# Devoir maison 1

Étienne Toussaint

10 décembre 2015

## 1 Disclaimer

For the whole assignment i confused the order in the tilling, in the definition, you wrote :
$t =< left, bot, right, top >$

I did a mistake and used the convention :
$t =< right, top, left, bot >$

I also assumed that having a compution of length $B$ meant having a terminating run of length $B$ whether is was accepting or rejecting one.

## 2 Exercice 1

The acceptance problem in polynomial time nondeterministicTuring machines can be formalize this way.
Input :
An NTM Machine $M$, an integer $n$ and a polynomial function $p$.
Question :
Does there exist a run on $M$ accepting $n$ in less or equal than $p(n)$ steps.

### 2.1 The machine and B

Suppose $M =< Q, \delta, Q_0, \sum >$.
We will write *accept* and *reject* the special states of $M$, $ACCEPT$ and $REJECT$ the special states of $M_1$

Let $M_1 =< \{Q \cup \{accept, reject\} \times [\![0; p(n)]\!]\} \cup \{(reject, p(n)+1)\}, \delta', \{(q_0, 0) | q_0 \in Q_0\}, \sum >$

$\forall (q, i) \in \{Q \times [\![0; p(n)-1]\!]\}$ :
$\delta'((q,i),u) = \{((p, i+1), v, arrow) | \exists (p, v, arrow) \in \delta(q, u)\}$

$\forall (q, i) \in \{Q \times \{p(n)\}\}$
$\delta'((q,i),u) = ((reject, p(n)+1), u, \downarrow)$

$\forall i \in [\![0; p(n)-1]\!]$
$\delta'((accept, i), u) = ((accept, i+1), u, \downarrow)$

$\forall i \in [\![0; p(n)]\!]$
$\delta'((reject, i), u) = ((reject, i+1), u, \downarrow)$

$\forall u$
$\delta((accept, p(n)), u) = (ACCEPT, u, \downarrow)$

$\forall u$
$\delta((reject, p(n)+1, u) = (REJECT, u, \downarrow)$

Let consider a machine $M_2$ which ignore is input, copy $n$ on his band and then execute $M_1$.

Intuitivly we built a non deterministic machine $M_2$ which will simulate $< M, n >$ but will at best stop on an accepting exectution after $B$ steps and to avoid any rejected computation of length $B$ it will REJECT after $B+1$ steps where $B = p(n) + l(n) + 1$ if we consider the copy of $n$ on the band take $l(n)$ steps.

So $B = p(n) + l(n) + 1$.

## 2.2 If and only if

Let's show that if $M_2$ computes $\epsilon$ in $B$ steps then there exists a run on $M$ accepting $n$ in less or equal than $p(n)$ steps.

Let's consider a run $r$ such that $M_2$ computes $\epsilon$ in $B$ steps.

First let's show $r$ reachs $ACCEPT$.
By absurd, let 's suppose $r$ reachs $REJECT$, then by definition of the machine it had to reach $(reject, p(n)+1)$ on the previous step. As each step of the machine increment the second argument, the run has done $p(n)+2$ steps from $(q_0, 0)$ to $REJECT$. It also has done the $l(n)$ steps required to copy the data. So it has $p(n)+2+l(n)$ steps and $p(n)+2+l(n) \neq B$

As $r$ reachs $ACCEPT$ state it had to reached $(accept, p(n))$ before. Let's call $i$ the minimal value where $\exists t | state(r(i)) = (accept, t)$ Considere the restrition of $r$ call $r' = r([\![l(n); i]\!])$ First such a restriction exists because $i > l(n)$ by construction of the machine. Then $r'$ is an accepting run for $< M, n >$ and $|r'| \leqslant p(n)$

Clearly $|r'| \leqslant p(n)$ as $|r'| \leqslant B - l(n) - 1$.
Moreover $\forall j \leqslant |r'|$ if $r(j) = (w', (q, j), aw)$ and $r(j+1) = (w'b, (p, j+1), w)$ then there exists $((p, j+1), b, \rightarrow) \in \delta'((q, j), a)$. Hence by construction of $\delta'((q, j), a)$ there exists $(p, b, \rightarrow) \in \delta(q, a)$
Same can be said for $\{\leftarrow, \downarrow\}$.

Let's show that if there exists a run $r$ on $M$ that accept $n$ in less or equal than $p(n)$ steps then there exists a run that compute $\epsilon$ on $M_2$ in $B$ steps.

Let's consider $r'$ a run on $M_2$ the first $l(n)$ steps being trivial let's ignore them.

Let's build $r'$ such that $\forall j \leqslant |r|$ if $r(j) = (w, p, v)$ then $r'(j + l(n)) = (w, (p, j), v)$
Such construction is possible because if $r(0) = (\epsilon, q_0, v)$ then $q_0 \in Q_0$ so $(q_0, 0) \in Q_0 \times \{0\}$

hence $r'(l(n)) = (\epsilon, (q_0, 0), v)$ is a valid state of the machine $M_2$.
If $r(j) = (w, q, av)$ and $r(j+1) = (w, p, bv)$ and $r'(j + l(n)) = (w, (q, j), av)$ is a valid state of the machine $M_2$ Then $r'(j + l(n) + 1) = (w, (p, j+1), bv)$ is also a valid state. Indeed if $r(j+1) = (w, p, bv)$ and $r'(j + l(n)) = (w, (q, j), av)$ then $(p, b, \downarrow) \in \delta(q, a)$ so by construction of $\delta'$, $((p, j+1), b, \downarrow) \in \delta((q, j), a)$, hence $r'(j + l(n) + 1) = (w, (p, j+1), bv)$ is a valid state. (Same can be said for $\{\leftarrow, \rightarrow\}$).
Moreover as $r(|r| - 1) = (w, accepte, v)$, $r'(l(n) + |r| - 1) = (w, (accept, |r| - 1), v)$

$\forall j \geqslant |r| + l(n) \wedge æ \leqslant p(n) + l(n)$
$r'(j) = (w, (accept, j - l(n)), v)$ easy by construction of $M_2$ which only incremente when on *accept*
Finally as $r'(p(n) + l(n)) = (w, (accept, p(n)), v)$, $r'(p(n) + l(n) + 1) = (w, ACCEPT, v)$, hence $M_2$ computes $\epsilon$ in $B = p(n) + l(n) + 1$ steps on $r'$.

### 2.3 Remarks

As $p(n)$ is finite there is only a finite set of states and transitions in $M_2$ moreover $l(n)$ only depend on $n$ and can be easily calculate if we build the machine which copy input. Moreover the construction of $M_2$ and $B$ knowing $M$, $n$ and $p$ is clearly in polynomial time as we have to build $(p(n) + 2) * Q$ states and $p(n)$ transition for each transition of $M$.

## 3 Exercice 2

### 3.1 Question 1

In order to show that the tilling problem is in NP, we can show that there exists a polynomial algorithm that check if the given assignement $\Theta$ is a tillement.

Input : A set of Tile $T$ , a grid size $N > 0$ encoded in unary, and $\Theta = N \times N \rightarrow T$ an assignment.

Question : Is $\Theta$ a valid tilling by $T$ for the grid $N \times N$

```
checking_till(T,N,Theta) {
  for(i = 0; i<N-1 ; i++){
    for(j = 0; j<N-1; j ++){
      if(Theta(i,j) not in T) return false;
      else if (Theta(i,j)[left] <> Theta(i,j)[right]) return false;
      else if (Theta(i,j)[bot] <> Theta(i,j+1)[top]) return false;
    }
  }
  return true;
}
```

checking_till does at most $3 * (N-1)^2$ operations then it is polynomial.
And obtaining a tile in Theta can be done easily in linear time among $n$
Hence the tilling probleme is in NP.

### 3.2 Question 2

Input : A one-tape nondeterministic Turing machine $M$ and a time bound $B > 0$ encoded in unary.
Question : Does $M$ have a computation of length $B$ on the empty input $\epsilon$ ?

Input : A tile set $T$ and a grid size $N > 0$ encoded in unary.
Question : Does there exist a tiling of the $N \times N$ square grid by $T$ ?

We tweak the machine $M$ such that we create a machine $M'$ which emulate $B-1$ steps of $M$ and then go to a special $q_{loose}$ state if the next state of the $M$ computation wasn't *accept* or *reject*, on $q_{loose}$, $M'$ loops indefinetly. This construction is easily buildable in polynamial time in the way as the one in the previous question (As we have to count steps). Such machine verify that $M'$ admit a computation of lenght $B$ on $\epsilon$ if and only if $M$ admit a computaton of lenght $B$ on $\epsilon$ Moreover every computation that doesn't end in $B$ steps, land on $q_looose$ after $B$ steps. We will now assmume that the Machine considered is like $M'$.

Let $M =< Q, \delta, q_0, \sum, \{b, \$\} >$

We build the set of tile $T$.
First $N = B + 1$, each row represent a state of the machine. ie. $B$ charactere from the alphabet and a state. As the computation of $M$ is bouded by $B$, their will be at most $B$ different character from $b$.

For each $a \in \sum \cup \{b, \$\}$ we add to $T$
$< f, a, f, a >$

The intuition behind $f$ is the creation of a special color to force any element of the state to remain the same between two row (exept those concern by the transtion).

For each $q \in Q, a \in \sum \cup \{b, \$\}$ if $(p, c, \downarrow) \in \delta(q, a)$ and $p \neq q_{loose}$ then we add to $T$
$< q_p, p, f, q >$ and (head tile)
$< f, c, q_p, a >$ (right tile)

The intuition here is to create a Tile which force her left side to remain unchange thanks to $f$, make the state move from $q$ to $p$ and warn the right tile representing $a$ that a transition from $q$ to $p$ as occured, then we add an another tile which reprensent the fact that $a$ can become $c$ when a $q$ to $p$ transition occured. If $q$ can not become $p$ when reading $a$ then the tilling will stop there as there won't be any available tile to match $q_p$ and $a$.

For each $q \in Q, a \in \sum \cup \{b, \$\}$ if $(p, c, \rightarrow) \in \delta(q, a)$ and $p \neq q_{loose}$ then we add to $T$
$< q_c^{\rightarrow}, c, f, q >$ and (head tile)
$< f, p, q_c^{\rightarrow}, a >$ (right tile)

The intuition here is to create a Tile which will make the head move on and write a $c$ at her place. This also warn her right tile that it moves from $q$ to $c$. It also add a tile representing the new head on the state $p$. If $q$ can not become $c$ when reading $a$ then the tilling will stop as $< f, p, q_c^{\rightarrow}, a >$ wont exist.

For each $q \in Q, a \in \sum \cup \{b, \$\}$ if $(p, c, \leftarrow) \in \delta(q, a)$ and $p \neq q_{loose}$ then we add to $T$
For each $d \in \sum \cup \{b, \$\}$
$< q_c^a, d, q_c^a \times d, q >$ and (head tiles)
$< q_c^a \times d, p, f, d >$ and (left tiles)

$< f, c, q_c^a, a >$ (right tile)

Here it's a bit tricky has we have to imagine both what the left and right tile are. In order to be able to do so we have to create many colors. If there isn't a transition $(p, c, \leftarrow) \in \delta(q, a)$ then $< q_c^a \times d, p, f, d >$ won't exists. If $a$ wasn't the right tile then we can't apply $< f, c, q_c^a, a >$, if $d$ wasn't the left tile then we can't apply $< q_c^a \times d, p, f, d >$.

We already create one tile for each element of $\sum \cup \{b, \$\}$, 2 tiles for each $\downarrow$ transition, 2 tiles for each $\rightarrow$ transition, and $2 * |\sum \cup \{b, \$\}| + 1$ tiles for each $\leftarrow$ transition. Moreover we also have at most create : $|\sum \cup \{b, \$\}| + |Q| * |Q| + |\sum \cup \{b, \$\}| * |Q| + |\sum \cup \{b, \$\}|^2 * |Q| + |\sum \cup \{b, \$\}|^3 * |Q|$ different colors.
Both of those values are polynomial.

We now have have to constrain the first row thanks to $t_0$ to be $\epsilon q_0 \$ b....b$ in order so we introduce a special color $i$ only use on the first row and special tiles : $t_0 =< \$, q_0, i, i >$ , $< i, \$, \$, i >$ ,and $< i, b, i, i >$. Here we have a bit of loose in generality because i can't find a way to deal with a set of possible initial states. however we can always add multiple $\epsilon$ transition (ie. which doesn't change the tape) from $q_0$ to the other member of the initial set.

As a formal proof will take too long, i will try to convince you.

If $M$ have a compution of length $B$ on $\epsilon$. Consider such computation $r$, as we can add as many column on the first row of the tilling we add $B - 1$ tile of $b$, then we can respect the computation $r$ to choose the tilling. It is always possible as we build the set of tile $T$ in order to. We can do so $B$ times resulting in $B + 1 \times B + 1$ tilling. It 's important to notice that we won't need more $b$ than $B$ cause as the computation is bound by $B$ there is at most $B \rightarrow$ transition.

Considere there is no computation of lenght $B$ on $\epsilon$. Suppose there exists a $B+1 \times B+1$ tilling with the $T$, Then inductivly we can prove that each row of the tilling can represent a state of the machine. (Indeed it's clear for the first one as $t_0$ and $i$ insure it, and as to put a till above the head we have to respect condition on $\delta$ it also true for the row above) Then if we are able to tile $B$ row that mean that is able to perform $B - 1$ steps. Moreover, as we didn't add any tile for $q_{loose}$ if we are able to tile the last row then, the last row contain a tile representing *accept* or *reject*. (If we didn't tweak the machine $M$ beforehand any computation longer then $B$ which induce a valid tilling) Hence the run represent by the row is a valid run for $M$ of lenght $B$ which computs $\epsilon$ (as the first row has to be of form $q_0 \$ bbb...bb$).

The building of $T$ can be performed in polynomial has, there is a polynomial number of tiles in $T$, and $N = B + 1$ is compute linearly.

As a conclusion the tilling problem is NP-Hard, hence it is NP-Complete.

# 4 Exercice 3

## 4.1 Question 1

From an NFA $A$ with $|Q|$ states we can construct a DFA which recognize the same language with $2^{|Q|}$ states. Then if $[\![E]\!] \neq \mathbb{N}$ there exists a path from the initial state $q_0$ to

a non terminal state $p$. As there is $2^{|Q|}$ states, such a path (not a walk) is $\leq 2^{|Q|}$ hence there exists $n \leq 2^{|Q|}$ such that $n \notin [\![E]\!]$

### 4.2 Question 2

(a)

We want to show that $A^n[q, q'] \geq 1$ iff $q'$ is reachable from $q$ in $n$ steps.

Inductivly :
By definition of $A$, $A[q, q'] \geq 1$ iff $q'$ is reachable from $q$ in 1 step.

Suppose that $A^n[q, q'] \geq 1$ iff $q'$ is reachable from $q$ in $n$ steps.
Then $A^{n+1}[q, q'] = \sum_{p \in Q} A^n[q, p] * A[p, q']$
Then $A^{n+1}[q, q'] \geq 1$ iff $\exists p \in Q$ such that $A^n[q, p] \geq 1 \wedge A[p, q'] \leq 1$
Hence $A^{n+1}[q, q'] \geq 1$ iff there exists a state $p$ such there exist a path in $n$ step from $q$ to $p$ and a path in 1 step from $p$ to $q'$.
Hence $A^{n+1}[q, q'] \geq 1$ iff there exists a path from $q$ to $q'$ in $n + 1$ steps.

So immediatly we have that $(I.A^n)[q] \geq 1$ iff $\exists p \in I$ and there exists $a$ path from $p$ to $q$ in $n$ steps Finaly we have that $(I.A^n.F^T) \geq 1$ iff $\exists p \in I$ and $q \in F$ such there exists a path from $p$ to $q$ in $n$ steps. Hence $(I.A^n.F^T) \geq 1$ iff $n \in [\![E]\!]$

(b)

Instead of determining wether $A^{2^k}[i, j] > 0$ i will show that compute $A^{2^k}$ in polynomial time. Consider the algorithm ;

```
exp2(A,k) {
  A2 = A * A;
  A2K = A2;
  for(i = 1; i<k ; i++)
    A2K = A2K * A2;
  return A2K;
}
```

Then computing $A2$ cost $|Q|^3$ and each step of the iteration cost $|Q|^3$, hence computing $A2K$ cost $k * |Q|^3$. Checking wheter $A^{2^k}[i, j] > 0$ is just a matter of accesing data which can be done linearly. So determining whether $A^{2^k}[i, j] > 0$ can be in polynomial time on $k$ and $|Q|$

### 4.3 Question 3

We are able to compute $A^{2^k}$ with $k \leq |Q|$ in polynomial time among $|Q|$ as seen in question (2.$b$) and because $k \leq |Q|$. We are also able to compute $A^{2^k+1}$ with $k < |Q|$ in polynomial time among $|Q|$ by computing $A^{2^k}$ and applying the matricial produt which is polynomial too in order to obtain $A^{2^k+1}$.
As applying (2.$a$) is polynomial in $|Q|$ too as it is matricial product, the checking of any work $n \leq 2^{|Q|}$ can be done in polynomial time among $|Q|$.

As learn from question (1) if there is a solution there is a solution $\leq 2^{|Q|}$.
Hence we just have to "guess" the right $n \leq 2|Q|$.
Hence the problem is NP-Easy

# 5 Exercice 4

## 5.1 Question 1

$$E_0 = (1^{p_{0,0}})^*.(\sum_{i=1}^{p_{0,0}-1} 1^i) + \epsilon$$

## 5.2 Question 2

$$E_T = \sum_{i=0}^{N}\sum_{j=0}^{N}\left[(1^{p_{i,j}})^*.(\sum_{t=n}^{p_{i,j}-1} 1^t)\right]$$

We will take care during the construction of the rare case where exists $p_{i,j} = n$

## 5.3 Question 3

If $z = ap + k$ and $z = bq + l$ where $q$ and $p$ are prime number.
Then $ap - bq = l - k$
As $p$ and $q$ are prime $\exists u, v \in \mathbb{N}$ such that $up - vq = 1$.
Then $a = u * (l - k)$ and $b = v * (l - k)$.

Applying this we obtain that.

$$E_H = \sum_{i=0}^{N-1}\sum_{j=0}^{N}\left[\sum_{l>k|t_k[right]\neq t_l[left]}\left((1^{p_{i,j}\times u_0\times(l-k)})^*.1^k\right) + \sum_{k>l|t_k[right]\neq t_l[left]}\left((1^{p_{i+1,j}\times v_0\times(k-l)})^*.1^l\right)\right]$$

where $u_0 = min\{u|up_{i,j} - vp_{i+1,j} = 0\}$ and $v_0 = min\{v|up_{i,j} - vp_{i+1,j} = 0\}$

Indeed by construction of $u_0$ :
$(1^{p_{i,j}\times u_0\times(l-k)})^*.1^k mod[p_{i,j}] = k)$ and
$(1^{p_{i,j}\times u_0\times(l-k)})^*.1^k mod[p_{i+1,j}] = l)$

Same way we obtain :

$$E_V = \sum_{i=0}^{N}\sum_{j=0}^{N-1}\left[\sum_{l>k|t_k[top]\neq t_l[bot]}\left((1^{p_{i,j}\times u_0\times(l-k)})^*.1^k\right) + \sum_{k>l|t_k[top]\neq t_l[bot]}\left((1^{p_{i,j+1}\times v_0\times(k-l)})^*.1^l\right)\right]$$

where $u_0 = min\{u|up_{i,j} - vp_{i,j+1} = 0\}$ and $v_0 = min\{v|up_{i,j} - vp_{i+1,j} = 0\}$

## 5.4 Question 4

The construction of $E_0$ is linear among $n$ if we consider we know as many big enough prime number as we need and that they only depend on $n$.

The construction of $E_T$ is polynomial among $N$ as we have to "sum" over $N$.
ie. $\Theta(N^2 * n)$

Same as the knowing of prime number we can assume that we already know $u_0$ and $v_0$ for each couples $(p_{i,j}, p_{i+1,j})$ and $(p_{i,j}, p_{i,j+1})$ Then we have to "sum" over $N$ and check for all $k, l$ whether if $t_k[top] \neq t_l[bot]$ the worst case we be that there is no matching.
ie. $\Theta(N^2 * n^2)$

Hence the construction of each expression can be computed in polynomial time.

Moreover we clearly have the fact that $T$ admit a tilling of size $N$ iff
$\exists z \notin E_0 + E_T + E_H + E_V$

$\Longleftarrow$ is immediate by the construction of a $\Theta$
Such that $\Theta(i, j) = t_k$ iff $zmod[p_{i,j}] = k$ where $z \notin E_0 + E_T + E_H + E_V$

$\Longrightarrow$ suppose there is a tilling then there exist $z$
Such that $zmod[p_{0,0}] = 0$ as $\Theta(0, 0) = t_0$
$\forall i, j,\ zmod[p_{i,j}] < n$ as $\Theta(i, j) \in T$
$\forall i, j,\ zmod[p_{i,j}] = k \wedge zmod[p_{i+1,j}] = l \rightarrow t_k[right] = t_l[left]$ as $\Theta$ is a valid tilling
$\forall i, j,\ zmod[p_{i,j}] = k \wedge zmod[p_{i,j+1}] = l \rightarrow t_k[right] = t_l[left]$ as $\Theta$ is a valid tilling

Hence $z \notin E_0 + E_T + E_H + E_V$