Transducers
Session 10: One proof done well (hopefully)

Aliame LOPEZ
TA mail*  Course page†  Exercises page‡

May 8, 2024

1 Blind again

Exercise 1 (Pumping lemma for regular functions). Let \( f \) be a regular function. Prove that there exists \( N \geq 0 \) such that for all \( w \in A^* \) with \( |w| \geq N \), there exist \( v_0, v_1 \in A^*, u \in A^+, n \geq 0, \alpha_0, \ldots, \alpha_n \in B^*, \beta_1, \ldots, \beta_n \in B^+ \) such that \( w = v_0u^Xv_1 \) and

\[
f(v_0u^{X+1}v_1) = \alpha_0\beta_1X\alpha_1\ldots\beta_nX\alpha_n \quad \text{for all} \; X \geq 0.
\]

▷ Hint 1
▷ Solution 1 (Self-contained proof)

Exercise 2 (Prefixes is not blind). Our goal is to prove that the function prefixes is not computable by a polyblind function.

1. Let \( f_1, \ldots, f_n \) be regular functions. Is it possible that \( f_1(w)f_2(w)\cdots f_n(w) \) computes a factor of prefixes(\( w \)) with a number of hashes that tends to \( +\infty \) as \( |w| \) grows?

2. Let \( f \) be a regular function. Is it possible that \( f(w)^{|w|} \) computes a factor of prefixes(\( w \)) with a number of hashes that tends to \( +\infty \) as \( |w| \) grows?

3. Using an induction on the polyblind depth and leveraging the pumping lemma of regular functions prove that the function prefixes is not polyblind.

▷ Hint 2
▷ Hint 3
▷ Hint 4
▷ Solution 2 (Self-contained proof)

2 Compression

Exercise 3 (Straight line program evaluation). Let \( e \) be the evaluation function from straight line programs to strings. Is \( e \) a polyregular function?

▷ Solution 3 (Solution)

*ad.lopez@uw.edu.pl
†https://www.mimuw.edu.pl/~bojan/2023-2024/przekształcenia-automatowe-transducers
‡https://aliaumel.github.io/transducer-exercises/
Exercise 4 (Efficient compression). What is the minimal size (number of instructions) needed to express prefixes($a^n$) as a straight line program?

▷ Hint 5
▷ Solution 4 (Solution)

Exercise 5 (Straight-line homomorphic programs). A function $f$ is straight-line homomorphic if there exists a polynomial time algorithm $P$ such that for all straight line program $X$, $f(e(X)) = e(P(X))$, where $e$ is the expansion function.

1. Prove that prefixes is not straight-line homomorphic.
2. Prove that regular functions are straight-line homomorphic.

▷ Solution 5 (Solution)

3 Cheat-Sheet

Definition 1 (The prefixes function). The prefixes function is defined inductively as follows prefixes($w$) is the list of non-empty prefixes of $w$ separated by hashes. For instance, prefixes(abc) = a#ab#abc.

Definition 2 (Composition by substitution). Let $f$ be a function from $\Sigma^*$ to \{1, \ldots, k\}^*$, and $g_1, \ldots, g_k$ be functions from $\Sigma^* \rightarrow \Gamma^*$. The composition by substitution of $f$ by $g_1, \ldots, g_k$ is the function

$$\text{cbs}(f, g_1, \ldots, g_k)(w) = \text{map}(\lambda x. g_x(w))(f(w)).$$

Definition 3 (Polyblind functions). The class of polyblind functions is defined as the smallest class of functions containing the regular functions and closed under composition by substitution. The polyblind depth of a function is the smallest $k$ such that the function can be obtained by composition by substitution of nesting depth at most $k$.

Definition 4 (Straight line program). A straight line program over an alphabet $\Sigma$ is a finite sequence of instructions of the form $x_i := u$ where $u$ is a single letter, or $x_i := x_j x_k$ with $i > j, k$. The value of a straight line program is the value of the last variable.

References

A Hints

**Hint 1** (Exercise 1 Idempotent transition monoid). Look at idempotent words in the transition monoid of the function \( f \).

**Hint 2** (Exercise 2 For the first). Note that \( f_1(w) \ldots f_n(w) \) is of linear output size.

**Hint 3** (Exercise 2 For the second). Notice that if \( f(w) \) outputs a word with at least two hashes, then \( f(w)^2 \) cannot be a factor of \( \text{prefixes}(w) \). If it has only one hash, then \( f(w)^X = (f(w)^2)^{X/2} \) and we conclude similarly for even \( X \)’s.

**Hint 4** (Exercise 2 For the third). The statement is clear for regular functions. Let us now consider a function obtained by a composition by substitution. Leveraging the pumping lemma for regular functions, conclude that some factor of \( \text{prefixes}(w) \) should be computed by a function lower \( \text{polyblind depth} \).

**Hint 5** (Exercise 4 What about variables containing two hashes?). Let \( x_i \) be a variable in a straight line program that evaluates to \( \text{prefixes}(a^n) \) and that contains a string with two hashes. Can it be used twice?
B Solutions

Solution 1 (Solution to Exercise 1). One version of the full proof is given by [Dou23, Proposition 2.16].

Solution 2 (Solution to Exercise 2). A complete proof of the result can be found in [Dou23, Proposition 3.14].

Solution 3 (Solution to Exercise 3). The function $e$ is not polyregular, because $e$ can have exponential size output, for instance by compressing $a^n$ into $O(\log(n))$ operations.

Solution 4 (Solution to Exercise 4). We prove that a straight line program that evaluates to a factor of prefixes($a^n$) having $k$ hashes must contain at least $k$ variables evaluating to words containing at least two hashes.

Let us first remark that every variable in the straight line program containing more than one hash can only be used once, as otherwise the output word contains two hash-separated words of the same size, which contradicts the definition of prefixes($a^n$).

For the base case, if the factor contains at least one hash, then the straight line program must contain at least one instruction.

For the induction, let $x_\ell$ be the last variable of the straight line program. It cannot be a constant assignment because the factor contains at least two hashes. Therefore, $x_\ell = x_i x_j$ for some $i, j < \ell$. By the induction hypothesis, $x_i$ and $x_j$. If $x_i$ contains at least two hashes, then it can only be used once, and $x_i \neq x_j$, otherwise, $x_j$ contains a factor of prefixes($a^n$) with at least $k$ hashes, but of smaller size and we proceed by induction. Now, because $x_i$ and $x_j$ are distinct variables, and because every variable containing at least two hashes can only be used once, we can partition the variables of the straight line program into three sets: the ones containing at most one hash, the ones containing at least two hashes used to build $x_i$, and the ones containing at least two hashes used to build $x_j$. Using the induction hypothesis we conclude as desired.

Solution 5 (Solution to Exercise 5). It is clear that prefixes is not straight-line homomorphic because prefixes($a^n$) cannot be compressed in less than $n$ instructions, while $a^n$ can be compressed in $O(\log(n))$ instructions. If prefixes were straight-line homomorphic, then the compression of prefixes($a^n$) would be doable in $O(P(\log(n)))$ instructions.

For the second part. Let $f$ be computed by a copyless SST with states $Q$, registers $\{1, \ldots, n\}$, and output function $F$. We construct our program $P$ as follows.

For every states $q_1, q_2 \in Q$, for every variable $x_i$ in the straight line program for the input word, for every register $r \in \{1, \ldots, n\}$, we create two variables $y_{i,q_1,r,in}$ and $y_{i,q_2,r,out}$. These are meant to encode the transitions of the SST on the word $x_i$ if it were to start in state $q_1$, and end up in state $q_2$, with initial values of the registers being $y_{i,q_1,r,in}$, and with the final values of the registers (after reading $x_i$) stored in $y_{i,q_2,r,out}$.

Now, it is easy to write the straight line program that uses these variables to simulate the SST on the input word. For a single letter $x_i := a$, this is just a transition of the SST. For a concatenation $x_i := x_j x_k$, we use the intermediate variables $y_{j,q_1,r,out}$ to simulate the transition of the SST on $x_j$, and then use the intermediate variables $y_{k,q_2,r,in}$ to simulate the transition of the SST on $x_k$.

This new straight-line program is constructed in polynomial time.