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

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)