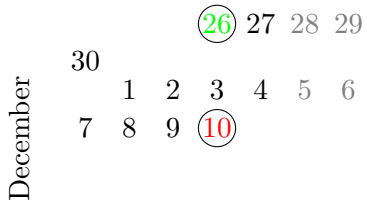
Tiling Problems and Unary Regular Expressions

Home assignment to hand in before or on December 10, 2015.



Electronic versions (PDF only) can be sent by email to ⟨schmitz@lsv.ens-cachan.fr⟩, paper versions should be handed in on the 10th or put in my mailbox at LSV, ENS Cachan. **No delays**. The numbers in the margins next to exercises are indications of time and difficulty, not necessarily of the points you might earn answering them.

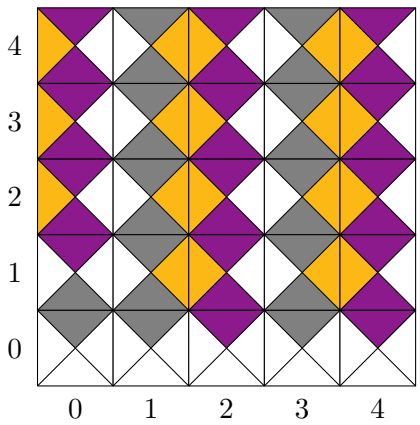
In this home assignment, we are interested in *tiling problems* as a means to obtain NP-complete problems with very simple descriptions. In a second part, we apply these results to prove the coNP-completeness of the universality problem for regular expressions over a *unary alphabet*, i.e. an alphabet of cardinal 1.

1 Square Tiling

The objective in a tiling problem is to cover a grid using a finite set of square tiles. Each tile can be reused as many times as we want. The four edges of each tile are coloured, so that adjacent tiles must have matching colours; importantly, tiles cannot be rotated.

Example 1. For instance, assume we are given the following set of tiles:

We can use this set to tile the 5×5 square grid as follows:



Note that there is however no tiling of the 3×3 square grid using the subset of tiles $\{t_0, t_1, t_2, t_3\}$: t_1 and t_2 can then only be used on the topmost row, and t_3 only on the rightmost column, and we cannot fill the 2×2 bottom-left square using only t_0.

Definition 1 (Tile Set and Tiling)**.** Let $D \stackrel{\text{def}}{=} \{\triangleright, \triangle, \triangleleft, \triangledown\}$. Formally, a *tile set* is represented as a tuple $\mathcal{T} = \langle C, T, t_0 \rangle$ where C is a finite set of colours, $T \subseteq C^D$ is a set of tiles (described by their left, bottom, right, and top colours), and $t_0 \in T$ is a distinguished start tile. For instance, the tile set of Example 1 can be defined by $C \stackrel{\text{def}}{=} \{\bigcirc, \bullet, \bullet, \bullet\}$ and $T \stackrel{\text{def}}{=} \{t_0, t_1, t_2, t_3, t_4\}$ where

$$t_0 \stackrel{\text{def}}{=} \langle \bigcirc, \bigcirc, \bigcirc, \bullet \rangle , \qquad t_1 \stackrel{\text{def}}{=} \langle \bigcirc, \bigcirc, \bigcirc, \bullet \rangle , \qquad t_2 \stackrel{\text{def}}{=} \langle \bigcirc, \bullet, \bigcirc, \bullet \rangle ,$$
$$t_3 \stackrel{\text{def}}{=} \langle \bigcirc, \bullet, \bullet, \bullet \rangle , \qquad t_4 \stackrel{\text{def}}{=} \langle \bullet, \bullet, \bigcirc, \bullet \rangle .$$

Given a positive integer $N > 0$, a *tiling* of the $N \times N$ square grid by $\mathcal{T}$ is an assignment $\theta : N \times N \to T$ such that the bottom-left square is tiled by t_0:

$$\theta(0, 0) = t_0 ,$$

two horizontally adjacent tiles must have matching right and left colours:

$$\forall 0 \leq i < N - 1 . \forall 0 \leq j < N . \theta(i, j)[\triangleleft] = \theta(i + 1, j)[\triangleright] ,$$

and two vertically adjacent tiles must have matching top and bottom colours:

$$\forall 0 \leq i < N . \forall 0 \leq j < N - 1 . \theta(i, j)[\triangledown] = \theta(i, j + 1)[\triangle] .$$

[3] **Exercise 1** (Time-Bounded Computation Problem)**.** Show that the following decision problem is NP-hard by a reduction from acceptance in polynomial time nondeterministic Turing machines:

input: A one-tape nondeterministic Turing machine $\mathcal{M}$ and a time bound $B > 0$ encoded *in unary*.

question: Does $\mathcal{M}$ have a computation of length B on the empty input ε?

Exercise 2 (Square Tiling Problem)**.** Let us show that the following decision problem is NP-complete:

input: A tile set $\mathcal{T}$ and a grid size $N > 0$ encoded *in unary*.

question: Does there exist a tiling of the $N \times N$ square grid by $\mathcal{T}$?

[1] 1. Show that the square tiling problem is in NP.

[3] 2. Reduce the time-bounded computation problem to the square tiling problem, thereby showing its NP-hardness. The idea of the reduction is for the top colours of the jth row in a tiling of the $N \times N$ square to represent the jth configuration of a Turing machine.

2 Universality of Unary Regular Expressions

As seen during the lectures, the universality problem for regular expressions, i.e. whether $L(E) = \Sigma^*$ when given a regular expression E over some alphabet Σ as input, is PSPACE-complete. Here we are however interested in a particular case: that of a *unary alphabet* $\Sigma \stackrel{\text{def}}{=} \{1\}$.

In this case, a *unary regular expression* is a term E defined by the abstract syntax

$$E ::= \varnothing \mid 1 \mid E + E \mid E \cdot E \mid E^* \ .$$

Because the language of such an expression E is a set of words each of form 1^n for some n, it is more convenient to define its semantics $[\![E]\!] \subseteq \mathbb{N}$ as the set of lengths n of words in its language: $[\![E]\!] \stackrel{\text{def}}{=} \{n \mid 1^n \in L(E)\}$. This can also be defined inductively on E by:

$$[\![\varnothing]\!] \stackrel{\text{def}}{=} \varnothing \ , \qquad [\![1]\!] \stackrel{\text{def}}{=} \{1\}$$
$$[\![E + F]\!] \stackrel{\text{def}}{=} [\![E]\!] \cup [\![F]\!] \ , \qquad [\![E \cdot F]\!] \stackrel{\text{def}}{=} \{m + n \mid m \in [\![E]\!] \wedge n \in [\![F]\!]\} \ ,$$
$$[\![E^*]\!] \stackrel{\text{def}}{=} \{0\} \cup \{\lambda n \mid \lambda \in \mathbb{N} \wedge n \in [\![E]\!]\} \ .$$

The *universality problem* for unary regular expressions is then:

input: A unary regular expression E.

question: Does $[\![E]\!] = \mathbb{N}$?

In the following, we prove the coNP-completeness of this problem. To facilitate matters, we actually prove the NP-completeness of the more intuitive *non universality problem*, which asks instead whether $[\![E]\!] \neq \mathbb{N}$.

Exercise 3 (NP-Easiness)**.** In order to prove that the non universality problem is in NP, we shall use the fact that any regular expression E can be converted into a non deterministic finite automaton $\mathcal{A} = \langle Q, \Sigma, \delta, I, F \rangle$ with $|Q| = |E| + 1$ states and $L(\mathcal{A}) = L(E)$.

[2] 1. Show that, if $[\![E]\!] \neq \mathbb{N}$, then there exists $n \leq 2^{|E|}$ such that $n \notin [\![E]\!]$.

2. Consider the transition matrix of $\mathcal{A}$: it is a square matrix A of dimension $|Q|$ with $A[q, q'] \stackrel{\text{def}}{=} 1$ if $(q, 1, q') \in \delta$ and $A[q, q'] \stackrel{\text{def}}{=} 0$ otherwise. We similarly view I and F as vectors in $\{0, 1\}^{|Q|}$.

[2] (a) Show that $n \in [\![E]\!]$ if and only if $I \cdot A^n \cdot F^{\mathrm{T}} > 0$.

[2] (b) Show that, for any $0 \leq i, j < |Q|$, whether $A^{2^k}[i, j] > 0$ can be computed in time polynomial in $|Q|$ and k.

[1] 3. Conclude.

Exercise 4 (NP-Hardness)**.** In order to show the NP-hardness of the non universality problem, we are going to reduce from the square tiling problem. Consider a tile set $\mathcal{T}$ with tiles $T = \{t_0, \ldots, t_{n-1}\}$ and a grid size $N > 0$.

By the Prime Number Theorem, we can find distinct prime numbers $p_{i,j} \geq n$ for $0 \leq i, j < N$ such that $\max_{i,j} p_{i,j}$ is polynomial in N and n. The idea of the reduction is then to represent a tiling θ of $N \times N$ by $\mathcal{T}$ as an integer $z \in \mathbb{N}$ such that $z \mod p_{i,j} = k$ if and only if $\theta(i, j) = t_k$. We are going to build a unary regular expression E such that, if $z \notin [\![E]\!]$, then z describes a tiling.

[1] 1. Provide a regular expression E_0 such that $z \in [\![E_0]\!]$ if and only if $z \mod p_{0,0} \neq 0$.

[1] 2. Provide a regular expression E_T such that $z \in [\![E_T]\!]$ if and only if there exist $0 \leq i, j < N$ such that $z \mod p_{i,j} \notin \{0, \ldots, n-1\}$.

[3] 3. Provide a regular expression E_H of such that $z \in [\![E_H]\!]$ if and only if there exist $0 \leq i < N - 1$ and $0 \leq j < N$ such that $z \mod p_{i,j} = k$, $z \mod p_{i+1,j} = \ell$, and $t_k[\triangleleft] \neq t_\ell[\triangleright]$.

   Provide similarly a regular expression E_V such that $z \in [\![E_V]\!]$ if and only if there exist $0 \leq i < N$ and $0 \leq j < N - 1$ such that $z \mod p_{i,j} = k$, $z \mod p_{i,j+1} = \ell$, and $t_k[\triangledown] \neq t_\ell[\triangle]$.

[1] 4. Show that these expressions can be computed in polynomial time and conclude.