PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE
ESCUELA DE INGENIERIA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACION

Complexity Theory - IIC3242
Homework 3
Deadline: Tuesday, 24th of April

1 Queries on the Web [6 points]

Web databases. A *web database* is directed graph where each edge is labelled by a label from some finite alphabet Σ. Formally, a web database over an alphabet Σ can be defined as a tuple $G = (V, E)$, where V is a finite set of nodes and $E \subseteq V \times \Sigma \times V$ is a finite set of edges labelled by a letter from Σ. An example of a web database is given in the following image:

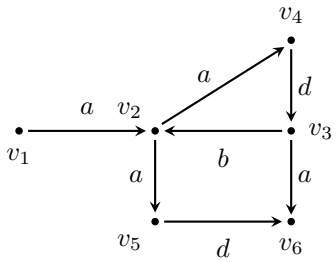

Figure 1: Here the nodes are $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ and the edges are $E = \{(v_1, a, v_2), (v_2, a, v_5), \ldots\}$.

Web queries. In order to explore a web database we will use the language of *web queries* that allows us to see how the nodes of the graph are connected. Formally, a *web query* q is an expression of the form

$$q = x \xrightarrow{\pi} y \wedge e(\pi)$$

where e is a regular expression over Σ. Here x and y are called node variables, and π a path variable. The intended meaning of a web query q is to extract all pairs (x, y) of nodes in G that are connected by a path π such that the word formed by the edge labels along this path belongs to the language of e. We formalize these concepts below.

Semantics of web queries. A path in a web database $G = (V, E)$ is a sequence

$$P = v_1, a_1, v_2, a_2, v_3 \ldots, a_k, v_{k+1},$$

where each v_i is a node in V and $a_i \in \Sigma$ such that $(v_i, a_i, v_{i+1}) \in E$ is an edge of this web database. The label of the path P above is defined as $\lambda(P) = a_1 \cdots a_k$, that is, as the word that is obtained by concatenating the letters on the edges of P. Let $q = x \xrightarrow{\pi} y \wedge e(\pi)$ be a web query. We define the *evaluation of q over the web graph G*, denoted by $\text{EVAL}(q, G)$ as the set of all pairs (v, v') of nodes in G such that there is some path P

starting with v and ending with v' such that $\lambda(P) \in L(e)$, where $L(e)$ denotes the language defined by the regular expression e.

A quick example. Consider the web database G from the above figure. If we define our web query as $q = x \xrightarrow{\pi} y \wedge ad^*(\pi)$ it is easy to see that $\text{EVAL}(q, G) = \{(v_1, v_2), (v_2, v_4), (v_2, v_3), (v_2, v_5), (v_2, v_6), (v_3, v_6)\}$. For example, (v_2, v_6) is in the answer to our query because for $P = v_2 a v_5 d v_6$ we have that $\lambda(P) = ad$ is in the language of the expression ad^*.

Computing the answer. In order to make web queries useful we need to be able to compute them. One way of doing this would be to design an algorithm that, when given q and G as its input, computes the set $\text{EVAL}(q, G)$. Since we are dealing with decision problems we will be interested in the following language (which can then be used to compute the entire answer):

$$\text{WEBCOMP} = \{\langle \Sigma, q, G, (v, v') \rangle \mid (v, v') \in \text{EVAL}(q, G), \text{ where } q \text{ is a query over } \Sigma \text{ and } G \text{ a graph over } \Sigma\}.$$

As a warm-up you are asked to prove the following:

a) [2 point] Show that the language WEBCOMP is NLOGSPACE-complete. A high level description will do. (1.8 points for the upper bound, 0.2 for the lower bound)

More expressive web queries. Web queries allow us to search web databases using paths, but we are often interested in more complicated patterns occurring in the database. For instance, we might want to check if there are two different paths connecting two nodes, if there are three nodes connected into a clique labelled a certain way, or if there are two paths with different labels that are of the same length. For this we use *conjunctive web queries*. Formally, conjunctive web queries are expressions of the form:

$$\varphi = \exists x_1 \ldots \exists x_n \Big(\bigwedge_{i=1 \ldots \ell} \big(u_i \xrightarrow{\pi_i} z_i \wedge e_i(\pi_i)\big) \wedge \bigwedge_{(j,k) \in S} \lambda(\pi_j) = \lambda(\pi_k) \Big)$$

where $Var = \{x_1, \ldots, x_n\}$ are all the node variables appearing in the expression φ (and are pairwise distinct), $u_i, z_i \in Var$, for $i \in \{1, \ldots, \ell\}$, e_i are regular expressions, $\{\pi_1, \ldots, \pi_\ell\}$ are path variables in φ, and $S \subseteq \{1, \ldots, \ell\}^2$ is a set of pairs of numbers in $\{1, \ldots, \ell\}$. Intuitively, conjunctive web queries test whether some pattern defined using (non conjunctive) web queries holds true, and in addition, they allow us to test if some paths carry the same label.

Formally, we will say that the query φ above is true over a web database $G = (V, E)$ if there exists a function $v : \{x_1, \ldots, x_n\} \mapsto V$ and there exist paths $P_1, \ldots P_n$ in G such that:

1. For all $i \in \{1, \ldots \ell\}$, the path P_i starts in $v(u_i)$ and ends in $v(z_i)$
2. For all $i \in \{1, \ldots \ell\}$, $\lambda(P_i) \in L(e_i)$
3. For all $(j, k) \in S$ it holds that $\lambda(P_j) = \lambda(P_k)$.

Some examples. Consider the following conjunctive web query:

$$\varphi = \exists x \exists y_1 \exists y_2 \Big(\big(x \xrightarrow{\pi_1} y_1 \wedge ad^*(\pi_1)\big) \wedge \big(x \xrightarrow{\pi_2} y_2 \wedge a^* d^*(\pi_2)\big) \wedge \big(y_1 \xrightarrow{\pi_3} y_2 \wedge a(\pi_3)\big) \wedge \lambda(\pi_1) = \lambda(\pi_2) \Big).$$

The query asks if there are nodes x, y_1 and y_2, with the following four properties:

1. x is connected with y_1 by a path P_1 labelled by (a word in) ad^*,
2. x is connected with y_2 by a path P_2 labelled by (a word in) $a^* d^*$,
3. y_1 is connected to y_2 by a path P_3 labelled a,
4. And $\lambda(P_1) = \lambda(P_2)$.

When evaluated over the web database G from Figure 1, this query will be true, because for the valuation v that assigns $v(x) = v_2, v(y_1) = v_3$ and $v(y_2) = v_6$ we have paths $P_1 = v_2av_4dv_3$, $P_2 = v_2av_5dv_6$ and $P_3 = v_3av_6$ that satisfy φ.

The evaluation problem. Similarly as for ordinary web queries, here we are interested in the following problem:

$$\text{CONJWEBCOMP} = \{\langle \Sigma, G, \varphi \rangle \mid \text{ the query } \varphi \text{ is true over } G, \text{ where both } q \text{ and } G \text{ are over } \Sigma\}.$$

For the final part of this assignment you are asked to show the following:

b) [4 points] Prove that the problem CONJWEBCOMP is $PSPACE$-hard. Note that you are only asked to show the lower bound.

Hint: A good approach is to do a reduction from the problem of the intersection of regular expression which is known to be PSPACE-complete [1]. Given some alphabet Σ, the intersection of regular expressions $\bigcap\texttt{REG}$ is defines as follows: given a number m, and a sequence $R_1, \ldots, R_m$ of regular expressions, determine whether there is a word w over Σ such that $w \in L(R_1) \cap L(R_2) \cap \cdots \cap L(R_m)$. Formally, this problem is defined as follows:

$$\bigcap\texttt{REG} = \{\langle \Sigma, m, R_1, \ldots, R_m \rangle \mid m \text{ is a number, } R_1, \ldots, R_m \text{ are regular expressions over } \Sigma, \text{ and } L(R_1) \cap \cdots \cap L(R_m) \neq \emptyset\}$$

Note that here m is part of the input. It is not difficult to see that for each fixed m the problem becomes solvable in linear time.

Remark: In fact $\bigcap\texttt{REG}$ is PSPACE-complete for a fixed alphabet Σ, which makes all of our bounds hold even when Σ is fixed and not part of the input.

References

[1] M. R. Garey, David S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.*, W. H. Freeman 1979, ISBN 0-7167-1044-7