

Assignment #6: Termination Detection

DUE: Friday, May 25th

Introduction

Consider the termination detection problem as discussed in chapter 10 of the notes. In this question, you are given 2 new algorithms to solve this problem.

Your task is to determine whether each algorithm is correct. If the algorithm is correct, prove its correctness. If it is incorrect, give an example of how it can fail.

Background: A Correct Algorithm

There is a separate *detector* process, different from the other processes of the network. For convenience, we refer to processes of the network as *processes* and to the detector as *the detector*.

There are first-come-first-served channels from each process to the detector. A process sends a message to the detector when the process transits from active to idle. The message from the process to the detector contains the number of messages received by the process on each input channel, and the number of messages sent by the process on each output channel, when the process becomes idle.

For example, a process with incoming channels d and e , and outgoing channels f and g , sends a message of the form

$$\{id, inputs\{(d, r.d), (e, r.e)\}, outputs\{(f, s.f), (g, s.g)\}\}$$

where id is the unique identifier of this process, $r.d$ and $r.e$ are the numbers of messages received on these channels, and $s.f$ and $s.g$ are the numbers of messages sent on these channels.

The detector keeps integer variables $in.c$ and $out.c$ associated with each channel c . Initially, for all channels c , $in.c = 0$ and $out.c = 0$. When the detector receives a message

$$\{id, inputs\{(d, n_1), (e, n_2)\}, outputs\{(f, n_3), (g, n_4)\}\}$$

it sets $in.d = n_1$, $in.e = n_2$, $out.f = n_3$, and $out.g = n_4$.

The detector sets a variable **done** to TRUE when it has received a message from all processes and for all channels c : $in.c = out.c$. Initially, **done** is FALSE. The detector claims that the computation has terminated if **done** holds.

This algorithm is correct and is covered in the notes.

An Incorrect Algorithm

A process sends a message to the detector when the process transits from active to idle. The message from the process to the detector contains the *total* number of messages received by the process, and the *total* number of messages sent by the process, when the process becomes idle.

That is, a process p sends a message of the form

$$\{id, total_rcvd.p, total_sent.p\}$$

where $total_rcvd.p$ and $total_sent.p$ are the total numbers of messages received and sent by process p respectively.

The detector keeps integer variables $in.p$ and $out.p$ associated with each process p . Initially, for all processes p , $in.p = 0$ and $out.p = 0$. When the detector receives a message

$$\{id, total_rcvd.p, total_sent.p\}$$

it sets $in.p = total_rcvd.p$, and $out.p = total_sent.p$.

The detector sets a variable **done** to TRUE when it has received a message from all processes and $\sum_p in.p = \sum_p out.p$. Initially, **done** is FALSE. The detector claims that the computation has terminated if **done** holds.

This algorithm is more efficient than the previous one, because the messages sent to the detector are much shorter. Unfortunately, it is incorrect. The detector can claim the computation has terminated when in fact this is not the case.

The Problem

This problem consists of considering two alternative algorithms for detecting termination in the given system. In each case, you will be presented with an algorithm, and asked if it is correct. If it is correct, you are to prove the algorithm. If it is incorrect, you are to give a counterexample that demonstrates the algorithm's failure. (If you can demonstrate the algorithm's failure, *there is no need to correct the algorithm*).

Consider a hybrid of the two algorithms discussed above. Suppose the message sent to the detector consists of the *total* number of messages received, as well as the number of messages *sent to each process*.

For example, a process p with outgoing channels to processes q and r sends a message of the form

$$\{id, total_rcvd.p, outputs\{(q, s(p, q)), (r, s(p, r))\}\}$$

where $total_rcvd.p$ is the total number of messages received by process p , and $s(p, q)$, and $s(p, r)$ are the numbers of messages sent on channels from p to processes q and r respectively.

The detector keeps an integer variable $in.p$ associated with each process p and integer variable $out(p, q)$ associated with each pair of process p, q . Initially, for all processes p, q , $in.p = 0$ and $out(p, q) = 0$. When the detector receives a message

$$\{id, total_rcvd.p, outputs\{(q, s(p, q)), (r, s(p, r))\}\}$$

it sets $in.p = total_rcvd.p$, $out(p, q) = s(p, q)$, and $out(p, r) = s(p, r)$.

The detector sets a variable **done** to **TRUE** when it has received a message from each process and for all processes p : $in.p = \sum_q out(q, p)$. Initially, **done** is **FALSE**. The detector claims that the computation has terminated if **done** holds.

PART I

For part I of this question, answer the following.

1. Is this algorithm correct? Answer yes or no.
2. If the algorithm is correct, prove the correctness of the algorithm. If the algorithm is incorrect, give a counterexample.

PART II

Consider the analogous hybrid, where each process sends to the detector (when it becomes idle) the total number of messages it has *sent*, and the number of messages it has *received from each process*. The detector keeps integer variables $out.p$ and $in(p, q)$, where $out.p$ is the total number of messages sent by process p , and $in(p, q)$ is the number of messages received on the channel from p to q (*i.e.*, messages received by process q from process p). The detector sets **done** to **TRUE** when it has received a message from each process and for all processes p : $out.p = \sum_q in(p, q)$. Initially, **done** is **FALSE**. The detector claims that the computation has terminated if **done** holds.

For part II of this question, answer the following.

1. Is this algorithm correct? Answer yes or no.
2. If the algorithm is correct, prove the correctness of the algorithm. If the algorithm is incorrect, give a counterexample.