Logic in CS Autumn 2024

Problem Sheet 7

S. Krishna

1. (a) Let A be a DFA accepting the language L. Is the reverse of all the strings accepted by $L(A)$ regular?

   (b) Let L be a regular language over $\{a, b, c\}$. Define the projection of L with respect to $\{b, c\}$ denoted $L \downarrow \{b, c\}$ as the language

   $$\{w' \mid w' \text{ is obtained from } w \in L \text{ after deleting all occurrences of symbol } a\}$$

   Is $L \downarrow \{b, c\}$ regular?

   (c) Show that every NFA can be converted into an equivalent one with a single accepting state.

   (d) Let $N = (Q, \Sigma, \delta, q_0, F)$ be a DFA. Construct an automaton $N_1 = (Q, \Sigma, \delta_1, q_0, F_1)$ as follows:

   - $F_1 = F \cup \{q_0\}$
   - Define δ_1 such that for any $q \in Q$ and $a \in \Sigma \cup \{\epsilon\}$,

   $$\delta_1(q, a) = \delta(q, a) \text{ for } q \notin F \text{ or } a \neq \epsilon$$

   $$\delta_1(q, a) = \delta(q, a) \cup \{q_0\} \text{ for } q \in F \text{ and } a = \epsilon$$

   Is $L(N_1) = (L(N))^*$?

   (e) Let L be a regular language. Is the language $L_{\frac{1}{2}}$, the set of first halves of strings in L regular? Formally,

   $$L_{\frac{1}{2}} = \{x \mid \exists y, |x| = |y|, xy \in L\}$$

   (f) Let L be a regular language. Is the language $\mathsf{Cuberoot}(L)$ defined as $\{w \mid w^3 \in L\}$ regular?

   (g) Let L be a regular language. Consider the language L' defined as

   $$\{w \in L \mid \text{no proper prefix of } w \text{ is in } L\}$$

   Show that L' is regular.

   (h) For any language L, define $cycle(L) = \{uv \mid vu \in L\}$ as the set of all cyclic shifts of words accepted by L. As an example, if $abcd \in L$, then all its cyclic shifts $abcd, dabc, cdab, bcda$ are also in L. Show that if L is regular, so is $cycle(L)$.

Solution

(a) Yes it is regular, just reverse all the transitions in A and switch initial and final states. [This will give a NFA which accepts L_{rev}. If there are multiple final states, create a new final state q' and add ε transitions from q' to all the states (which were final in A).]

(b) Yes it is regular, simply replace all a-edges with ϵ-edges, and we get an NFA which accepts $L \downarrow \{b, c\}$.

(c) Strip the accepting rights of final states, add an ϵ-edge from them to a single final state and make it accepting.

(d) No, it's possible for $L(N_1)$ to contain words not in $(L(N))^*$ since q_0 is now an accepting state, which implies any closed walk at q_0 which doesn't pass through any F state also gets accepted.

(e) **DFA Construction:**
Let a DFA accepting L be $A = (Q, \Sigma, \delta, q_0, F)$. Then we can construct the DFA accepting $L_{\frac{1}{2}}$: $B = (Q \times 2^Q, \Sigma, \delta_B, (q_0, F), F_B)$ where

- Transition function: $\delta_B((q, S), a) = (\delta(q, a), T)$ where $T = \{p \in Q : \exists\, b \in \Sigma \text{ such that } \delta(p, b) \in S\}$
- Final states: $F_B = \{(q, S) : q \in S\}$

Invariant (to be proved using induction on $|w|$) for correctness of DFA:

$\hat{\delta_B}((q_0, F), u) = (\hat{\delta}(q_0, u), S_u)$ where $S_u = \{q \in Q : \exists v \in \Sigma^*, |v| = |u|, \delta(q, v) \in F\}$

[That is, the second component of the pair keeps track of possible states from which on accepting a word of same length as the input, we reach one of the final states.] Using this invariant, we can show that $w \in L \iff w$ is accepted by the DFA, to complete the proof of correctness.

(f) **Method 1 (Using properties of regular languages) :** Since L is regular, there exists a DFA $(Q, \Sigma, \delta, q_0, F)$ whose language is L. We denote the regular language accepted by the DFA $(Q, \Sigma, \delta, \alpha, F')$ by $L(\alpha, F')$. Then, we claim that

$$\mathsf{Cuberoot}(L) = \bigcup_{q_1, q_2 \in Q} \Big(L(q_0, \{q_1\}) \bigcap L(q_1, \{q_2\}) \bigcap L(q_2, F) \Big)$$

Since the finite intersection and finite union of regular languages is regular, Cuberoot(L) is regular.

Method 2 (NFA construction): We need to construct an automaton which can simultaneously start from the initial state q_0 as well as the "one-third" and "two-third" state q_1 and q_2 of D, and parallelly parse three occurrences of w, from q_0 to q_1, from q_1 to q_2 and from q_2 to some accepting state.

We also need to remember what this "one-third" and "two-third" state was. Thus, the machine will work on $Q \times Q \times Q \times Q \times Q$ state space.

The NFA which accepts $\mathsf{Cuberoot}(L)$ is:

$$N = (Q \times Q \times Q \times Q \times Q, \Sigma, \Delta, Q_0, F')$$

where

$$Q_0 = \{(q_0, q_1, q_2, q_1, q_2) \mid q_1, q_2 \in Q\}$$

$$F' = \{(q_1, q_2, q_f, q_1, q_2) \mid q_f \in F, q_1, q_2 \in Q\}$$

$$\Delta = \{((q_l, q_m, q_n, q_1, q_2), a, (q'_l, q'_m, q'_n, q_1, q_2)) \mid \delta(q_i, a) = q'_i \; \forall i \in \{l, m, n\}\}$$

(g) Consider $A(L)$ and create an NFA from it by removing all outgoing edges from all final states. The language of this NFA is L'.

(h) Since L is regular, there exists a DFA $(Q, \Sigma, \delta, q_0, F)$ whose language is L. We denote the regular language accepted by the DFA $(Q, \Sigma, \delta, \alpha, F')$ by $L(\alpha, F')$. Then, we claim that

$$cycle(L) = \bigcup_{q_i \in Q} (L(q_i, F) \cdot L(q_0, \{q_i\}))$$

- $L_1 \cdot L_2 = \{w_1 w_2 : w_1 \in L_1, w_2 \in L_2\}$ (definition of concatenation)

The proof of the claim is straightforward, showing w is in the LHS if and only if it is in the RHS.

$$\begin{aligned} w \in cycle(L) &\iff \exists u, v \text{ s.t. } w = vu \text{ and } \hat{\delta}(q_0, uv) \in F \\ &\iff \exists u, v, q_i \text{ s.t. } w = vu, \hat{\delta}(q_0, u) = q_i, \hat{\delta}(q_i, u) \in F \\ &\iff \exists q_i \text{ s.t. } w \in L(q_i, F) \cdot L(q_0, \{q_i\}) \\ &\iff w \in \bigcup_{q_i \in Q} (L(q_i, F) \cdot L(q_0, \{q_i\})) \end{aligned}$$

Since the concatenation of two regular languages is regular (*prove this by NFA construction.*) and finite union of regular languages is regular, $cycle(L)$ is regular.

2. Let $L_n = \{x \mid x \text{ is a binary number that is a multiple of } n\}$. Show that for all $n \geq 1$, L_n is regular.

Solution

By simulating binary division, we construct a DFA M with n states that recognizes the language L_n. M has n states, each of which keeps track of the n possible remainders in the division process. The start state is the only accept state and corresponds to remainder 0.

The input string is fed into M starting from the most significant bit. For each input

bit, M doubles the remainder that its current state records and then adds the input bit. Its new state is the sum modulo n. We double the remainder because that corresponds to the left shift of the computed remainder in the long division algorithm. If an input string ends at the accept state (corresponding to remainder 0), the binary number has no remainder on division by n and is therefore a member of L_n.
The formal definition of M is:

$$M = (\{q_0, q_1, \ldots, q_{n-1}\}, \{0,1\}, \delta, q_0, \{q_0\}),$$

where $q_0, q_1, \ldots, q_{n-1}$ are the states representing the possible remainders modulo n, and $\{0,1\}$ is the input alphabet.
For each $q_i \in Q$ and $b \in \{0,1\}$, the transition function δ is defined as:

$$\delta(q_i, b) = q_j, \quad \text{where} \quad j = (2i + b) \mod n.$$

This function models the division process by updating the state based on the remainder after shifting and adding the current input bit.
Thus, M accepts an input string if and only if the binary number represented by the string is divisible by n (i.e., ends at state q_0).

3. Recall that we defined an angelic acceptance condition for NFAs in class : a word w is accepted whenever it has atleast one accepting run. Under this, we showed that the languages accepted by NFAs are regular. Consider the following *devilish* acceptance condition, which says that an NFA M accepts a word x iff every possible computation of M on x ends in an accept state. Show that NFAs with the devilish acceptance condition recognize the class of regular languages.

Solution

We have to show that for any regular language L, there exists an NFA with devilish acceptance and the language accepted by a devilish NFA is a regular language.

The former can be proved by constructing a DFA $(Q, \Sigma, \delta, q_0, F)$ whose language is L. Since a DFA has only one possible run for each word, it satisfies the devilish acceptance condition.

From a given NFA with devilish acceptance condition $(Q, \Sigma, \delta, Q_0, F)$, we can construct a DFA $(2^Q, \Sigma, \Delta, Q_0, 2^F)$ such that the language accepted by the NFA and the DFA is same.

$$\Delta(S, a) = \bigcup_{s \in S} \delta(s, a)$$

A word x is accepted by the DFA if and only if $\Delta(Q_0, x) \in 2^F$ and hence this word x is accepted by the devilish NFA.

4. Write second order logic formulae to capture the following:

   (a) There is a path from node s to node t in the graph. The signature is $\tau = \{E\}$.

   (b) Every bounded non empty set has a least upper bound. The signature is $\tau = \{\leq\}$

Solution

(a) We will construct the set of nodes reachable from s, and assert that t belongs to this set. Let:

$$\varphi(X, s) = [X(s) \wedge \forall u \forall v \, (X(u) \wedge E(u, v) \rightarrow X(v))]$$

$\varphi(X, s)$ asserts that X contains all the nodes reachable from s — it may contain other nodes also (to see this, note that $\varphi(V, s)$ is always true). We want X to be the set containing precisely the nodes reachable from s.

Note that this is the "smallest" set satisfying φ, i.e., it is a subset of every set satisfying φ. The required formula therefore becomes:

$$\exists X[\varphi(X, s) \wedge \forall Y(\varphi(Y, s) \rightarrow \forall u(X(u) \rightarrow Y(u))) \wedge X(t)].$$

(b) Let $\varphi(X, t)$ denote that t is an upper bound of X, i.e.,

$$\varphi(X, t) = \forall x \, [X(x) \rightarrow x \leq t],$$

the required sentence is then:

$$\forall X[\exists x X(x) \rightarrow \exists t(\varphi(X, t) \wedge \forall r(\varphi(X, r) \rightarrow t \leq r))].$$

5. Let Σ be a finite alphabet. The atomic formulae in MSO defined over Σ^* are $x = y, x < y, S(x, y), X(x)$ and $Q_a(x)$, $a \in \Sigma$. Consider the following logic called MSO_0 having atomic formulae of the following forms:

$$\mathsf{Sing}(X), X \subseteq Y, X < Y, S(X, Y), Q_a(X)$$

where

- $\mathsf{Sing}(X)$ means that X is a SO variable of cardinality 1;
- $X \subseteq Y$ means that every element of the SO variable X is contained in the SO variable Y;
- $X < Y$ means that SO variables X, Y have cardinality 1, and that the element in Y is greater than the element in X;
- $S(X, Y)$ means that SO variables X, Y have cardinality 1, and Y contains the successor of the element in X; and,
- $Q_a(X)$ means that all positions in X are decorated by $a \in \Sigma$.

If φ is an atomic formula in MSO, then $\varphi \wedge \varphi, \neg\varphi, \varphi \vee \varphi$, $\forall x \ \varphi$ and $\forall X \ \varphi$ are formulae in MSO. Similarly, if φ is an atomic formula in MSO_0, then, $\varphi \wedge \varphi, \neg\varphi, \varphi \vee \varphi$ and $\forall X \ \varphi$ are

formulae in MSO_0.
Compare the expressiveness of MSO and MSO_0.

Solution

We will show that MSO and MSO_0 are equally expressive. Firstly, we will inductively define a transformation function $\tau : \text{MSO}_0 \mapsto \text{MSO}$, which gives an equivalent formula in MSO for every MSO_0 formula. The atomic formulae of MSO_0 can also be expressed in MSO as follows:

$$\tau(\text{Sing}(X)) = \exists x[X(x) \wedge \forall y(X(y) \rightarrow y = x)]$$

$$\tau(X \subseteq Y) = \forall x[X(x) \rightarrow Y(x)]$$

$$\tau(X < Y) = \tau(\text{Sing}(X)) \wedge \tau(\text{Sing}(Y)) \wedge \exists x \exists y[x < y \wedge X(x) \wedge Y(y)]$$

$$\tau(S(X,Y)) = \tau(\text{Sing}(X)) \wedge \tau(\text{Sing}(Y)) \wedge \exists x \exists y[S(x,y) \wedge X(x) \wedge Y(y)]$$

$$\tau(Q_a(X)) = \forall x[X(x) \rightarrow Q_a(x)]$$

Now, by induction hypothesis if we have τ defined for φ and ψ, we have

$$\tau(\neg\varphi) = \neg\tau(\varphi)$$

$$\tau(\varphi \wedge \psi) = \tau(\varphi) \wedge \tau(\psi)$$

$$\tau(\varphi \vee \psi) = \tau(\varphi) \vee \tau(\psi)$$

$$\tau(\forall X \varphi) = \forall X \tau(\varphi)$$

Therefore, MSO is at least as expressive as MSO_0.

Now, we need to show that every MSO formula has an equivalent MSO_0 formula. However, MSO_0 has no first-order variables, and so we have to map the first-order variables in the MSO formula to appropriate second-order MSO_0 variables. We will map the first-order variables to second-order variables corresponding to singleton sets. For this we define another function $\lambda : \text{MSO} \mapsto \text{MSO}_0$, inductively. The mappings of the atomic MSO_0 formulae are (here X and Y are the second-order variables corresponding to the first-order variables x and y):

$$\lambda(x = y) = \text{Sing}(X) \wedge \text{Sing}(Y) \wedge X \subseteq Y \wedge Y \subseteq X$$

$$\lambda(x < y) = X < Y$$

$$\lambda(S(x,y)) = S(X,Y)$$

$$\lambda(H(x)) = \text{Sing}(X) \wedge X \subseteq H$$

[a]

$$\lambda(Q_a(x)) = \text{Sing}(X) \wedge Q_a(X)$$

Now using induction hypothesis, let $\lambda(\varphi)$ and $\lambda(\psi)$ be defined. Then,

$$\lambda(\neg\varphi) = \neg\lambda(\varphi)$$

$$\lambda(\varphi \wedge \psi) = \lambda(\varphi) \wedge \lambda(\psi)$$

$$\lambda(\varphi \vee \psi) = \lambda(\varphi) \vee \lambda(\psi)$$

$$\lambda(\forall X \varphi) = \forall X \lambda(\varphi)$$

$$\lambda(\forall x\, \varphi(x)) = \forall X[\text{Sing}(X) \implies \lambda(\varphi(X))]$$

Therefore, every MSO formula has an equivalent MSO_0 formula, which implies that they are equally expressive.

[a] Here H is a second order variable and $H(x)$ is the predicate that $x \in H$

6. For the formula $\exists x \forall y(x < y \to Q_a(y))$ give an equivalent MSO_0 formula. Also draw the equivalent DFA following the steps done in class.

Solution

We can write the formula:

$$\exists x \forall y[x < y \to Q_a(y)]$$

as:

$$\neg\forall x \neg\forall y[x < y \to Q_a(y)].$$

Applying the λ transformation discussed in the previous problem, the MSO_0 equivalent of this would be:

$$\neg\forall X[\text{Sing}(X) \to \neg\forall Y(\text{Sing}(Y) \to [X < Y \to \text{Sing}(Y) \wedge Q_a(Y)])].$$

This can be simplified to:

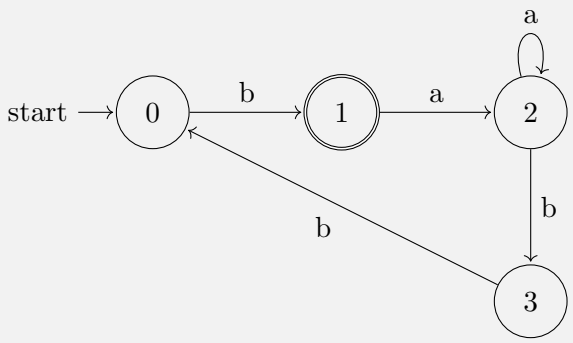
$$\neg\forall X[\text{Sing}(X) \to \neg\forall Y(X < Y \to Q_a(Y))].$$

Drawing the automaton is left as an exercise.

7. Consider the following NFA $N = (\{0, 1, 2, 3\}, \{a, b\}, \Delta, \{0\}, \{1\})$ with $\Delta(0, b) = \{1\}$, $\Delta(1, a) = \{2\}$, $\Delta(2, a) = \{2\}$, $\Delta(2, b) = \{3\}$ and $\Delta(3, b) = \{0\}$. Write an MSO formula with two SO variables that characterizes $L(N)$.

Solution

The automata described in the question can be visualised as:



Observe that $L(N) = b(a + b^3)^*$. This language is actually first-order definable, as can be seen in Problem Sheet 6, Question 5, and so we can write it as a monadic second-order (MSO) formula as well by adding two valid clauses of the form $\exists X[\forall y \neg X(y)]$ to trivially add two MSO variables.