Masters Algorithm

chris.rossouw5

March 2021

1 Positive Algorithm

Algorithm 1: Automaton Coverage

---

input : An LR Graph $LR_G = (V, E, v_0, v_{acc})$
output : A test suite covering all edges in LR_G

$reduction\_path(u \dashrightarrow_{A/|\gamma|} v) = \iota(p \in E_{\dashrightarrow}^{|\gamma|}, vert(p) = v \ldots u) \circ (u \dashrightarrow_{A/|\gamma|} v) \circ \iota(v \rightarrow_A v')$
$embed(red\_path) = complete\_path \mid complete\_path = v_0 \ldots \circ p \circ \ldots v_{acc}, complete\_path\ is\ shallowest\ imbedding\ of\ red\_path$
$test\_suite \leftarrow \varnothing$
for $e \in E_{\dashrightarrow}$ **do**
  $complete = embed(reduction\_path(e))$
  $test\_suite.\cup \{wp(complete)\}$
end
return $test\_suite$

---

2 Negative Stack Mutations

We define equivalent stack sequences as $wp(p) = wp(q) \implies vert(p)\ is\ equivalent\ to\ vert(q)$ for reduction paths p and q.

2.1 Deletion

Let s be a reduction path such that $pre \cup s \cup post$ is a valid path over an automaton graph LR_G.

We may delete s if $vert(post)$ is not equivalent to the stack sequences corresponding to any reduction paths originating from the last vertex in pre

2.2 Insertion

We may insert a stack sequence $vert(s)$ of a reduction path s after any node v such that $u \rightarrow_A v$ is the last edge visited before v and $vert(s)$ is not equivalent to a stack sequence corresponding to any reduction path originating from v.

2.3 Substitution

We may substitute a reduction path $p|vert(p) = u...v$ by a reduction path s if $vert(s)$ is not equivalent to any stack sequences corresponding to reduction paths originating from u