ToC: Homework 2 [150 points]

Due: Saturday, October 2nd, by midnight

Reading

Please read Lectures 8, 9, 11, 12, 13, and 14 of Kozen.

How to submit

Please put your solution in a folder whose name is your Hawkid. If you worked with a partner please include a file called `partner.txt` that lists the Hawkids for you and your partner. Only one partner should submit their assignment. You will both receive the same grade.

How to get help

Please post questions to ICON Discussions, if possible (as opposed to emailing us directly), so that our answers can benefit others who likely have similar questions. You may also come to our Zoom office hours, listed on ICON under Pages - Office Hours.

Haskell code

I am copying the current Haskell source files that I am using in class, to the directory for this homework. These are there to help you, and the `WriteRegexp.hs` file depends on some of them. They do not contain any problems for you to solve.

1 Writing regular expressions [50 points]

For each of the following sets of strings, write a regular expression in Haskell to describe it. Please put your answers in `WriteRegexp.hs`. One tool you can use if you want to try to check your answer is the `checkRegexp` function found in `Regexp.hs`. This takes a regular expression and a string, tells whether or not the string is in the meaning of the regular expression. So you could check your answers by testing them with some inputs, if you wish.

[10 points each]

1. The set of all strings that either start with an A and then have zero or more Bs, or else start with a B and then have zero or more As.
2. The set of all strings that repeat the following zero or more times: an A, then **one** or more Bs (not zero), then a C.

3. The set of all strings that start with zero or more characters that are either A or B, then have two Bs, and then have zero or more Cs.
4. The set of all strings that start with either zero or more As or one or more Bs, then have zero or more characters that are either A or C.
5. The set of all strings that repeat the following zero or more times: zero or more characters that are B or C, then an A.

2 Understanding regular expressions [50 points]

For the following problems, you can just puzzle things out by staring at automata, or you can use the Haskell code I am including to help you, if you want. To be able to use those tools, though, you would need (correct) answers (in Haskell) to the parts of Problem 1. [25 points each]

1. The files named `mystery1.pdf` through `mystery5.pdf` show minimal DFAs recognizing the languages described in Problem 1. Please match up the PDFs with the parts of Problem 1, to show which automaton corresponds to which language. Please just note in a comment in your `WriteRegexp.hs` file which mystery file goes with each Regexp.
2. The file named `intersect.pdf` shows the minimal DFA accepting the intersection of two of the sets described in Problem 1. Which two? You can put another comment for the two Regexps in `WriteRegexp.hs`, to indicate your answer.

3 Epsilon closure [20 points]

Please put your solution in `Epsclosure.hs`.

1. Fill in the definition of `epsClosure` so that it takes in an NFA with epsilon-transitions (of type `Eps.Nfa` in the code), and produces an NFA without epsilon-transitions (`Nfa`). The way to do this is as follows:
   (a) The states of the new NFA are the same as the original one.
   (b) Compute the reflexive-transitive closure of the `eps` relation which gives the epsilon-transitions. There is a function in `Relation.hs` to compute this.
   (c) The new NFA's transition relation compose that closure of `eps` with the the original NFA's transition relation, for each input letter.
   (d) The start states are the same.
   (e) The final states are all those states which can reach a final state of the original NFA, using that closure of `eps`. I used `Data.List`'s `nub` function to drop duplicates when computing this.

   I am supplying `nfa.pdf` and `epsclosure.pdf` as examples. Notice that `epsclosure.pdf` has some unreachable states. This is expected with the above algorithm.

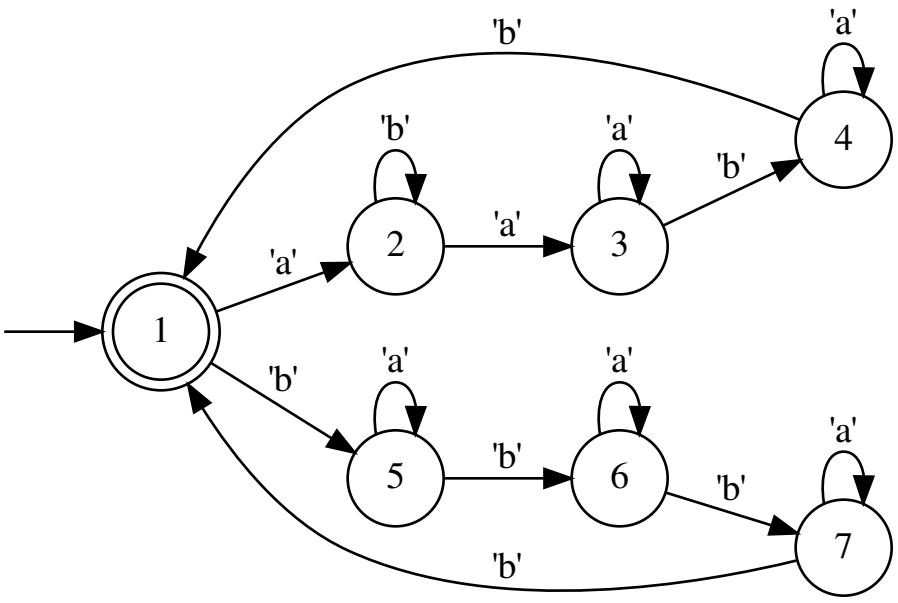

Figure 1: Automaton for problem about respected equivalence relations

4 Equivalence relations respected by automata [30 points]

These are set-theory problems. You can put your solution in a file named `Respect.?` where `?` is for a standard file type like text file, Word document, or image format if you are submitting a scan or photo of a written solution.

1. For each of the following relations, indicate whether or not its equivalence-closure (that is, the relation we get by taking the reflexive, transitive, symmetric closure of the given one) is respected by the automaton in Figure 1. Let S denote the set of states of that automaton (so $\{1, \ldots, 7\}$). [3 points each]

   (a) $\{(x, y) \mid x \in S,\ y \in S\}$

   (b) $\{(3, 6)\}$

   (c) $\{(4, 7)\}$

   (d) $\{\ \}$

   (e) $\{(3, 5), (2, 6), (4, 7)\}$

2. Find the smallest relation whose equivalence closure gives the coarsest relation respected by the automaton in Figure 1. [15 points]