Examen du cours «Analyse d'algorithmes» (1ère session)

25 janvier 2006

Rappel : seule une feuille A4 de notes manuscrites est autorisée pendant l'examen. Tout autre document est interdit.

Le barême donné est indicatif. Soyez précis et concis. On rappelle que le soin apporté aux justifications demandées est déterminant. Veillez à ne pas confondre votre copie et vos feuilles de brouillon, merci !

1 Questions de cours

Exercice 1 (4 points)
Pour chaque affirmation ci-dessous, vous indiquerez, en le justifiant, si elle est vraie ou fausse. **Une réponse correcte sans justification ne rapportera aucun point**.

1. Tout problème de décision peut-être résolu par un algorithme.
2. Toute machine de Turing est une procédure effective.
3. Tout langage régulier est décidé :
   (a) par une machine de Turing
   (b) par un automate fini
4. Deux algorithmes en $\mathcal{O}(n)$ prennent le même temps de calcul pour toute valeur de n.
5. Un langage régulier est décidé en temps linéaire en la taille du mot d'entrée.
6. La simulation déterministe d'une machine de Turing non-déterministe préserve sa complexité :
   (a) temporelle
   (b) spatiale

2 Automates finis

On définit l'opérateur *soustraction préfixe* sur les langages de la façon suivante :

$$u^{-1}L = \{v \mid uv \in L\}$$

pour tout mot u et pour tout langage L sur un alphabet Σ. $u^{-1}L$ est donc le langage des mots de L qui commencent par u, privés du préfixe u. Par exemple, $ab^{-1}\{ab, abcd, cab, c\} = \{\varepsilon, cd\}$.

L'opérateur de soustraction préfixe a les propriétés suivantes :

1. $\varepsilon^{-1}L = L$
2. $a^{-1}\emptyset = \emptyset$
3. $a^{-1}\{\varepsilon\} = \emptyset$
4. $a^{-1}\{a\} = \{\varepsilon\}$
5. $b^{-1}\{a\} = \emptyset$
6. $a^{-1}(L \cup M) = (a^{-1}L) \cup (a^{-1}M)$

7. $a^{-1}(LM) = (a^{-1}L)M$ si $\varepsilon \notin L$
8. $a^{-1}(LM) = (a^{-1}M) \cup (a^{-1}L)M$ si $\varepsilon \in L$
9. $a^{-1}L^* = a^{-1}LL^*$

pour tous symboles $a, b \in \Sigma$ distincts et pour tous langages $L, M \subseteq \Sigma^*$.

Exercice 2 (2 points)
Prouver les propriétés 1, 3 et 8 ci-dessus

Exercice 3 (6 points)
Dans cet exercice, pour simplifier l'écriture des langages, un singleton $\{a\}$ est simplement noté a. Nous considérons le langage $L = (a \cup b)^* ab (a \cup b)^*$ sur l'alphabet $\Sigma = \{a, b\}$. Voici le résultat de l'application de l'opérateur de soustraction préfixe sur L pour les symboles de Σ :

$$\begin{aligned} & a^{-1}\left((a \cup b)^* ab (a \cup b)^*\right) \\ &= \left(a^{-1}(ab(a \cup b)^*)\right) \cup \left(a^{-1}(a \cup b)(a \cup b)^* ab (a \cup b)^*\right) \\ &= (b(a \cup b)^*) \cup ((a \cup b)^* ab (a \cup b)^*) \end{aligned}$$

et :

$$\begin{aligned} & b^{-1}\left((a \cup b)^* ab (a \cup b)^*\right) \\ &= \left(b^{-1}(ab(a \cup b)^*)\right) \cup \left(b^{-1}(a \cup b)(a \cup b)^* ab (a \cup b)^*\right) \\ &= \emptyset \cup \varepsilon(a \cup b)^* ab (a \cup b)^* \\ &= (a \cup b)^* ab (a \cup b)^* \end{aligned}$$

1. Le tableau ci-dessous est construit de la façon suivante. Le langage L est placé dans la colonne de gauche. On écrit, sur la même ligne du tableau, $a^{-1}L$ dans la colonne a et $b^{-1}L$ dans la colonne b. Notons que $b^{-1}L = L$, mais $a^{-1}L \neq L$. On écrit alors $a^{-1}L$ dans la colonne de gauche, et on complète la ligne associée comme précédemment. Ce processus se poursuit tant que de nouveaux langages sont calculés : ils sont ajoutés au fur et à mesure dans la colonne de gauche. Compléter ce tableau.

	a	b
$(a \cup b)^* ab (a \cup b)^*$	$(b(a \cup b)^*) \cup ((a \cup b)^* ab (a \cup b)^*)$	$(a \cup b)^* ab (a \cup b)^*$
$(b(a \cup b)^*) \cup ((a \cup b)^* ab (a \cup b)^*)$	$\cdots$	$\cdots$
$\cdots$	$\cdots$	$\cdots$

Remarque : $b(a \cup b)^* \subset (a \cup b)^*$

2. Le tableau précédent définit les états (colonne de gauche) et les transitions d'un automate fini. Dessinez-le et définissez ses états initiaux et accepteurs. Justifier votre réponse.

3. Quelles sont les propriétés d'un automate fini obtenu par cette méthode ? Justifier votre réponse.

3 Preuve d'algorithmes

Nous considérons le while-program suivant qui calcule le nombre d'occurrences d'un élément `e` dans une liste `l`.

```
n := 0; l' := l;
while (l' != NULL) do
  if (elm(l') = e) then
    n := n+1
  endif;
  l' := next(l')
endwhile
```

Le TAD (Type Abstrait de Données) `list` fournit les opérations suivantes :

```
NULL : list
elm : list -> element
next : list -> list
```

où `NULL` désigne la liste vide, `elm` renvoie l'élément stocké dans la cellule en tête de liste, et `next` renvoie la liste privée de sa première cellule. Nous disposons de plus de la fonction :

```
nbocc : element * list -> int
```

qui indique le nombre d'occurrences d'un élément dans une liste.

Exercice 4 (6 points)
Notre but est de prouver que ce while-program est **correct**, c'est à dire qu'il satisfait les spécifications :

- Pré-cond : `vrai`
- Post-cond : `n=nbocc(e,l)`

Répondre aux questions suivantes :

1. Donner un invariant pour la boucle `while` qui permet de prouver la correction. Vous expliquerez votre choix.
2. Calculer la preuve de ce while-program et donner les obligations de preuve que vous générez.
3. Indiquer, en le justifiant, si vos obligations de preuve sont satisfaites. Conclure sur la correction du while-program.

Exercice 5 (2 points)
Nous nous intéressons maintenant à la **terminaison** de ce while-program.

1. Que devez-vous prouver ?
2. Proposer une mesure en justifiant votre choix. Vous pouvez, si vous le souhaitez, ajouter une opération au TAD `list` (nous ne demandons pas la preuve de terminaison).

Règles de la logique de Hoare

Règle de conséquence :

$$\frac{P' \Rightarrow P, \{P\}\, S\, \{Q\}\ ,\ Q \Rightarrow Q'}{\{P'\}\, S\, \{Q'\}}$$

Règle d'affectation :

$$\frac{}{\{P[e/x]\}\, x{:=}e\, \{P\}}$$

Règle de séquence :

$$\frac{\{P\}\, S_1\, \{Q\}\ ,\ \{Q\}\, S_2\, \{R\}}{\{P\}\, S_1;S_2\, \{R\}}$$

Règle du «`si`» :

$$\frac{\{P \wedge B\}\, S_1\, \{R\}\ ,\ \{P \wedge \neg B\}\, S_2\, \{R\}}{\{P\}\, \texttt{si}\ B\ \texttt{alors}\ S_1\ \texttt{sinon}\ S_2\ \texttt{finsi}\, \{R\}}$$

Règle du «`tantque`» :

$$\frac{\{P \wedge B\}\, S\, \{P\}}{\{P\}\, \texttt{tantque}\ (\texttt{B})\ \texttt{faire}\ \texttt{S}\ \texttt{fintantque}\, \{P \wedge \neg B\}}$$