Prof. Ryan Cotterell

Assignment #1

03/03/2022 - 21:57h

Question 1: Weighted Languages

(a) In $\mathbb{R}_{+,\times}\langle\langle\{a,b\}\rangle\rangle$, simplify the expression $3ab \oplus 4ab \oplus (5ba \otimes 7)$.

(b) In $\mathbb{R}_{\min,+}\langle\langle\{a,b\}\rangle\rangle$, simplify the expression $3ab \oplus 4ab \oplus (5ba \otimes 7)$.

(c) In general, we write ST as an abbreviation for $S \otimes T$. There is a danger of confusion here, since $3ab$ is ambiguous: it could mean either (i) the function that maps $ab \mapsto 3$ and maps everything else $\varepsilon \mapsto 0$, or (ii) the product $3 \otimes a \otimes b$. Show that there is actually no danger because the power series (i) and (ii) are equal.

Question 2: Two Basic Definitions

This question requires you to implement two basic utilities in `rayuela` that correspond to definitions in the course notes.

i) We say a WFSA $\langle R, \Sigma, Q, \delta, \lambda, \rho\rangle$ is **deterministic** if, for every state–letter pair $(q, a) \in Q \times \Sigma$, the transition function $\delta(q, a, q') \neq \mathbf{0}$ for at most one element $q' \in Q$ and, furthermore, there are no ε transitions in the automaton. Fill in the code marked in `fsa.py` to check whether the input FSA is deterministic.

ii) We say a WFSA $\langle R, \Sigma, Q, \delta, \lambda, \rho\rangle$ is **pushed** if, for every $q \in Q$, the following equation holds:

$$\bigoplus_{(a,q')\in\Sigma\times Q} \delta(q, a, q') \oplus \rho(q) = \mathbf{1} \tag{1}$$

Fill in the code marked in `fsa.py` to check whether the input FSA is pushed.

Question 3: Trimming a Weighted Finite-State Automaton

This question concerns itself with optimizing a WFSA by removing unnecessary states. We call this operation **trimming**. Importantly, trimming is not minimization because we will *not* merge any states; merging states will be discussed in lecture 4. We will walk you through the four steps necessary to trim a WFSA.

i) It is a well known fact that the regular languages are closed under **reversal**. You are going to implement a constructive proof of this fact. Design an algorithm that reverses a FSA. Prove correctness of your algorithm, i.e., show that for any string $\mathbf{x} \in \Sigma^*$ accepted by the original automaton, $\mathbf{x}^{\mathrm{R}}$ is accepted by its reversal. Fill in the code in `fsa.py`, copied below, to compute the reversal

ii) Given a WFSA $\langle R, \Sigma, Q, \delta, \lambda, \rho \rangle$, a state $q \in Q$ is called **accessible** if it is reachable from a state q' with $\lambda(q') \neq \mathbf{0}$. Design an algorithm to compute the set of accessible states in an automaton.

iii) Given a WFSA $\langle R, \Sigma, Q, \delta, \lambda, \rho \rangle$, a state $q \in Q$ is called **co-accessible** if a state q' with $\rho(q') \neq \mathbf{0}$ is reachable from q. Design an algorithm to compute the set of co-accessible states in an automaton.

iv) Provide an algorithm to trim a WFSA. Explain why removing accessible and co-accessible states is the most we can remove. Explain why reversal is a useful subroutine in the construction.

Question 4: Union, Concatenation and Kleene Closure

Regular languages are famously closed under union and concatenation. In this question, you will implement algorithms for computing the union and concatenation of two arbitrary WFSAs. In so doing, you are providing a constructive proof of the aforementioned closure properties.

i) Give an algorithm to compute the **union** of two WFSAs. Prove its correctness and analyze its runtime complexity. Implement your algorithm in `rayuela` in the marked location.

ii) Give an algorithm to compute the **concatenation** of two WFSAs. Prove its correctness and analyze its runtime complexity. Implement your algorithm in `rayuela` in the marked location.

iii) Give an algorithm to compute the **Kleene closure** of a WFSA. Prove its correctness and analyze its runtime complexity. Implement your algorithm in `rayuela` in the marked location.