Diskret Matematik og Formelle Sprog: Problem Set 4

Due: Wednesday March 15 at 12:59 CET .
Submission: Please submit your solutions via *Absalon* as a PDF file. State your name and e-mail address close to the top of the first page. Solutions should be written in LaTeX or some other math-aware typesetting system with reasonable margins on all sides (at least 2.5 cm). Please try to be precise and to the point in your solutions and refrain from vague statements. Make sure to explain your reasoning. *Write so that a fellow student of yours can read, understand, and verify your solutions.* In addition to what is stated below, the general rules for problem sets stated on *Absalon* always apply.
Collaboration: Discussions of ideas in groups of two to three people are allowed and indeed, encouraged—but you should always write up your solutions completely on your own, from start to finish, and you should understand all aspects of them fully. It is not allowed to compose draft solutions together and then continue editing individually, or to share any text, formulas, or pseudocode. Also, no such material may be downloaded from or generated via the internet to be used in draft or final solutions. Submitted solutions will be checked for plagiarism.
Grading: A score of 120 points is guaranteed to be enough to pass this problem set.
Questions: Please do not hesitate to ask the instructor or TAs if any problem statement is unclear, but please make sure to send private messages—sometimes specific enough questions could give away the solution to your fellow students, and we want all of you to benefit from working on, and learning from, the problems. Good luck!

1 (60 p) Consider the regular expression $(a(bb|c^*)d^*)^*$ and determine which of the strings below belong to the language generated by this regular expression. Motivate your answers briefly but clearly by explaining for each string how it can be generated or arguing why it is impossible.

1. $bbacccd$
2. $abbdaccc$
3. $aaaccc$
4. $abbcccd$
5. $abbddcccddd$
6. $abbabba$

Solution: We give 10 p per fully correct and satisfactorily motivated answer. It stands to reason that any such answer will have to involve a discussion of how how to interpret the regular expression (in order to argue why accepted strings are accepted), so let us start by explaining this.

Because of the outermost star *, the regular expression $(a(bb|c^*)d^*)^*$ accepts a concatenation of zero or more strings on the following form:

- First comes exactly one character a.

- Then comes either the string bb or zero or more repetitions of the character c.
- Finally, we have zero or more repetitions of the character d.

With this in mind, we get the following answers:

1. $bbacccd$ ✗ (No string accepted by the regular expression can start with the character b.)
2. $abbdaccc = (abbd)\big(ac^3d^0\big)$ ✓
3. $aaaccc = \big(ac^0d^0\big)\big(ac^0d^0\big)\big(ac^3d^0\big)$ ✓
4. $abbcccd$ ✗ (No string accepted by the regular expression can have a character b directly followed by a character c—there has to be a character a in between.)
5. $abbddcccddd$ ✗ (No string accepted by the regular expression can have a d directly followed by a c—there has to be an a in between.)
6. $abbabba = \big(abbd^0\big)\big(abbd^0\big)\big(ac^0d^0\big)$ ✓

2 (90 p) Consider the following context-free grammars, where a, b, c are terminals, S, T, U are non-terminals, and S is the starting symbol.

Grammar 1:

$$S \rightarrow TaT \tag{1a}$$
$$T \rightarrow bT \tag{1b}$$
$$T \rightarrow ccT \tag{1c}$$
$$T \rightarrow \tag{1d}$$

Grammar 2:

$$S \rightarrow TaT \tag{2a}$$
$$T \rightarrow bTb \tag{2b}$$
$$T \rightarrow U \tag{2c}$$
$$U \rightarrow cU \tag{2d}$$
$$U \rightarrow \tag{2e}$$

Grammar 3:

$$S \rightarrow aTa \tag{3a}$$
$$T \rightarrow TbT \tag{3b}$$
$$T \rightarrow U \tag{3c}$$
$$U \rightarrow cUc \tag{3d}$$
$$U \rightarrow \tag{3e}$$

Which of these grammars generate regular languages? For each grammar, write a regular expression that generates the same language (and explain why), or argue why the language generated by the grammar is not regular. In your regular expressions, please use *only* the concatenation, alternative (|) and star (*) operators, and not the syntactic sugar extra operators that we just mentioned in class but never utilized.

Solution: We give 30 p per fully correct and satisfactorily motivated answer.

In **Grammar 1**, the non-terminal T generates any (possibly empty) alternation of b and cc, which corresponds to the regular expression $(b|cc)^*$. The first production $S \to TaT$ just means that somewhere in the string produced there has to appear a single a. It follows that the language generated by Grammar 1 is captured by the regular expression $(b|cc)^*a(b|cc)^*$.

Grammar 2 can generate strings on the form $ab^nc^mb^n$ for $m, n > 0$ by first applying $S \to TaT$ and then getting rid of the first T by using $T \to U$ and $U \to$. The second T will now have to generated a balanced number of characters b since the only way of generating this character is by using $T \to bTb$, and the productions $T \to U$ and $U \to cU$ can then produce a non-zero number of characters c in between the two sequences of bs. It follows by the same line of reasoning that the grammar *cannot* generate, for example, strings $ab^nc^mb^{n+1}$. In order to distinguish between strings $ab^nc^mb^n$ and $ab^nc^mb^{n+1}$ we need to be able to count, and this is nothing that finite automata can do. Hence, the language generated by Grammar 2 is *not* regular.

Note that although you do not need to prove that finite automata cannot count, but can refer to Mogesen's notes for this, some care is nevertheless needed to get a correct counting argument. To see one example of a faulty argument, note that it is also true that Grammar 2 can generate strings $(bb)^na(bb)^n$, but it is hard to get a counting argument out of this since also strings $(bb)^ma(bb)^n$ for $m \neq n$ can be generated. Hence, for this type of strings we cannot claim that counting is required, other than for keeping track of odd or even numbers of b, which a regular expression can certainly do.

In **Grammar 3**, the first production $S \to aTa$ makes sure that the string will start and end with a, and that everything in between these two characters a will be generated by the non-terminals T and U. Looking at the non-terminal U, the productions $U \to cUc$ and $U \to$ just mean that U produces an even number of characters c (possibly zero). In view of this, the productions $T \to TbT$ and $T \to U$ will just generate any (possibly empty) alternation of b and cc, which corresponds to the regular expression $(b|cc)^*$. We can conclude that the whole language generated by Grammar 3 is captured by the regular expression $a(b|cc)^*a$.

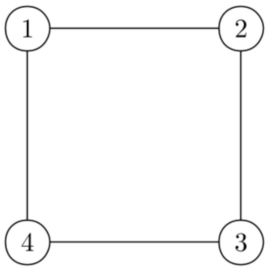
3 (150 p) We have learned in class about matrix multiplication, but there is also a way of multiplying matrices (or vectors) by just a number, which is called *scalar multiplication*. To multiply a matrix A by a number c, we multiply each entry $a_{i,j}$ in the matrix with the number c so that

$$c \cdot A = c \cdot \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix} = \begin{pmatrix} c \cdot a_{1,1} & c \cdot a_{1,2} & \cdots & c \cdot a_{1,n} \\ c \cdot a_{2,1} & c \cdot a_{2,2} & \cdots & c \cdot a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ c \cdot a_{m,1} & c \cdot a_{m,2} & \cdots & c \cdot a_{m,n} \end{pmatrix}$$

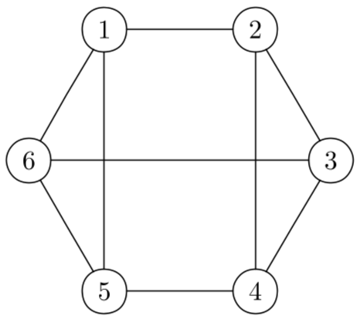
is the result of the scalar multiplication.

An intriguing phenomenon that sometimes arises is that for some pairs of matrices and vectors matrix multiplication and scalar multiplication give the same result. As an example of this, we have

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ 2 \\ 2 \end{pmatrix} = 2 \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \tag{4}$$



(a) Graph with adjacency matrix as in (4).



(b) Graph with adjacency matrix as in (5).

Figure 1: Two example regular graphs in Problem 3.

and another example is

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 3 \\ 3 \\ 3 \\ 3 \\ 3 \\ 3 \end{pmatrix} = 3 \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad . \tag{5}$$

When for a square matrix A there is a vector $\vec{x}$ (with not all entries equal to zero) and a number λ such that

$$A \cdot \vec{x} = \lambda \cdot \vec{x} \ , \tag{6}$$

such a vector $\vec{x}$ is called an *eigenvector* of the matrix A with *eigenvalue* λ. We see that the matrix in (4) has the all-ones vector as eigenvector with eigenvalue 2, and the matrix in (5) also has the all-ones vector as eigenvector but with eigenvalue 3. In this problem, we want to develop our skills of matrix multiplication by studying such eigenvalues and eigenvectors, and also to establish some non-obvious connections between eigenvalues and -vectors on the one hand and graphs on the other hand.

We say that an undirected, simple graph is *d-regular* if every vertex is incident to exactly d edges, or, in other words, has exactly d neighbours. For two illustrations of this, the graph in Figure 1a is 2-regular and the graph in Figure 1b is 3-regular. Now, a fun fact is that the 2-regular graph in Figure 1a has the adjacency matrix in (4), which has eigenvalue 2, and the 3-regular graph in Figure 1b has the adjacency matrix in (5) with eigenvalue 3. Your task is to show that this is not a coincidence, and to derive some other interesting connections between d-regular graphs and the eigenvalues and -vectors of their adjacency matrices.

3a (10 p) If $\vec{x}$ is an eigenvector of A corresponding to some eigenvalue λ, then so is $c \cdot \vec{x}$ for any $c \neq 0$. Explain why this is so.

Hint: This should be easy—just use the definitions above.

Solution: Suppose that A is an $n \times n$ matrix, just to fix the dimension. By definition, coordinate i in the vector $A\vec{x}$ is $\sum_{j=1}^{n} a_{i,j}x_j$. If λ is an eigenvalue with eigenvector $\vec{x}$, it holds that $\sum_{j=1}^{n} a_{i,j}x_j = \lambda x_i$. But then for the vector $c\vec{x}$ we get that coordinate i in the vector $A(c \cdot \vec{x})$ is $\sum_{j=1}^{n} a_{i,j}(cx_j) = c\sum_{j=1}^{n} a_{i,j}x_j = c \cdot (\lambda x_i) = \lambda(c \cdot x_i)$, so clearly $c \cdot \vec{x}$ is also an eigenvector.

Just explaining briefly that $A(c\vec{x}) = c(A\vec{x}) = c \cdot (\lambda \cdot \vec{x}) = \lambda(c \cdot x_i)$ is also fine.

3b (20 p) Show that if G is a d-regular graph, then its adjacency matrix A_G always has d as an eigenvalue.

Solution: For this and the following problems, an important first observation is that if we identify the vertices $v_1, v_2, \ldots, v_n$ of G with the integers $1, 2, \ldots, n$ (as in Figure 1), then the ith entry of $A_G\vec{x}$ is

$$\left(A_G\vec{x}\right)_i = \sum_{j=1}^{n} a_{i,j}x_j = \sum_{j \in N(i)} x_j \ , \tag{7}$$

where as usual $N(\cdot)$ denotes the set of neighbours of a vertex. This is so since $a_{i,j}$ is 1 when vertices i and j are neighbours and is 0 otherwise.

If we follow the examples in the problem statement and let $\vec{x} = \mathbf{1}$ be the all-ones vector, then we see from (7) that $\left(A_G\mathbf{1}\right)_i = \sum_{j \in N(i)} 1$ will simply count the number of neighbours of vertex i. Since all vertices have d neighbours in a d-regular graph, we have $\left(A_G\mathbf{1}\right)_i = d = d \cdot 1$, and so it follows that $\mathbf{1}$ is an eigenvector with eigenvalue d.

3c (20 p) Show that if G is a d-regular graph, then its adjacency matrix A_G can never have an eigenvalue λ such that $|\lambda| > d$.

Hint: Suppose that there is an eigenvector $\vec{x}$ with eigenvalue λ such that $|\lambda| > d$. Rescale the entries in $\vec{x}$ by some $c \neq 0$ so that the largest entry has value 1 and all other entries have absolute value at most 1. Consider the product $A_G \cdot \vec{x}$, focus on a largest entry in $\vec{x}$, and argue by contradiction.

Solution: Follow the hint and rescale the vector $\vec{x}$, so that $x_i = 1$ for the largest entry i and $|x_j| \leq 1$ for all j. Then we get

$$\left|\left(A_G\vec{x}\right)_i\right| = \left|\sum_{j \in N(i)} x_j\right| \leq \sum_{j \in N(i)} |x_j| \leq \sum_{j \in N(i)} 1 \leq d < |\lambda| \ , \tag{8}$$

and so $|\lambda|$ is just too large to possibly be the absolute value of an eigenvalue of A_G.

3d (30 p) Show that if the d-regular graph G is connected, so that there is a path between any two vertices u and v in $V(G)$, then any eigenvector $\vec{x}$ of the adjacency matrix A_G with eigenvalue d must have all entries equal (i.e., $\vec{x}$ is the all-ones vector or some multiple of the all-ones vector).

Hint: Suppose $\vec{x}$ is an eigenvector of A_G with eigenvalue d in which not all entries are equal. Rescale the entries in $\vec{x}$ by some $c \neq 0$ so that the largest entry has value 1 and all other entries have absolute value at most 1, and consider the product $A_G \cdot \vec{x}$.

Solution: Again we rescale the vector $\vec{x}$ so that the largest entry has value 1 and so that $|x_j| \leq 1$ holds for all j. Suppose that i^* is a coordinate such that $x_{i^*} = 1$ but there is a neighbour j^* of i^* such that $x_{j^*} < 1$. (If there are no such i^* and j^*, then all entries in the vector are the same   note that we are using here the assumption that the graph is connected.) We now have

$$\left(A_G\vec{x}\right)_{i^*} = \sum_{j \in N(i^*)} x_j < \sum_{j \in N(i^*)} 1 = d \ , \tag{9}$$

where we get a strict inequality since $x_j \leq 1$ holds for all j and for the particular neighbour j^* we have strict inequality $x_{j^*} < 1$. This shows that $\vec{x}$ is not an eigenvector of A_G with eigenvalue d.

3e (30 p) Show that if the d-regular graph G is *not* connected, so that there exist two vertices u and v in $V(G)$ with no path between them, then there is in fact is an eigenvector $\vec{x}$ of the adjacency matrix A_G with eigenvalue d in which not all entries are equal.

Hint: Consider the different connected components of G and use them to define interesting vectors for which d is an eigenvalue.

Solution: Fix some connected component C and define a vector $\vec{x}$ such that $x_i = 1$ for $i \in C$ and $x_i = 0$ for $i \notin C$. We know from (7) that $\left(A_G\vec{x}\right)_i = \sum_{j \in N(i)} x_j$. Since for every vertex i it holds that i and $N(i)$ lie in the same connected component, we get for all $i' \notin C$ that $\left(A_G\vec{x}\right)_{i'} = \sum_{j \in N(i')} x_j = \sum_{j \in N(i')} 0 = 0 = d \cdot x_{i'}$. For vertices $i'' \in C$ we get that C must contain all the d neighbours, and so in this case it holds that $\left(A_G\vec{x}\right)_{i''} = \sum_{j \in N(i'')} x_j = \sum_{j \in N(i')} 1 = d = d \cdot x_{i'}$. Hence, $\vec{x}$ is an eigenvector with eigenvalue d.

3f (40 p) We say that an undirected graph $G = (V, E)$ is *bipartite* if there is a bipartition $V = V_1 \dot{\cup} V_2$ (where $\dot{\cup}$ denotes *disjoint union*, so that $V_1 \cup V_2 = V$ but $V_1 \cap V_2 = \emptyset$) such that any edge in G has one endpoint in V_1 and one endpoint in V_2, but there are no edges connecting vertices in V_1 to each other or vertices in V_2 to each other. (Just to give examples for this definition, it is not hard to verify that the graph in Figure 1a is bipartite but that the graph in Figure 1b is not.)

Show that if the d-regular graph G is bipartite and the adjacency matrix A_G has an eigenvector $\vec{x}$ with eigenvalue λ, then $-\lambda$ is also an eigenvalue for A_G.

Hint: Consider the bipartition $V = V_1 \dot{\cup} V_2$ and use it to modify the eigenvector $\vec{x}$ in some interesting way. (This connection between bipartiteness and negated eigenvalues is actually an if and only if—it holds that $-\lambda$ is also an eigenvalue only if G is bipartite—but you definitely do not need to prove this.)

Solution: Suppose that G is a bipartite d-regular graph with bipartition $V = V_1 \dot{\cup} V_2$, and that $\vec{x}$ is an eigenvector with eigenvalue λ. Define the vector $\vec{y}$ by

$$y_i = \begin{cases} x_i & \text{if } i \in V_1, \\ -x_i & \text{if } i \in V_2. \end{cases} \tag{10}$$

We are again going to use that $\left(A_G\vec{y}\right)_i = \sum_{j \in N(i)} y_j$ according to (7).

Consider first a vertex $i \in V_1$. Note that for all its neighbours $j \in N(i)$ it holds that $j \in V_2$ because of bipartiteness, and so $y_j = -x_j$ by our construction of $\vec{y}$ in (10). This means that

$$\left(A_G\vec{y}\right)_i = \sum_{j \in N(i)} y_j = \sum_{j \in N(i)} -x_j = -\lambda x_i = (-\lambda) \cdot y_i \tag{11}$$

(where we used that $\sum_{j \in N(i)} x_j = \lambda x_i$ by assumption, since $\vec{x}$ is an eigenvector with eigenvalue λ).

In exactly the same way, for $i \in V_2$ we have that $y_i = -x_i$ but that the neighbours $j \in N(i)$ have vector entries $y_j = x_i$ since $j \in V_1$. The calculation

$$\left(A_G\vec{y}\right)_i = \sum_{j \in N(i)} y_j = \sum_{j \in N(i)} x_j = \lambda x_i = (-\lambda) \cdot y_i \tag{12}$$

completes the proof that $\vec{y}$ is an eigenvector with eigenvalue $-\lambda$.

Note that Problems 3d and 3e say that a d-regular graph G is connected if and only if the only eigenvectors of the adjacency matrix A_G with eigenvalue d are multiples of the all-ones vector, and Problem 3f says that G is bipartite only if the eigenvalues of A_G are symmetric with respect to 0. We are in fact only scratching the surface here, in that the eigenvalues of A_G can tell us much more about the properties of G. Perhaps the most important connection is that the second largest eigenvalue λ_2 in absolute value is a measure of how well-connected the graph is if the gap between d and $|\lambda_2|$ is large, then G is an *expander graph* in which there are short paths between any two vertices. These and other highly nontrivial facts are further studied in *spectral graph theory*.