Transducers

Session 9: Linear Regular Functions
Version: v0.0.10

Aliaume LOPEZ
TA mail[*] Course page[†] Exercises page[‡]

June 24, 2024

1 Lambda-calculus

Exercise 1 (Basic Training). Write lambda-terms for the following functions:

- ☐ *lowercase*
- ☐ *expandtabs*
- ☐ *sort*
- ☐ *swap*

Exercise 2 (How to duplicate). Prove that square (without underscores) is definable in the lambda calculus.

Exercise 3 (Subject Reduction). Prove that (context) beta reduction and eta expansion preserve typing using the classical subject reduction technique.

Exercise 4 (What is the expressive power of the lambda-calculus?). Prove that the lambda-calculus using linear regular functions without squaring is strictly less expressive than the same lambda-calculus with squaring. What are the kind of functions computed?

2 Pebble transducers

Exercise 5 (Squaring for atoms). Let f be the function that maps w to the string $\prod_{i,j \text{ lex}} w[i]w[j]$.

1. Prove that it is definable by an atomic pebble transducer.
2. Prove that it is not definable by an atomic 2-pebble transducer.

▷ Hint 1

Exercise 6 (Composition of pebbles ?). Prove that k-pebble transducers are not closed under composition.
▷ Hint 2

[*] `ad.lopez@uw.edu.pl`
[†] `https://www.mimuw.edu.pl/~bojan/2023-2024/przeksztalcenia-automatowe-transducers`
[‡] `https://aliaumel.github.io/transducer-exercices/`

Exercise 7 (I was blind all along). We define blind models as models where the pebble cannot be read except for the last one. In the pebble model, the machine starts in the last position of the head of the caller and returns to this position when it pops.

1. Prove that the list of suffixes can be produced by a blind pebble transducer.
2. Prove that the composition of blind pebble transducers can express squaring.
3. Prove that blind pebble transducer are strictly less expressive than pebble transducers, because they cannot express squaring (this is quite difficult).
4. Conclude that blind pebble transducers are not closed under composition.

▷ Hint 3

3 Composition of functions

Exercise 8 (Replacing map reverse?). What happens if one replaces the combinator `map-reverse` by `square-map-reverse` in the definition of polyregular functions?

Exercise 9 (Weaker squaring). What happens if we replace `square` by `square-without-underlines` in the definition of poylregular functions?

4 For Transducers

Exercise 10 (Bounded output). Prove that it is decidable whether a for-transducer has bounded output.

Exercise 11 (Decidability of unnested). Give an algorithm that decides if a for-transducer f can be realised by an unnested for-transducer.

Exercise 12 (For transducers with string variables). Prove the equivalence between for-transducers and for-transducer with string variables having a single-use property, that cannot be iterated over.

Exercise 13 (Self-reduction?).

Exercise 14 (Evalutation of polyregular functions). Let $P_{k,d}$ be the collection of for-programs with nesting at most k, and using at most d distinct boolean variables. Prove that the evaluation function $e\colon P_{k,d} \times \Sigma^* \to \Gamma^*$ is polyregular. What is the increase in the number of loops? Of pebbles?

5 Reductions modulo polyregular functions

A polyregular reduction of a problem A into a problem B is a polyregular function f that maps instances of A into instances of B, and such that $x \in A$ if and only if $f(x) \in B$.

Exercise 15 (A canonical P-complete language). Let A be the language of strings (M, w, n) such that M is the code of a non-deterministic turing machine, and $M(w)$ accepts in time at most n, where n is written in unary.

1. Prove that the language A is NP-complete with respect to polyregular reductions.
2. Prove that the SAT problem is NP-complete with respect to polyregular reductions.
3. Prove that the 3SAT problem is NP-complete with respect to polyregular reductions.

A Hints

Hint 1 (Exercise 5 Number of outputs). Remark that the nested transducer has a bound on the size of its ouptut. Indeed, for any fixed i, if the output is too large, one letter repeats (because of the shape of the output) a lot, which means that the head goes a lot of time to some position, and if it is higher than the number of states, the automata loops. As a consequence, the output is at most linear.

Hint 2 (Exercise 6 Size issues). Just compose square twice.

Hint 3 (Exercise 7 Squaring by composition). Let f be the function that lists suffixes. We can post-compose f to underline the first letter of every suffix. Then, we can post-compose this function to prepend to any underlined letter, all the underlined letters that come before it in the reverse order. Finally, we can use a map reverse operation to put everything in the correct ordering.