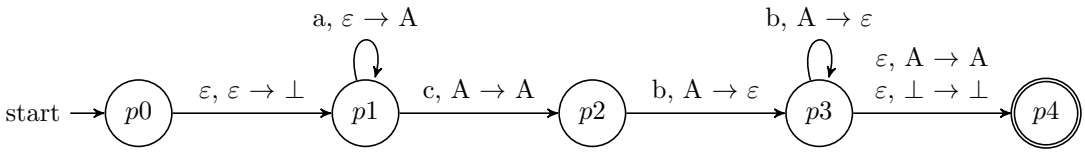
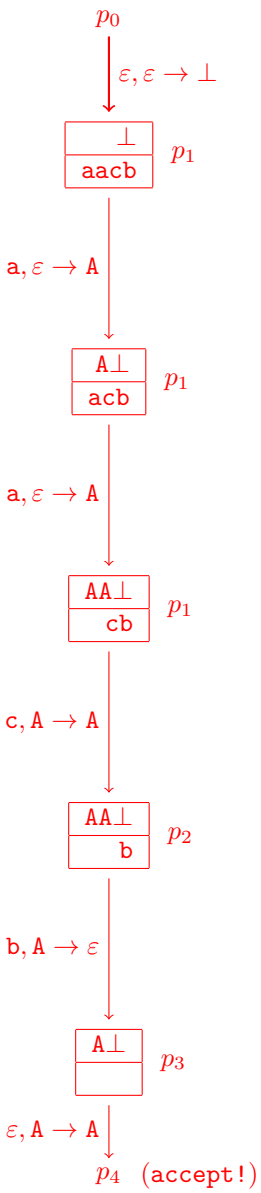
Langages Formels
Série 8 Correction - Automates à Pile

17 Novembre 2025

Pensez à justifier vos réponses.

1. Donner l'arbre d'exécution pour le mot *aacb*. Déterminez le langage accepté par cet automate à pile, puis construisez une grammaire Hors-Contexte qui génère le même langage :



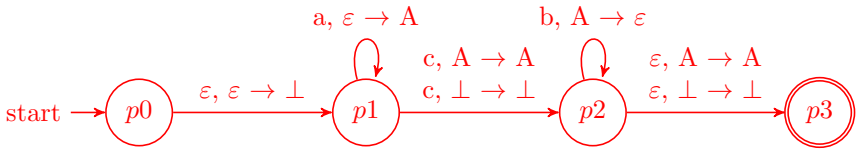


Depuis l'état initial, on commence par ajouter un symbole pour le fond de la pile. Ensuite, pour chaque "a" lu (en $p1$), on ajoute un A dans la pile. On doit en lire au moins un avant de continuer, car pour aller à $p2$, il faut lire un "c" dans le mot et un "A" sur le dessus de la pile (qu'on remet sur le dessus de la pile). Ensuite, chaque fois qu'on lit un b (au moins un pour atteindre $p3$), on dépile un A, puis on peut aller dans l'état final soit si on voit le symbole du fond de pile (on a lu autant de "b" que de "a"), soit s'il reste des A dans la pile (on a lu moins de "b" que de "a").
On a donc le langage $L = \{a^i c b^j \mid i \geq j \geq 1\}$.

Une grammaire Hors-Contexte générant ce langage serait $G = \langle \{S\}, \{a, b, c\}, S, P \rangle$ avec les règles : $P = \{S \to aSb \mid aS \mid acb\}$

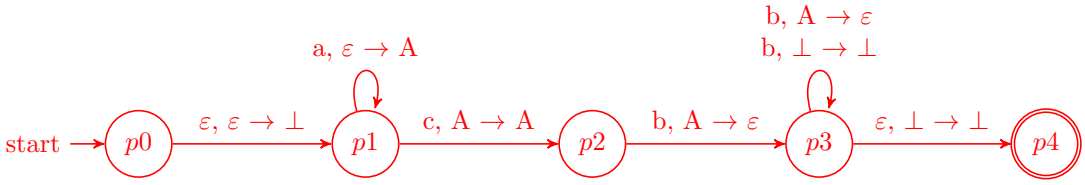
2. Construisez des automates à pile pour les langages suivants, sur l'alphabet $\Sigma = \{a, b, c\}$:

   - $L_1 = \{w = a^i c b^j \mid i \geq j \geq 0\}$.
     Voici un automate à pile acceptant L_1 :

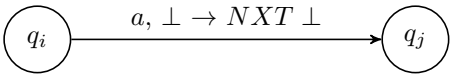


   - $L_2 = \{w = a^i c b^j \mid 1 \leq i \leq j\}$.

     Voici un automate à pile acceptant L_2 :



3. Equivalence entre modèles permettant d'ajouter un ou plusieurs éléments dans la pile : voici une transition dans un modèle ou l'on autoriserait d'ajouter plusieurs symboles dans la pile en même temps (avec a dans l'alphabet du langage, et $\bot, N, X, T$ dans l'alphabet de la pile) :



   A partir du modèle initial où une transition ne peut ajouter qu'un seul élément à la fois dans la pile, montrez qu'on peut simuler ce modèle (indice : utiliser plusieurs transitions et états intermédiaires).

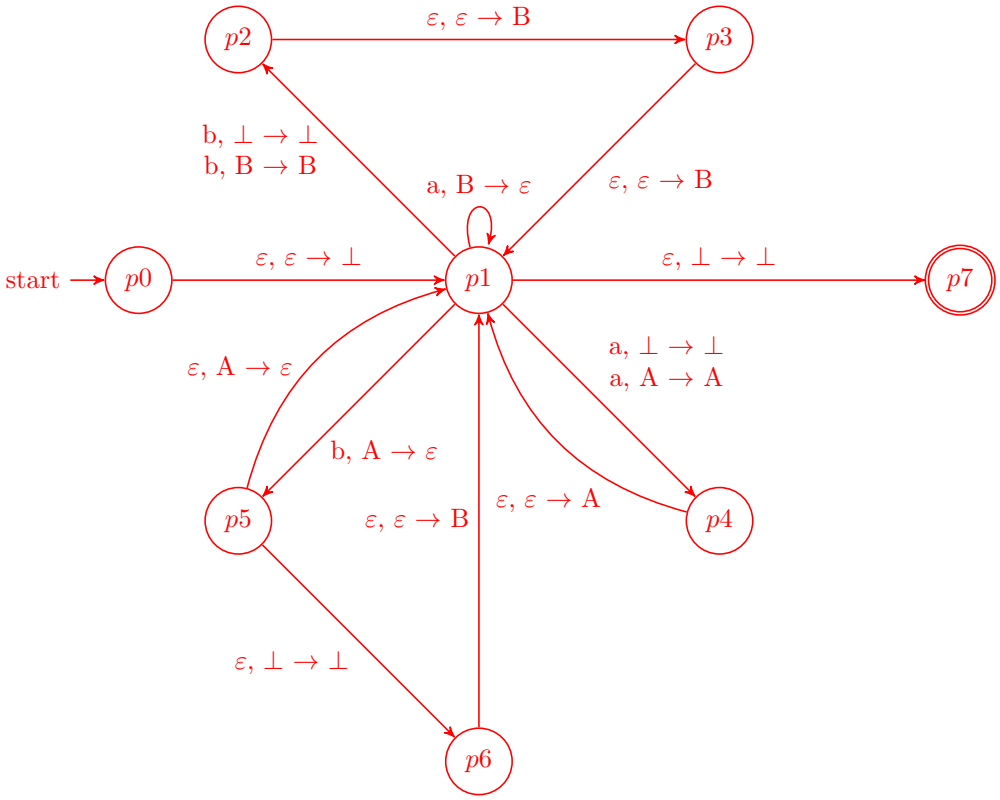
   On peut construire la structure suivante qui est équivalente :

   A partir de cet exercice, on s'autorise donc à utiliser ce modèle étendu avec des transitions ajoutant plusieurs caractères à la fois dans la pile.

4. Créez un automate à pile pour les langages suivants :

   - $L_3 = \{w \in \{a, b\}^* \mid |w|_a = 2|w|_b\}$

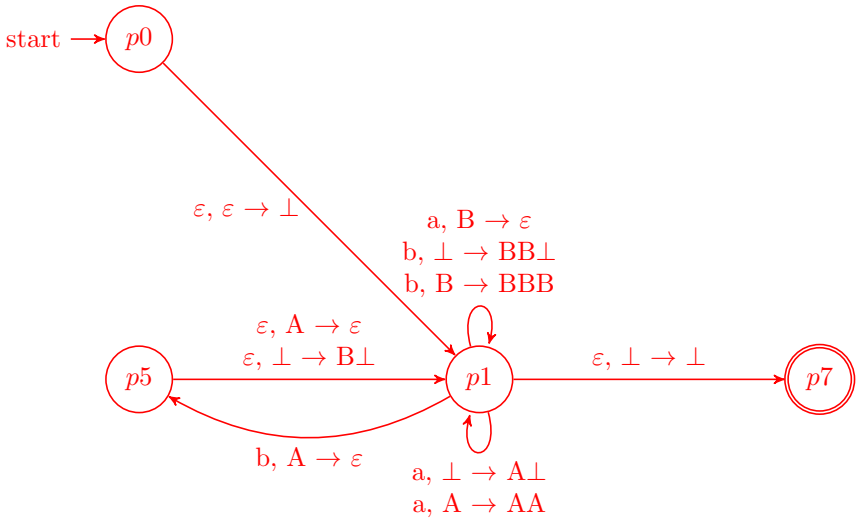
Ici, on veut accepter les mots contenant deux fois plus de a que de b (nombre de "a" = 2 fois le nombre de "b"). Cela signifie donc que pour chaque b dans le mot, on devra avoir deux a. Les caractères pouvant être dans n'importe quel ordre, on peut donc lire des "a" d'avance puis les compenser par des "b", ou lire des "b" d'avance puis les compenser par des "a", et répéter ou mélanger ces processus. Il va donc falloir gérer tout cela dans l'automate.
Voici un automate qui accepte ce langage (en se limitant à un caractère ajouté à la pile à la fois) :



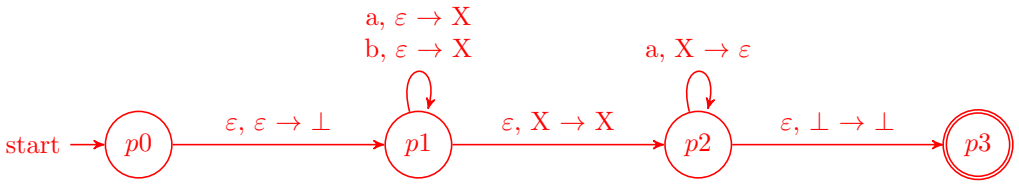
Cet automate utilise deux symboles de pile :
Les A symboliseront des "a" d'avance dans la pile, on écrira un A dans la pile pour chaque "a" d'avance (et chaque "b" lu plus tard retirera deux A de la pile).
A l'inverse, pour chaque "b" d'avance, on écrira deux B dans la pile. Cela permettra d'enlever un B pour chaque "a" lu plus tard (donc pour compenser le "b", qui ajoute deux B dans la pile, on devra lire deux "a").
On notera que l'automate est construit de telle manière qu'on a jamais des A et des B en même temps dans la pile : on ne peut empiler des A que sur des A ou le symbole du fond de la pile, et on ne peut empiler des B que sur des B ou sur le symbole du fond de la pile.
Enfin, dans le cas ou on lit un "b" mais qu'on a qu'un seul A dans la pile : on avait donc encore un seul "a" d'avance : ce sera un des deux

"a" qui compensent le "b", il faut donc enlever le A, puis ajouter un seul B dans la pile (car il ne manque plus qu'un "a" pour compenser).

Et voici un automate qui accepte ce langage avec la même lmogique décrite ci-dessus, mais simplifié en utilisant le fait qu'on peut ajouter plusieurs caractères dans la pile sur une transition:



- $L_4 = \{w = uv \mid u \in \{a,b\}^+,\ v \in a^+\ ,\ |u| = |v|\}$
  Ici, on veut accepter les mots qui ont deux moitiés non vides (de même taille) tels que la première moitié contient des a et/ou des b, et la deuxième moitié uniquement des a.
  Pour cela, on va empiler un symbole X en comptant indifféremment des a et/ou des b. Puis permettre de passer à n'importe quel moment dans un second état, ou on dépilera tous les symboles en lisant uniquement des a (pour assurer que la deuxième moitié ne contienne que des a, et soit de même longueur que la première moitié).



5. Soit le langage $L_{Pokemon} = \{Raichu, Sabelette, Maraiste, Lucario, Dedenne, Pohm\}$. Montrez que les deux langages suivants sont Hors-Contextes, en créant des automates à pile les acceptant :

   Vous pouvez simplifier les automates en utilisant seulement l'initiale de chaque Pokémon (Raichu = R, Sabelette = S, etc).

- $L_{MoreEvolved} = \{w \in (L_{Pokemon})^* \mid w$ contient strictement plus de Pokemon evolues que de Pokemon non-evolues$\}$, sachant que Raichu, Maraiste et Lucario sont des pokémon évolués (et les trois autres des non-évolués).

  Voici un automate acceptant $L_{MoreEvolved}$ (les E dans la pile comptent les évolués lus d'avance, les N dans la pile les non-évolués lus d'avance, et on a jamais les deux à la fois dans la pile) :

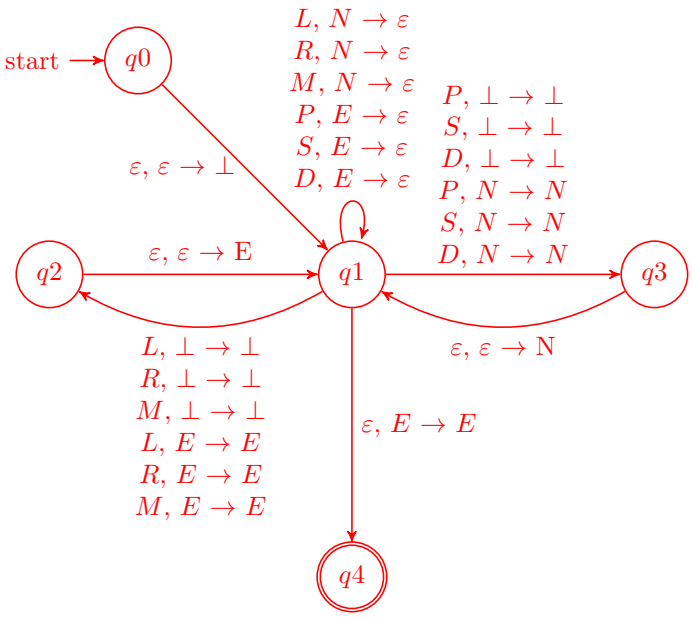


Figure 1: Automate acceptant $L_{MoreEvolved}$

- $L_{EquityGroundElectric} = \{w \in (L_{Pokemon})^* \mid w$ contient autant de Pokemon de type Sol que de type Electric$\}$, sachant que Raichu, Dedenne et Pohm sont de type Electric, que Sabelette et Maraiste sont de type Sol, et que Lucario n'est d'aucun des deux types.

  Voici un automate acceptant $L_{EquityGroundElectric}$ (principe assez similaire : on compte les type electrik ou sol d'avance par des E et G respectivement dans la pile) :

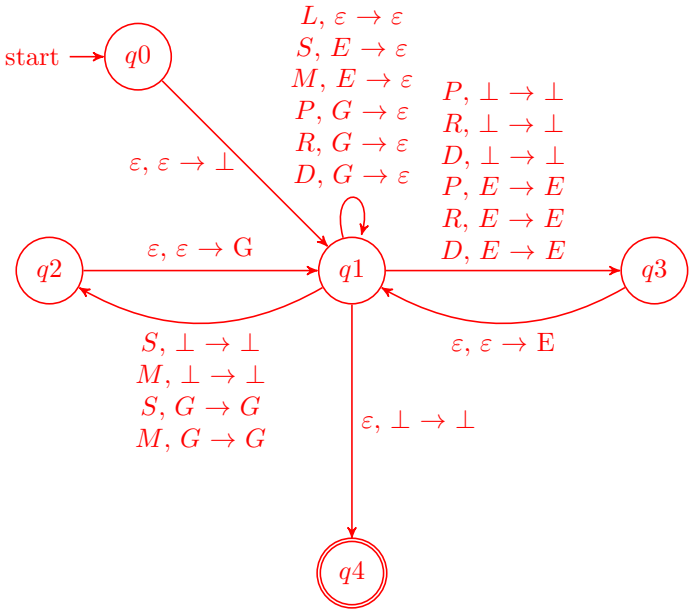
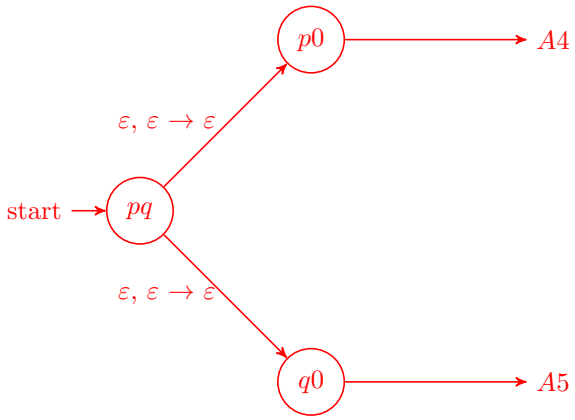


Figure 2: Automate acceptant $L_{MoreEvolved}$

Note : pour ces deux automates, on pourrait également simplifier cet automate en mettant plusieurs caractères dans la pile à la fois : cela permettrait de remplacer les boucles depuis q1 passant par q2 et q3 par des boucles directement sur q1.

6. Soit L_4, L_5 deux langages hors-contexte quelconques, et A_4, A_5 deux automates à pile qui les acceptent respectivement. Que peut-on dire en général sur les langages suivants, sont-ils forcément hors-contextes ?
   Si oui, montrez comment construire un automate à pile acceptant le résultat à partir des deux automates à pile acceptant L_4 et L_5 (indiquez clairement quels changements sont opérés sur les automates A_4 et A_5 si pertinent).
   Si non, donnez un contre-exemple où le/les langage(s) initial/aux est/sont Hors-Contexte, mais où le résultat de l'opération n'est pas Hors-Contexte.

   - $L_6 = L_4 \cup L_5$
     L'union de deux langages Hors-Contexte est Hors-Contexte :
     A partir des deux automates A4 et A5, on peut construire l'union avec la structure suivante, ou p_0 et q_0 sont les états initiaux des deux automates (et en recopiant bien sûr l'entièreté des deux automates) :

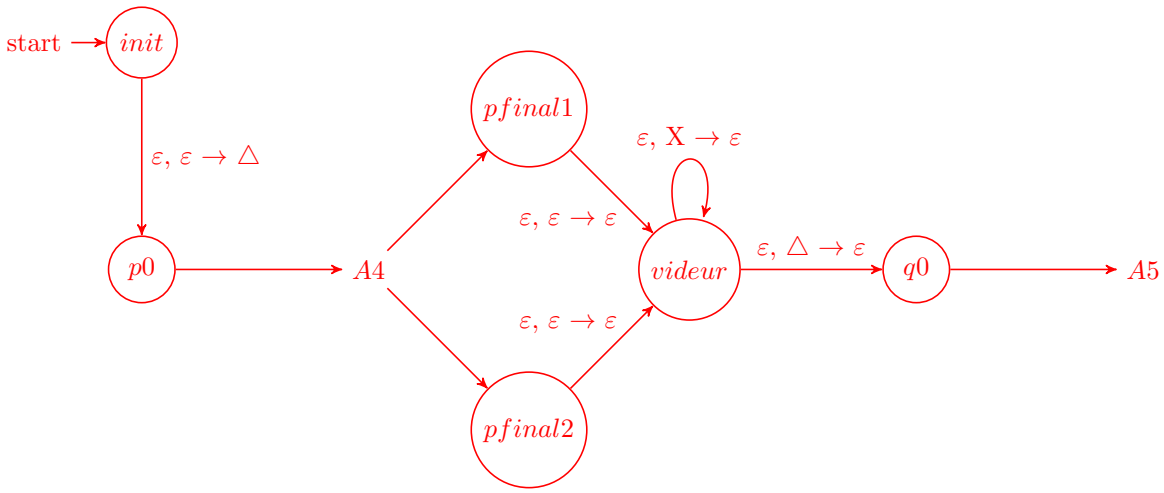


(*Note : on aurait aussi pu le montrer via les grammaires Hors-Contexte.*
Soit $G_1 = \langle N_1, T_1, S_1, P_1 \rangle$ et $G_2 = \langle N_2, T_2, S_2, P_2 \rangle$ les deux grammaires Hors-Contextes acceptant les deux langages. On commence par renommer tous les Non-terminaux de G_2 et adapter les règles correspondantes pour qu'ils soient tous différents de ceux de G_1 (y compris l'axiome) si c'est nécessaire (on veut donc N_1 et N_2 qui sont disjoints). Et soit S un nouveau non-terminal n'appartenant ni à N_1 ni à N_2. Il suffit alors de construire la grammaire suivante :
$G_3 = \langle N_1 \cup N_2 \cup S, T_1 \cup T_2, S, P_1 \cup P_2 \cup \{S \to S_1, S \to S_2\} \rangle$.
On prend donc les deux grammaires ensemble, et on ajoute depuis le nouvel axiome S la possibilité d'aller soit vers l'axiome de la première grammaire, S_1, soit vers l'axiome de la seconde grammaire, S_2.
)

- $L_7 = L_4 \cap L_5$
  En général, l'intersection de deux langages Hors-Contexte n'est pas forcément Hors-Contexte.
  Par exemple, si $L_4 = \{a^n b^n c^i \mid n, i \geq 0\}$ et $L_5 = \{a^n b^i c^n \mid n, i \geq 0\}$, alors l'intersection sera $L_7 = \{a^n b^n c^n \mid n \geq 0\}$, qui n'est pas Hors-Contexte (On le montrera via le deuxième lemme de l'étoile pour langages Hors-Contexte).

- $L_8 = L_4 \cdot L_5$
  La concaténation de deux langages Hors-Contexte est Hors-Contexte. Soit les deux automates A_4 et A_5 acceptant L_4 et L_5. On construit la concaténation de la manière suivante : on commence par ajouter un nouveau symbole dans l'alphabet de pile, notons le $\triangle$, différent de ceux déjà utilisés par A_4 et A_5. Ce nouveau symbole sera placé avant A_4, comme nouveau symbole de fond de pile (sous l'éventuel symbole de fond de pile utilisé par A_4).
  Ensuite, on rend les états finaux de la machine M_1 non finaux (ici représentés par pfinal1 et pfinal2, on peut en avoir un nombre variable), puis on les "branche" sur un état intermédiaire (le "videur"), et on branche cet état intermédiaire sur l'état initial de A_5 (ici représenté par q0). La différence par rapport aux automates finis est qu'ici on doit vider la pile avant de commencer la machine A_5.

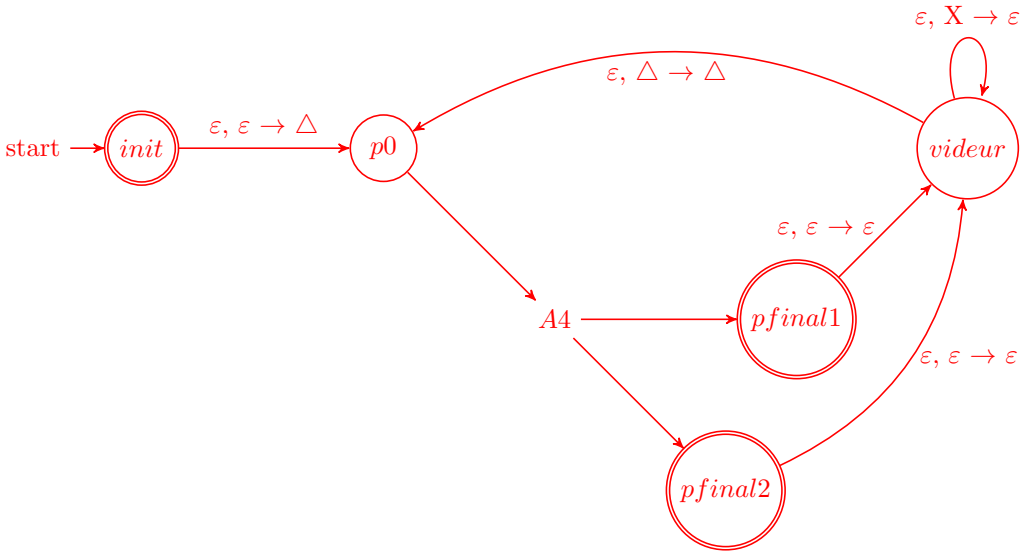
C'est ce que fait l'état "videur" : on a une transition pour chaque caractère de pile possible dans A_4 qui enlève ce caractère. Ici, on note X qui représente n'importe quel symbole de l'alphabet de la pile pour A_4. Quand on a terminé, on retrouve alors le nouveau symbole de pile vide qu'on avait mis avant A_4, le $\triangle$, et on sait qu'on a vidé la pile et qu'on peut commencer A_5 :



- $L_9 = L_4^*$

  L'étoile de Kleene sur un langage Hors-Contexte crée un langage Hors-Contexte :

  Soit A_4 la machine acceptant L_4. Il suffit de permettre de revenir à l'état initial en vidant la pile depuis les états finaux pour créer L^+ (de la même manière que pour la concaténation, on ajoute un nouveau symbole de pile vide, $\triangle$, pour savoir quand on a fini de vider la pile entre deux répétitions), et d'ajouter le mot vide via un état supplémentaire pour obtenir L^* (ici, $p0$ est l'état initial de la machine A_4 pour L_4, $pfinal1$ et $pfinal2$ ses états finaux, et X représente n'importe quel symbole de la pile de la machine A_4) :



- $L_{10} = \overline{L_4}$

  En général, le complémentaire d'un langage Hors-Contexte n'est pas forcément Hors-Contexte.
  Par exemple, soit $\Sigma = \{a, b, c\}$, et $L_4 = \{w \in \Sigma^* \mid |w|_a \neq |w|_b\} \cup \{w \in \Sigma^* \mid |w|_a \neq |w|_c\}$ (c'est à dire, L_4 contient tous les mots ayant un nombre de a différent de celui des b ou différent de celui des c), alors le complémentaire sera $L_{10} = \overline{L_4} = \{w \in \Sigma^* \mid |w|_a = |w|_b = |w|_c\}$. Et ce langage n'est pas Hors-Contexte (preuve très similaire à celle pour $a^n b^n c^n$, ce qu'on verra via le second lemme de l'étoile pour langages Hors-Contextes).