Discrete Mathematics

Exercises 9 – Solutions with Commentary

Marcelo Fiore Ohad Kammar Dima Szamozvancev

14. On inductive definitions

1. Let L be the subset of $\{a, b\}^*$ inductively defined by the axiom $\frac{}{\varepsilon}$ and rule $\frac{u}{aub}$ for $u \in \{a, b\}^*$.

   a) Use *rule induction* to prove that every string in L is of the form $a^n b^n$ for some $n \in \mathbb{N}$.

   > We prove that for every string s in the set L inductively defined by the axiom and rule, there exists a natural number n such that $s = a^n b^n$.
   >
   > **Axiom** $\frac{}{\varepsilon}$ The string s must be the empty string ε, and for $n = 0$ we have that $\varepsilon = a^0 b^0$.
   >
   > **Rule** $\frac{u}{aub}$ Let $s = aub$ for some string u and assume the Ⓘⓗ: there exists a natural number k such that $u = a^k b^k$. Then, $s = aub \overset{\text{IH}}{=} aa^k b^k b = a^{k+1} b^{k+1}$ so the witness $n = k + 1$ satisfies the required property.

   b) Use *mathematical induction* to prove that for all $n \in \mathbb{N}$, $a^n b^n \in L$.

   > **Base case**: $n = 0$. The string $a^0 b^0$ is the empty string ε, which is an element of L by the defining axiom.
   >
   > **Inductive step** $n = k + 1$. Assume the (IH): $a^k b^k \in L$. We prove that the string $a^{k+1} b^{k+1}$ is in L as well. By definition of string repetition, $a^{k+1} b^{k+1} = aa^k b^k b$. The (IH) states that $a^k b^k \in L$, and the rule can be applied to deduce that $aa^k b^k b \in L$ as well.

   c) Conclude that $L = \{a^n b^n \mid n \in \mathbb{N}\}$.

   > In the previous two parts we have shown that every string of L is of a particular form $a^n b^n$ for $n \in \mathbb{N}$, and that every string of this form is in L. Thus, we have the subset inclusions $L \subseteq \{a^n b^n \mid n \in \mathbb{N}\}$ and $\{a^n b^n \mid n \in \mathbb{N}\} \subseteq L$, proving that the sets are equal.

   d) Suppose we add the string a to L to get $L' = L \cup \{a\}$. Is L' closed under the axiom and rule? If not, characterise the strings that would be in the smallest set containing L' that is closed under the axiom and rule.

   > The resulting language L' would not be closed: we can use the rule to generate the strings aab, $aaabb$, $a^{n+1} b^n$, which are not of the required form and therefore are not already part of the language. The closure of L' under the rule and axiom would therefore be $\{a^n b^n \mid n \in \mathbb{N}\} \cup \{a^{n+1} b^n \mid n \in \mathbb{N}\}$.

2. Suppose $R \colon X \nrightarrow X$ is a binary relation on a set X. Let $R^\dagger \colon X \nrightarrow X$ be inductively defined by the following axioms and rules:

$$\frac{}{(x,x)\in R^\dagger}\ (x\in X) \qquad \frac{(x,y)\in R^\dagger}{(x,z)\in R^\dagger}\ (x\in X \text{ and } y\,R\,z)$$

a) Show that $R^\dagger$ is reflexive and that $R\subseteq R^\dagger$.

> We show that $R^\dagger$ is reflexive by giving a derivation of $(x,x)\in R^\dagger$ for all $x\in X$. This is simply the first axiom defining the relation.
>
> Next, we show that for all $(x,y)\in R$, $(x,y)\in R^\dagger$ by providing a derivation:
>
> $$\frac{\dfrac{}{(x,x)\in R^\dagger}}{(x,y)\in R^\dagger}\ (x\in X \text{ and } x\,R\,y)$$

b) Use rule induction to show that $R^\dagger$ is a subset of

$$S \triangleq \left\{(y,z)\in X\times X \;\middle|\; \forall x\in X.\,(x,y)\in R^\dagger \implies (x,z)\in R^\dagger\right\}$$

Deduce that $R^\dagger$ is transitive.

> We show that for all $(y,z)\in R^\dagger$, we have that for all $x\in X$ such that $(x,y)\in R^\dagger$, $(x,z)\in R^\dagger$ by rule induction.
>
> **Axiom** $\frac{}{(y,y)\in R^\dagger}$ We clearly have $(x,y)\in R^\dagger$ implying $(x,y)\in R^\dagger$, as required.
>
> **Rule** $\frac{(x,y)\in R^\dagger}{(x,z)\in R^\dagger}$ with ① $y\,R\,z$. Assume the ⒾⒽ: $(x,y)\in S$. To show $(x,z)\in S$, let $w\in X$ be an element and suppose that ② $(w,x)\in R^\dagger$; we prove $(w,z)\in R^\dagger$ by giving a derivation:
>
> $$\frac{\dfrac{\text{IH}}{(w,y)\in R^\dagger}}{(w,z)\in R^\dagger}\ (w\in X \text{ and ① } y\,R\,z)$$
>
> where the ⒾⒽ $(x,y)\in S$ is applied to the assumption ② $(w,x)\in R^\dagger$ to deduce $(w,y)\in R^\dagger$, as required.
>
> To prove that $R^\dagger$ is transitive, we need to show that $(x,y),(y,z)\in R^\dagger$ implies $(x,z)\in R^\dagger$. Since $R^\dagger\subseteq S$, we also have $(y,z)\in S$, which, by definition of S and the assumption $(x,y)\in R^\dagger$ implies $(x,z)\in R^\dagger$.

c) Suppose that $T : X \nrightarrow X$ is a reflexive and transitive binary relation and that $R\subseteq T$. Use rule induction to show that $R^\dagger\subseteq T$.

> We show that for all $(x,y)\in R^\dagger$, $(x,y)\in T$ by rule induction.
>
> **Axiom** $\frac{}{(y,y)\in R^\dagger}$ Since T is reflexive, we have that $(y,y)\in T$.

> **Rule** $\frac{(x,y)\in R^\dagger}{(x,z)\in R^\dagger}$ with ① $y\,R\,z$. Assume the ⒾⒽ: $(x,y)\in T$. Since $R\subseteq T$, we also have $(y,z)\in T$ from ①; then, since T is transitive, we deduce $(x,z)\in T$ using the ⒾⒽ, which is what we were meant to prove.

d) Deduce from above that $R^\dagger$ is equal to R^*, the reflexive-transitive closure of R.

> In parts (a) and (b) we showed that $R^\dagger$ is reflexive and transitive; in part (a) we also proved that $R\subseteq R^\dagger$. Finally, part (c) established that $R^\dagger$ is smaller than any other reflexive-transitive superset of R, which is the universal characterisation of the reflexive-transitive closure of R.

3. Let L be a subset of $\{a,b\}^*$ inductively defined by the axiom and rules (for $u\in\{a,b\}^*$):

$$\frac{}{ab}\qquad\qquad \frac{au}{au^2}\qquad\qquad \frac{ab^3u}{au}$$

a) Is ab^5 in L? Give a derivation, or show that there isn't one.

> The string ab^5 is indeed in L, as witnessed by the following derivation:
>
> $$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{}{ab}}{ab^2}}{ab^4}}{ab^8}}{ab^5}$$

b) Use rule induction to show that every $u\in L$ is of the form ab^n with $n=2^k-3m\geq 0$ for some $k,m\in\mathbb{N}$.

> **Axiom** $\frac{}{ab}$ We have that $ab=ab^1$ and $1=2^k-3m$ for $k=m=0$.
>
> **Rule** $\frac{au}{au^2}$ If by the IH we have that $u=b^{2^l-3n}$, then $u^2=b^{2^{l+1}-3\cdot(2n)}$, so au^2 is of the required form with $k=l+1$ and $m=2n$.
>
> **Rule** $\frac{ab^3u}{au}$ If by the IH we have that $b^3u=b^{2^l-3n}$, then by removing the first 3 bs we have that $u=b^{2^l-3(n+1)}$; thus, au is of the required form with $k=l$ and $m=n+1$.

c) Is ab^3 in L? Give a derivation, or show that there isn't one.

> If ab^3 were in L, by part (b) it must be of the form ab^{2^k-3m} for some $k,m\in\mathbb{N}$. This is not possible however, since $2^k-3m=3\iff 2^k=3(m+1)$ would require $3(m+1)$ to be a power of 2; but the only prime factor of 2^k is 2 so it can't a multiple of 3.

d) Find an explicit characterisation of the elements of the language as a set comprehension, and prove (along the lines of §14.1) that it coincides with the inductively defined set L.

We claim that $L = \left\{ ab^{2^k - 3m} \,\middle|\, 2^k - 3m \geq 0 \right\}$. We've already shown the $\subseteq$ direction, proving that every string in L is of the appropriate form. We now show that every string of the appropriate form has a derivation; namely, that for all $k \in \mathbb{N}$,

$$\forall m \in \mathbb{N}.\ 2^k - 3m \geq 0 \Longrightarrow ab^{2^k-3m} \in L$$

which we prove by mathematical induction on k.

Base case: $k = 0$. The only m for which the hypothesis $2^0 - 3m \geq 0$ is satisfied is $m = 0$, and for this we have a derivation of $ab^{2^0-3\cdot 0} = ab \in L$ by the axiom.

Inductive step: $k = l + 1$. Assume the ⒾⒽ:

$$\forall m \in \mathbb{N}.\ 2^l - 3m \geq 0 \Longrightarrow ab^{2^l-3m} \in L$$

and prove $\forall m \in \mathbb{N}.\ 2^{l+1} - 3m \geq 0 \Longrightarrow ab^{2^{l+1}-3m} \in L$ by nested mathematical induction on m.

Inner base case: $m = 0$. By the ⒾⒽ we have that $ab^{2^l} \in L$, and by applying the rule $\frac{au}{au^2}$ we can derive $ab^{2^{l+1}} \in L$.

Inner inductive step: $m = n + 1$. If, by the nested IH we have that $ab^{2^{l+1}-3n} \in L$, then by applying the rule $\frac{ab^3u}{au}$ we can derive $ab^{2^{l+1}-3n-3} = ab^{2^{l+1}-3(n+1)} \in L$.

15. On regular expressions

1. Find regular expressions over $\{0, 1\}$ that determine the following languages:

   a) $\{u \mid u$ contains an even number of 1's$\}$

   We should only be able to add 1s in pairs, so we take the regex $(0|10^*1)^*$.

   b) $\{u \mid u$ contains an odd number of 0's$\}$

   After requiring one 0, we ask for an even number of 0s: $1^*0(1|01^*0)^*$.

2. Show that $b^*a(b^*a)^*$ and $(a|b)^*a$ are equivalent regular expressions, that is, a string matches one iff it matches the other. Your reasoning should be rigorous but can be informal.

   First note that any string u matching $b^*a(b^*a)^*$ is a concatenation $u = u_1u_2\cdots u_n$ of one or more (i.e. $n \geq 1$) strings in $\{a, b\}^*$ matching b^*a. Each u_i ends with an a and hence (because $n \geq 1$), so does u. Therefore u matches $(a|b)^*a$.

   Conversely, if u matches $(a|b)^*a$ it is a string in $\{a, b\}^*$ ending with an a: looking at the occurrences of a in u, we can express u as $u = b^{n_1}ab^{n_2}a\cdots b^{n_k}a$ for some $k \geq 1$ and some $n_1, \ldots n_k \geq 0$; and hence u matches $b^*a(b^*a)^*$.

   ♪ Equivalence of regular expressions is more difficult to establish in general – reasoning by "observation" or pattern analysis like above does not scale to more complicated regexes.

> The question will be revisited, however, in the second half of the course, using some additional developments that will allow us to check equivalence of regular expressions in finite time.

3. Extend the concrete syntax, abstract syntax, parsing relation of regular expressions, and the matching relation between strings and regular expressions with the following constructs:

   a) $r?$: matches the regex r zero or one times. For example, $ab?c$ is matched by ac and abc, but not $abbc$.

   b) r^+: matches the regex r one or more times. For example, ab^+c is matched by abc and $abbbbc$, but not ac.

> We extend the alphabet Σ' with the symbols ? and $^+$ and the concrete syntax with the following rules:
>
> $$\frac{r}{r?} \qquad\qquad \frac{r}{r^+}$$
>
> The abstract syntax is extended with the unary constructors *Opt* and *Plus*, and parsing is modified as follows:
>
> $$\frac{r \sim R}{r? \sim Opt(R)} \qquad\qquad \frac{r \sim R}{r^+ \sim Plus(R)}$$
>
> Finally, we add the following axioms and rules to the matching relation:
>
> $$\frac{}{(\varepsilon, r?)} \qquad \frac{(u, r)}{(u, r^+)} \qquad \frac{(u, r)}{(u, r?)} \qquad \frac{(u, r) \qquad (v, r^+)}{(uv, r^+)}$$

Show that $(r^+)?$ is equivalent to r^*. Is that the case for $(r?)^+$ as well?

> The regex $(r^+)?$ matches either the empty string ε, or one or more repetitions of a string matched by r. Combined, it matches zero or more repetitions of a string matched by r, which is precisely the meaning of r^*.
>
> The regex $(r?)^+$ matches one or more repetitions of either the empty string, or a string matched by r. In particular, it matches the empty string (if all the repetitions are empty), and any nonzero number of occurrences of r. Again, this is the same as the meaning of r^*.
>
> ♪ This question involved adding two new constructs to our regex syntax, and as the last part showed, the system is now "non-orthogonal" in that certain regexes are interderivable. This is not necessarily a problem – many formal systems exhibit this form of redundancy – but it does make the inductively defined syntax larger which may complicate reasoning about the system (for example, every recursive definition or inductive proof on regexes now has two extra cases that would be covered by existing ones). Thus we may also reasonably choose to define $r?$ and s^+ as "syntactic sugar" (extra notation added for convenience)

abbreviating $r|\epsilon$ and ss^*, respectively. These, and other derived operators and patterns form the basis of practical regex engines used widely in text processing applications.

16. On finite automata

1. For each of the two languages mentioned in §15.1 (string containing an even number of 1's or an odd number of 0's), find a DFA that accepts exactly that set of strings.

We can construct both with only two states each, corresponding to whether we have seen an even or odd number of 1's or 0's and making the appropriate state accepting.

$$\begin{tikzcd} {} \arrow[r] & \text{e1s} \arrow[loop above, "0"] \arrow[r, bend left, "1"] & \text{o1s} \arrow[loop above, "0"] \arrow[l, bend left, "1"] \end{tikzcd} \qquad \begin{tikzcd} {} \arrow[r] & \text{e0s} \arrow[loop above, "1"] \arrow[r, bend left, "0"] & \text{o0s} \arrow[loop above, "1"] \arrow[l, bend left, "0"] \end{tikzcd}$$

2. Given an NFA$^\varepsilon$ $M = (Q, \Sigma, \Delta, s, F, T)$, we write $q \overset{u}{\Rightarrow} q'$ to mean that there is a path in M from state q to state q' whose non-ε labels form the string $u \in \Sigma^*$. Show that $L = \left\{ (q,u,q') \,\middle|\, q \overset{u}{\Rightarrow} q' \right\}$ is equal to the subset of $Q \times \Sigma^* \times Q$ inductively defined by the axioms and rules:

$$\frac{}{(q,\varepsilon,q)} \qquad \frac{(q,u,q')}{(q,u,q'')} \text{ if } q' \xrightarrow{\varepsilon} q'' \text{ in } M \qquad \frac{(q,u,q')}{(q,ua,q'')} \text{ if } q' \xrightarrow{a} q'' \text{ in } M$$

Hint: recall the method from §14.1. for showing that a language defined via set comprehension is equal to an inductively defined set: first show that L is closed under the rules and axioms, then show that every string in L has a derivation.

($\subseteq$) We show that every element (q,u,q') of the inductively defined set L satisfies $q \overset{u}{\Rightarrow} q'$ by rule induction.

Axiom $\frac{}{(q,\varepsilon,q)}$ We can always transition from a state to itself without consuming any symbols, so $q \overset{\varepsilon}{\Rightarrow} q$ holds vacuously.

Rule $\frac{(q,u,q')}{(q,u,q'')}$ where $q' \xrightarrow{\varepsilon} q''$ in M. The IH states that $q \overset{u}{\Rightarrow} q'$. If there is an ε-transition from q' to q'', we can make one further step without consuming a symbol, so the overall string formed by the non-ε labels will still be u – hence, $q \overset{u}{\Rightarrow} q''$, as required.

Rule $\frac{(q,u,q')}{(q,ua,q'')}$ where $q' \xrightarrow{a} q''$ in M. The IH states that $q \overset{u}{\Rightarrow} q'$. If there is a transition from q' to q'' labelled with a, we can make a further step that extends the recognised string with the symbol a – hence, $q \overset{ua}{\Rightarrow} q''$, as required.

($\supseteq$) We show that we can derive $(q,u,q') \in L$ whenever $q \overset{u}{\Rightarrow} q'$ in M by mathematical induction on the length $|u| \in \mathbb{N}$ of the string u.

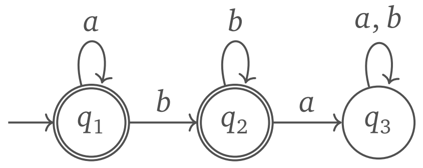
Base case: $|u| = 0$, so $u = \varepsilon$. If the number of non-ε symbols in the path is 0, all of the steps must have been ε-transitions. Such paths can be captured by the axiom, and any number of applications of the first rule to transition between states without consuming an input.

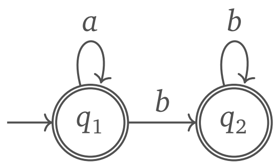
Indutive step: $|u| = k+1$, so $u = va$ for $a \in \Sigma$ and $|v| = k$. Assume the (IH): we have a path $q \overset{v}{\Longrightarrow} q'$ and therefore $(q, v, q') \in L$. If we have a path labelled va, there must be a transition from q' to q'' labelled by a; then, the second rule can be applied to $(q, v, q') \in L$ and $q' \xrightarrow{a} q''$ to deduce $(q, va, q') \in L$, as required.

$\boxed{♪}$ Even though strings are no different from finite lists of symbols of the alphabet, they are formally elements of Σ^*, not a set inductively defined with an axiom for the empty string, and a rule for "consing" a symbol to the string. Performing induction on the length of the string simulates the kind of (structural) induction one would perform on an OCaml-style list.

3. The example of the subset construction given on Slide 58 constructs a DFA with eight states whose language of accepted strings happens to be $L(a^* b^*)$. Give an "optimised" DFA with the same language of accepted strings, but fewer states. Give an NFA with even fewer states that does the same job.

The simplified NFA and DFA are as follows:





The main difference is that the DFA needs to handle the symbol a occurring in state q_2, which would mean seeing an occurrence of a after a b which disqualifies the string from being in $L(a^* b^*)$. The usual way of marking this as an invalid input in a DFA is to transition into a state from which it is impossible to reach an accepting state; upon any further input just stays stuck in q_3.