

---

step(currentBsState)

---

```
1: currentContext  $\leftarrow$  context[currentBsState]
2: for each state in states[currentContext] do
3:   tran  $\leftarrow$  trOut[state]
4:   while tran  $\neq$  NIL do
5:     if isEventPresent(currentContext, tran) and isBufferFree(currentContext, tran) then
6:       newContext  $\leftarrow$  createNewContext(currentContext, tran)
7:       newTransition  $\leftarrow$  createNewTransition(currentContext, tran)
8:       item  $\leftarrow$  contextSearch(newContext, ctHashMap)
9:       if item  $\neq$  NIL then
10:        destinationBsState  $\leftarrow$  subValue[item]
11:        dest[newTransition]  $\leftarrow$  destinationBsState
12:       else
13:        createNewState(newContext, tran)
14:        step(destinationBsState)
15:       end if
16:     end if
17:   end while
18: end for
```

---

---

takeEventFromBuffer(*context*, *action*)

---

```
1: l  $\leftarrow$  link[action]
2: pos  $\leftarrow$  index[l]
3: return buffer[context][pos]
```

---

---

isEventPresent(*context*, *transition*)

---

```
1: actionRequest  $\leftarrow$  actIn[transition]
2: eventBuffer  $\leftarrow$  takeEventFromBuffer(context, actionRequest)
3: eventRequest  $\leftarrow$  event[actionRequest]
4: return (actionRequest = NIL or eventBuffer = eventRequest)
```

---

---

isBufferFree(context,transition)

---

```
1: actionProduced  $\leftarrow$  actOut[transition]
2: while actionProduced  $\neq$  NIL do
3:   eventBuffer  $\leftarrow$  takeEventFromBuffer(context,actionProduced)
4:   if eventBuffer  $\neq$  NIL then
5:     return FALSE
6:   end if
7:   actionProduced  $\leftarrow$  next[actionProduced]
8: end while
9: return TRUE
```

---

---

createNewContext(context,transition)

---

```
1: newContext  $\leftarrow$  initializeContext()
2: state  $\leftarrow$  dest[transition]
3: actionRequest  $\leftarrow$  actIn[transition]
4: eventRequest  $\leftarrow$  event[actionRequest]
5: if eventRequest  $\neq$  NIL then
6:   eventBuffer  $\leftarrow$  NIL
7: end if
8: actionProduced  $\leftarrow$  actOut[tran]
9: while actionProduced  $\neq$  NIL do
10:  l2  $\leftarrow$  link[actionProduced]
11:  pos2  $\leftarrow$  index[l2]
12:  buffer[newContext][pos2]  $\leftarrow$  actionProduced
13:  actionProduced  $\leftarrow$  next[actionProduced]
14: end while
15: return newContext
```

---

---

createNewTransition(context,transition,currentBsState)

---

```
1: newTransition  $\leftarrow$  initializeTransition()
2: obs[newTransition]  $\leftarrow$  obs[transition]
3: rel[newTransition]  $\leftarrow$  rel[transition]
4: src[newTransition]  $\leftarrow$  currentBsState
5: addTransition(newTransition)
6: return newTransition
```

---

---

createNewState(context,transition)

---

```
1: destinationBsState  $\leftarrow$  initializeState()
2: context[destinationBsState]  $\leftarrow$  context
3: if isFinal(context) then
4:   final[destinationBsState]  $\leftarrow$  TRUE
5: else
6:   finale[destinationBsState]  $\leftarrow$  FALSE
7: end if
8: addState(destinationBsState)
9: dest[transition]  $\leftarrow$  destinationBsState
10: addContextToHashMap(context)
```

---

---

dfs(state)

---

```
1: color[state] ← GRAY
2: transitionsIncoming ← trIn[state]
3: while transitionsIncoming ≠ NIL do
4:   stateSource ← scr[transitionsIncoming]
5:   if color[stateSource] = WHITE then
6:     dfs[stateSource]
7:   end if
8:   transitionsIncoming ← next[transitionsIncoming]
9: end while
10: color[state] ← BLACK
```

---

---

prune(network)

---

```
1: {consideriamo solo reti con un solo automa, infatti il pruning avviene solo sulle
2:  network generate dal sistema}
3: autom ← automatons[network]
4: totalState ← states[autom]
5: while totalState ≠ NIL do
6:   if final[totalState] = TRUE then
7:     dfsVisit(totalState)
8:   end if
9:   totalState ← next[totalState]
10: end while
11: while totalStat ≠ NIL do
12:   if color[totalState] = WHITE then
13:     removeTheState(autom, totalState)
14:   end if
15:   totalState ← next[totalState]
16: end while
```

---

---

isTransitionObservable(context, transition)

---

```
1: label ← NIL
2: currentObservation ← currentObs[context]
3: {Controllo sulla presenza della lista di osservazioni}
4: if currentObservation ≠ NIL then
5:   label ← currentObervation
6:   transitionLabel ← obs[transition]
7:   idObsarvation ← id[transitionLabel]
8: end if
9: if transitionLabel ≠ NIL and (label ≠ NIL or idLabel ≠ idObsarvation) then
10:  return FALSE
11: else
12:  return TRUE
13: end if
```

---