Apprentissage d'un langage régulier avec L^*

1- Représentation des automates

1. 1. (S_1, T_1) est **correcte** et **complète**. Tout proposition sur tout les elements d'un ensemble vide est vraie.

   2. (S_2, T_2) est **correcte**, en effet, a et ε ne sont pas T_2-équivalents ($b \in \mathcal{L}$, ab $\notin \mathcal{L}$).Mais, elle **n'est pas complète** car ab (forme ua, u = "a" $\in S_2$, a= "b"$\in \Sigma$), n'est T équivalent avec aucun mot $v \in S_2$:

   $$v = \varepsilon, b \in T_2 : \text{abb} \notin \mathcal{L}, b \in \mathcal{L}$$

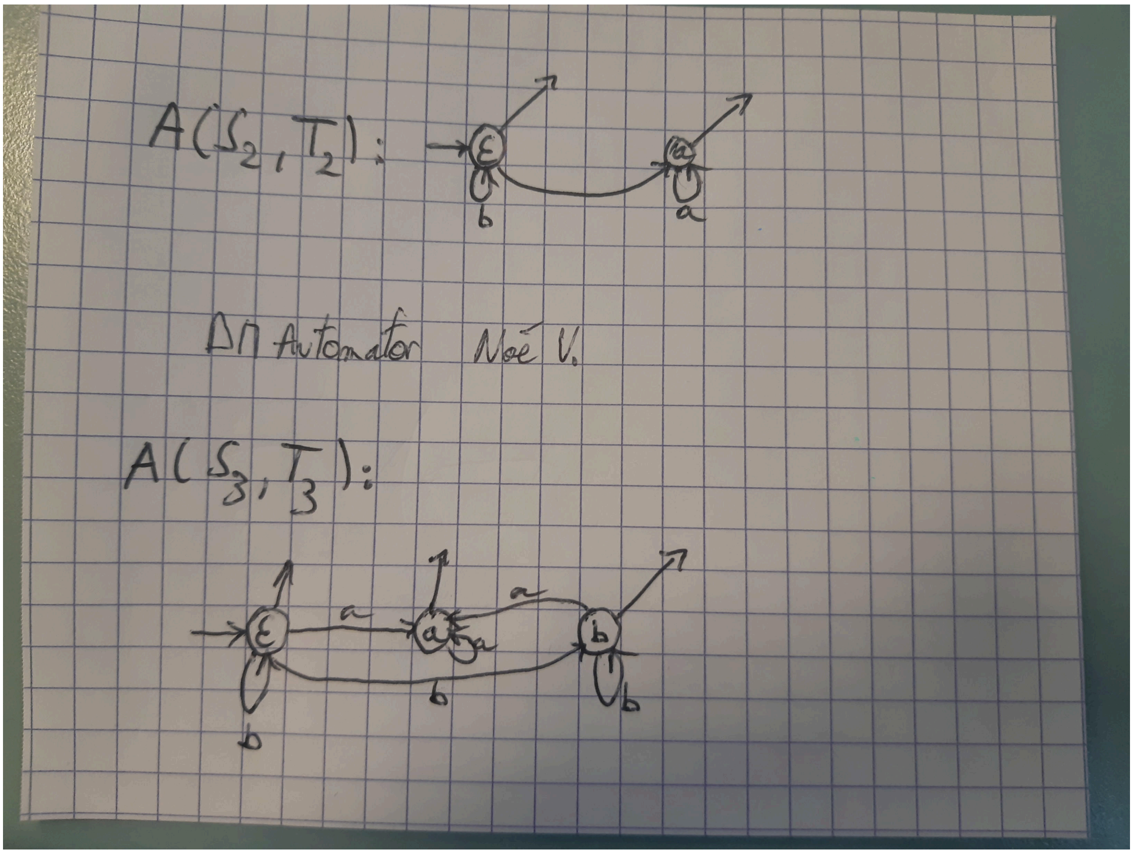
   $$v = a, \varepsilon \in T_2 : \text{ab} \notin \mathcal{L}, a \in \mathcal{L}$$

   3. (S_3, T_3) n'est pas correcte car "b" et "ε" sont T_3-équivalents, en effet:
      - pour $w = b \in T_3$, $\varepsilon.b \in \mathcal{L}, b.b \in \mathcal{L}$
      - pour $w = \varepsilon \in T_3$, $\varepsilon \in \mathcal{L}, b \in \mathcal{L}$

      Cependant elle n'est pas complète il n'existe aucun mot de S qui soit T_3-équivalent avec ab. (Même preuve que précedemment à laquelle on ajoute le cas $v = b, \varepsilon \in T_3$, ab $\notin \mathcal{L}, b \in \mathcal{L}$).

2. cf automator.ml

3. 1. $A(S_1, T_1)$ est non représenté puisque vide (aucun état).

   2. 

4. On peut construire $A(S, T)$, à l'aide d'appels à `oracle` en deux étapes:
   - D'abord pour déterminer les états finaux.
   - Ensuite pour construire les transitions à l'aide de la regle: $u \xrightarrow{a} v$ si ua et v sont T-equivalents. En effet, on peut utiliser `oracle` pour déterminer cette équivalence (cf implémentation de make_t_equivalence).

   Pour connaitre les états finaux on fait $|S|$ appels à `oracle`.

   Soit f une fonction qui à un couple de mots donne vrai s'ils sont T-équivalents. Pour chaque couple ordonné (u,v) de sommets, il est nécessaire de faire $|\Sigma|$ appels f.

   Or cette fonction f fait $2 * |T|$ appels à `oracle` (selon l'implémentation proposée). Ainsi, la construction nécéssite $2 * |S|^2 * |\Sigma| * |T|$ appels à `oracle` pour construire les transitions.

   Ainsi pour construire $A(S, T)$ il est nécessaire de faire : $\boldsymbol{2 * |S|^2 * |\Sigma| * |T| + |S|}$ **appels à oracle.**

   Définissons l'ensemble des questions posée à l'oracle Q sous la forme d'un ensemble de mots.

$$Q = Q_S \cup Q_T$$

   où $Q_S = S$ correspond à l'ensemble des sommets dont on vérifie s'ils sont terminaux et Q_T correspond à l'ensmble des questions posée pour construire les transitions.

$$Q_T = \{w \mid w \in (S \cup S.\Sigma).T\}$$

   d'où

$$|Q| = |S| + |(S \cup S.\Sigma).T| - |S \cap (S \cup S.\Sigma).T|$$

   or $S \cap (S \cup S.\Sigma).T = S$ car $\varepsilon \in T$ (si T non vide).

$$|Q| = |S| + |(S \cup S.\Sigma).T| - |S| = |(S \cup S.\Sigma).T|$$

   Si T est vide alors $|Q| = |S|$ car $Q_T = \emptyset$.

   On pose donc $|S|$ ou $|(S \cup S.\Sigma).T|$ questions différentes à `oracle` en fonction respectivement de si T est vide ou non.

5. cf automator.ml

6. cf automator.ml

7. Soit (S, T) une paire correcte et complète. Soit $A = A(S, T)$ l'automate associé.

   Montrons que A est déterministe et complet.

   On suppose que $\varepsilon \notin \Sigma$, ainsi δ ne contient pas d'ε-transition.

   Rappelons qu'on a $A = (Q, I, \Sigma, F, \delta)$ avec:
   - $Q = S$
   - $I = \{\varepsilon\}$
   - $F = S \cap \mathcal{L}$
   - $\delta = \left\{(q, c, q'), (q, q') \in S^2, q.c \underset{T}{\sim} q'\right\}$

   où $\underset{T}{\sim}$ représente la relation de T-équivalence.

   Montrons que A est **déterministe** càd :

$$\forall q \in Q, \forall a \in \Sigma, |\{q', (q, a, q') \in \delta\}| \leq 1$$

Par l'absurde:

Supposons qu'il existe $q, q_1, q_2 \in Q, q_1 \neq q_2, a \in \Sigma$ tels que:

$$(q, a, q_1) \in \delta \wedge (q, a, q_2) \in \delta$$

alors $q.a \underset{T}{\sim} q_1$ et $q.a \underset{T}{\sim} q_2$. Soit, par définition de la T-équivalence,

$$\forall w \in T, q_1.w \in \mathcal{L} \Leftrightarrow q.a.w \in \mathcal{L} \Leftrightarrow q_2.w \in \mathcal{L}$$

Ainsi, $q_1 \underset{T}{\sim} q_2$, or $q_1 \neq q_2$ donc (S, T) n'est pas correcte. ABSURDE !

$\rightarrow$ A est donc **déterministe**.

Montrons que A est **complet**:

La paire (S, T) est complète ainsi:

$$\forall u \in S, \forall a \in \Sigma, \exists v \in S \text{ tq } u.a \underset{T}{\sim} v$$

Ainsi, au vus de la construction de A :

$$(u, a, v) \in \delta$$

Donc $\forall q \in Q, \forall a \in \Sigma, \exists q' \in Q \text{ tq } (q, a, q') \in \delta \rightarrow$ A est **complet**.

8. Soit N le nombre de résiduels de $\mathcal{L}$. Supposons que :

$$|S| > N$$

Ainsi, $\exists u, v \in S, u \neq v$ tels que $u^{-1}.\mathcal{L} = v^{-1}.\mathcal{L}$ (Lemme des pigeonniers/tirroirs). Ainsi par définition de la T-équivalence :

$$u \underset{T}{\sim} v$$

Donc (S,T) n'est pas correcte. ABSURDE

Ainsi $\boldsymbol{|S| \leq N}$

9. On pose $C_{T,S} : \Sigma^* \rightarrow S$ l'appplication qui à un mot associe sa classe d'équivalence pour $\underset{T}{\sim}$ intersectée avec S.

Soit

$$S_{n+1} = \begin{cases} S_n \cup \{a_n\} \text{ si } a_n \text{ existe} \\ S_n \text{ sinon} \end{cases}, S_0 = S$$

où $a_n = q.a \text{ tq } C_{T,S_n}(q.a) = \emptyset$

Montrons que $\forall n \in \mathbb{N}, (S_n, T)$ est correcte par récurrence.

D'abord, $S_0 = S$ donc (S, T) est correcte.

Supposons que (S_k, T) est correcte pour un $k \in \mathbb{N}$ fixé quelconque.

Si $\nexists a_k$, alors $S_{k+1} = S_k$ donc (S_{k+1}, T) est correcte.

Si $\exists a_k$ alors par définition a_k n'est T-équivalent avec aucun élement de S_k. De plus (S_k, T) est correcte. Donc (S_{k+1}, T) est correcte.

Ainsi, $\forall n \in \mathbb{N}$, **(S_n, T) est correcte.**

Remarquons que $(|S_n|)_{n\in\mathbb{N}}$ est strictement croissante tant qu'a_n existe et constante dès qu'il n'existe plus aucun a_n. Or cette suite est majorée par N, donc elle converge forcément. Donc $(|S_n|)_{n\in\mathbb{N}}$ converge.

Or $(S_n)_{n\in\mathbb{N}}$ est croissante pour l'inclusion ($S_n \subset S_{n+1}$). Ainsi S_n converge et est constante dès un rang $R \in \mathbb{N}$.

On pose donc $S' = S_R$.

On a donc que (S', T) est correcte, montrons qu'elle est complête.

Par l'absurde:

Supposons que $\exists q \in S', a \in \Sigma$ tq $C_{T,S'} = \emptyset$ alors on a $S_{R+1} = S_R \cup \{q.a\}$. ABSURDE

car S_R est la limite de $(S_n)_{n\in\mathbb{N}}$.

Ainsi (S', T) est complète.

On pourra donc calculer S' en répetant les calculs des S_n successifs jusqu'à convergence.

10. cf automator.ml

11. Non traitée

2- Algorithme d'apprentissage

12. Lors de l'initialisation $(S, T) = (\{\varepsilon\}, \{\varepsilon\})$. Cette paire est évidemment correcte (Il n'y a qu'un mot dans S).

La question 9. garantie que S' est aussi correcte car elle n'ajoute à S' aucun élement T-équivalent à un autre.

Enfin, les étapes 3 et 4 ne modifient pas la paire (S, T).

Ainsi, les étapes 1 à 4 (incluse) conserve la correction de la paire.

Montrons que l'étape 5 le fait aussi.

Soit (S,T) la paire avant l'execution de l'étape 5, (S,T') la paire obtenue en ajoutant w et tout ses suffixes à T.

On suppose que (S,T) est correcte, montrons que (S,T') l'est aussi, par l'absurde.

Supposons que (S,T') ne soit pas correcte.

Ainsi $\exists u, v \in S$ tq $u \underset{T'}{\sim} v$, donc

$$\forall w \in T', u.w \in \mathcal{L} \Leftrightarrow v.w \in \mathcal{L}$$

Or $T \subset T'$ donc on a:

$$\forall w \in T, u.w \in \mathcal{L} \Leftrightarrow v.w \in \mathcal{L}$$

Donc (S, T) n'est pas correcte, ABSURDE.

Ainsi, l'étape 5 conserve bien la correction de la paire.

On a donc montré que toutes les étapes de l'algorithme conservent la correction de la paire et que celle-ci est initialisée correcte. Ainsi **(S, T) est correcte tout au long de l'éxecution de l'algorithme.**

13. cf automator.ml

14. Observons la trace d'execution suivante:

(∗ Magic Happens ∗: I become a computer)

- Initialisation :

  $(S, T) = (\{\varepsilon\}, \{\varepsilon\})$
- Tour 0 : $(S, T) = (\{\varepsilon\}, \{\varepsilon\})$

  S est complète: $S \leftarrow S' = S$

  L'oracle donne bbb comme contre exemple non reconnu par $A(S, T)$ (qui ne reconnait rien car $T \cap \mathcal{L} = \emptyset$ donc $F = \emptyset$).

  $T \leftarrow \{b.b.b, b.b, b, \varepsilon\}$
- Tour 1 : $(S, T) = (\{\varepsilon\}, \{b.b.b, b.b, b, \varepsilon\})$

  S n'est pas complète (car $\varepsilon.b \underset{T'}{\nsim} \varepsilon$).

  $S' = \{\varepsilon, b, b.b, b.b.b\}$

  On fait une requête d'équivalence sur $A(S, T)$:

  Cela donne l'automate (Figure 1).

  La requête d'équivalence réussit !

  $\rightarrow$ Fin de l'éxecution.

(∗ Magic Happens again ∗: vuelvo a ser humano)

L'algorithme a bien nécéssité $|\{0, 1\}| = 2$ tours (deux requêtes d'équivalence) pour déterminer A(S,T) qui selon le résultat de la question 11. est minimal.

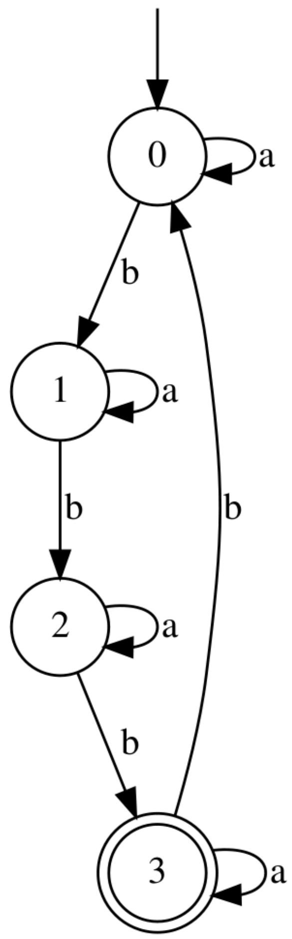


Figure 1: A(S,T) par automator

15. cf automator.ml

16. cf automator.ml

17. On suppose que (S, T') est complète.
    1. Soit $A = (Q, I, F, \Sigma, \delta) = A(S, T)$ et $A' = (Q', I', F', \Sigma, \delta')$

       On remarque tout d'abord que $Q' = S = Q$. Ainsi $F = F'$ et $I = I'$.

       De plus les paires étant complêtes et correctes, A et A' sont déterministes complets de même nombre de sommets donc $|\delta| = |\delta'|$.

       Montrons que $\delta' \subset \delta$.

       Soit:

$$\exists (q, a, q') \in \delta'$$

Alors $q.a \underset{T'}{\sim} q'$ donc $\forall w \in T', q.a.w \in \mathcal{L} \Leftrightarrow q'.w \in \mathcal{L}$

Or $T \subset T'$ donc:

$$\forall w \in T, q.a.w \in \mathcal{L} \Leftrightarrow q'.w \in \mathcal{L}$$

Donc $(q, a, q') \in \delta$.

Ainsi $\delta' \subset \delta$ et $|\delta'| = |\delta|$ donc $\boldsymbol{\delta' = \delta}$

2. $\Rightarrow$ Supposons $\varepsilon \xrightarrow{w} w', w' \in \mathcal{L}$ dans A'. Alors, $w' \in F' = F$ donc... Non rédigée.

3. Non rédigée.

18. Lors de l'étape 2, on étend S. Ainsi sa taille croit d'au moins 1 (on ajoute au moins un sommet). Or, la question 12 donne que (S, T) est toujours correcte. Ainsi selon la question 8. :

    $|S| \leq N$, où N est le nombre de résiduels du langage.

    On peut donc borner le nombre de passage par l'étape 2 par N.

19. Soit A l'automate (reconnaissant $\mathcal{L}$) à minimiser.

    On propose d'appliquer l'algorithme L^*, avec les oracle suivants:
    - Oracle d'appartenance de w: on verifie si w est reconnu par A
    - Oracle d'équivalence:

      soit $\mathcal{L}'$ le langage reconnu par $A(S, T)$. On construit à partir de A et A(S,T), l'automate reconnaisant :

      $$\mathcal{L}'' = (\mathcal{L} \cup \mathcal{L}') \setminus (\mathcal{L} \cap \mathcal{L}')$$

      On peut le construire à l'aide de l'automate produit et en prenant $F'' = (F \cup F') \setminus (F \cap F')$.

      On verifie ensuite que $\mathcal{L}'' = \emptyset$ en vérifitant que F'' est vide dans l'automate émondé. (càd aucun état final accessible).

      Si $\mathcal{L}'' = \emptyset$ alors $\mathcal{L} = \mathcal{L}' \to$ A et A' sont équivalents. Sinon ils ne le sont pas.

20. Soit N le nombre de résiduels de $\mathcal{L}$, soit K la borne de la taille des contre-exemples.

    Observons d'abord la complexite de la vérification de la T-équivalence.

    Remarquons que la taille de T augmente d'au plus K élements à chaque tours de boucle, or il y a au plus N tours de boucles. Ainsi on a :

    $$|T| \leq N * K$$

    Or la vérification de la T équivalence entre deux mots revient faire $2 * |T|$ tests d'appartenance à $\mathcal{L}$ (demande à l'utilisateur en $O(1)$).

    Ainsi la vérification de la T-équivalence est donc bornée en $O(N * K)$.

    Remarquons que pour le test de completude on fait de nombreux appels à oracle déja fait lors des tours précédent (en fait, on en fait seulement K nouveaux appels à oracle pour chaque tests de T-équivalence). À l'aide de la mémoisation le test de completude est donc en

    $$O(|S|^2 * |\Sigma| * K)$$

. Car on fait K nouveaux tests (parmis les N*K tests) pour chaque couple $(u, (v.a)) \in S * (S * \Sigma)$.

Ainsi le test de complétude est en $O(N^2 * |\Sigma| * K)$.

Cependant ce test n'est pas vraiment effectué. On réalise seulement la complétion (qui reviendra, si la paire est complète, à ne rien faire).

Cette opération est en $O(N * |S| * |\Sigma| * K) = O(N^2 * K * |\Sigma|)$

En effet, on effectue au plus N recherches de nouveaux états (car $|S| < N$), recherche qui revient à $|\Sigma| * |S|$ tests de T-équivalence, or à l'aide de la mémoïsation, les tests de T équivalences ne coutent qu'au plus K appels à oracle en $O(1)$.

Ainsi l'opération ligne 2 a une compléxite en $O(N^2 * K * |\Sigma|)$

Selon la question 3., la création de l'automate est en $O(N^2 * |\Sigma| * K)$ et la requête d'équivalence es en $O(1)$.

Ainsi l'opération ligne 3 a une compléxite en $O(N^2 * K * |\Sigma|)$

L'opération ligne 4 a une complexité en $O(1)$.

L'opération ligne 5 a une complexité en $O(K)$.

Ainsi, au global l'algorithme a une complexité en :

$$O(N^3 * K * |\Sigma|)$$

Plus le temps/la force de faire beaucoup plus...