$\mathbb{L}$angages $\mathbb{A}$utomates $\mathbb{N}$on-déterminisme $\mathbb{G}$rammaires $\mathbb{A}$ttibuées et $\mathbb{G}$énératives $\mathbb{E}$xpressions régulières

$\mathbb{C}$orrection $\mathbb{P}$artielle de $\mathbb{P}$rogrammes

«Ceci n'est pas un cours de Langage C++»

3 Langage reconnus par un automate non-déterministe, opérations sur les langages, opérations sur les automates

3.1 Motivation

- Utilisation des automates (dés 1950 : S.C.Kleene, C.Shannon, J.Mc Carthy) :
  - conception de circuits, de protocoles, de menu de navigation, de digicode, de politique de sécurité
  - reconnaissance et recherche de mots dans un texte : bonne complexité de l'algorithme de recherche $O(m^3 + t)$
  - compilation : analyse lexicale
- Utilisation des grammaires – le quotidien des informaticiens – cours RICM4 :
  - définition des langages de programmation ou de réprésentations de données
  - analyse syntaxique
- Récente utilisation des automates : l'ordinateur quantique (Travaux de B.Boigelot et P.Wolper) Un variable quantique peut avoir plusieurs valeurs à la fois. Par exemple deux entiers quantiques $\hat{n}$ et $\hat{m}$ peuvent avoir chacun une infinité de valeurs dans $\mathbb{N}$: $\hat{n} = \{1, 21, 321, 4321, \ldots\}$ et $\hat{m} = \{0, 2, 4, 6, 8, \ldots\}$. L'addition de $\hat{n}$ et de $\hat{m}$ donne alors une valeur quantique, c'est-à-dire une valeur parmi l'ensemble des valeurs possibles de $n + m$ avec $n \in \hat{n}$ et $m \in \hat{m}$:

$$\hat{n} + \hat{m} = \{n + m \mid n \in \hat{n}, m \in \hat{m}\} = \left\{ \begin{array}{c} 1+0, 21+0, 321+0, 4321+0, \ldots \\ 1+2, 21+2, 321+2, 4321+2, \ldots \\ 1+4, 21+4, 321+4, 4321+4, \ldots \\ \vdots \end{array} \right\}$$

  En représentant $\hat{n}$ et $\hat{m}$ par des automates, B.Boigelot et P.Wolper[1] parviennent à calculer l'automate qui correspond à $\hat{n} + \hat{m}$, ce qui revient à faire une infinité d'opérations en parallèle.

3.2 Langage

- Alphabet, mot, opération sur les mots : longueur $|\omega|$, projection ω/s, concaténation $\omega_1.\omega_2$
- Langage, opérations sur les langages : union, intersection, restriction, complémentaire
- Concaténation de langage $L_1 \cdot L_2$, généralisation à la fermeture de Kleene L^*

3.3 Automates reconnaisseurs à (nombre d') états finis (AEF)

- Construction à l'aide des catégories de langages $\{\epsilon, \omega, \omega a, \omega ab, \omega aba\}$
- Construction d'un automate complétement spécifié $spec(A)$

[1] B. Boigelot and P. Wolper. Representing Arithmetic Constraints with Automata : An Overview. Proceedings of the 18th International Conference on Logic Programming, volume 2401, Lecture Notes in Computer Science, pages 1-19, July 2002, Springer-Verlag.

– Représentation graphique et sous forme de table de transitions
– Principe d'exécution d'un automate, condition d'acceptation d'un mot

3.4 Opérations sur les langages / Opérations sur les automates

3.4.1 Langage complémentaire / Automate complémentaire

Si l'automate A est déterministe et complétement spécifié alors le complémentaire du langage reconnu par A est le langage reconnu par l'automate $inv(A)$ obtenu à partir de A en inversant les états accepteurs/non-accepteurs.

$\overline{\mathscr{L}(A)} = \mathscr{L}(A^C)$ *où* $A^C \stackrel{def}{=} inv(det(spec(A)))$

Remarque L'Exercice 2.3 donne un exemple de construction incorrecte du complémentaire.

3.4.2 Intersection de langages / Produit d'automates

L'automate produit simule une exécution simultanée des automates A_1 et A_2.

$\mathscr{L}(A_1) \cap \mathscr{L}(A_2) = \mathscr{L}(A_1 \times A_2)$

Principe de construction
– Les états de $A_1 \times A_2$ sont des couples (q_1, q_2) avec $q_1 \in Etat(A_1)$ et $q_2 \in Etat(A_2)$. Lors d'une reconnaissance q_1 indique l'état courant de A_1 et q_2 l'état courant de A_2.
– Les états accepteurs de $A_1 \times A_2$ sont des couples, notés $(q_1^a, q_2^a)^a$ où $q_1 \in Acc(A_1)$ et $q_1 \in Acc(A_2)$.
– Les états initiaux de $A_1 \times A_2$ sont les couples ${}_i(_iq_1, {}_iq_2)$ formés d'un état inital de A_1 et d'un état inital de A_2.
– Les transitions de $A_1 \times A_2$ sont construites de la manière suivante

si $q_1 \xrightarrow{l} q_1' \in Trans(A_1)$ et $q_2 \xrightarrow{l} q_2' \in Trans(A_2)$

alors $(q_1, q_2) \xrightarrow{l} (q_1', q_2') \in Trans(A_1 \times A_2)$

En bref,

$Etat(A_1 \times A_2) \stackrel{def}{=} Etat(A_1) \times Etat(A_2)$
$Init(A_1 \times A_2) \stackrel{def}{=} Init(A_1) \times Init(A_2)$
$Accept(A_1 \times A_2) \stackrel{def}{=} Accept(A_1) \times Accept(A_2)$
$Trans(A_1 \times A_3) \stackrel{def}{=} \{(q_1, q_2) \xrightarrow{l} (q_1', q_2') \mid l \in \Sigma, q_1 \xrightarrow{l} q_1' \in Trans(A_1), q_2 \xrightarrow{l} q_2' \in Trans(A_2)\}$

Remarques La construction du produit est correcte pour les automates non-déterministes et partiellement spécifiés. L'automate produit qu'on obtiendra sera alors non-déterministe et partiellement spécifié.

Lors de la construction de $A_1 \times A_2$ on prend soin de ne pas construire les états (q_1, q_2) inutiles, c'est-à-dire deux qui ne sont pas accessibles depuis les états initiaux de $A_1 \times A_2$.

Applications voir Exercice 2.1

Q1. Donnez l'automate A des nombres binaires paires, c'est-à-dire l'automate qui reconnaît le langage formé des mots sur $\Sigma = \{0, 1\}$ qui se terminent par 0.

$\mathscr{L}(A) = \{0, 10, 100, 110, 1000, 1010, 1100, 1110, \ldots\}$

Q2. Construisez le complémentaire de A, donnez les premiers mots du langage reconnu par A^C.

Q3. Construisez le produit $A \times A^C$, que reconnaît cet automate ?

3.4.3 Union de langages / Somme d'automates

$$\mathscr{L}(A_1) \cup \mathscr{L}(A_2) = \mathscr{L}(A_1 + A_2) \text{ où } A_1 + A_2 = \left(A_1^C \times A_2^C\right)^C$$

Rappel sur les ensembles : $E \cup F = \overline{\overline{E} \cap \overline{F}}$ donc $\mathscr{L}(A_1) \cup \mathscr{L}(A_2) = \overline{\overline{\mathscr{L}(A_1)} \cap \overline{\mathscr{L}(A_2)}} = \mathscr{L}\big(\ \overline{\overline{A_1} \times \overline{A_2}}\ \big)$
Pour faire l'union de deux langages $\mathscr{L}(A_1), \mathscr{L}(A_2)$, on construit la somme des automates A_1, A_2. Si les automates A_1 et A_2 sont déterministes et complétement spécifiés pour construire la somme $A_1 + A_2$ il suffit d'appliquer l'algorithme de construction du produit d'automates puis d'utiliser les règles suivantes pour déterminer les états accepteurs de $A_1 \times A_2$.
Soit q_1 un état de A_1 et q_2 un état de A_2 :

A_1	A_2	$\overline{A_1}$	$\overline{A_2}$	$\overline{A_1} \times \overline{A_2}$	$\overline{\overline{A_1} \times \overline{A_2}}$
q_1^a	q_2^a	q_1	.....	.....	$(q_1, q_2)^a$
q_1	q_2	q_1^a	q_2^a	.....	.....
q_1	q_2^a	.....	.....	.....	$(q_1, q_2)^a$
q_1^a	q_2	.....	.....	(q_1, q_2^a)	.....

Conclusion : un état (q_1, q_2) de $A_1 \times A_2$ est accepteurs *si et seulement si* $q_1 \in Acc(A_1)$ ou $q_2 \in Acc(A_2)$.

Remarque Cet algorithme est incorrecte si les automates A_1 et A_2 ne sont pas déterministes et complétement spécifiés. En effet dans ce cas la construction du complémentaire est incorrecte. Nous verrons dans un prochain cours un autre moyen de construire la somme de deux automates.

3.4.4 Restriction d'un langage

Rappel sur les ensembles : $E \setminus F = E \cap \overline{F}$ donc on peut construire l'automate qui reconnaît le langage $\mathscr{L}(A_1) \setminus \mathscr{L}(A_2)$ puisqu'on sait effectuer les opérations $\cap$ et complémentaire sur les langage reconnu par un automate.