**Rules**

1. When active, machine(i+1) keeps the token; when passive, it hands over the token to machine(i)
2. A machine sending a message makes itself black
   1. A machine receiving a message becomes active
3. When machine(i+1) propagates the probe, it hands over a black token to machine(i) if it is black itself, whereas while being white it hands over the color of the token unchanged
4. After the completion of an unsuccessful probe, machine(0) initiates a next probe.
5. Machine(0) initiates a probe by making itself white and sending a white token to machine(i-1)
6. Upon transmission of the token to machine(i), machine(i+1) becomes white. (Note that its original color may have influence the color of the token.)

**Vocabulary**

Color = {black, white}

Token = {blackToken, whiteToken, noToken}

Machine

Computation

color : Machine X Computation → Color

token : Machine X Computation → Token

terminated : Computation → Boolean

static next : Machine → Machine

monitored isActive : Machine X Computation → Boolean

monitored blackTokenEvent, whiteTokenEvent, sendMessageEvent

**Model**

ReactOnEvents( m : Machine , c : Computation ) ≡

if blackTokenEvent(m,c) then

token(m,c) := blackToken

if whiteTokenEvent(m,c) then

token(m,c) := whiteToken

if sendMachineEvent(m,c) then

color(m,c) := black

InitializeMachine ( m : Machine, c : Computation ) ≡

token(m,c) := noToken

color(m,c) := white

RegularMachineProgram ( m : Machine ) ≡

(∀ c ∈ Computation with ¬terminated(c) )

ReactOnEvents(m,c)

if ¬isActive(m,c) ∧ ¬token(m,c)=noToken

InitializeMachine(m,c)

if color(m,c) = black

ForwardToken(m, blackToken, nextMachine(m), c)

else if color(m,c) = white

ForwardToken(m, token(m,c), nextMachine(m), c)

SupervisorMachineProgram ( m : Machine ) ≡

(∀ c ∈ Computation with ¬terminated(c) )

ReactOnEvents(m,c)

if ¬isActive(m,c) ∧ ¬token(m,c)=noToken

terminated(c) := true

if (terminated(c) for ∀ c ∈ Computation)

ReportGlobalTermination

else

InitializeMachine(m,c)

ForwardToken(m, whiteToken, nextMachine(m), c)

Initial State ≡

(∀ c ∈ Computation)

terminated(c) := false

(∃ machine0 ∈ Machine) (program(machine0) = SupervisorMachineProgram) ∧ token(machine0) = blackToken) ∧

(∀ m ∈ Machine) (m≠machine0 ⇒ program(m) = RegularMachineProgram)

(∀ m ∈ Machine) (color(m) = white)

**Things Left Abstract**

* blackTokenEvent
* whiteTokenEvent
* sendMessageEvent
* ForwardToken
* ReportGlobalTermination
* Computation processing

**Assumptions**

* Passing tokens, sending messages, becoming active/passive happen immediately (each event incurs no lag time)
* Computations take random amounts of time to complete
* Machines send messages with random probability
* All machines are initially active
* No concept of data concurrency control
* Each token is held by only one machine at a time at any given instant