**Distributed Termination Detection  
CoreASM Specification**

Darius Yao  
Dominic Renaud

Table of Contents

[1. List of Vocabulary 3](#_Toc321217060)

[2. Machine Configurations and Initial States 4](#_Toc321217061)

[3. Event Mechanisms and Assumptions 4](#_Toc321217062)

[3.1. Starting the Process 4](#_Toc321217063)

[3.2. Transferring the Token 4](#_Toc321217064)

[3.3. Receiving the Token 4](#_Toc321217065)

[3.4. Waking Up Another Machine 5](#_Toc321217066)

[3.5. Changing a machine colour to black 5](#_Toc321217067)

[3.6. Putting a machine to sleep 5](#_Toc321217068)

[3.7. Global Termination 5](#_Toc321217069)

[4. Testing the CoreASM Model and CoreASM Pitfalls 5](#_Toc321217070)

[5. Validation Argument 6](#_Toc321217071)

[6. Expanding the Algorithm for Multiple Computations 6](#_Toc321217072)

# 1. List of Vocabulary

Universes:

Agent m0: The supervisor machine for termination detection.

Agents m1 to m8: A cluster of eight machines that computations are run on.

Computation c1 to c3: Three separate computations that are run on the machine cluster.

Enumerations:

Colour: The colour of a machine, either black or white.

Token: The colour of the token currently held by the machine, either black or white.

Status: The status of a machine, active or asleep.

Functions:

colour: When provided with an agent and a computation, it returns the colour of the machine.

token: When provided with an agent and a computation, it returns the colour of the machine’s token.

Status: When provided with an agent and a computation, it returns the status of the machine.

number: When provided with an agent, it returns the machine’s number. When a machine wakes another machine up, this number is used to determine whether the current machine has woken up a machine before or after it.

BlackTokenEvent: Given an agent and a computation, it returns a boolean value. If it is true, a black token will be created on the machine.

WhiteTokenEvent: Given an agent and a computation, it returns a boolean value. If it is true, a white token will be created on the machine.

# 2. Machine Configurations and Initial States

In our ASM model, our machines are connected in a circular ring in descending order. Machine 8 is followed by machine 7, machine 7 is followed by machine 6, and so on. Machine 0 is followed by machine 8. In the initial state, all machines have the following properties:

* The machine is active
* The machine is white colour
* The token is undefined (all machines do not possess a token)
* BlackTokenEvent and WhiteTokenEvent are both false

# 3. Event Mechanisms and Assumptions

## 3.1. Starting the Process

When the supervisor machine is active, it creates a white token and sends it out for a probing test. After this initial token pass, it falls asleep and waits for the token to return.

## 3.2. Transferring the Token

When a machine has fallen asleep and the token needs to be passed on to the next machine, the colour of the machine is examined. If the machine is black, the colour of the token is changed to black, otherwise the machine is white, and the colour of the token is unchanged. The token is then passed to the rule ForwardToken. Forward Token inspects the colour of the token and generates either a BlackTokenEvent or WhiteTokenEvent for the next machine. After the token event is created, the machine’s current token is discarded, and the machine reverts to the white colour.

## 3.3. Receiving the Token

When a machine does not possess a token, it repeatedly calls the rule ReactToEvents. ReactToEvents checks the Boolean values of BlackTokenEvent and WhiteTokenEvent. If the former is true, a black token will be created on the machine. If the latter is true, a white token will be created. Note that both events cannot be true at the same time. After receiving a token, both BlackTokenEvent and WhiteTokenEvent are set to false, which prevent the machine from receiving tokens multiple times.

## 3.4. Waking Up Another Machine

As long as a machine is active, it has a probability of waking up another machine. We have set the probability of wake up to 30%, and the machine to be woken up is chosen at random. We make an assumption that the supervisor machine cannot be woken up.

## 3.5. Changing a machine colour to black

Because the machines are connected in descending order, if a machine wakes up another machine that has a larger number, then it indicates that a previous machine has awoken. The current machine’s colour will be changed to black as a result.

## 3.6. Putting a machine to sleep

After attempting to wake another machine up (see section 3.3), machines are set to have a 32% chance to fall asleep. If the probability fails, the machine program will loop. This loop allows one machine to possibly wake up multiple machines, which simulate the real situation more accurately.

## 3.7. Global Termination

Global termination status is reached when the supervisor machine receives a white token. Otherwise, the supervisor machine creates another white token and the token probing process is restarted.

# 4. Testing the CoreASM Model and CoreASM Pitfalls

The ASM model does not take inputs; therefore to change the configuration the code must be modified. Inside the initRule block, documentation is included to explain how to change the machine ring configuration and the probability changes for machine wake up and sleep.

Due to the way the random function works in CoreASM, the program will loop infinitely if the sleep chance is lower than the wake up chance. This occurs because the function random generates the same number for both the ‘random < wakeUpChance’ and the ‘random < sleepChance’ statements. If the sleep chance is less than the wake up chance, then it follows that the machine will only ever go to sleep when it also wakes up another machine. If the machine is the lowest in number (i.e. last in the sequence), the machine it wakes up must be larger in number and therefore m1 will turn black. This will cause the probing process to restart.

# 5. Validation Argument

This model is based on the termination detection algorithm for distributed computations by Dijkstra, Feijen, and van Gasteren. The aim is to detect whether a cluster of distributed machines have all finished its computations and have terminated, thereby reaching a global termination state.

The primitive version is to introduce a supervisor machine, m0, which injects a token into the cluster of machines. A machine holds onto the token as long as the machine is active. The machine can only pass the token to another machine when it has reached a sleep state. If the token has passed through every machine in the cluster, then it follows that all the machines have reached a sleep state and global termination is achieved.

However, in real situations, an active machine is able to send a message that prompts another machine to wake up. This creates two scenarios:

1. If a machine wakes up another machine that has not yet been traversed by the token, then the token is still valid.
2. If a machine wakes up another machine that has already been traversed by the token, then there is a contradiction. A machine the token has determined to be asleep is now awake.

To deal with the second situation, a special flag is asserted to indicate an invalid token. One strategy is to introduce a colour scheme. All machines and the token initially start with the white colour. When a machine generates a wake up signal with case 2’s conditions above, it changes to the black colour. When the token reaches this black machine, the token’s colour changes to black and becomes invalid. Global termination is now achieved when the token traverses all the machines while staying white in colour.

# 6. Expanding the Algorithm for Multiple Computations

Multiple computations can be handled by expanding our data space. If we run ‘n’ computations on our cluster of machines, each machine will be running ‘n’ processes (one for each computation). Each machine can utilize an array to maintain information on each of its processes. Each element of the array will contain the status (active or asleep) and the colour (black or white) of the process. Our supervisor machine will need to send out ‘n’ tokens to verify termination for each of the ‘n’ computations taking place. The termination of one computation will be independent of each other because each computation has its own token probe.

We had difficulty creating the data structures necessary to resemble this model in CoreASM, so we can only leave a description of the model we had envisioned.