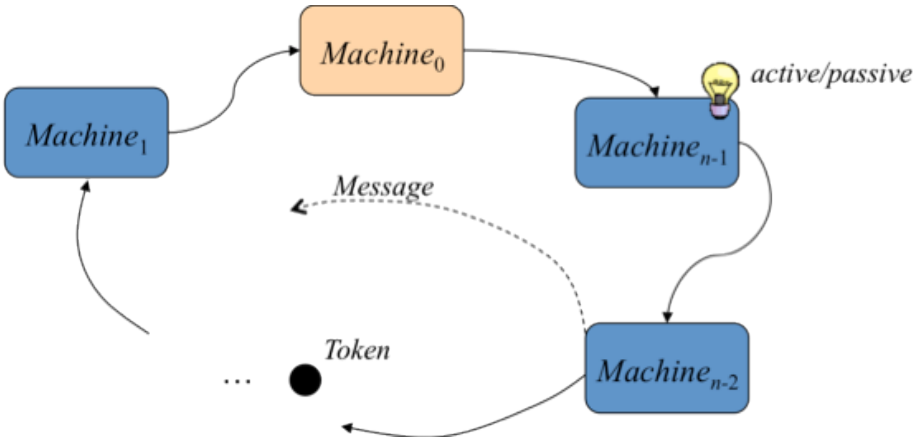
Due Date: At the START of class on Wednesday, November 28, 2012.

DISTRIBUTED TERMINATION DETECTION



Problem Description

In this project we consider a generalization of Dijkstra's original protocol for distributed termination detection of computations running on a cluster of machines, which communicate asynchronously by sending messages over some network interconnecting the machines. The original protocol proposed by Dijkstra, as described in [1] and discussed in class, is meant to detect the stable state of global termination for a single distributed computation running on a cluster consisting of n machines. In contrast, the version we consider here will allow to detect the stable state of global termination for any number $k \geq 1$ of independent distributed computations running concurrently (in any possible partial order) on the same cluster consisting of n machines. That is, for each of the k computations the termination of this computation will be detected independent of the state of any other computation running on the same cluster of machines. For simplicity, we may assume here that n is a fixed parameter of the protocol, although the protocol should in principle work for any number n of machines.

Logical Organization

1. In a first step, abstractly **model** the functional requirements that define the generalized version of the distributed termination detection protocol in terms of an Abstract State Machine model formally specifying the following aspects:
   - the *vocabulary* introducing the name and type of all universes (domains), functions and predicates used in your model,
   - the *machine program* for each type of computational agent (i.e., ASM agent),
   - any *assumptions* you make about interactions with the operational system environment, any
   - any *assumptions* you make about the initial machine state.

2. In a second step, **refine** your ASM model of the generalized distributed termination detection protocol as necessary for deriving an executable prototype described in terms of a CoreASM model. For testing your CoreASM model, provide a small collection of **sample inputs** as test cases to show that the model runs as expected with at least $k \geq 4$ independent distributed computations concurrently running on a network with a reasonable number n of machines, for some $n \geq 8$.
3. In a third step, **analyze** your model and provide a chain of reasoning (amounting to an informal proof) that your protocol indeed detects the global state of termination independently for each distributed computation running on the cluster. The description should be concise and conclusive.
4. Finally, identify and briefly describe at least one *safety property* and at least one *liveness property* of the protocol you have designed.

Basic Idea

While there may be more than one way of how to solve the problem in general, the basic idea for extending the original protocol so as to meet the new requirements is relatively simple and straightforward. Think about how to represent and distinguish multiple termination detection probes, one for each individual computation running on the machine cluster. For the ASM model you may assume a level of abstraction directly comparable to the version of the protocol discussed in class. By doing so, the description of the resulting protocol should only be insignificantly more complex.

Solution

A good solution for this assignment will be a comprehensive **technical description** that, besides correctness and completeness of the formal specification, also bears in mind the following aspects and provides:

- An ASM model of the protocol that complies to the ASM syntax and semantics as illustrated in class;
- A vocabulary providing a list of declarations for the sets, functions and predicates used in the model, clearly indicating which functions/predicates you consider monitored functions/predicates;
- A proper definition of the initial state insofar as it matters for the operation of the protocol; for instance, for the executable version this also includes the definition of your network in addition to the initial configuration of the machines;
- A concise and clear description of the event mechanisms used in your protocol, specifically, what assumptions do you make how actions trigger events both locally on any of the machines and globally over the distributed network environment;
- A clear and conclusive argumentation why you think you protocol works correctly;
- A brief explanation for how to test the executable version of your protocol, including the test data and test environment (such as the sample network) used so that one can easily repeat your experiments;
- A printed hard copy with the description of your solution (Steps 1-4);
- An executable CoreASM file of the rapid prototype that also includes the collection of test cases.
- Consult with the TA about further specifics regarding the format of the executable model.

We also ask you to submit an electronic copy of your course project (other than the CoreASM code) to TurnItIn.com under "Course Project: Termination Detection".

Reference

Edsger W. Dijkstra et al. Derivation of a Termination Detection Algorithm for Distributed Computations. *Information Processing Letters 16*: 217–219.