**Ring-Based Distributed Election**

(Formalized Specification)

**Pi::E**

**{SYNOPSIS}**

Basic variant of the algorithm for distributed election over logical communication ring.

**{ASSUMPTIONS}**

No failures of any kind (neither processes nor channels) are allowed, by condition.

However, the algorithm is not sensitive to failures - in the event of a failure and subsequent recovery, the algorithm restarts and the coordinator is established again. Moreover, both requirements for this class of distributed algorithms are observed - E1 (safety) and E2 (feasibility, liveness).

Multiple concurrent elections are allowed.

**ALGORITHM 1**: Declarative Part of *Pi::E*

**{SYSTEM CONSTANTS}**

String MSG\_ELECTION // type of message “election”

String MSG\_ELECTED // type of message “elected”

PID i // process identifier; 0 ≤ i < n

**{MESSAGES}**

<msg\_election, i>

<msg\_elected, i>

**{SET OF STATES}**

<State> := {NON\_PARTICIPANT, PARTICIPANT}

**{INTERNAL STATE SPACE}**

State state // current process E state

PID pidCoordinator // “coordinator” process identifier

**ALGORITHM 2**: Event Handlers of *Pi::E*

**OnInit:**

state := NON\_PARTICIPANT

pidCoordinator := NULL

**OnStart:**

state := PARTICIPANT

pidCoordinator := NULL

Send <msg\_election, i>

**OnReceiptOf <msg\_election, j>:**

**If** j > i

state := PARTICIPANT

pidCoordinator := NULL

Send <msg\_election, j>

**Else If** j < i

**If** state = NON\_PARTICIPANT

state := PARTICIPANT

pidCoordinator := NULL

Send <msg\_election, i>

**End If**

**Else**

state := NON\_PARTICIPANT

pidCoordinator := i

Send <msg\_elected, i>

**End If**

**OnReceiptOf <msg\_elected, j>:**

**If** j ≠ i

state := NON\_PARTICIPANT

pidCoordinator := j

Send <msg\_elected, j>

*{Local End}*

ME::OnAfterElection()

**Else**

*{Global End}*

ME::OnAfterElection()

**End If**