R_{\mathcal{T}}$ is a function, we write $\mathcal{T}(u) = v$ instead of $\mathcal{T}(u) = \{v\}$.

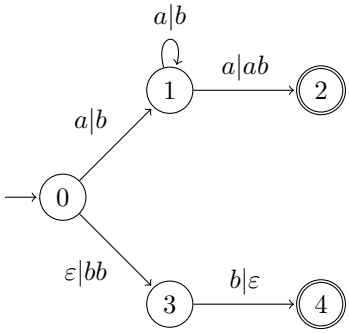


Figure 1 A transducer recognizing the relation $(a,b)^+(a,ab) + (b,bb)$

▶ **Definition 1.9** (Equivalence between transducers)**.** Two transducers $\mathcal{T}_1$ and $\mathcal{T}_2$ are said to be equivalent if they represent the same relation, *ie* if $\mathcal{R}_{\mathcal{T}_1} = \mathcal{R}_{\mathcal{T}_2}$.

Now, we define some classes of transducers.

▶ **Definition 1.10** (Real-time, letter-to-letter and sequential transducer)**.** A transducer is said to be *real-time* if it is labeled over $A \times B^*$. It is said to be letter-to-letter when it is labeled over $A \times B$ (those transducers are very useful to model reactive systems).

A transducer is said to be *sequential* if it is real-time, if it has a unique initial state and if the automaton obtained by projecting away the output words on the transitions of $\mathcal{T}$ is deterministic.

▶ Remark. A sequential transducer is necessarily functional.

We can relax this definition to subsequential transducers, which are allowed to output something from their final states.

▶ **Definition 1.11** (Subsequential transducer)**.** A *subsequential transducer* $\mathcal{T}$ over $A \times B^*$ is a sequential transducer $\mathcal{D} = (Q, E, \{i_0\}, F)$ extended with a function $\rho : F \to B^*$. It computes the function $f_{\mathcal{T}} : \left\{ \begin{array}{rcl} \mathrm{dom}(\mathcal{D}) & \to & B^* \\ u & \mapsto & v\rho(p) \text{ where } i_0 \xrightarrow[\mathcal{D}]{u|v} p \in F \end{array} \right\}$. We will also call them *deterministic transducers*, or DFT.

▶ **Definition 1.12** ($\bullet_{\mathcal{T}}$)**.** Let $\mathcal{T} = (Q, E, I, F)$ be a transducer. We define an operator $\bullet_{\mathcal{T}} : Q \times A^* \to \mathfrak{P}(Q)$, where $\mathfrak{P}(Q)$ is the set of the parts of Q:

- $\forall p \in Q, p \bullet_{\mathcal{T}} \varepsilon = \{p\}$
- $\forall p \in Q, \forall u \in A^+, p \bullet_{\mathcal{T}} u = \{q \in Q \mid \exists v \in B^*, p \xrightarrow[\mathcal{T}]{u|v} q\}$

When $\mathcal{T}$ is subsequential, we denote $p \bullet_{\mathcal{T}} u = q$ instead of $p \bullet_{\mathcal{T}} u = \{q\}$.

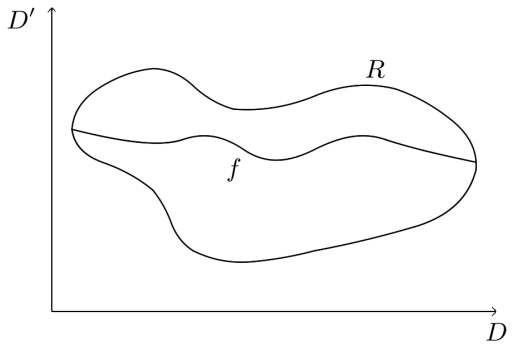


Figure 2 Uniformization of R by f

We can also cite another class of transducers, the 2-ways. A 2-way transducer is a transducer allowed to go back and forth in the input: we add to each transition a direction of reading, either *right* or *left*, and the head moves accordingly. We won't define them formally, since we will only quote them as an example.

In the rest of this paper, we will assume that our transducers are real-time.

1.2.3 Relations

Here, we define some notations over the relations.

▶ **Definition 1.13** (Relations and prefixes)**.** For $(w, x) \in \Sigma^* \times T^*$ and $\mathcal{R} \subseteq A^* \times B^*$, we define $(w, x)\mathcal{R} = \{(wu, xv) \in \Sigma^* A^* \times T^* B^* \mid (u, v) \in \mathcal{R}\}$. In particular, it allows us to add or delete prefixes to the words in $\mathrm{dom}(\mathcal{R})$ or $\mathrm{Im}(\mathcal{R})$.

1.2.4 Synthesis

This notion is thoroughly presented in [22].

▶ **Definition 1.14** (Realizability)**.** Let $R \subseteq X \times Y$ be a relation of domain $D \subseteq X$. Let $f : D \to Y$ be a total function. f is said to realize R if $\forall X \in D, (X, f(X)) \in R$. In other words, $\mathrm{dom}(f) = \mathrm{dom}(R)$ and f is a sub-relation: $f \subseteq R$.

▶ **Definition 1.15** (Uniformization)**.** Let $\mathcal{R}$ be a class of relations, $\mathcal{F}$ a class of functions, and $R \in \mathcal{R}$. We say that R is uniformizable in $\mathcal{F}$ if $\exists f \in \mathcal{F}$ which realizes R.

We now define the decision problem $\mathsf{Church}(\mathcal{R}, \mathcal{F})$ we will try to decide for some instances.

▶ **Definition 1.16** ($\mathsf{Church}(\mathcal{R}, \mathcal{F})$)**.** Let $\mathcal{R}$ be a class of relations, $\mathcal{F}$ a class of functions. $\mathsf{Church}(\mathcal{R}, \mathcal{F})$ is the following decision problem:

Input: $R \in \mathcal{R}$ (finitely represented)

Output: Is R uniformizable in $\mathcal{R}$.

The synthesis problem asks to generate (a finite representation of) f if such an f exists.

2 Uniformization with regards to k-inclusion

2.1 Semantics with origin

Here, we introduce a stronger definition of equivalence between transducers: the outputs also have to be (almost) synchronized. This notion was introduced in [4].

▶ **Definition 2.1** (Origin). Let $\mathcal{T}$ be a (real-time) transducer over $A \times B^*$, and $P = p_0 \xrightarrow{u_0|v_0} p_1 \xrightarrow{u_1|v_1} \cdots \xrightarrow{u_n|v_n} p_{n+1}$ a run in T. The *origin function* $o : \{1, \dots, |v|\} \to \{1, \dots, |u|\}$ of P is defined by $o(j) = i$ such that $|v_0 \cdots v_{i-1}| < j \leq |v_0 \cdots v_i|$, *ie* $\forall j \in \{1, \dots, |v|\}$, $v[j]$ was outputed while reading u_i.

If $\mathcal{T}$ is subsequential, then we extend $o : \{1, \dots, |v|\} \to \{1, \dots, |u|+1\}$, with the convention that the position of a word outputted at a final state is $|u| + 1$.

The relation with origin associated with T is then:
$R_o = \{(u, v, o) \mid \text{there exists a run } P \text{ in } T \text{ labeled by } u|v \text{ with } o \text{ as origin function}\}$

By relaxing these new semantics, we define the k-inclusion of two transducers $\mathcal{T}_1$ and $\mathcal{T}_2$: $\mathcal{T}_1$ can produce its output with at most k delay with respect to the output $\mathcal{T}_2$.

▶ **Definition 2.2** (k-inclusion). Let $\mathcal{T}_1, \mathcal{T}_2$ be two transducers over $A \times B^*$.
We say that $\mathcal{T}_1 \subseteq_k \mathcal{T}_2$ when

$$\forall (u, v, o) \in R_o(\mathcal{T}_1), \exists (u, v, o') \in R_o(\mathcal{T}_2), \forall j \in \{1, \dots, |v|\}, |o(i) - o'(i)| \leq k$$

2.2 The k-uniformization problem

Here, we want to decide whether it is possible to uniformize a nondeterministic transducer $\mathcal{T}$ by a deterministic transducer which is k-included in $\mathcal{T}$.

▶ **Definition 2.3** ($\mathsf{Church}_k(\mathrm{NFT}, \mathrm{DFT})$). Let $\mathcal{T}$ be a nondeterministic transducer. Does there exist $\mathcal{D}$ a deterministic transducer such that $\mathrm{dom}(\mathcal{D}) = \mathrm{dom}(\mathcal{T})$ and $\mathcal{D} \subseteq_k \mathcal{T}$?

▶ **Example 2.4.** We study in fig. 3 the transducer $\mathcal{E}$ which is 2-uniformizable, but not 1-uniformizable: on an input aax, where $x \in \{a, b\}$, any uniformizer must wait to know the input x to output anything. Indeed, $R_{\mathcal{E}}(aaa) = \alpha\alpha$, $R_{\mathcal{E}}(aab) = \beta\beta$ and $\alpha\alpha \wedge \beta\beta = \varepsilon$.

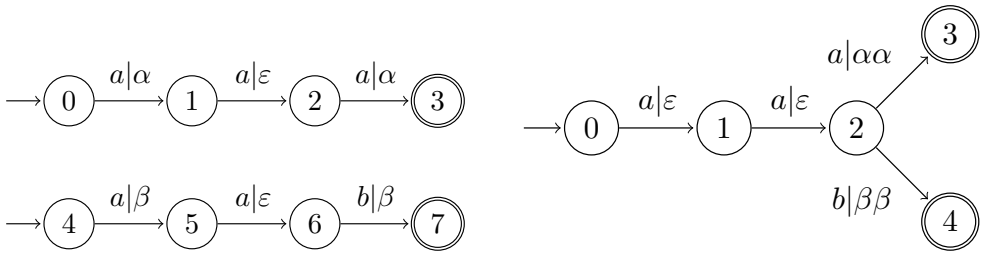


Figure 3 The transducer $\mathcal{E}$ and a 2-uniformizer for $\mathcal{E}$

▶ **Theorem 2.5** ($\mathsf{Church}_k(\mathrm{NFT}, \mathrm{DFT})$). *For all $k \in \mathbb{N}$, $\mathsf{Church}_k(\mathrm{NFT}, \mathrm{DFT})$ is in ExpTime.*

▶ Remark. Here, k is not part of the input.

2.3 The proof

We resort to game theory: we are going to build a two-players safety game over a graph involving players I (for *input*) and O (for *output*) where *Player O* has a winning strategy iff $\mathcal{T}$ is uniformizable.

2.3.1 Safety games

In this section, we follow the introduction given in [14], but we restrict to finite, turn-based safety games. In game theory, safety games involve two players, *Player O* and *Player I*. The protagonist *Player O* has to stay in safe states, whereas the antagonist *Player I* tries to lead him to unsafe states.

▶ **Definition 2.6** (Arena). An *arena* is a directed labeled graph $\mathcal{A} = (V = V_O \uplus V_I, T, \Sigma_I \cup \Sigma_O)$, where:

- V_O is the set of *O-vertices*, which belongs to *Player O*.
- V_I is the set of *I-vertices*, which belongs to *Player I*.
- Σ_I, Σ_O are two alphabets, used to label the transitions.
  ▶ Remark. Transitions are labeled to improve the readability of our proof.
- $T \subseteq \underbrace{(V_I \times \Sigma_I \times V_O)}_{\text{actions of } I} \uplus \underbrace{(V_O \times \Sigma_O \times V_I)}_{\text{actions of } O}$ are the *transitions*, or *moves*.
  ▶ Remark. Since the game is turn-based, transitions go from V_α to $V_{\overline{\alpha}}$.
  There must be no dead end for O: every state owned by O must have at least one outgoing transition.

Now, we describe the progress of the game. In the following, to talk equally about *Player O* and *Player I*, we will denote *Player* α, his opponent being *Player* $\overline{\alpha}$.

▶ **Definition 2.7** (Play). A token is placed on an initial vertex $v_0 \in V$. Then, when v is an α-vertex ($\alpha \in \{O, I\}$), *Player* α choses a transition, and moves the token to the end of this transition. The game keeps going until *Player I* decides to stop. He has to, otherwise he loses. Thus, a finite play of length n is a word $\pi = \pi_0\pi_1 \cdots \pi_n \in (V_I V_O)^* V_I$ such that $\pi_0 = v_0$, $\pi_n \in V_I$ and for all $0 \leq i \leq n-1$, $(\pi_i, \pi_{i+1}) \in T$.

We finally add a winning condition.

▶ **Definition 2.8** (Safety game). Let π be a play of length n. A safety game is an arena extended with a set of *unsafe states* U. *Player I* wins when he leads *Player O* to an unsafe state, *ie* when $\pi \cap U \neq \emptyset$. Otherwise, *Player O* wins.

We now define the notion of strategy:

▶ **Definition 2.9** (Strategies). A *strategy* λ_α for Player α is a mapping that maps any finite play whose last state v is in V_α to a state v' such that $(v, v') \in T$.

The *outcome* of a strategy λ_α of Player α is the set:

$$\mathsf{Outcome}_G(\lambda_\alpha) = \underbrace{\left\{\pi = \pi_0 \dots \pi_n \in V^* \;\middle|\; \forall 0 \leq j \leq n-1, \pi_j \in S_\alpha \Rightarrow \pi_{j+1} = \lambda_\alpha(\pi_0 \dots \pi_j)\right\}}_{\text{finite plays}}$$
$$\cup \underbrace{\left\{\pi = \pi_0\pi_1 \cdots \in V^\omega \;\middle|\; \forall j \in \mathbb{N}, \pi_j \in S_\alpha \Rightarrow \pi_{j+1} = \lambda_\alpha(\pi_0 \dots \pi_j)\right\}}_{\text{infinite plays}}$$

A strategy λ_α for Player α is *winning* if α always wins using this strategy, whatever his opponent plays. For *Player I*, it means that $\mathsf{Outcome}_G(\lambda_I) \subseteq V^*UV^*$, and for *Player O*, we get $\mathsf{Outcome}_G(\lambda_I) \cap V^*UV^* = \emptyset$.

▶ **Theorem 2.10** (Determinacy). *A class of games $\mathcal{G}$ is said to be* determined *if $\forall G \in \mathcal{G}$, either* Player I *or* Player O *has a winning strategy. Finite safety games are determined.*

Proof. This is a consequence of Zermelo's theorem. ◀

▶ **Theorem 2.11** (Winning strategies). *Let G be a finite safety game. We can decide in linear time if* Player O *has a winning strategy. Otherwise,* Player I *has a winning strategy.*

Proof. A proof of this theorem can be found in [14]. ◀

Now, we can prove our statement.

Proof. Let $\mathcal{T} = (S, E, I, F)$ be a nondeterministic transducer.

We first describe the safety game.

2.3.2 Reduction to a safety game

The vertices of this safety game are $\mathfrak{P}(S) \times \mathfrak{P}(S) \times \mathcal{F}\left(S \times \{0, \dots, k\}, \mathfrak{P}(\Sigma^{Mk})\right) \times \{I, O\}$ where $\mathfrak{P}(S)$ denotes the set of the parts of S, $\mathcal{F}(A, B)$ denotes the set of the functions from A to B and where M is the length of the longest output of a transition of $\mathcal{T}$. A state (P, Q, d, C) corresponds to:

- P is the current set of states we can reach from the initial states with regards to the inputs **and** outputs that have already been chosen.
- Q is the current set of states we can reach from the initial states only with regards to the inputs.
- d represents our delays: $d(q, i)$ is the set of words from which we have to chose one word to output in the following i steps to keep q as a valid state (otherwise, it will mean we haven't been able to catch up). For convenience, we will also represent the elements of the set $d(q, \cdot)$ by annotated words $\delta_q \in \Delta_q = d(q,0)^{(0)} \times d(q,1)^{(1)} \times \cdots \times d(q,k)^{(k)} \subseteq \mathcal{V} = (B + \mathbb{N}_k)^* + (B^{-1} + \mathbb{N}_k)^*$.

  We define three operations over these words: for $w \in B^*$, $w^{-1}\delta_q$ is the annotated word obtained by eliminating the longest prefix of w (without taking the annotations into account). $w^{-1}\Delta_q$ is then the natural extension of this function. strip(δ_q) is δ_q without its annotations. Now, lshift(δ_q) is δ_q where every annotation $d^{(i)}$ has been replaced by $d^{(i-1)}$. We can similarly extend lshift to Δ_q.
- $C \in \{I, O\}$ is the current player.

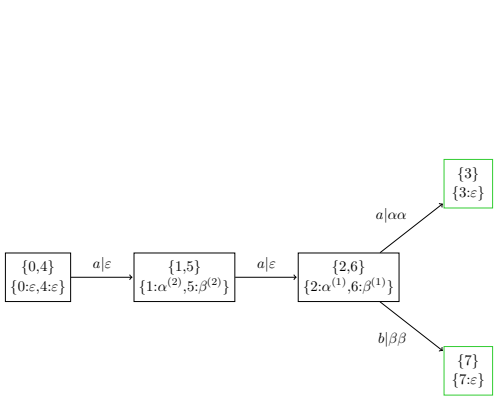
Now, we describe the edges: *Player I* can

- Input a letter, by playing $(P, Q, d, I) \xrightarrow{u} (P', Q', d', O)$ where:
  - $P' = P$
  - $Q' = Q \bullet u$
  - $d' = d$

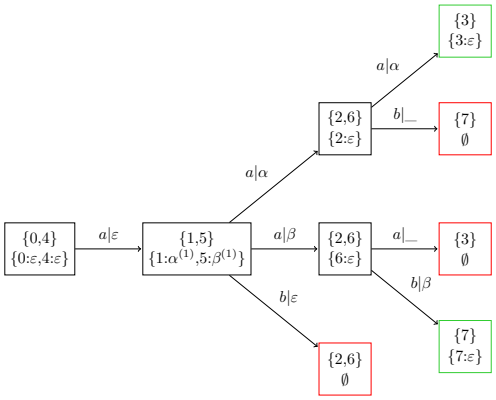
  Then, Player O has to chose a word $v \in B^{2Mk}$ to output.

  ▶ Remark. Player O won't need outputs longer than $2Mk$, because the delays to cover are at most of length Mk, and O is not allowed to be more than k steps ahead of I.

  She plays $(P', Q', d', O) \xrightarrow{\mathcal{T}, v} (P", Q", d", I)$ where:
  - We update d: $\left(\Delta_q := \bigcup_{p \xrightarrow[\mathcal{T}]{u|w} q} \text{lshift}\left(v^{-1}\Delta_p w^{(k+1)}\right)\right) \bigcap \mathcal{V}$
  - $P" = (P \bullet u) \backslash \{q \in (P \bullet u) | \delta_q = \emptyset\}$ (the states which delays are outdated)
  - $Q" = Q'$
- End his input. He has to end his input eventually, otherwise he loses.



(a) A compact representation of the safety game of the 2-uniformization of fig. 3a



(b) The safety game correponding to the 1-uniformization of fig. 3a

This game is finite, since there is a finite number of vertices, and the number of transitions is also bounded: at each step, I choses a letter in a (finite) alphabet, and O choses a word of length at most $2Mk$.

The unsafe states for O are then the (P, Q, d, I) such that $P \cap F = \emptyset$ and $Q \cap F \neq \emptyset$.

Let us show that *Player* O has a winning strategy iff $\mathcal{T}$ is uniformizable:

2.3.3 A winning strategy is a uniformizer

First, if O has a winning strategy, then we build a uniformizer $\mathcal{D} = (S', E', I', F')$ for $\mathcal{T}$:

- The states of $\mathcal{D}$ are $S' = \mathfrak{P}(S) \times \mathfrak{P}(S) \times \mathcal{F}\left(S \times \{0, \dots, k\}, \mathfrak{P}(\Sigma^{Mk})\right)$.
- For any state (P, Q, d), for any u, the transition labeled by u is $(P, Q, d) \xrightarrow{u|v} (P \bullet u, Q \bullet u, d')$, where v is O's answer to this move of I, and d' the updated d.
- The initial state is $I' = (I, I, \varepsilon)$.
- The final states F' are the states such that P and Q contains a final state.
- We extend our transducer with $\rho : \left\{ \begin{array}{ccc} F' & \to & B^* \\ (P, Q, d, I) & \mapsto & \text{strip}(\delta_q) \text{ where } q \in P \cap F \end{array} \right\}$

Clearly by construction, our transducer $\mathcal{D}$ is subsequential. Now, let us show that it is a proper uniformizer, *ie* $\mathcal{D} \subseteq_k \mathcal{T}$. Let $u = u_0 u_1 \dots u_n \in A^*$. We examine what happens when I plays successively $u_0, u_1, \dots, u_n$.

If $u \notin \text{dom}(\mathcal{T})$ then, even if there is a path labeled by u in $\mathcal{D}$, this path ends in $(P, I \bullet u, d)$, where $(I \bullet u) \cap F = \emptyset$: u isn't recognized by $\mathcal{D}$.

If $u \in \text{dom}(\mathcal{T})$, then the game ends in $(P_{n+1}, Q_{n+1}, d_{n+1}, I)$. Since $Q_{n+1} = (I \bullet u) \cap F \neq \emptyset$, we have $P_{n+1} \cap F \neq \emptyset$, otherwise we would be in an unsafe state, and we assumed that O prevented that. Thus, u is accepted by $\mathcal{D}$: let $(u, v, o) \in R_o(\mathcal{D})$ be the triple associated to u by $\mathcal{D}$. $\mathcal{D}$ being deterministic, (u, v, o) is unique.

Now, let $(P_i, Q_i, d_i, I)_{0 \leq i \leq n+1}$ be the I-states visited by the play, and $(v_i)_{0 \leq i \leq n+1}$ the corresponding outputs. We then build a run in $\mathcal{T}$ labeled by $u|v$ which satisfies the delay condition: we start from $q_{n+1} = f \in P_{n+1} \cap F$. At each step, we chose q_i an origin of a transition to q_{i+1} responsible for $\delta_{q_{i+1}}$: q_i verifies $\Delta_{q_i} \cap v_i \delta_{q_{i+1}} w_i \neq \emptyset$ where $q_i \xrightarrow{u_i|w_i} q_{i+1}$. We then chose a $\delta_{q_i} \in \Delta_{q_i} \cap v_i \Delta_{q_{i+1}} w_i$. Thus, we have a run $q_0 \xrightarrow{u_0|v'_0} q_1 \xrightarrow{u_1|v'_1} \cdots \xrightarrow{u_n|v'_n} q_n$ in $\mathcal{T}$.

We show by induction on $0 \leq i \leq n$ that for all $0 \leq i \leq n, v_0 \dots v_i \delta_{q_{i+1}} = v'_0 \dots v'_i$

- If $i = 0$, then $\delta_{q_1} = v_0^{-1} \varepsilon v'_0$, and then $v_0 \delta_{q_1} = v'_0$

- If $1 \leq i \leq n$, then we have $\delta_{q_{i+1}} = v_i^{-1}\delta_{q_i}v'_i$. By induction, we have $v_0 \dots v_{i-1}\delta_{q_i} = v'_0 \dots v'_{i-1}$, so $v_0 \dots v_{i-1}v_i\delta_{q_{i+1}} = v_0 \dots v_{i-1}v_iv_i^{-1}\delta_{q_i}v'_i = v_0 \dots v_{i-1}\delta_{q_i}v'_i = v'_0 \dots v'_{i-1}v'_i$

Thus, we have $v_0 \dots v_n\delta_{q_{n+1}} = v'_0 \dots v'_n$. Since $v_{n+1} = \delta_{q_{n+1}}$, we finally get $v_0 \dots v_{n+1} = v'_0 \dots v'_n$: the outputs are the same.

Moreover, the delays are respected: let o' be the origin function of the run labeled by u in $\mathcal{D}$. We show that $\forall 0 \leq i \leq n, o'(i) \leq o(i) + k$: let $0 \leq i \leq n$. We examine when v'_i is outputted. We have $\delta_{q_{i+1}} = \text{lshift}(v_i^{-1}\delta_{q_i}v_i'^{(k+1)})$. Let j be the first moment when the last letter of v'_i is outputted. Since the delay associated with v'_i decreases from one at each new letter of input, we have $j \leq i+k$, so v'_i is outputted at most at step $i+k$ (otherwise we would have $\delta_{q_j} \notin (B + \mathbb{N}_k)^* + (B^{-1} + \mathbb{N}_k)^*$, so our run wouldn't be valid), and so $o'(i) \leq o(i) + k$. Consequently, $|o'(i) - o(i)| \leq k$.

We have shown that if O has a winning strategy, then $\mathcal{T}$ is k-uniformizable.

2.3.4 A uniformizer is a winning strategy

Conversely, if $\mathcal{T}$ is k-uniformizable, then let $\mathcal{D}$ be a deterministic transducer which uniformizes $\mathcal{T}$. We build from $\mathcal{D} = (S', E, \{i_0\}, F')$ a winning strategy of our game: at each input a of Player I from a state $(P, I \bullet u, d, I)$ which leads to $(P, (I \bullet u) \bullet a, d, O)$, we answer by outputting the word w labelling the transition $(i_0 \bullet u) \xrightarrow{a|w} q$ in $\mathcal{D}$ if such transition exists, and ε otherwise.

Now, let $u_0, \dots, u_n$ be a sequence of inputs given by I. We denote $u = u_0 \dots u_n$, and $(P_i, Q_i, d_i)_{1 \leq i \leq n+1}$ the sequence of states owned by O visited by the play. We denote by (P_0, Q_0, d_0) the initial state (I, I, ε). Let us show that all states are safe.

- If $u \in \text{dom}(\mathcal{T}) = \text{dom}(\mathcal{D})$, then let $i_0 \xrightarrow{u_0|v'_0} p_1 \xrightarrow{u_1|v'_1} \cdots \xrightarrow{u_{n-1}|v'_{n-1}} p_n \xrightarrow{u_n|v'_n} f_{n+1} \xrightarrow{v'_{n+1}}$ be the run labeled by u in $\mathcal{D}$ and o' its origin function. Now, let $q_0 \xrightarrow{u_0|v_0} \dots \xrightarrow{u_n|v_n} q_{n+1}$ a run in $\mathcal{T}$, with o as origin function, such that $v'_0 \dots v'_n v'_{n+1} = v_0 \dots v_n = v$ and $\forall j \in \{1, \dots, |v|\}, |o(i) - o'(i)| \leq k$. Such run exists because $\mathcal{D}$ uniformizes $\mathcal{T}$.

  We now show that delays are respected:

  ▶ **Lemma 2.12.** $\forall 0 \leq i \leq n+1, q_i \in P_i \wedge (v'_0 \dots v'_{i-1})^{-1}v_0^{(k-(i-1))}v_1^{(k-(i-2))} \dots v_{i-1}^{(k)} \in \Delta_{q_i}$

  Proof. This technical proof can be found in annex at page 22. ◀

  From this lemma, we can deduce that O stays in safe states: we never have $P_i \cap F = \emptyset$ and $Q_i \cap F \neq \emptyset$.

We have shown that the strategy we built was winning.

The equivalence between a winning strategy and a uniformizer shows that, by building and solving this game, we can decide $\mathsf{Church}_k(\text{NFT}, \text{DFT})$. The size of this game is $2^{2|Q|} \times \left(|\Sigma|^{lk}\right)^{|Q| \times k}$. Thanks to 2.11, this game can be solved in linear time, so this leads us to an exponential execution time: $\mathsf{Church}_k(\text{NFT}, \text{DFT})$ is in ExpTime.

▶ Remark. If k is part of the input, $\mathsf{Church}_k(\text{NFT}, \text{DFT})$ is in 2-ExpTime.

◀

3 Uniformization of unions of deterministic transducers

We first introduce the notion of delays between words. It is closely related to the notion of longest common prefix. This notion is defined for two words in [11], but here, we adopt a slightly different point of view, and generalize it to sets of words. In this section, we prove

some lemmas to help the reader get a better understanding of the notion. These lemmas will also help us in the proof of our main result.

3.1 Delays

▶ **Definition 3.1** (Delays). For two words $u, v \in \Sigma^*$, we denote by $D(u,v) = v^{-1}u$ the delay between u and v. We can also extend it to a set of words, by examining the delays towards the longest common prefix: For $v_1, \cdots, v_l \in \Sigma^*$, we define $\Delta(\{v_1, \ldots, v_l\})$: $\left\{ \begin{array}{rcl} \{1, \ldots, l\} & \to & B^* \\ i & \mapsto & \left(\bigwedge_{j=1}^{l} v_j\right)^{-1} v_i \end{array} \right\}$

▶ **Lemma 3.2** (Sets of words and delays). *Let $\{u_i\}_{i\in I}, \{v_i\}_{i\in I} \in {\Sigma^*}^I$ be two finite sets of words, and let $w, x \in \Sigma^*$. If $\forall i \in I, D(u_i, w) = D(u_i v_i, x)$, then $\forall i, j \in I, D(u_i, u_j) = D(u_i v_i, u_j v_j)$.*

Proof. We assume that $\forall i \in I, D(u_i, w) = D(u_i v_i, x)$. Then, let $i, j \in I$. We have $\left\{ \begin{array}{l} w^{-1}u_i = x^{-1}u_i v_i \\ w^{-1}u_j = x^{-1}u_j v_j \end{array} \right.$, so $\left\{ \begin{array}{l} u_i v_i = x w^{-1} u_i \\ (u_j v_j)^{-1} = u_j^{-1}(x w^{-1})^{-1} \end{array} \right.$, which means that $(u_j v_j)^{-1} u_i v_i = u_j^{-1}(x w^{-1})^{-1}(x w^{-1}) u_i$, which finally leads to $D(u_i, u_j) = D(u_i v_i, u_j v_j)$. ◀

▶ **Lemma 3.3** (Stable delays). *$\forall u, v, w, x \in \Sigma^*$, we have $D(u,w) = D(uv, wx) \Leftrightarrow \forall l, m \in \mathbb{N}, D(uv^l, wx^l) = D(uv^m, wx^m)$*

Proof. We show by induction on $l \in \mathbb{N}$ that $\forall l \in \mathbb{N}, D(uv^l, wx^l) = D(u,w)$

- If $l = 0$, then we trivially have the identity.
- Let $l \geq 0$. $D(uv^{l+1}, wx^{l+1}) = x^{-l}x^{-1}w^{-1}uvv^l = x^{-l}D(uv,wx)v^l = x^{-l}D(u,w)v^l = x^{-l}w^{-1}uv^l = D(uv^l, wx^l) = D(u,w)$

The converse is given by $l = 0$ and $m = 1$. ◀

▶ **Lemma 3.4** (Delays and prefixes). *$\forall u, v, w, x \in \Sigma^*$ such that $|v| \neq 0$ or $|x| \neq 0$, $D(u,w) = D(uv, wx) \Rightarrow (u \preccurlyeq w \wedge uv \preccurlyeq wx) \vee (w \preccurlyeq u \wedge wx \preccurlyeq uv)$*

Proof. Let $u, v, w, x \in \Sigma^*$ such that $|v| \neq 0$ or $|x| \neq 0$, and $D(u,w) = D(uv, wx)$.

- If $w \preccurlyeq u$, then $\exists u' \in \Sigma^*, u = wu'$, *ie* $D(u,w) = u'$. Then, $D(uv, wx) = (wx)^{-1}uv = u'$, so $uv = wxu'$: $wx \preccurlyeq uv$
- Symmetrically, if $u \preccurlyeq w$, then $uv \preccurlyeq wx$.
- The third case is impossible: u and w cannot mismatch. Otherwise, $\exists \delta \in \Sigma^*, \exists u'w' \in \Sigma^*, \exists a, b \in \Sigma, u = \delta a u', w = \delta b w'$, so $D(u,w) = w^{-1}u = w'^{-1}b^{-1}au'$. Then, $D(uv, wx) = x^{-1}w'^{-1}b^{-1}au'v$, so, since $|v| \neq 0$ or $|x| \neq 0$, $|D(uv,wx)| > |D(u,v)|$: $D(u,v) \neq D(uv,wx)$.

◀

The next lemma states that if a loop leads to different delays, then it can be used to produce infinitely many different delays.

▶ **Lemma 3.5** (Infinitary condition). *$\forall u, v, w, x \in \Sigma^*$, $D(u,w) \neq D(uv,wx)$ if and only if $\left\{D(uv^i, wx^i) \;\middle|\; i \geq 0\right\}$ is infinite.*

Proof. We first notice that if $v = x = \varepsilon$, then $\forall i \in \mathbb{N}, D(u,w) = D(uv^i, wx^i)$ and the equivalence holds. We now assume that $v \neq \varepsilon$ or $x \neq \varepsilon$. We treat the different cases:

- u and w mismatch: $\exists \delta \in \Sigma^*, \exists u'w' \in \Sigma^*, \exists a, b \in \Sigma, u = \delta a u', w = \delta b w'$, so $D(u,w) = w^{-1}u = w'^{-1}b^{-1}au'$. Then $\forall i \in \mathbb{N}, D(uv^i, wx^i) = x^{-i}w^{-1}uv^i = x^{-i}w'^{-1}a^{-1}bu'v^i$. Consequently, $\forall i \in \mathbb{N}, |D(uv^i, wx^i)| = i \cdot |x| + |w'| + 2 + |u'| + i \cdot |v|$: since $v \neq \varepsilon$ or $x \neq \varepsilon$, $\{|D(uv^i, wx^i)|\}_{i\in\mathbb{N}} \xrightarrow[i\to\infty]{} \infty$, so it takes inifinitely many values: $\left\{D(uv^i, wx^i) \;\middle|\; i \geq 0\right\}$ is infinite.
- $w \preccurlyeq u$: $\exists u' \in \Sigma^*, u = wu'$, *ie* $D(u,w) = w^{-1}u = u'$. We have $\forall i \in \mathbb{N}, D(uv^i, wx^i) = x^{-i}w^{-1}uv^i = x^{-i}u'v^i$. Then, $\forall i \in \mathbb{N}, |D(uv^i, wx^i)| \geq |i \cdot |v| - |u'| - i \cdot |x|| \geq |i \cdot |v| - i \cdot |x|| - |u'|$.
  - If $|v| \neq |x|$, then $|i \cdot |v| - i \cdot |x|| - |u'| \xrightarrow[i\to\infty]{} \infty$, and then $\{|D(uv^i, wx^i)|\}_{i\in\mathbb{N}} \xrightarrow[i\to\infty]{} \infty$: $\left\{D(uv^i, wx^i) \;\middle|\; i \geq 0\right\}$ is infinite.
  - Else, we show that there exists $l \geq 0$ such that uv^l and wx^l mismatch:
    * If uv and wx mismatch then $l = 1$ is suitable.
    * Otherwise, since $w \preccurlyeq u$ and $|v| = |x|$, we have $wx \preccurlyeq uv$. Let $v' \in \Sigma^*$ be such that $uv = wxv'$, *ie* $D(uv, wx) = v'$. Now, let $l \geq 1$ such that $|x|^l > |v'| = |u'|$. We have $uv^{l+1} = wxv'v^l = wu'v^{l+1}$.

      We can't have $wx^{l+1} \preccurlyeq uv^{l+1}$, otherwise $\begin{cases} wx^{l+1} \preccurlyeq wxv'v^l \\ \qquad\quad \preccurlyeq wu'v^{l+1} \end{cases}$ *ie* $\begin{cases} x^l \preccurlyeq v'v^l \\ x^lx \preccurlyeq u'v^{l+1} \end{cases}$ and so, since $|x|^l \geq |v'| = |u'|$, $\begin{cases} v' \preccurlyeq x^l \\ u' \preccurlyeq x^l \end{cases}$, which means $u' = v'$, *ie* $D(u,w) = D(uv, wx)$

    Since uv^l and wx^l mismatch, we already showed that $\{D(uv^{li}, wx^{li}) \mid i \geq 0\}$ is infinite. Since $\{D(uv^{li}, wx^{li}) \mid i \geq 0\} \subseteq \{D(uv^i, wx^i) \mid i \geq 0\}$, $\{D(uv^i, wx^i) \mid i \geq 0\}$ is also infinite.

Conversely, if $D(u,w) = D(uv, wx)$ then, thanks to lemma 3.3, we get $\forall i \in \mathbb{N}, D(uv^i, wx^i) = D(u,w)$, so $\left\{D(uv^i, wx^i) \;\middle|\; i \geq 0\right\}$ is finite. ◀

▶ **Lemma 3.6** (Increasing delays). *$\forall u, v, w, x \in \Sigma^*$, we have $D(u,w) \neq D(uv, wx) \Leftrightarrow \forall l \neq m, D(uv^l, wx^l) \neq D(uv^m, wx^m)$*

Proof. If $\exists l < m, D(uv^l, wx^l) = D(uv^m, wx^m)$, then, since $\{D(uv^i, wx^i)\}_{i\in\mathbb{N}}$ can be defined by induction $\{D(uv^i, wx^i)\}_{i\in\mathbb{N}} = \begin{cases} D(u,w) \text{ if } i = 0 \\ x^{-1}D(uv^{i-1}, wx^{i-1})v \text{ if } i > 0 \end{cases}$ we can deduce that $\{D(uv^i, wx^i)\}_{i\geq l}$ is periodic, and takes finitely many values. Consequently, $\{D(uv^i, wx^i)\}_{i\in\mathbb{N}}$ takes finitely many values, which contradicts lemma 3.5.

The converse is given by $l = 0$ and $m = 1$. ◀

Here, we shed light on the relation between delays and their generalization.

▶ **Lemma 3.7** (Delays). *$\forall \{v_i\}_{i\in I}, \{w_i\}_{i\in I} \in \Sigma^{*I}, \Delta(\{v_i\}_{i\in I}) = \Delta(\{v_iw_i\}_{i\in I}) \Leftrightarrow \forall m, n \in I, D(v_m, v_n) = D(v_mw_m, v_nw_n)$*

Proof. The direct implication is given by lemma 3.2. Let us show the converse.

In the following, we will denote $\delta = \bigwedge_{i\in I} v_i$ and $\delta' = \bigwedge_{i\in I} v_iw_i$. Here, we assume that $\exists l \in I, w_l \neq \varepsilon$, otherwise the lemma is trivial. So, thanks to lemma 3.4, we have, by comparing to w_l, $\forall i, j \in I$, either $v_i \preccurlyeq v_j$ and $v_iw_i \preccurlyeq v_jw_j$, or $v_j \preccurlyeq v_i$ and $v_jw_j \preccurlyeq v_iw_i$. Consequently, $\exists n \in I, \delta = v_n$, $\exists p \in I, \delta' = v_pw_p$. Moreover, since $v_n \preccurlyeq v_p$, we get $v_nw_n \preccurlyeq v_pw_p$, so, since $\delta' = v_pw_p \preccurlyeq v_nw_n$, we finally have $\delta = v_n$ and $\delta' = v_nw_n$. Then, $\forall j \in I$, $\Delta(\{v_i\}_{i\in I})(j) = \delta^{-1}v_j = v_n^{-1}v_j = D(v_j, v_n) = D(v_jw_j, v_nw_n) = (v_nw_n)^{-1}v_jw_j = \delta'^{-1}v_jw_j = \Delta(\{v_iw_i\}_{i\in I})(j)$. In conclusion, $\Delta(\{v_i\}_{i\in I}) = \Delta(\{v_iw_i\}_{i\in I})$. ◀

▶ **Lemma 3.8** (Generalized delays). *This lemma is a version of 3.6 for generalized delays: let* $\{v_i\}_{i\in I}, \{w_i\}_{i\in I} \in {\Sigma^*}^I$. *If* $\Delta(\{v_i\}_{i\in I}) \neq \Delta(\{v_i w_i\}_{i\in I})$, *then* $\forall l, m \in \mathbb{N}, \Delta(\{v_i w_i^l\}_{i\in I}) \neq \Delta(\{v_i w_i^m\}_{i\in I})$

Proof. We use lemmas 3.6 and 3.7. ◀

▶ **Definition 3.9** (Relations and prefixes: an extension). Here, we define a relation similarly to what we did for the subsequential transducers: the following can be seen as the definition of an output function for the inital states of a nondterministic transducer.

For a transducer $\mathcal{T} = (Q, E, I, F)$ over $A^* \times B^*$, a set of states $\{q_1, \cdots, q_k\} \subseteq Q$ and a function $I : \{q_1, \cdots, q_k\} \to \Sigma^*$, we define the relation over $A^* \times \Sigma^* B^*$

$$IR_{\mathcal{T}} = \left\{ (u, v) \in A^* \times \Sigma^* B^* \;\middle|\; v = I(q_i)w \text{ where } q_i \xrightarrow[\mathcal{T}]{u|w} f \text{ is a succesful path} \right\}$$

▶ **Lemma 3.10** (Uniformizing continuations). *Let* $\mathcal{T} = \bigcup_{i=1}^{n} D_j$ *be the union of* l *deterministic transducers. Let* $u_1, u_2 \in A^*$ *such that* $\begin{cases} D_1 & p_1 & \xrightarrow{u_1|v_1} & q_1 & \xrightarrow{u_2|w_1} & r_1 \\ \vdots & \vdots & & \vdots & & \vdots \\ D_l & p_l & \xrightarrow{u_1|v_l} & q_l & \xrightarrow{u_2|w_l} & r_l \end{cases}$ *We define*

$I_1 : \left\{ \begin{array}{rcl} \{q_1, \cdots, q_l\} & \to & B^* \\ q_i & \mapsto & \Delta(\{v_j\}_{1\le j\le l})(i) \end{array} \right\}$, $I_2 : \left\{ \begin{array}{rcl} \{r_1, \cdots, r_l\} & \to & B^* \\ q_i & \mapsto & \Delta(\{v_j w_j\}_{1\le j\le l})(i) \end{array} \right\}$.

If $I_1 \bigcup_{j=1}^{l} D_j$ *is uniformizable, then so is* $I_2 \bigcup_{j=1}^{l} D_j$.

Proof. The idea is to use a uniformizer for $I_1 \bigcup_{j=1}^{l} D_j$, and start from $q = i \bullet u_1$. ◀

In the following, we generalize the Twinning Property defined by Choffrut, which characterizes the subsequential functions (see e.g [2]).

3.2 Generalized Twinning Property

In this whole section, we assume without loss of generality that all states are coaccessible. Here, we give a characterization of the uniformizable transducers in ∪DFT: when delays explode, we must be able to drop some runs and recursively uniformize the chosen subset.

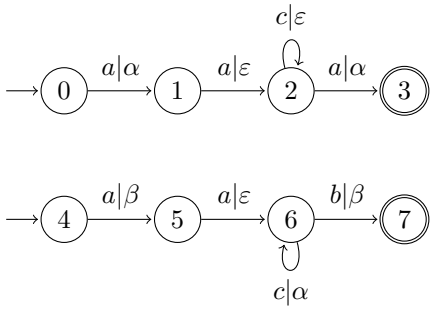


Figure 4 A union of two deterministic transducers which is not uniformizable

▶ **Example 3.11.** We invite the reader to try and grasp the necessity of dropping a run with fig. 4: the loops over states 2 and 6 forces the uniformizer to drop either the first transducer

or the second. Otherwise, it will have to memorize infinitely many α to output, which is not possible. However, it can't, since dropping the first will make him lose the possibility to cover the words finishing by a, and conversely. Thus, this transducer isn't uniformizable. A sufficient condition here would be that one of the two cover all the continuations after the state 2 (resp. 6).

The GTP is a generalization of this observation.

▶ **Definition 3.12** (Generalized Twinning Property). Let $\mathcal{T} = \bigcup_{i=1}^{n} D_i$ be the union of n deterministic transducers over $A \times B^*$. Let $I_1 : \left\{ \begin{array}{rcl} \{q_1, \cdots, q_k\} & \to & B^* \\ q_i & \mapsto & \Delta(\{v_1, \cdots, v_l\})(i) \end{array} \right\}$ and $I_2 : \left\{ \begin{array}{rcl} \{q_1, \cdots, q_k\} & \to & B^* \\ q_i & \mapsto & \Delta(\{v_1 w_1, \cdots, v_l w_l\})(i) \end{array} \right\}$. We say that $\mathcal{T}$ satisfies the Generalized Twinning Property if:

$$\forall u_1, u_2 \in A^*, \text{ if } \left\{ \begin{array}{llllll} D_{i_1} & p_1 & \xrightarrow{u_1|v_{1,1}} & q_1 & \xrightarrow{u_2|v_{2,1}} & q_1 \\ \vdots & \vdots & & \vdots & & \vdots \\ D_{i_k} & p_k & \xrightarrow{u_1|v_{1,k}} & q_k & \xrightarrow{u_2|v_{2,k}} & q_k \\ \\ \forall j \notin \{i_1, \ldots, i_k\}, \text{there is no run over } u_1u_2 \text{ in } D_j & & & & & \end{array} \right. \quad \text{where } p_1, \ldots, p_k$$

are initial, and if delays mismatch, *ie* $I_1 \neq I_2$, then there exists $P \subsetneq \{i_1, \ldots, i_k\}$ such that:

1. $\bigcup_{j=1}^{k} \mathrm{dom}(\mathcal{T}_{q_j}) = \bigcup_{j \in P} \mathrm{dom}(\mathcal{T}_{q_j})$
2. $I_1' \bigcup_{j \in P} D_j$ is uniformizable, where $I_1' : \left\{ \begin{array}{rcl} \{q_j\}_{j \in P} & \to & B^* \\ q_i & \mapsto & \Delta(\{v_{1,j}\}_{j \in P})(i) \end{array} \right\}$

▶ **Theorem 3.13** (Generalized Twinning Property). *Let $\mathcal{T} = \bigcup_{i=1}^{d} D_i$ be the union of d deterministic transducers over $A \times B^*$. $\mathcal{T}$ is uniformizable by a subsequential transducer* iff *it satisfies the Generalized Twinning Property.*

In the proof, we will need the following lemmas:

3.2.1 Preliminary lemmas

▶ **Lemma 3.14** (A bit of combinatorics on words). *Let $u_1, u_2, u_3, v_1, v_2, v_3 \in \Sigma^*$. If $u_1 u_2^l u_3 = v_1 v_2^l v_3$ holds for infinitely many l, then it is true for all l.*

Proof. This lemma is a consequence of the Fine and Wilf theorem, see e.g. [1], and of results proven in [13].

To give a quick sketch of the proof: we first show that u_2 and v_2 are conjugate, and then we just have to examine the different cases. ◀

▶ **Lemma 3.15** (Prefixes). *For any subsequential relation, we can either delete a common prefix or add any word. Formally, let $\mathcal{T}$ be a subsequential transducer over $A \times B^*$, and $v \in B^* + B^{-1^*}$. If $(\varepsilon, v) R_{\mathcal{T}} \subseteq A^* \times B^*$, then it is recognized by a subsequential transducer.*

We first give a quick but non constructive proof:

Proof. Here, we recall the distance over A^* of definition 1.6: $\forall u, v \in A^*, d(u,v) = |u| + |v| - 2|u \wedge v|$. Now, we state a characterization of subsequential relations:

▶ **Definition 3.16** (Bounded variation)**.** We say that f has bounded variation when:

$$\forall k \geq 0, \exists K \geq 0, \forall (u,v) \in \mathrm{dom}(f), d(u,v) \leq k \Rightarrow d(f(u), f(v)) \leq K$$

Thanks to **Proposition 7** of [2], we know that f is subsequential *iff* it has bounded variation. $R_{\mathcal{T}}$ is subsequential, so it has bounded variation. For all $(u,v) \in \mathrm{dom}(\mathcal{T})$, we get

$$\begin{aligned} d((\varepsilon,\delta)f_{\mathcal{T}}(u), (\varepsilon,\delta)f_{\mathcal{T}}(v)) &= |(\varepsilon,\delta)f_{\mathcal{T}}(u)| + |(\varepsilon,\delta)f_{\mathcal{T}}(v)| - 2|(\varepsilon,\delta)f_{\mathcal{T}}(u) \wedge (\varepsilon,\delta)f_{\mathcal{T}}(v)| \\ &= |f_{\mathcal{T}}(u)| + |f_{\mathcal{T}}(v)| - 2|f_{\mathcal{T}}(u) \wedge f_{\mathcal{T}}(v)| + 2|\delta| \end{aligned}$$

with the following convention: if $\delta \in B^{-1^*}, |\delta| = -|\delta|_{B^{-1^*}}$, since δ removes letters (here, the fact that δ is prefix to $f_{\mathcal{T}}(u)$ and $f_{\mathcal{T}}(v)$ is crucial).

Now, let $k \geq 0$, and $K \geq 0$ the associated modulus of variation. We have: $\forall (u,v) \in \mathrm{dom}(\mathcal{T}), d(u,v) \leq k \Rightarrow d((\varepsilon,\delta)f_{\mathcal{T}}(u), (\varepsilon,\delta)f_{\mathcal{T}}(v)) \leq K + 2|\delta|$, so $(\varepsilon,\delta)f_{\mathcal{T}}$ admits $K + 2|\delta|$ as a modulus of variation for k: $(\varepsilon,\delta)f_{\mathcal{T}}$ has bounded variation.

In conclusion, $(\varepsilon,\delta)f_{\mathcal{T}}$ is subsequential. ◀

▶ Remark. A second proof, where we show how the corresponding transducer can be built, is presented in the appendix at page 22.

We can now prove our statement.

Proof. 3.2.2 The condition is necessary

We assume that $\mathcal{T}$ is uniformizable by a subsequential transducer $\mathcal{D} = (Q, E, \{i_0\}, F, \rho)$.

Now, let $u_1, u_2 \in A^*$ such that $\begin{cases} D_{i_1} \quad p_1 \xrightarrow{u_1|v_{1,1}} q_1 \xrightarrow{u_2|v_{2,1}} q_1 \\ \vdots \quad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ D_{i_k} \quad p_k \xrightarrow{u_1|v_{1,k}} q_k \xrightarrow{u_2|v_{2,k}} q_k \\ \\ \forall j \notin \{i_1, \ldots, i_k\}, \text{there is no run over } u_1u_2 \text{ in } D_j \end{cases}$ and $I_1 \neq I_2$, *ie* $\Delta(\{v_{1,1}, \ldots, v_{1,k}\}) \neq \Delta(\{v_{1,1}v_{2,1}, \ldots, v_{1,k}v_{2,k}\})$.

Let $P = \{i_1, \ldots, i_k\} \backslash \{i | D(w_1, v_{1,i}) \neq D(w_1w_2, v_{1,i}v_{2,i})\}$.

A particular case: the uniformizer also loops

We first assume that $\mathcal{D}$ and the D_i loop simultaneously: $i_0 \xrightarrow[\mathcal{D}]{u_1|w_1} q \xrightarrow[\mathcal{D}]{u_2|w_2} q$ for some $q \in Q$. Let us show that P satisfies the needed properties:

Dropping runs We first show that $P \subsetneq \{i_1, \ldots, i_k\}$:

$\Delta(\{v_{1,1}, \ldots, v_{1,k}\}) \neq \Delta(\{v_{1,1}v_{2,1}, \ldots, v_{1,k}v_{2,k}\})$, so, thanks to lemma 3.7, we can find m, n such that $D(v_{1,m}, v_{1,n}) \neq D(v_{1,m}v_{2,m}, v_{1,n}v_{2,n})$. We show that either $D(w_1, v_{1,m}) \neq D(w_1w_2, v_{1,m}v_{2,m})$, or $D(w_1, v_{1,n}) \neq D(w_1w_2, v_{1,n}v_{2,n})$: if $D(w_1, v_{1,m}) = D(w_1w_2, v_{1,m}v_{2,m})$ *ie* $v_{1,m}^{-1}w_1 = v_{2,m}^{-1}v_{1,m}^{-1}w_1w_2$, then $v_{2,n}^{-1}v_{1,n}^{-1}w_1w_2 = v_{2,n}^{-1}v_{1,n}^{-1}v_{1,m}v_{2,m}v_{1,m}^{-1}w_1 \neq v_{1,n}^{-1}v_{1,m}v_{1,m}^{-1}w_1 = v_{1,n}^{-1}w_1$. So $D(w_1, v_{1,n}) \neq D(w_1w_2, v_{1,n}v_{2,n})$. Consequently, either $m \notin P$, or $n \notin P$. It implies that $P \subsetneq \{i_1, \ldots, i_k\}$.

The continuations are the same Now, we prove that $\bigcup_{j=1}^{k} \mathrm{dom}(\mathcal{T}_{q_j}) = \bigcup_{j\in P} \mathrm{dom}(\mathcal{T}_{q_j})$.

The following lemma is stronger than needed, but will be useful after. It implies that the continuations are the same.

▶ **Lemma 3.17** (Choice). $\forall (u_3, w_3) \in R_{\mathcal{D}_q}, \exists i \in P, \exists v_{3,i} \in B^*, (u_3, v_{3,i}) \in R_{D_{iq_i}} \wedge \forall k \in \mathbb{N}, w_1 w_2^k w_3 = v_{1,i} v_{2,i}^k v_{3,i}$

In other words, for all continuations, the uniformizer choses one deterministic transducer among the ones in P and sticks to it.

Proof. We define a sequence $\{u_l\}_{l\in\mathbb{N}} \in \{1,\dots,k\}^{\mathbb{N}}$: $\forall l \in \mathbb{N}, u_l = i \in \{1,\dots,k\}, w_1 w_2^l w_3 = v_{1,i} v_{2,i}^l v_{3,i}$. $\{u_l\}$ takes its values in a finite set $\{1,\dots,k\}$, so $\exists i \in \{1,\dots,k\}, u_l = i$ for infinitely many l. Consequently, for infinitely many l, $v_{1,i} v_{2,i}^l v_{3,i} = w_1 w_2^l w_3$. Thanks to lemma 3.14, we can deduce that this identity holds for all $l \in \mathbb{N}$. For $l = 0, 1$, we get $v_{1,i} v_{3,i} = w_1 w_3$ and $v_{1,i} v_{2,i} v_{3,i} = w_1 w_2 w_3$.

We just have to show that $i \in P$: to this end, we show that if $v_{1,i} v_{3,i} = w_1 w_3$ and $v_{1,i} v_{2,i} v_{3,i} = w_1 w_2 w_3$, then $i \in P$. Indeed, $w_3 = w_1^{-1} v_{1,i} v_{3,i} = w_2^{-1} w_1^{-1} v_{1,i} v_{2,i} v_{3,i}$, and then $w_1^{-1} v_{1,i} = w_2^{-1} w_1^{-1} v_{1,i} v_{2,i}$, which means that $D(w_1, v_{1,i}) = D(w_1 w_2, v_{1,i} v_{2,i})$. In conclusion, $i \in P$. ◀

Consequently, $\forall (u_3, w_3) \in R_{\mathcal{D}_q}, \exists i \in P, (u_3, v_{3,i}) \in R_{D_{iq_i}}$, otherwise $\mathcal{D}$ couldn't find an image to stick to. We get $\forall u_3 \in \mathrm{dom}(\mathcal{D}_q), \exists i \in P, u_3 \in \mathrm{dom}(D_{iq_i})$, which means that $\bigcup_{j\in P} \mathrm{dom}(\mathcal{T}_{q_j}) \supseteq \mathrm{dom}(\mathcal{D}_q) \supseteq \bigcup_{i=1}^{k} \mathrm{dom}(\mathcal{T}_{q_j})$ (because $\mathcal{D}$ uniformizes $\mathcal{T}$).

The rest is uniformizable We finally show that $\mathcal{R} = I_1' \bigcup_{i\in P} D_i$ is uniformizable. We want to uniformize the following relation, where $\delta = \left(\bigwedge_{j\in P} v_{1,j}\right)$:

$$\begin{aligned}\mathcal{R} &= \left\{(u,v) \;\middle|\; v = \Delta(\{v_{1,i}\}_{i\in P})(i)w \text{ where } q_i \xrightarrow[D_{i\in P}]{u|w} f\right\} \\ &= \left\{(u,v) \;\middle|\; v = \delta^{-1} v_{1,i} w \text{ where } q_i \xrightarrow[D_{i\in P}]{u|w} f\right\}\end{aligned}$$

Let $(u_3, w_3) \in R_{\mathcal{D}_q}$. By virtue of lemma 3.17, $\exists i \in P, \forall k \in \mathbb{N}, w_1 w_2^k w_3 = v_{1,i} v_{2,i}^k v_{3,i}$. In particular, $w_1 w_3 = v_{1,i} v_{3,i}$. Consequently, $\delta^{-1} w_1 w_3 = \delta^{-1} v_{1,i} v_{3,i}$, and, since δ is a prefix of $v_{1,i}$, we get $\delta^{-1} w_1 w_3 \in B^*$: $\delta^{-1} w_1 R_{\mathcal{D}_q} \subseteq A^* \times B^*$, so, thanks to lemma 3.15, $\delta^{-1} w_1 R_{\mathcal{D}_q}$ is recognized by $\mathcal{D}'$ a subsequential transducer. Let us show that $\mathcal{D}'$ realizes $\mathcal{R}$:

First, $\mathrm{dom}(\mathcal{D}') = \mathrm{dom}(\mathcal{D}) = \mathrm{dom}(\mathcal{D}_q) = \bigcup_{i=1}^{k} \mathrm{dom}(\mathcal{T}_{q_i}) = \mathrm{dom}(\mathcal{R})$

Now, if $(u_3, v) \in \delta^{-1} w_1 R_{\mathcal{D}_q}$, then let $w_3 \in I_{q,\mathcal{D}}$ such that $v = \delta^{-1} w_1 w_3$. Since $(u_3, w_3) \in R_{\mathcal{D}_q}$, lemma 3.17 yields $\exists i \in P, \forall k \in \mathbb{N}, w_1 w_2^k w_3 = v_{1,i} v_{2,i}^k v_{3,i}$. Then, $\exists i \in P, w_1 w_3 = v_{1,i} v_{3,i}$ *ie* $\exists i \in P, v = \delta^{-1} v_{1,i} v_{3,i}$, with $q_i \xrightarrow[D_{i\in P}]{u_3|v_{3,i}}$. This precisely means $(u_3, \delta^{-1} v_{1,i} v_{3,i}) \in \mathcal{R}$, so we can deduce that $(u_3, v) \in \mathcal{R}$.

In conclusion, $\mathcal{D}'$ realizes $\mathcal{R}$.

The general case

Now, we have to treat the general case: $\mathcal{D}$ and the D_i doesn't necessarily loop simultaneously, and we have $i_0 \xrightarrow[\mathcal{D}]{u_1|w_1} r \xrightarrow[\mathcal{D}]{u_2|w_2} s$. However, by pumping, $\exists l, m \in \mathbb{N}$ such that

$$\begin{cases} D_{i_1} \quad p_1 \xrightarrow{u_1u_2^l|v_{1,1}v_{2,1}^l} q_1 \xrightarrow{u_2^m|v_{2,1}^m} q_1 \\ \quad\vdots \quad\quad \vdots \qquad\qquad\quad \vdots \qquad\quad\; \vdots \\ D_{i_k} \quad p_k \xrightarrow{u_1u_2^l|v_{1,k}v_{2,k}^l} q_k \xrightarrow{u_2^m|v_{2,k}^m} q_k \\ \mathcal{D} \quad\; p \xrightarrow{u_1u_2^l|w_1'} q \xrightarrow{u_2^m|w_2'} q \\ \forall j \notin \{i_1, \dots, i_k\}, \text{there is no run over } u_1u_2 \text{ in } D_j \end{cases}$$

Here, $\mathcal{D}$ and the D_i loops simultaneously over u_2^m. Thus, $\exists P \subsetneq \{i_1, \cdots, i_k\}$ such that:

1. $\bigcup_{j=1}^{k} \mathrm{dom}(\mathcal{T}_{q_j}) = \bigcup_{j\in P} \mathrm{dom}(\mathcal{T}_{q_j})$
2. $I_1'' \bigcup_{j\in P} D_j$ is uniformizable, where $I_1'' : \left\{ \begin{array}{rcl} \{q_1, \cdots, q_k\} & \to & B^* \\ q_i & \mapsto & \Delta(\{v_{1,j}v_{2,j}^l\}_{j\in P})(i) \end{array} \right\}$

We just have to show that the P we defined suits us. We already showed that $P \subsetneq \{i_1, \cdots, i_k\}$ and the continuations are the same. We just have to realize $\mathcal{R} = I_1' \bigcup_{j\in P} D_j = \left\{(u, v) \mid v = \delta^{-1}v_{1,i}w \text{ where } q_i \xrightarrow[D_{i\in P}]{u|w} f\right\}$ With what preceeds, we know that $\mathcal{R}' = I_1'' \bigcup_{j\in P} D_j = \left\{(u, v) \mid v = \delta'^{-1}v_{1,i}v_{2,i}^l w \text{ where } q_i \xrightarrow[D_{i\in P}]{u|w} f\right\}$ (where $\delta' = \bigwedge_{j\in P} v_{1,j}v_{2,j}^l$) is uniformizable. Since $\Delta(\{v_{1,j}v_{2,j}^l\}_{j\in P}) = \Delta(\{v_{1,j}v_{2,j}^l v_{2,j}^m\}_{j\in P})$, we get, thanks to lemma 3.7, $\Delta(\{v_{1,j}\}_{j\in P}) = \Delta(\{v_{1,j}v_{2,j}^l\}_{j\in P})$, *ie* $I_1'' = I_1'$. Consequently, $\mathcal{R} = \mathcal{R}'$: $\mathcal{R}$ is uniformizable.

The GTP is thus necessary.

3.2.3 The condition is sufficient

We now prove that it is sufficient: we extend the subset construction with delays and, when they explode, we branch to a uniformizer of a subset which preserves the delays.

Bounding the delays

▶ **Lemma 3.18** (Catching up the delays). *If the GTP holds, then, for all $u \in A^*$, if we have*

$$\begin{cases} D_{i_1} \quad p_1 \xrightarrow{u|w_1} q_1 \\ \quad\vdots \quad\quad \vdots \qquad\quad \vdots \\ D_{i_k} \quad p_k \xrightarrow{u|w_k} q_k \\ \forall j \notin \{i_1, \dots, i_k\}, \textit{there is no run over } u \textit{ in } D_j \end{cases} \quad \textit{and if } \exists i \in \{1, \dots, k\}, |\Delta(\{w_j\}_{1\le j\le k})(i)| >$$

Mn^d (where d is the number of deterministic transducers in $\mathcal{T}$), then $\exists P \subsetneq \{i_1, \cdots, i_k\}$ s.t.

1. $\bigcup_{j=1}^{k} \mathrm{dom}(\mathcal{T}_{q_j}) = \bigcup_{j\in P} \mathrm{dom}(\mathcal{T}_{q_j})$
2. *$I \bigcup_{j\in P} D_j$ is uniformizable, where* $I : \left\{ \begin{array}{rcl} \{q_1, \cdots, q_k\} & \to & B^* \\ q_i & \mapsto & \Delta(\{w_j\}_{j\in P})(i) \end{array} \right\}$

Proof. Let $i \in \{1,\dots,k\}$ such that $|\Delta(\{w_1,\cdots,w_k\})(i)| > Mn^d$. Then $|w_i| > Mn^d$, so $|u| > n^d$. Consequently, all the D_i loops simultaneously, *ie*

1. $\exists u_1,u_2,u_3 \in A^*$ such that $|u_1u_3| \leq n^d$, $u = u_1u_2u_3$
2. $\exists v_{1,1},\cdots,v_{1,k},v_{2,1},\cdots v_{2,k},v_{3,1},\cdots v_{3,k} \in B^*$ such that $\forall i \in \{1,\dots,k\}, w_i = v_{1,i}v_{2,i}v_{3,i}$

and
$$\begin{cases} D_{i_1} \quad p_1 \xrightarrow{u_1|v_{1,1}} q_1 \xrightarrow{u_2|v_{2,1}} q_1 \xrightarrow{u_3|v_{3,1}} r_1 \\ \quad\vdots \quad\quad \vdots \quad\quad\quad\quad \vdots \quad\quad\quad\quad \vdots \\ D_{i_k} \quad p_k \xrightarrow{u_1|v_{1,k}} q_k \xrightarrow{u_2|v_{2,k}} q_k \xrightarrow{u_3|v_{3,k}} r_k \\ \\ \forall j \notin \{i_1,\dots,i_k\}, \text{there is no run over } u_1u_2u_3 \text{ in } D_j \end{cases}$$

Now, we have $\Delta(\{v_{1,1},\cdots,v_{1,k}\}) \neq \Delta(\{v_{1,1}v_{2,1},\cdots,v_{1,k}v_{2,k}\})$, otherwise $\Delta(\{w_1,\cdots,w_k\}) = \Delta(\{v_{1,1}v_{2,1}v_{3,1},\cdots,v_{1,k}v_{2,k}v_{3,k}\}) = \Delta(\{v_{1,1}v_{3,1},\cdots,v_{1,k}v_{3,k}\})$, and then, since $|u_1u_3| \leq n^d$, we get $\forall i \in \{1,\dots,k\}, |\Delta(\{w_1,\cdots,w_k\})| \leq Mn^d$. Consequently, we can apply the generalized twinning property: let $P \subsetneq \{i_1,\cdots,i_k\}$ such that:

1. $\bigcup_{j=1}^{k} \mathrm{dom}(\mathcal{T}_{q_j}) = \bigcup_{j\in P} \mathrm{dom}(\mathcal{T}_{q_j})$
2. $\left(\Delta(\{v_{1,j}\}_{j\in P}) \bigcup_{j\in P} D_j\right)$ is uniformizable

Thanks to lemma 3.10, $\Delta(\{w_j\}_{j\in P}) \bigcup_{j\in P} D_j$ is uniformizable. ◀

Building a uniformizer

The construction We now build a uniformizer $\mathcal{U} = (Q,E,\{i\},F)$ for $\mathcal{T} = \bigcup_{i=1}^{n} D_i$ by extending the subset constructions with delays:

- The states are tuples $(p_1 : \delta_1,\cdots,p_k : \delta_k)$, where the $p_i \in Q_i^\circ = Q_i \cup \{\{\emptyset\}\}$ and $\delta_i \in B^{Mn^k}$. We represent it by $Q = (Q_1^\circ \times \cdots \times Q_k^\circ) \times \left(B^{Mn^k}\right)^k$
- $i = (i_1 : \varepsilon,\cdots,i_k : \varepsilon)$
- F is the set of the tuples that contain at least one final state: $F = \{t \in Q \mid \exists i \in \{1,\dots,k\}, \pi_i(t) \in F_i\}$, where π_i is the i-th projection.

For readability concerns, for building the transitions, we will consider the D_i', which are the completed D_i, where the sink state is denoted by $\{\emptyset\}$.

The transitions are:

Let $\delta = \delta_1 \wedge \cdots \wedge \delta_k \wedge v_1 \wedge \cdots \wedge v_k$, where for all $i \in \{1,\dots,k\}$, $p_i \xrightarrow[D_i']{a|v_i} q_i$ (we can notice that if there is no run labeled by a in D_i or if $p_i = \{\emptyset\}$, then we have $p_i \xrightarrow[D_i']{a|\varepsilon} \{\emptyset\}$).

- If $\forall i \in \{1,\dots,k\}$, $|\delta^{-1}\delta_i v_i| \leq Mn^k$, then we add $(p_1 : \delta_1,\cdots,p_k : \delta_k) \xrightarrow[\mathcal{U}]{a|\delta} (q_1 : \delta^{-1}\delta_1 v_1,\cdots,q_k : \delta^{-1}\delta_k v_k)$
- Else, it means that the delays exceeded Mn^k. So, thanks to lemma 3.18, $\exists P \subsetneq \{i_1,\cdots,i_k\}$ such that:
  1. $\bigcup_{j=1}^{k} \mathrm{dom}(\mathcal{T}_{q_j}) = \bigcup_{j\in P} \mathrm{dom}(\mathcal{T}_{q_j})$
  2. $\Delta(\{\delta_i v_i\}_{j\in P}) \bigcup_{j\in P} D_j$ is uniformizable

  Then, let $\mathcal{V}$ be a uniformizer for $\Delta(\{\delta_j v_j\}_{j\in P}) \bigcup_{j\in P} D_j$. We add the transition $(p_1 : \delta_1,\cdots,p_k : \delta_k) \xrightarrow[\mathcal{U}]{a|\varepsilon} i_{\mathcal{V}}$ where $i_{\mathcal{V}}$ is the initial state of $\mathcal{V}$.

Finally, for all final states $f = (p_1 : \delta_1, \cdots, f_i : \delta_i, \cdots, p_k : \delta_k)$, we define $\rho(f) = \delta_i$. If the tuple contains multiple final states, we just pick one.

Validity of our construction We now show that $\mathcal{U}$ is a proper uniformizer:

- Clearly by construction, $\mathcal{U}$ is deterministic.
- Now, we show that $\mathrm{dom}(\mathcal{T}) = \mathrm{dom}(\mathcal{U})$, and that $\mathcal{R}_\mathcal{U} \subseteq \mathcal{R}_\mathcal{T}$. Let $u = u_0 \cdots u_l \in \mathrm{dom}(\mathcal{T})$. If delays don't explode, then we have, for the same reasons as in theorem 2.5, $(i_1 : \varepsilon, \cdots, i_k : \varepsilon) \xrightarrow[\mathcal{U}]{u_0|w_0} (q_{0,1} : \delta_{0,1}, \cdots, q_{0,k} : \delta_{0,k}) \xrightarrow[\mathcal{U}]{u_1|w_1} \cdots \xrightarrow[\mathcal{U}]{u_l|w_l} (q_{l,1} : \delta_{l,1}, \cdots, f_i : \delta_{l,i}, \cdots, q_{l,k} : \delta_{l,k}) \xrightarrow[\mathcal{U}]{\delta_{l,i}}$, where $i_i \xrightarrow[D_i]{u_0|v_0} q_{1,i} \xrightarrow[D_i]{u_1|v_1} \cdots \xrightarrow[D_i]{u_l|w_l} f_i \xrightarrow[D_i]{w_{l+1}}$, and $w_0 \cdots w_l \delta_{l,i} = v_0 \cdots v_l v_{l+1}$. Otherwise, $(i_1 : \varepsilon, \cdots, i_k : \varepsilon) \xrightarrow[\mathcal{U}]{u_0|w_0} (q_{0,1} : \delta_{0,1}, \cdots, q_{0,k} : \delta_{0,k}) \xrightarrow[\mathcal{U}]{u_1|w_1} \cdots \xrightarrow[\mathcal{U}]{u_m|w_m} (q_{m,1} : \delta_{m,1}, \cdots, q_{m,k} : \delta_{m,k}) \xrightarrow[\mathcal{U}]{u_{m+1}|v_{m+1}} \mathcal{V}$, where $\mathcal{V}$ uniformizes $\mathcal{R}_\mathcal{V} = \Delta(\{\delta_j v_j\}_{j \in P}) \bigcup_{j \in P} D_j$. Now, since P covers all the continuations, $\exists i \in P, (u_{m+1} \cdots u_l, v_{m,i} v_{m+1,i} \cdots v_{l,i} v_{l+1,i}) \in \mathcal{R}_{D_{i_{q_i}}}$, which means that $(u_{m+1} \cdots u_l, \delta^{-1} \delta_{m,i} v_{m,i} v_{m+1,i} \cdots v_{l,i} v_{l+1,i}) \in \mathcal{R}_\mathcal{V}$. Consequently, if we add the beginning, we get $(u_0 \cdots u_m u_{m+1} \cdots u_l, w_0 \cdots w_m \varepsilon \delta^{-1} \delta_{m,i} v_{m,i} v_{m+1,i} \cdots v_{l,i} v_{l+1,i}) \in \mathcal{R}_\mathcal{U}$, and we know that $w_0 \cdots w_m \delta^{-1} \delta_{m,i} = v_{0,i} \cdots v_{m,i}$.
  We finally have $(u_0 \cdots u_m u_{m+1} \cdots u_l, w_0 \cdots w_m \varepsilon \delta^{-1} \delta_{m,i} v_{m,i} v_{m+1,i} \cdots v_{l,i} v_{l+1,i}) \in \mathcal{R}_\mathcal{U}$. Conversely, for similar reasons, if $(u, v) \in \mathcal{R}_\mathcal{U}$, then $(u, v) \in \mathcal{R}_\mathcal{T}$.

Thus, $\mathcal{T}$ is realized by $\mathcal{U}$. ◀

3.3 Decidability of the GTP

▶ **Theorem 3.19.** *The GTP is decidable.*

Proof. Since we can bound the delays by Mn^d in our proof, the GTP holds *iff* it holds for $u_i \in A^{Mn^d}$. Consequently, we just have to check it for every word of length less than Mn^d, and recursively. We conjecture that this verification can be done in PSpace. ◀

From this, we can deduce the main result of this work:

3.4 Main result

▶ **Theorem 3.20** (Church(∪DFT, DFT))**.** *Church*$(\cup DFT, DFT)$ *is decidable.*

4 Conclusion

4.1 Our contribution

In this work, we solved two subcases of Church(NFT, DFT): with $\mathsf{Church}_k(\mathrm{NFT}, \mathrm{DFT})$, we forced the uniformizer to output his answer within a certain delay. In Church(∪DFT, DFT), we focused on the union of deterministic transducers.

4.2 Future work

The study of the GTP gave us an intuition of a generalization to the case of the union of functional transducers Church(∪FFT, DFT). Thanks to [25], we know that $\bigcup_{i=1}^{k} FFT =$ k-valued, where the k-valued transducers are the transducers that represent the relations

with at most k images for a given word. Consequently, this would allow us to decide $\mathsf{Church}(k\text{-valued}, \mathrm{DFT})$

Next, we will study the general case: is it undecidable, as we conjectured, or will we be able to find a suitable algorithm?

We also need lower bounds. *A priori*, our upper bounds are tight: $\mathsf{Church}_k(\mathrm{NFT}, \mathrm{DFT})$ is ExpTime-complete, and $\mathsf{Church}(\cup\mathrm{DFT}, \mathrm{DFT})$ is PSpace-complete.

Finally, we will explore a more qualitative area, which is the measurement of the quality of uniformizers, by adding some constrains over the number of states, or by favoring some paths over others by adding a weight to the transitions.

References

1. J. Autebert, J. Berstel, and L. Boasson. Context-free languages and pushdown automata. In , *Handbook of formal languages, vol. 1*, pages 111–174. Springer-Verlag, 1997.
2. M.-P. Béal and O. Carton. Determinization of transducers over finite and infinite words. *Theoretical Computer Science*, 289(1):225–251, 2002.
3. J. Berstel. Transductions and context-free languages `http://www-igm.univ-mlv.fr/~berstel/`, Dec. 2009.
4. M. Bojanczyk. Transducers with origin information. *ICALP*, abs/1309.6124, 2013.
5. J. R. Buchi and L. H. Landweber. Solving Sequential Conditions by Finite-State Strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969.
6. A. Church. Logic, arithmetic and automata. In *Int. Congr. Math.*, pages 23–35, Stockholm, 1962.
7. E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In , *Logics of Programs — Proceedings 1981 (LNCS Volume 131)*, pages 52–71. Springer-Verlag: Heidelberg, Germany, 1981.
8. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, Cambridge, Massachusetts, 1999.
9. R. de Souza. Uniformisation of two-way transducers. In *LATA*, pages 547–558, 2013.
10. R. Ehlers. Symbolic bounded synthesis. In *Computer Aided Verification, 22nd International Conference, CAV 2010, Edinburgh, UK, July 15-19, 2010. Proceedings*, volume 6174 of *LNCS*, pages 365–379. Springer, 2010.
11. E. Filiot, O. Gauwin, P.-A. Reynier, and F. Servais. Height-bounded memory visibly pushdown transductions. In *Submitted*, 2011.
12. E. Filiot, N. Jin, and J.-F. Raskin. Antichains and compositional algorithms for LTL synthesis. *Formal Methods in System Design*, 39(3):261–296, 2011.
13. E. Filiot, J.-F. Raskin, P.-A. Reynier, F. Servais, and J.-M. Talbot. On functionality of visibly pushdown transducers. *CoRR*, abs/1002.1443, 2010.
14. E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, Logics, and Infinite Games*, volume 2500 of *LNCS*. Springer, 2002.
15. M. Holtmann, L. Kaiser, and W. Thomas. Degrees of lookahead in regular infinite games. *Logical Methods in Computer Science*, 8(3), 2012.
16. B. Jobstmann and R. Bloem. Optimizations for LTL synthesis. In *FMCAD*, pages 117–124. IEEE Computer Society, 2006.
17. B. Jobstmann, S. Galler, M. Weiglhofer, and R. Bloem. Anzu: A tool for property synthesis. In *Computer Aided Verification (CAV)*, pages 258–262, 2007.
18. A. Pnueli. The temporal logic of programs. In *focs77*, pages 46–57, 1977.
19. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *ACM Symposium on Principles of Programming Languages (POPL)*. ACM, 1989.

20 Queille and Sifakis. A temporal logic to deal with fairness in transition systems. In *FOCS: IEEE Symposium on Foundations of Computer Science (FOCS)*, 1982.

21 S. Schewe and B. Finkbeiner. Bounded synthesis. In *Automated Technology for Verification and Analysis*, volume 4762 of *LNCS*, pages 474–488. Springer Berlin / Heidelberg, 2007.

22 W. Thomas. Facets of synthesis: Revisiting church's problem. In , *FOSSACS*, volume 5504 of *LNCS*, pages 1–14. Springer, 2009.

23 W. Thomas. Synthesis and some of its challenges. In *CAV*, page 1, 2012.

24 M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *lics86*, pages 332–344, 1986.

25 A. Weber. Decomposing finite-valued transducers and deciding their equivalence. *SIAM Journal on Computing*, 22(1):175–202, 1993.

Appendix

A Proofs

A.1 Proof of lemma 2.12

▶ **Lemma 1.1.** $\forall i \in \{0,\dots,n+1\}, q_i \in P_i$ *and* $(v'_0 \dots v'_{i-1})^{-1} v_0^{(k-(i-1))} v_1^{(k-(i-2))} \dots v_{i-1}^{(k)} \in \Delta_{q_i}$

Proof. We show the property by induction on n:

- If $i = 0$, then $q_0 \in P_0 = I$, and we have nothing to check for the output.
- If $0 \leq i \leq n$, then we are in (P_i, Q_i, d_i, I), Player I inputs u_i, and O answers v'_i, which leads to $(P_{i+1}, Q_{i+1}, d_{i+1}, I)$. Since $q_i \xrightarrow[\mathcal{T}]{u_i|v_i} q_{i+1}$ is a valid transition, $D_{q_{i+1}} = \text{lshift}\left(v_i'^{-1} \Delta_{q_i} v_i^{(k+1)}\right)$ is a candidate for being in $\Delta_{q_{i+1}}$. By the induction hypothesis, $(v'_0 \dots v'_{i-1})^{-1} v_0^{(k-(i-1))} \dots v_{i-1}^{(k)} \in \Delta_{q_i}$, so $v_i'^{-1} (v'_0 \dots v'_{i-1})^{-1} v_0^{(k-(i-1))} \dots v_{i-1}^{(k)} v_i^{(k+1)} \in v_i'^{-1} \Delta_{q_i} v_i^{(k+1)}$, from which we get $(v'_0 \dots v'_i)^{-1} v_0^{(k-(i-1))} \dots v_i^{(k+1)} \in v_i'^{-1} \Delta_{q_i} v_i^{(k+1)}$, so $(v'_0 \dots v'_i)^{-1} v_0^{(k-i)} \dots v_i^{(k)} \in \text{lshift}\left(v_i'^{-1} \Delta_{q_i} v_i^{(k+1)}\right) = D_{q_{i+1}}$. Moreover, since we have $\forall j \in \{0,\dots,|v|\}, |o(j) - o'(j)| \leq k$, the last letter of v'_i has at most k delay towards v, so $(v'_0 \dots v'_i)^{-1} v_0^{(k-(i+1-1))} \dots v_i^{(k)} \in \mathcal{V}$: $q_{i+1} \in P_{i+1}$ and $(v'_0 \dots v'_i)^{-1} v_0^{(k-(i+1-1))} \dots v_i^{(k)} \in \Delta_{q_{i+1}}$

◀

A.2 A constructive proof for lemma 3.15

▶ **Lemma 1.2** (Prefixes). *For any subsequential relation, we can either delete a common prefix or add any word. Formally, let $\mathcal{T}$ be a subsequential transducer over $A \times B^*$, and $v \in B^* + B^{-1^*}$. If $(\varepsilon, v) R_{\mathcal{T}} \subseteq A^* \times B^*$, then it is recognized by a subsequential transducer.*

Proof. We first show how we can delete a common prefix.

▶ **Lemma 1.3** (Deleting prefixes). *Let $\mathcal{T}$ be a subsequential transducer over $A \times B^*$. If δ is a common prefix to the words in $I_{\mathcal{T}}$, then $(\varepsilon, \delta^{-1}) R_{\mathcal{T}}$ is recognized by a subsequential transducer.*

Proof. We will build a subsequential transducer $\mathcal{T}'$ that recognizes this relation.

Let us show that for all $\delta \in B^*$, a call to SpreadDifference(p, δ) leads to $R_{\mathcal{T}'_{(p,\delta)}} \leftarrow (\varepsilon, \delta^{-1}) R_{\mathcal{T}_p}$.

Require: δ is a common prefix to the words in $I_{\mathcal{T}}$
Ensure: $\mathcal{T}'$ recognizes $(\varepsilon, \delta^{-1})R_{\mathcal{T}}$
procedure REMOVE($(\delta, \mathcal{T})$)
$\mathcal{T}' \leftarrow (Q \times B^{nl}, E' = \emptyset, \{(i, \delta)\}, F \times B^{nl}, \rho')$
procedure SPREADDIFFERENCE((p, δ))
if (p, δ) hasn't already been visited **then**
if p is accepting in $\mathcal{T}$ **then**
$\rho'((p, \delta)) \leftarrow \delta^{-1}\rho(p)$
for all $p \xrightarrow[\mathcal{T}]{a|x} q$ **do**
$E' \leftarrow E' \cup \left\{ (p, \delta) \xrightarrow[\mathcal{T}']{a|(\delta\wedge x)^{-1}x} (q, (\delta \wedge x)^{-1}\delta) \right\}$
SPREADDIFFERENCE($(q, (\delta \wedge x)^{-1}\delta)$)
SPREADDIFFERENCE((i, δ))
return $\mathcal{T}'$

We prove it by induction on the distance needed to compensate δ, *ie* the least n such that for all path $p \xrightarrow[\mathcal{T}]{u_0|v_0} p_1 \xrightarrow[\mathcal{T}]{u_1|v_1} \cdots \xrightarrow[\mathcal{T}]{u_{n-1}|v_{n-1}} p_n$ and $p \xrightarrow[\mathcal{T}]{u_0|v_0} p_1 \xrightarrow[\mathcal{T}]{u_1|v_1} \cdots \xrightarrow[\mathcal{T}]{u_{n-2}|v_{n-2}} p_{n-1} \xrightarrow[\mathcal{T}]{v_{n-1}}$, $\delta \preccurlyeq v_0 \cdots v_{n-1}$

- If $n = 0$, then $\delta = \varepsilon$, and SPREADDIFFERENCE just copies recursively $\mathcal{T}$. It terminates, because every state is visited at most once. We then have $R_{\mathcal{T}'_{(p,\varepsilon)}} \leftarrow R_{\mathcal{T}_p}$.
- Let $n \geq 0$. Let $p \xrightarrow[\mathcal{T}]{u_0|v_0} q$ be a transition. $E' \leftarrow E' \cup \left\{ (p, \delta) \xrightarrow[\mathcal{T}']{u_0|(\delta\wedge v_0)^{-1}v_0} (q, (\delta \wedge v_0)^{-1}\delta) \right\}$.
  The distance needed to compensate $(\delta \wedge v_0)^{-1}\delta$ is at most $n-1$, so, by the induction hypothesis, $R_{\mathcal{T}'_{(q,(\delta\wedge v_0)^{-1}\delta)}} \leftarrow \left(\varepsilon, ((\delta \wedge v_0)^{-1}\delta)^{-1}\right) R_{\mathcal{T}_q}$.
  Now, let $(u_0 \cdots u_l, v_0 \cdots v_{l+1}) = (u, v) \in R_{\mathcal{T}_p}$ such that $p \xrightarrow[\mathcal{T}]{u_0|v_0} q \xrightarrow[\mathcal{T}]{u_1|v_1} \cdots \xrightarrow[\mathcal{T}]{u_l|v_l} f_{l+1} \xrightarrow[\mathcal{T}]{v_{l+1}}$ is a run from p in $\mathcal{T}$. We show that $(u, \delta^{-1}v) \in R_{\mathcal{T}'_{(p,\delta)}}$. $(u_1 \cdots u_l, v_1 \cdots v_{l+1}) \in R_{\mathcal{T}_q}$, so $(u_1 \cdots u_l, ((\delta\wedge v_0)^{-1}\delta)^{-1}v_1 \cdots v_{l+1}) \in R_{\mathcal{T}'_{(q,(\delta\wedge v_0)^{-1}\delta)}}$. Consequently, $(u_0 \cdots u_l, (\delta \wedge v_0)^{-1}v_0((\delta \wedge v_0)^{-1}\delta)^{-1}v_1 \cdots v_{l+1}) \in R_{\mathcal{T}'_{(p,\delta)}}$.
  We have to examine $(\delta \wedge v_0)^{-1}v_0((\delta \wedge v_0)^{-1}\delta)^{-1}v_1 \cdots v_{l+1}$. Here, the notation $^{-1}$ is not enough, and we have to examine two cases:
  - $v_0 \preccurlyeq \delta : \delta \wedge v_0 = v_0$. Thus, $(\delta\wedge v_0)^{-1}v_0((\delta\wedge v_0)^{-1}\delta)^{-1}v_1 \cdots v_{l+1} = v_0^{-1}v_0\delta^{-1}v_0v_1 \cdots v_{l+1}$ $= \delta^{-1}v$: $(u, \delta^{-1}v) \in R_{\mathcal{T}'_{(p,\delta)}}$
  - $\delta \preccurlyeq v_0 : \delta \wedge v_0 = \delta$. We get $(\delta\wedge v_0)^{-1}v_0((\delta\wedge v_0)^{-1}\delta)^{-1}v_1 \cdots v_{l+1} = \delta^{-1}v_0\delta^{-1}\delta v_1 \cdots v_{l+1}$ $= \delta^{-1}v$: $(u, \delta^{-1}v) \in R_{\mathcal{T}'_{(p,\delta)}}$

  This property holds $\forall (u, v) \in R_{\mathcal{T}_p}$ and $\forall q$ such that $p \xrightarrow[\mathcal{T}]{u_0|v_0} q$. Moreover, if p is accepting, then we also have $p \xrightarrow[\mathcal{T}]{\rho(p)}$, *ie* $(\varepsilon, \rho(p)) \in R_{\mathcal{T}_p}$. Since $(p, \delta) \xrightarrow[\mathcal{T}']{\delta^{-1}\rho(p)}$, $(\varepsilon, \delta^{-1}\rho(p)) \in R_{\mathcal{T}'_{(p,\delta)}}$.
  In conclusion, $(\varepsilon, \delta^{-1})R_{\mathcal{T}_p} \subseteq R_{\mathcal{T}'_{(p,\delta)}}$.
  Conversely, if $(u, v) \in R_{\mathcal{T}'_{(p,\delta)}}$, $u|v$ labels a path $(p, \delta) \xrightarrow[\mathcal{T}']{u_0|(\delta\wedge v_0)^{-1}v_0} (p_1, (\delta\wedge v_0)^{-1}\delta) \xrightarrow[\mathcal{T}']{u_1|v_1} \cdots \xrightarrow[\mathcal{T}']{u_l|v_l} (f_{l+1}, \delta_{l+1}) \xrightarrow{\delta_{l+1}^{-1}\rho(f_{l+1})}$, and so $(u, v) \in (\varepsilon, \delta^{-1})R_{\mathcal{T}_p}$
  Thus, $R_{\mathcal{T}'_{(p,\delta)}} \leftarrow (\varepsilon, \delta^{-1})R_{\mathcal{T}_p}$

Now, since δ is a common prefix to the images of the words in $\mathrm{dom}(\mathcal{T}_p)$, the distance needed to compensate it is at most n. Otherwise, we would have a run $p \xrightarrow[\mathcal{T}]{u_1|v_1} q \xrightarrow[\mathcal{T}]{u_2|v_2} q \xrightarrow[\mathcal{T}]{u_3|v_3} f \xrightarrow[\mathcal{T}]{v_4}$ such that $\delta \preccurlyeq v_1v_2v_3v_4$ but $\delta \not\preccurlyeq v_1v_3v_4$, however both $v_1v_3v_4$ and $v_1v_2v_3v_4$ are in $I_{p,\mathcal{T}}$. ◀

We now prove the counterpart, *ie* that we can add prefixes:

▶ **Lemma 1.4** (Adding prefixes). *Let $\mathcal{T}$ be a subsequential transducer over $A \times B^*$. For all $v \in B^*, (\varepsilon, v)R_{\mathcal{T}}$ is recognized by a subsequential transducer.*

Proof. We will just have to output v before entering $\mathcal{T}$. Thus, we create a new subsequential transducer $\mathcal{T}'$ by adding a state i'_0 to $\mathcal{T}$. This state is now the initial state, and for all transitions $i_0 \xrightarrow[\mathcal{T}]{a|x} p$, we add $i'_0 \xrightarrow[\mathcal{T}']{a|vx} p$. If i_0 is final, then $i_0 \xrightarrow[\mathcal{T}]{x}$, and we add $i'_0 \xrightarrow[\mathcal{T}']{vx}$. Now, we have $i'_0 \xrightarrow[\mathcal{T}']{a|vx} p \xrightarrow[\mathcal{T}']{u|y} f \xrightarrow[\mathcal{T}']{z}$ iff $i_0 \xrightarrow[\mathcal{T}]{a|x} p \xrightarrow[\mathcal{T}]{u|y} f \xrightarrow[\mathcal{T}]{z}$, because a path from p cannot contain transitions from i'_0 since i'_0 only has outgoing transitions. Consequently, $(au, vxyz) \in R_{\mathcal{T}'}$ iff $(au, xyz) \in R_{\mathcal{T}}$, and $(\varepsilon, vx) \in R_{\mathcal{T}'}$ iff $(\varepsilon, x) \in R_{\mathcal{T}}$: $R_{\mathcal{T}'} = (\varepsilon, v)R_{\mathcal{T}}$. ◀

We just have to combine the two statements:

- If $v \in B^{-1^*}$, then $(\varepsilon, v)R_{\mathcal{T}} \subseteq A^* \times B^*$ iff v is a common prefix to the words in $I_{\mathcal{T}}$, so we get the result with lemma 1.3.
- If $v \in B^*$, then $(\varepsilon, v)R_{\mathcal{T}} \subseteq A^* \times B^*$, and we get the result with lemma 1.4.

◀

B Context of the internship

Lab and team My work was conducted in the Computer Science Departement of the ULB, in Brussels. I worked under the supervision of Emmanuel Filiot, in the Formal Methods and Verification Group.

Interactions I mainly interacted with Mr. Filiot, since he was often there to help me. I also discussed with J.F. Raskin, the team leader, and, at one point, this prevented me from going into a wrong direction. The post-doctoral students here allowed me to widen my point of view about what I was working on, telling me about the infinite case and about related notions (infinite trees, tree automata,...), but also about their work, which was more focused on game theory. In addition, I went to Mons and witnessed a PhD. defence about the implementation of Acacia+, a software designed to synthesize a system from LTL specifications.

Research All these interactions and my work gave me a greater insight about what a researcher was supposed to do, from the proof finding to the proof redaction, but also about all the non-reseach tasks: administration, exam-marking...