Cours – Application des langages réguliers

1 Les langages réguliers (c'est-à-dire les AEF et les expressions régulières) sont très utilisés

Ils sont utiles et utilisés par les théoriciens du fait que

1. qu'on sait effectuer toutes les opérations ensemblistes sur les automates : $\in, \notin, \cup, \cap, \text{-}, \subseteq, \stackrel{?}{=} \emptyset$
2. que ces opérations correspondent aux opérateurs logiques *satisfait*, *ne satisfait pas*, $\vee, \wedge, \neg, \Rightarrow, \nexists$
3. que les AEF sont équivalents à des formules de la logique monadique du second ordre (une extension de la logique usuelle) ; et qu'on sait ainsi décider si une formule est invalide (ou non) en cherchant si le langage de l'automate associée est vide (ou non).

Ils sont utiles et utilisés en pratique :

1. **pour rechercher efficacement des mots dans un texte à partir d'une expression régulière**.

   Par exemple, rechercher dans une distribution linux les packages dont le nom est de la forme `lib_?*0.?*.1`

   La recherche est en $O(m^3 + t)$ c'est-à-dire cubique dans la taille m du motif recherché et linéaire dans celle t du texte. L'algorithme est donc efficace puisqu'en général le motif est court et le texte est long.

2. **pour décrire des propriétés d'exécution : par exemple des politiques de sécurité.** Un AEF (appelé moniteur de sécurité) définit les séquences d'opérations autorisées. Par exemple, on peut interdire les connections de machine en machine par login successifs tel que

   $$\texttt{login}(m_1,m_2)\ ;\ \texttt{login}(m_2,m_3)\ ;\ \texttt{login}(m_3,m_4)\ ;\ \ldots$$

   et imposer que chaque login soit suivi d'un logout avant de refaire un login sur une autre machine, c'est-à-dire

   $$\texttt{login}(m_1,m_2)\ ;\ \texttt{logout}(m_2)\ ;\ \texttt{login}(m_1,m_3)\ ;\ \texttt{logout}(m_3)\ ;\ \ldots$$

   Pour cela on place sur la machine m_1 un moniteur qui bloque l'opération `login` si la séquence de login lougout n'appartient pas au langage défini par l'expression régulière

   $$\big(\texttt{login}(m,m')\ ;\ \texttt{logout}(m')\big)^*$$

   Puisque les opérations (union, intersection, complémentaire, privé de,...) sont définis sur les AEF on peut facilement combiner des politiques de sécurité. On peut ainsi définir :
   - une politique permissive qui permet tout ce que permettent les réseaux A et B en prenant l'union des politiques de sécurité
   - une politique restrictive en autorisant uniquement ce qui est permis par les deux réseaux : on fait alors l'intersection des politiques de sécurité

3. **pour reconnaître les mots-clefs et les identificateurs des langages de programmation.**

   Exemple Notons $L = \{a, \ldots, z\}$ l'ensemble des lettre minuscules ; $M = \{A, \ldots, Z\}$ l'ensembles des lettres majuscules ; et $C = \{0, \ldots, 9\}$ l'ensemble des chiffres.

   Dans de nombreux langages de programmation les identificateurs de variables sont définis par lexpression régulière $L.(L \cup M \cup C)^*$ et les constantes par $M.(L \cup M \cup C)^*$

4. **pour gérer des interactions complexes avec un utilisateur.** C'est la meilleure façon de programmer une interface de navigation dans un menu qui offre un grand nombre de commandes (comme ceux des téléphones portables par exemple). Les AEF présentent l'avantage d'être faciles à implémenter (voir section **??**), efficaces, faciles à modifier et d'éviter les erreurs de programmation. L'utilisation d'un automate remplace avantageusement un empilement de case ou de if-then-else

difficile à maintenir (comparez ce type d'implantation avec la proposition de la section **??**). Par ailleurs, du fait que sur les AEF on sait réaliser toutes les opérations ensemblistes, on dispose de nombreux algorithmes :

- pour echaîner des menus (concaténation),
- fusionner des menus (union),
- selectionner les commandes communes à deux menus (intersection),
- autoriser les commandes qui ne sont pas dans un autre menu (opération privé de),
- pour les optimiser (minimisation),
- les tester (en générant automatiquement les tests),
- les mettre au point et les vérifier : on sait générer les suites de commandes qui mènent à des états bloquants, éliminer les états inaccessibles, *etc*

Ils sont utilisés par exmeple dans les applications suivantes : analyse lexicale dans le compilateurs, gestion d'une partie des interactions sur les téléphones portables, les carte à puces, les controlleurs de périphériques électronique, les ascenseurs, l'électro-ménager, *etc*

2 Les langages réguliers sont limités

Certains langages échappent aux AEF, autrement dit il existe des langages qui ne sont pas reconnaissable par un AEF.

Exemple : le langage $\{a^n b^n \mid n \in \mathbb{N}\}$ n'est pas régulier ce qui signifie qu'il est impossible de reconnaître les expressions bien parenthésées de la forme (((((((...))))))) avec un AEF. Il s'agit d'un cas particulier où $a =$ `"("` et $b =$ `")"`.

Il est alors clair que les AEF ne peuvent suffir à définir la syntaxe d'un langage de programmation puisqu'il sera (entre autre) impossible de vérifier à l'aide d'un AEF que chaque `"("` est bien suivie d'une `")"`, idem pour les accolades et les crochets.

En pratique l'usage des AEF est limités à la reconnaissance des mots-clefs du langage de programmation.

On le voit il est nécessaire d'explorer les langages au delà des langages réguliers.