

CSE105 (Winter 2014): Homework 2

Instructor: Daniele Micciancio

Due on Wednesday, January 22, 2014.

Your solutions to this homework should consist of 16 files named `hw1a.jff`, `hw1b.jff`, `hw1c.jff`, `hw1d.jff`, `hw2a.jff`, `hw2b.jff`, `hw3a.jff`, `hw3aNFA.jff`, `hw3b.jff`, `hw3bNFA.jff`, `hw3c.jff`, `hw3cNFA.jff`, `hw4a.hs`, `hw4b.hs`, `hw5a.hs`, `hw5b.hs` (the last two files are optional) and a write up for the last problem. As usual, the files should be zipped and submitted using `turnin`. You can bring your writeup to class.

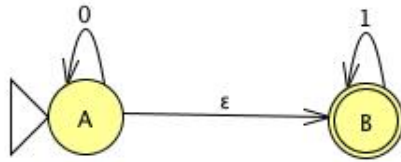
1 NFA Design

For each of the following languages, design an NFA with the required number of states. Draw your NFAs in JFLAP and save them as files `hw1a.jff`, ..., `hw1d.jff`.

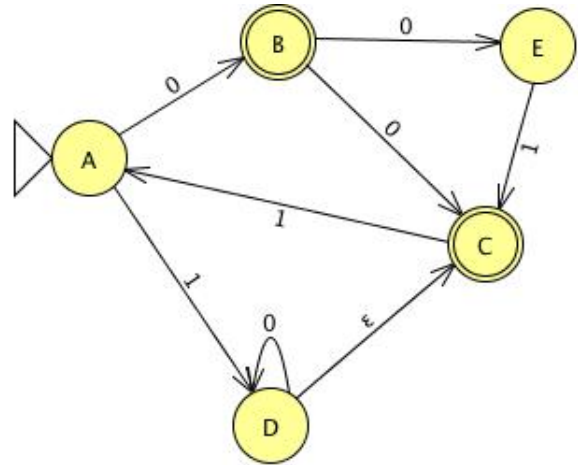
- (a) Sipser 1.7(b): The set of all strings over the alphabet $\Sigma = \{0, 1\}$ that contain the substring 0101.
- (b) Sipser 1.7(c): The set of all strings over the alphabet $\Sigma = \{0, 1\}$ that contain an even number of 0s, or contain exactly two 1s.
- (c) The set of all string over the alphabet $\Sigma = \{0, 1\}$ of length ≥ 1 , that does not contain the string 00 nor 11. You can do this in 3 states.
- (d) The set of all strings, over the alphabet $\Sigma = \{0, 1, 2\}$, of length ≥ 4 , that start with 01 and end with 10 and do not contain another 10 in the middle. For example, these strings are accepted: 0110, 01201110. Whereas these strings are rejected: 01, 010, 1101, 010210.

2 NFA to DFA conversion

Convert each of the following NFAs into a corresponding DFA using the procedure described in the textbook. You should perform the conversion by hand, and then save your DFAs using JFLAP as files `hw2a.jff` and `hw2b.jff`. Label the states of the resulting DFA with strings of the form AC, BCD, etc., corresponding to subsets of the original NFA (in alphabetical order), using X as the label for the empty set. You only need to include the “useful” states, i.e., the states that are reachable from the start state of the DFA. (If solved correctly, your DFAs should use 3 and 9 states respectively.)



(a)



(b)

3 Regular expressions

For each of the following languages, design a corresponding regular expression, and submit your solution following these instructions.

1. Write the regular expression in JFLAP, and save it as `hw3a.jff`, ..., `hw3c.jff`.
2. Convert each regular expression into an equivalent NFA using JFLAP “Convert to NFA / Do All” command, and store the result as `hw3aNFA.jff`, ..., `hw3cNFA.jff`. Notice that we are not asking you to perform the transformation by hand: all you have to do is to load each regular expression in JFLAP, and let JFLAP do all the work by first clicking on “Convert to NFA” and then “Do All”. We are asking you do this just to facilitate our testing of the regular expressions.
3. Submit all 6 files as part of your solution.

(a) All binary strings that include 110 as a substring. E.g., 110, 0101100 are in the language, while ϵ , 011, 1010 are not.

(b) All binary strings with an even number of 0s. E.g., ϵ , 100 and 01110 are in the language, while 1101 is not.

(c) All binary strings that do not include 000 as a substring.

4 Haskell

Most of the mathematical definitions and theorems studied in this course, have a direct translation into (useful) computer programs. For example, the mathematical definition of DFAs and NFAs easily translate into corresponding data structures. The recursive definition of the function computed by a DFA is easily translated into a computer program that given a DFA and an input string, outputs True or False. The proof that regular languages are closed under union (Theorem 1.25 in the textbook) gives an algorithm that on input two DFAs for languages L_1 and L_2 , produces a DFA for the language $L_1 \cup L_2$. Translating mathematical definitions and theorems into computer programs is especially easy when using a high level programming language. In this problem you are asked to experiment with the implementation of concepts related to this course using Haskell, a functional programming language that allows to define functions in a way that closely mimics the mathematical definitions. (We will soon post some material on the course webpage to get you started.) Specifically, we ask you to do the following:

- (a) Prove that regular languages are closed under complement by giving a program that on input a DFA for a language L , outputs a DFA for the language $\bar{L}$ (the complement of L .)
- (b) Prove that regular languages are closed under intersection by giving a program that on input two DFAs for languages L_1 and L_2 , produces a DFA for the intersection $L_1 \cap L_2$.

Submit your solutions as two files `hw4a.hs` and `hw4b.hs` following the guidelines (soon to be posted) on the course webpage. This problem is intentionally very easy, and it is intended mostly to get you acquainted with writing simple programs in haskell. You can get as much help as you want, by working with other students, posting questions on piazza, etc.

5 Closure properties

In this problem you are asked to prove some less trivial closure properties. It is highly recommended that you first prove the closure property and write down your proof using mathematical notation, and then translate it into a haskell program implementing the associated transformation. For this problem, submitting the haskell implementation is optional. Everybody should submit a proof written using mathematical notation.

(a) For any language $L \subseteq \Sigma^*$ and symbol $a \in \Sigma$, let $L - a = \{w \in \Sigma^* \mid aw \in L\}$. Prove that for any such L and a , if L is regular then also $L - a$ is regular by giving a transformation $T(M, a)$ that on input a DFA M with alphabet Σ and a symbol $a \in \Sigma$, outputs an NFA for the language $L - a$. Implement the transformation in haskell and submit it as file `hw5a.hs`.

(b) For any two languages $L_1, L_2 \subseteq \Sigma^*$, let $L_1 \setminus L_2 = \{w \in \Sigma^* \mid \exists u \in L_2. uw \in L_1\}$. Prove that for two languages L_1, L_2 , if both L_1 and L_2 are regular, then $L_1 \setminus L_2$ is also regular by giving a transformation $T(M_1, M_2)$ that on input two DFAs M_1, M_2 over the same alphabet Σ produces an NFA for the language $L_1 \setminus L_2$. Implement the transformation in haskell and submit it as file `hw5b.hs`.

Many of you are likely to find the last problem quite challenging. If you can do everything except the last problem, don't worry, you are still doing fine.