Automates et applications

Paul Patault

12 mars 2022

Résumé

Projet du cours *Automates et Applications*. Utilisation de la structure de donnée d'automates d'arbres pour vérifier la bonne formation de fichier type XML. Ces automates d'arbres seront générés automatiquement à partir d'un fichier *à la* DTD donné en entrée du programme. Celui-ci sera donc parsé puis compilé dans notre type.

1 Validation des documents

Section sans questions. Simple présentation du sujet, introduction des définitions et du travail attendu.

2 Validation top-down non-déterministe

2.1 Question

Un run d'un automate d'arbre $A = (Q, \delta, I, F, \Sigma)$ pour un arbre $t \in \mathcal{T}(\Sigma)$ est une fonction $r : dom(t) \to Q$ telle que $\forall p \in dom(t), (t(p), r(p), r(p1), r(p2)) \in \delta$. Un run est dit acceptant si et seulement si $r(\epsilon) \in I$.

2.2 Question

```
let rec validate_td a t p q =
    if label (t p) = '#' then
        true
```

```
    else
        let transition = label (t p), q in
        let l = List.filter ((=) transition) a.delta in (* modifier ici *)
        List.fold (fun acc q' ->
            acc ||
            (validate_td a t (p@[first_child (t p)]) q' &&
             validate_td a t (p@[next_sibling (t p)]) q')
        ) false l
```

2.3 Question

La complexité de l'expression $\exists$ `q` $\in I$ tel que `validate_td a t eps q` est $O(|\texttt{a}|^{|\texttt{t}|})$, où $|\texttt{a}|$ est le nombre de transitions de l'automate `a`. En effet, l'algorithme nous fait prendre au pire $|\texttt{a}|$ fois chaque arête de l'arbre `t`.

3 Validation bottom-up

3.1 Question

```
let rec validate_bu a t p =
    let lab = label (t p) in
    if lab = '#' then
        List.filter ((=) ('#',[],[]) a.trans in
    else
        let left = validate_bu a t (p@[first_child (t p)]) in
        let right = validate_bu a t (p@[next_sibling (t p)]) in
        let res = ref [] in
        List.iter (fun r -> List.iter (fun l ->
            let trans = lab, l, r in
            let possible_states = List.filter ((=) trans) a.trans in
            res <- possible_states :: !res;
        ) left) right
        res
```

3.2 Question

La complexité de l'expression (`validate_bu a t eps`) $\cap I$ est $O(|\texttt{t}|)$.

3.3 Question

TODO

4 Compilation

4.1 Question

En quelques mots, nous commençons par transformer l'arbre n-aire parsé par la librairie `Xml-light` en un arbre binaire (le code est situé dans le fichier `src/tree.ml`) avec un algorithme simple s'appuyant sur l'isomorphisme de ces deux structures présenté en fin de cours 3. Ainsi les fils directs de chaque nœud deviennent les fils droits du fils gauche de ce nœud et récursivement.

```
let n2bin input_n_tree =
    let rec aux = function
      | [] -> Leaf
      | node :: sibling -> Node (aux node, aux sibling)
    in
    match input_n_tree with
    | rac, childs -> Node (rac, aux childs, Leaf)
```

4.2 Question

L'automate d'arbre produit correspondant au type donné en entrée est de taille ...TODO...