

# Rapport de projet

paul.patault@universite-paris-saclay.fr

## 2. Validation top-down non-déterministe

### Question 1

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

### Question 2

```
let rec validate_td a t p q =
  let lab = label p in
  if lab = '#' then
    true
  else
    let l = get_trans a lab q in
    List.fold (fun acc elt ->
      acc || validate_td a t (elt p) q
    ) false l
  (* first_child(p), next_sibling(p) *)
```

### Question 3

La complexité de l'expression

$$\exists q \in I \text{ tel que } \text{validate\_td } a \ t \ \text{eps } q$$

est  $O(|a|^{|t|})$ . En effet, l'algorithme nous fait prendre  $|a|$  fois chaque arête de l'arbre  $t$ .

## 3. Validation bottom-up

### Question 1

### Question 2

### Question 3

## 4. Compilation

### Question 1

### Question 2