Problem Set for Tutorial 1 — TDDD14/TDDD85

1 Strings and Sets

Exercise 1. Let w be the string *abcde*.

1. Give all prefixes of w.
2. Give all suffixes of w.
3. Which strings are both prefixes and suffixes of w?

Exercise 2. Suppose $L_1 = \{carl, hugh, paul\}$ and $L_2 = \{smith, jones\}$. Enumerate the strings which belong to the language $L_3 = L_1 L_2$ (i.e., $L_3 = \{xy \mid x \in L_1, y \in L_2\}$).

Exercise 3. If L is a language, then L^n denotes the language obtained by concatenating L n times, where $L^0 = \{\varepsilon\}$ and for $n > 0$, $L^n = L \cdot L^{n-1}$. Furthermore, L^* denotes $\bigcup_{n=0}^{\infty} L^n$.

Given $L_1 = \{mor, far\}$ and $L_2 = \{s\}$, give examples of strings in the language $(L_1^2 L_2)^* L_1 \cup L_1^2$.

Solutions

Solution to Exercise 1.

1. ε, a, ab, abc, $abcd$, $abcde$.
2. ε, e, de, cde, $bcde$, $abcde$.
3. ε and the string itself, $abcde$.

Solution to Exercise 2. $carlsmith$, $carljones$, $hughsmith$, $hughjones$, $paulsmith$, $pauljones$.

Solution to Exercise 3. We proceed systematically and begin by describing

$$L_1^2 = \{mormor, morfar, farmor, farfar\}.$$

Next, $L_1^2 L_2$ is then formed by taking all strings in L_1^2 and appending an s, i.e.,

$$L_1^2 L_2 = \{mormors, morfars, farmors, farfars\}.$$

The set $(L_1^2 L_2)^* L_1$ then contains strings built up by zero or more strings from $L_1^2 L_2$ followed by a string from L_1, e.g., $mormorsmor$, $morfarsfar$. Last, we should not forget that $(L_1^2 L_2)^* L_1 \cup L_1^2$ also contains L_1^2 which we have already described.

2 Mathematical Proofs

Exercise 4. Let $A, B, C \subseteq \Sigma^*$ for some alphabet Σ. Prove that the following properties hold.

1. $A \cup B = B \cup A$ and $A \cap B = B \cap A$ ($\cup$ and $\cap$ are commutative).
2. $A(B \cup C) = AB \cup AC$ (set concatenation distributes over union).

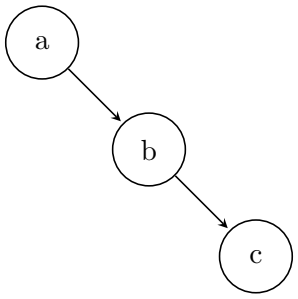
Exercise 5. Consider the set of all strings over the Boolean alphabet $\Sigma = \{0, 1\}$. Say that a string w is a *palindrome* if it reads the same forward and backward. For example, 00100 and 111 are palindromes, while 0101 is not.

1. Is ww for $w \in \{0, 1\}^*$ always a palindrome?
2. Is ww for $w \in \{0, 1\}^*$ a palindrome whenever w is a palindrome? Prove your claim.

Hint: do not use induction, it is easier to give direct proofs.

The following exercise can be skipped if you are already comfortable with proofs by mathematical induction. However, if you feel slightly unsure, we strongly recommend you to try it.

Exercise 6. The depth of a node v in a tree T is defined as follows: If v is the root node of T, then the depth of v is 0. Otherwise, v belongs to a subtree T' of the root of T (i.e., T' is a tree such that the root of T is the parent of the root of T'), and the depth of v in T is defined to be one more than the depth of v in T'. As an example, the depth of the node c in the tree below is 2.



The height of a tree is the largest depth of a node in the tree. A tree is called a binary tree if every node has either no children or exactly two children. (So the tree in the diagram is not binary). Suppose T is a binary tree of height k. Show that T has n nodes where n satisfies the condition $2k + 1 \leq n \leq 2^{k+1} - 1$.

Exercise 7. If Σ is an alphabet, then Σ^* denotes the language which comprises all strings that can be formed by using the symbols in Σ. For $x \in \Sigma^*$, let x^R denote x reversed and be defined recursively as follows:

1. If $x = \epsilon$, then $x^R = \epsilon$.
2. If $x = ay$ for some $a \in \Sigma$ and $y \in \Sigma^*$, then $x^R = y^R a$.

Let $|x|$ denote the length of a string x. Give a recursive definition of the length of a string and then show $|x| = |x^R|$ for all strings $x \in \Sigma^*$. Hint: a formal proof should use induction over the length of the string x.

Exercise 8. The set of all subsets of a set A is called the power set of A, denoted by 2^A.

1. Give 2^A for $A = \emptyset$.
2. Give 2^A for $A = \{a, b, c\}$.
3. Show by induction that the number of elements in 2^A is 2^n if the number of elements in A is n.

Solutions

Solution to Exercise 4. We are given that $A, B, C \subseteq \Sigma^*$ for some alphabet Σ. Here, we prove universal statements so it is not sufficient to make any particular assumptions on the sets A, B, and C.

1. To show that $\cup$ and $\cap$ are commutative, i.e., $A \cup B = B \cup A$ and $A \cap B = B \cap A$: for $\cup$, let $x \in A \cup B$. This means that $x \in A$ or $x \in B$. If $x \in A$, then it is also true that $x \in B \cup A$. Similarly, if $x \in B$, then $x \in B \cup A$. Hence, $A \cup B \subseteq B \cup A$. By a symmetric argument, $B \cup A \subseteq A \cup B$. Therefore, $A \cup B = B \cup A$. The $\cap$ case follows through similar arguments.

2. To show that set concatenation distributes over union, i.e., $A(B \cup C) = AB \cup AC$, we first prove that $A(B \cup C) \subseteq AB \cup AC$ and then that $AB \cup AC \subseteq A(B \cup C)$. First, let $w \in A(B \cup C)$. By definition this means there exists an $a \in A$ and a $b \in B \cup C$ such that $w = ab$. If $b \in B$, then $w = ab$ where $a \in A$ and $b \in B$, so $w \in AB$. Similarly, if $b \in C$, then $w = ab$ where $a \in A$ and $b \in C$, so $w \in AC$. Therefore, $A(B \cup C) \subseteq AB \cup AC$.

   Conversely, let $w \in AB \cup AC$. This implies that either $w \in AB$ or $w \in AC$. If $w \in AB$, then there exists an $a \in A$ and a $b \in B$ such that $w = ab$. Since $b \in B$, it follows that $b \in B \cup C$, and thus $w \in A(B \cup C)$. Similarly, if $w \in AC$, then there exists an $a \in A$ and a $c \in C$ such that $w = ac$. Since $c \in C$, it follows that $c \in B \cup C$, and thus $w \in A(B \cup C)$. Hence, $AB \cup AC \subseteq A(B \cup C)$.

   Combining both directions, we conclude that $A(B \cup C) = AB \cup AC$.

We remark that more elemental properties of the basic set operations can be established and that they form a *Boolean algebra* (if we exclude the star and asterisk operations, whose presence instead leads to a *Kleene algebra*).

Solution to Exercise 5.

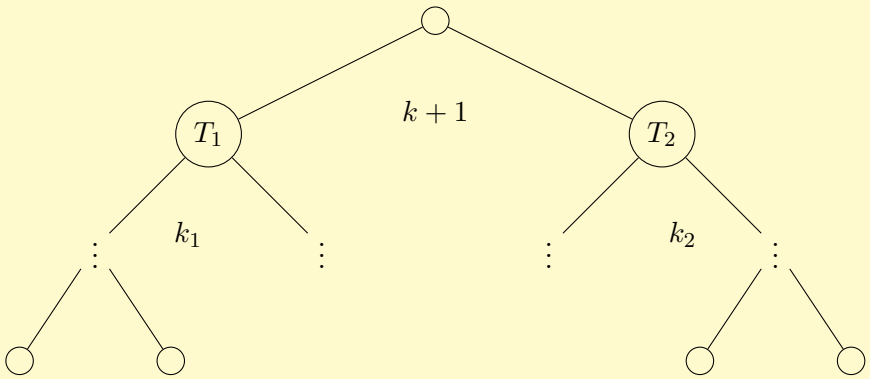
1. We suspect that it might not be true since it seems like a very strong property. We find no counter examples of strings of length $n = 0$ or $n = 1$, but for $n = 2$ we (e.g.) see that if $w = 01$ then $ww = 0101$ is not a palindrome.

2. We cannot find a counter example so we suspect that the claim is true. Assume that $w = b_1 \dots b_n \in \{0, 1\}^*$ is a string of length $n \geq 0$. We first realize that if $n = 0$ then the claim is trivially true (since ε is a palindrome, and since $\varepsilon\varepsilon = \varepsilon$) so we assume that $n \geq 1$. Otherwise, we are not allowed to make any particular assumptions on the string. Now, consider the string $ww = b_1 \dots b_n b_1 \dots b_n$. It follows that $b_1 = b_n$, $b_2 = b_{n-1} \dots b_n = b_1$, and we conclude that the string is indeed a palindrome.

Solution to Exercise 6. For $k \geq 0$ let IH(k) be the statement: 'the number of nodes n in a binary tree of height k satisfies the condition $2k + 1 \leq n \leq 2^{k+1} - 1$'. We will prove by

induction on k that IH(k) holds for all $k \geq 0$.

- **Base case**: For $k = 0$, a binary tree of height 0 consists of a single node, thus $n = 1$, which satisfies $2 \cdot 0 + 1 = 1 \leq n \leq 2^1 - 1 = 1$, so IH(0) holds.
- **Inductive Hypothesis**: Suppose IH(k) holds for some $k \geq 0$.
- **Inductive step**: Consider a binary tree of height $k + 1$ as in the figure below, where the two subtrees T_1 and T_2 has height $0 \leq k_1 \leq k$, respectively $0 \leq k_2 \leq k$. Then either $k_1 = k$ or $k_2 = k$. Let n_1 and n_2 be the numbers of nodes in T_1 and T_2, respectively. By the inductive hypothesis, $n_1 \leq 2^{k+1} - 1$ and $n_2 \leq 2^{k+1} - 1$. The total number of nodes in the tree is $n = 1 + n_1 + n_2 \leq 1 + 2(2^k + 1 - 1) = 2^{k+1} + 1 - 1$. Similarly, $n = 1 + n_1 + n_2 \geq 1 + (2k + 1) + 1 = 2(k + 1) + 1$. Thus, $2(k + 1) + 1 \leq n \leq 2^{k+2} - 1$, proving that IH($k + 1$) holds.



Solution to Exercise 7. The length of a string x, denoted $|x|$, can be defined recursively as follows:

1. If $x = \epsilon$, then $|x| = 0$.
2. If $x = ay$ for some $a \in \Sigma$ and $y \in \Sigma^*$, then $|x| = 1 + |y|$.

Let IH(k) be: $|x| = k$ if and only if $|x^R| = k$, i.e., $|x| = |x^R|$. Show that IH(k) holds for all $k \geq 0$.

1. **Base case**: IH(0) holds:

$$|x| = 0 \Leftrightarrow x = \epsilon \Leftrightarrow x^R = \epsilon \Leftrightarrow |x^R| = 0.$$

2. **Inductive hypothesis**: Suppose IH(k) holds for some $k \geq 0$.
3. **Inductive step**: Show that IH($k + 1$) then holds.

$$|x| = k + 1 \Leftrightarrow x = ay \text{ and } |y| = k$$

for some $a \in \Sigma$ and $y \in \Sigma^*$. The induction hypothesis implies $|y^R| = k$ and thus we have $|x^R| = |y^R a| = k + 1$.

What is missing? From the definition of $|\cdot|$ above, it does not immediately follow that the equality $|y^R a| = k + 1$ holds. We can explicitly show this by induction.

Let $\mathrm{IH}'(k)$ be: 'if $|x| = k$ then $|xa| = k + 1$, for any $x \in \Sigma^*$ and $a \in \Sigma$'.

1. **Base case**: $\mathrm{IH}'(0)$ holds.

   If $|x| = 0$, then $x = \epsilon$. Thus $|xa| = |a| = 1$.

2. **Inductive hypothesis**: Suppose $\mathrm{IH}'(k)$ holds for some $k \geq 0$.

3. **Inductive step**: Show that $\mathrm{IH}'(k + 1)$ then holds. If $|y| = k + 1$ then $y = bx$, where $|x| = k$. So $ya = bxa$. By the assumption, $|xa| = k + 1$. Hence $|bya| = |bxa| = k + 2$ by the definition of $|\cdot|$.

Solution to Exercise 8.

1. For $A = \emptyset$, the power set $2^A = \{\emptyset\}$ because the only subset of the empty set is the empty set itself.

2. For $A = \{a, b, c\}$, the power set 2^A includes the following subsets:

$$2^A = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}.\}$$

3. We now prove by induction that the number of elements in 2^A is 2^n if the number of elements in A is n.

   (a) **Base Case**: For $n = 0$, $A = \emptyset$, and $2^A = \{\emptyset\}$. Thus, the number of elements in 2^A is $1 = 2^0$, which holds true.

   (b) **Inductive Hypothesis**: Assume that for a set A with k elements, the power set 2^A contains 2^k subsets.

   (c) **Inductive Step**: Let $x \notin A$ and consider the number of subsets of $A \cup \{x\}$. Note that $A \cup \{x\}$ has cardinality $k + 1$. By the inductive hypothesis we know that $|2^A| = 2^k$, and for each $B \subseteq A$ we can obtain a new subset of $A \cup \{x\}$ via $B \cup \{x\}$. Hence, $2^{A \cup \{x\}} = 2^A \cup \{B \cup \{x\} \mid B \subseteq A\}$, and since $|\{B \cup \{x\} \mid B \subseteq A\}| = |\{B \mid B \subseteq A\}| = |2^A| = 2^k$ we conclude that $|2^{A \cup \{x\}}| = 2^k + 2^k = 2^{k+1}$.

3 Advanced Exercises

Exercise 9. A *monoid* is a tuple $(M, \cdot)$ where M is a set referred to as the *base set*, or *universe*, of the monoid, and where $\cdot : M \times M \to M$ is a binary operation on M such that the following properties hold.

- *Associativity:* For all $a, b, c \in M$, it holds that $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.
- *Identity Element:* There exists an element $e \in M$, known as the *identity element*, such that for every $a \in M$, $e \cdot a = a \cdot e = a$.

Prove that $(\Sigma^*, \cdot)$ is a monoid, where Σ is an arbitrary alphabet and where we write $\cdot$ for string concatenation.

Exercise 10. Let Σ be an alphabet and $L \subseteq \Sigma^*$ a language. Consider the relation $R_L \subseteq \Sigma^* \times \Sigma^*$ defined by: xR_Ly if and only if for all $z \in \Sigma^*$, $xz \in L \Leftrightarrow yz \in L$.

1. Show that R_L is an equivalence relation[1].
2. Consider the language $L = \{(01)^n | n \geq 0\}$ over the alphabet $\Sigma = \{0, 1\}$. Does R_l have a finite or infinite amount of equivalence classes[2].
3. The same question but for the language $L = \{0^n 1^n | n \geq 1\}$ over $\Sigma = \{0, 1\}$.

[1] An equivalence relation on a set is a relation that is reflexive (every element is related to itself), symmetric (if a is related to b, then b is related to a), and transitive (if a is related to b and b is related to c, then a is related to c).

[2] An equivalence class $[x]$ of a string $x \in \Sigma^*$ is defined as the set of all strings related to x, i.e., $[x] = \{y \mid xR_Ly\}$

Solutions

Solution to Exercise 9. This exercise is not as hard as one might believe at a first glance. We are simply being asked to verify that if we as our universe takes all strings over an alphabet Σ and consider the operation of concatenating two strings together, then this operator behaves roughly as multiplication. Since we have two properties to verify we work systematically and prove each property in turn.

- *Associativity:* Let $x, y, z \in \Sigma^*$ be strings. Then it clearly holds that $(x \cdot y) \cdot z = x \cdot (y \cdot z) = xyz$.
- *Identity Element:* we choose the empty string $\varepsilon \in \Sigma^*$. It follows that $\varepsilon \cdot x = x \cdot \varepsilon = x$ for any $x \in \Sigma^*$.

Why is it useful to know that $(\Sigma^*, \cdot)$ is a monoid? This formalizes the intuition that $\cdot$ behaves roughly as multiplication and, in this context, then that the empty string ε behaves as 1. With this intuition it is easy to e.g. see why $\varepsilon \cdot \varepsilon = \varepsilon$ since 1 multiplied by itself still yields 1.

Solution to Exercise 10.

1. To demonstrate that R_L is an equivalence relation, we must prove that R_L is reflexive, symmetric, and transitive.
   (a) **Reflexive:** For all $x \in \Sigma^*$, we must show xR_Lx. Choosing an arbitrary string $x \in \Sigma^*$, it is evident that for all $z \in \Sigma^*$, $xz \in L \Leftrightarrow xz \in L$. Thus, xR_Lx.
   (b) **Symmetric:** For all $x, y \in \Sigma^*$, we must show $xR_Ly \Rightarrow yR_Lx$. Given $x, y \in \Sigma^*$ such that xR_Ly, xR_Ly iff for all $z \in \Sigma^*$, $xz \in L \Leftrightarrow yz \in L$. We conclude that for all $z \in \Sigma^*$, $yz \in L \Leftrightarrow xz \in L$, i.e., yR_Lx.
   (c) **Transitive:** For all $x, y, w \in \Sigma^*$, we must show $xR_Ly \wedge yR_Lw \Rightarrow xR_Lw$. Choosing $x, y, w \in \Sigma^*$ such that xR_Ly and yR_Lw, and an arbitrary $z \in \Sigma^*$, if $xz \in L$, then xR_Ly implies $yz \in L$, which in turn implies $wz \in L$. Conversely, if $xz \notin L$, xR_Ly implies $yz \notin L$, which in turn implies $wz \notin L$. Thus, $xz \in L \Leftrightarrow wz \in L$, i.e., xR_Lw.
2. The equivalence classes constitute a partitioning of Σ^* and we want to know whether we have a finite or infinite amount of such classes. To determine the equivalence class for a string x, we analyze whether xR_Ly based on whether $xz \in L \Leftrightarrow yz \in L$. Note that this condition can be restated as $xz \in L \Rightarrow yz \in L$ and $xz \notin L \Rightarrow yz \notin L$. We then approach the question systematically and start by investigating small strings and hope that we see a pattern. Hence, we should first investigate the equivalence class $[\varepsilon]$, and if it turns out that $0 \notin [\varepsilon]$ then we continue with $[0]$, $[1]$, and so on.
   (a) $[\varepsilon]$**:** For $x = \varepsilon$, $xz = z \in L$ implies $z = (01)^n, n \geq 0$. Thus, if $z = (01)^n$ and $yz \in L$, it must be the case that $y = (01)^m, m \geq 0$ since $yz = (01)^m(01)^n = (01)^{m+n}$.

Conversely, if $xz = z \notin L$, then $z \neq (01)^n, n \geq 0$, and $yz \notin L$. Therefore, $[\varepsilon] = \{(01)^m | m \geq 0\}$.

(b) $[0]$**:** $x = 0$ and $xz = 0z \in L$ implies $z = 1(01)^n, n \geq 0$. If $z = 1(01)^n$ and $yz \in L$, then $y = (01)^m 0$ since $yz = (01)^m 01(01)^n = (01)^{m+n+1}$. Therefore, $[0] = \{(01)^m 0 | m \geq 0\}$.

(c) $[1]$**:** $x = 1$ implies that for any $z \in \Sigma^*$, $1z \notin L$, meaning all strings y in $[1]$ must satisfy $yz \notin L$ for any chosen z. Therefore, $[1] = \Sigma^* - ([\epsilon] \cup [0])$.

We conclude that $\Sigma^* = [\varepsilon] \cup [0] \cup [1]$ and thus that R_L has a finite amount of equivalence classes.

3. Here we get an infinite number of equivalence classes since each string of the form $0n$, $n \geq 0$, is only related to itself. This yields the following classes.

   - $[0^n] = \{0^n\}, n \geq 0$, meaning each string consisting of n zeros is only related to itself.
   - $[01] = \{0^n 1^n | n \geq 1\}$, indicating the class of strings with n zeros followed by n ones.
   - $[0^{k+1}1] = \{0^{k+n}1^n | n \geq 1\}, k \geq 1$, representing strings with a sequence of zeros followed by an equal number of ones, starting with at least one zero.
   - $[1] = \Sigma^* - ([01] \cup (\bigcup_{n=0}^{\infty}[0^n]) \cup (\bigcup_{n=2}^{\infty}[0^n 1]))$, capturing all strings not included in the previous classes.

This exercise suggests an interesting property of a language L: whether R_L has a finite or infinite amount of equivalence classes. At this stage in the course it is perhaps hard to appreciate exactly why this is useful, but, as we will see in the context of the Myhill-Nerode theorem, this turns out to give a complete and powerful description of languages in the sense that simple languages always have a finite amout of equivalence classes while (comparably) complicated languages always have an infinite number of equivalence classes. This, in turn, can be related to the automata perspective where a finite number of states means that the language can be encoded by a machine with a finite number of states.