1 Logic

**Exercise 1** (MSO Workout). Provide MSO transductions realizing the following functions:

1. *reverse*
2. *sort*
3. *swap-first-last*
4. *duplicate*

2 Atomisation and Deatomisation

**Exercise 2** (Deatomisation of Bimachines). Let \( A \) be an infinite set, and \( S \) be the set of permutations of \( A \). We define **atomic bimachines** with input alphabet \((\Sigma \cup A)\) and output alphabet \((\Gamma \cup A)\) via the existence of a finite monoid \( M \) and a morphism \( \mu: (\Sigma \cup A)^* \to M \) such that \( \mu(a) = 1_M \) for all \( a \in A \), together with a production function \( \pi: M \times (\Sigma \cup A) \times M \to (\Gamma \cup A)^* \), satisfying the following **equivariance property** (where \( \sigma \in S \) is lifted from permutations over \( A \) to an action over \((\Gamma \cup A)^* \) and \((\Sigma \cup A)^* \) in the natural way):

\[
\forall a \in A, \forall \sigma \in S, \pi(m, \sigma(a), n) = \sigma(\pi(m, a, n))
\]

An **atom coding function** is a function \( c: A \to \langle 1^k \rangle \), i.e., that maps every atom to a unique word of the form \( \langle 1^k \rangle \) for some \( k \in \mathbb{N} \). We say that a function \( f: (A \cup \Sigma)^* \to (A \cup \Gamma)^* \) is **deatomisable** if there exists a rational function \( f^\dagger: (\Sigma \cup \{\langle \cdot \rangle, 1\})^* \to (\Gamma \cup \{\langle \cdot \rangle, 1\})^* \) such that for all **atomic codes** \( c: A \to \langle 1^k \rangle \), the following commutes

\[
c \circ f = f^\dagger \circ c
\]

Prove that the following are equivalent:

1. A function \( f \) is **deatomisable**.
2. It is realisable by an **atomic bimachine**.

**Exercise 3** (Atomic Bimachines). Prove that the following are not computable by atomic bimachines.

---

*ad.lopez@uw.edu.pl
†https://www.mimuw.edu.pl/~bojan/2023-2024/przeksztalcenia-automatowe-transducers
‡https://aliaumel.github.io/transducer-exercices/
1. The reverse function.
2. The duplicate function.
3. The unzip function.

Conclude that those cannot be computed by rational functions.

3 Pumping Lemmas

**Exercise 4** (Pumping Bimachines). Let \( f \) be computed by a bimachine. We extend the function \( f \) by considering \( f(w_1[w]w_2) \) to be the word produced by the bimachine when reading \( w \), under the context \( w_1 \) and \( w_2 \).

1. Prove that \( f(w_1[w]w^X \times n^1[w]w_2) \) is of the form \( \alpha \beta^X \gamma \) for some \( \alpha, \beta, \gamma \), where \( n \) is the number of states of the automata in the bimachine.
2. Conclude that reverse, duplicate and unzip are not computable by bimachines.

**Exercise 5** (Pumping Sweeping Transducers). This exercise is based on the notion of sweeping transducers and their study done by Baschenis et al. [Bas+15] and Baschenis et al. [Bas+16].

Let \( f \) be computed by a sweeping transducer. Provide an appropriate pumping lemma for \( f \). Use this pumping argument to prove that \( \text{map-reverse} \) is not computable using a sweeping transducer.

**Exercise 6** (Pumping for sweeping 2DFTs). Provide a pumping lemma for sweeping 2DFTs. Conclude that \( \text{map-reverse} \) is not computable using a sweeping transducer.

**Exercise 7** (Sweeping Minimization). We define the sweeping number of a sweeping transducer the maximal number of sweeps it performs on any input words.

1. Prove that the sweeping number of a sweeping transducer is finite
2. Does there exist an algorithm that, given a transducer \( T \), computes its sweeping number?
3. Describe an algorithm that, given a sweeping transducer \( T \) and a number \( k \) with the promise that \( T \) can be realized by a sweeping transducer of sweeping number \( k \), constructs such a transducer.
4. Given a sweeping transducer \( T \) and a number \( k \), is it decidable whether \( T \) is realized by a sweeping transducer with sweeping number at most \( k \)?

\( \triangleright \) Hint 1

4 Well quasi orderings

**Exercise 8** (Well-Quasi-Ordered Image). Let \( f \) be a function from \( \Sigma^* \) to \( \Gamma^* \). We say that \( f \) generates a well-quasi-order whenever \( f(\Sigma^*) \) is well-quasi-ordered for the factor relation. We say that \( f \) generates a \( k \)-well-quasi-order whenever \( f(\Sigma^*) \) endowed (freely) with \( k \) distinguishing colours (unary predicates) is a well-quasi-order. Finally, we say that \( f \) generates an \( \infty \)-well-quasi-order whenever it generates a \( k \)-well-quasi-order for all \( k \in \mathbb{N} \).

1. Is it decidable whether the image of \( f \) is a well-quasi-ordering when \( f \) is computed by a 2DFT?
2. What about a bimachine? What about a Mealy Machine?
3. Prove that it is decidable whether \( f \) generates an \( \infty \)-well-quasi-ordering.
References


A Hints

Hint 1 (Exercise 7 Use effective continuity). Recall that if a function $f$ is computed by a 2DFT, then it is continuous, and even more: effectively continuous.