Automaton Description

Signatures

Symbols:

- $B = \{\mathsf{True}, \mathsf{False}\}$ boolean values;
- $\mathbf{N}$ natural numbers;
- D the domain defining the computation;
- $G = \mathbf{N} \times \mathbf{N}$ unique group numbers;
- Messages domain:

$$
\begin{aligned}
M \quad = \quad & \{\mathsf{Accept}\} \times G \\
& \cup \{\mathsf{AcceptAnswer}\} \times B \\
& \cup \{\mathsf{AreYouCoordinator}\} \\
& \cup \{\mathsf{AreYouCoordinatorAnswer}\} \times B \\
& \cup \{\mathsf{AreYouThere}\} \times G \\
& \cup \{\mathsf{AreYouThereAnswer}\} \times B \\
& \cup \{\mathsf{Invitation}\} \times \mathbf{N} \times G \\
& \cup \{\mathsf{Ready}\} \times \mathbf{N} \times D \\
& \cup \{\mathsf{ReadyAnswer}\} \times B \times G;
\end{aligned}
$$

- $\mathbf{P}(X)$ the set of all subsets of the set X.

Input: $\mathbf{Receive}(m)_{ji}$, $m \in M, j \in neighbors_i$

Output: $\mathbf{Send}(m)_{ij}$, $m \in M, j \in neighbors_i$

Internal: For each i in $neighbors$

$\mathbf{Check}_i$,
$\mathbf{Merge}_i$,
$\mathbf{Recovery}_i$,
$\mathbf{Timeout}_i$

States

Variable	Explanation
$accepted \in B$	used to checks whether the coordinator accepted the current process to be in its group.
$counter \in \mathbf{N}$	used to create a unique sequence number; does not lose its content in case of a failure.
$coordinator \in \mathbf{N}$	coordinator of the process's group.
$definition \in D$	the values for the computation to be done.
$group \in G$	pair (coordinator, group number).
$isCoordinator \in B$	used to check whether the coordinator is still alive.
$numAnswer \in \mathbf{N}$	counts the received ReadyAnswer messages.
$others \in \mathbf{P}(\mathbf{N})$	set of other coordinators in the net.
send	array of FIFO queues for messages; initially empty.
status	contains one of the values DOWN, ELECTION, MERGE, NORMAL, RECOVERY, REORGANIZATION; initially DOWN.
$up \in \mathbf{P}(\mathbf{N})$	set of processes in the group; used only if the process is a coordinator.
$upSet \in \mathbf{P}(\mathbf{N})$	temporarily stores the content of the variable up.

Transitions

1. **Recovery**$_i$: Recover from a failure.

   Precondition:

   $status = \mathsf{DOWN}$ or $status = \mathsf{RECOVERY}$

   Effect:

   $status = \mathsf{ELECTION}$

   $counter := counter + 1$

   $group := (i, counter)$

   $coordinator := i$

   $up := \{\}$

   $status := \mathsf{REORGANIZATION}$

   $definition :=$ Some definition

   $status := \mathsf{NORMAL}$

2. **Check**$_i$: Check for other coordinators.

Precondition:

$status = \mathsf{NORMAL}$

$coordinator = i$

Effect:

$others := \{\}$

for all j in $neighbors \setminus \{i\}$

Add ($\mathsf{AreYouCoordinator}$) to $send_j$.

WAIT for answers

if $others \neq \{\}$

WAIT for a time proportional to i.

$status := \mathsf{MERGE}$

3. **Merge$_i$**: Try to merge the members of other groups into own group.

Precondition:

$status = \mathsf{MERGE}$

$coordinator = i$

Effect:

$status = \mathsf{ELECTION}$

Stop computation.

$counter := counter + 1$

$group := (i, counter)$

$coordinator := i$

$upSet := up$

$up := \{\}$

for all j in $others \cup upSet$

Add $(\mathsf{Invitation}, i, group)$ to $send_j$.

WAIT for answers.

$status = \mathsf{REORGANIZATION}$

$numAnswer := 0$

for all j in up

Add $(\mathsf{Ready}, group, definition)$ to $send_j$.

WAIT for answers.

if $numAnswer < |up|$

$status := \mathsf{RECOVERY}$

else

$status := \mathsf{NORMAL}$

4. **Timeout**$_i$: Test whether the coordinator is still alive.

Precondition:

$status = \mathsf{NORMAL}$

$coordinator \neq i$

Effect:

Add $(\mathsf{AreYouThere}, group)$ to $send_{coordinator}$.

$isCoordinator := \mathsf{False}$

WAIT for answer

if $isCoordinator = \mathsf{False}$

$status := \mathsf{RECOVERY}$

5. **Send**$(m)_{ij}$: Put the message from the FIFO to the channel.

Precondition:

m is first on the FIFO $send_j$

Effect:

Remove head from $send_j$.

6. **Receive**($\mathsf{AreYouCoordinator}$)$_{ji}$: Get request from another coordinator.

Effect:

if ($status = \mathsf{NORMAL}$ and $coordinator = i$)

  Add ($\mathsf{AreYouCoordinatorAnswer}, \mathsf{True}$) to $send_j$.

else

  Add ($\mathsf{AreYouCoordinatorAnswer}, \mathsf{False}$) to $send_j$.

7. **Receive**(($\mathsf{AreYouCoordinatorAnswer}, answer$))$_{ji}$: Get response from neighbors whether or not they are coordinators themselves.

Effect:

if $answer = \mathsf{True}$

  $others := others \cup \{j\}$

8. **Receive**((AreYouThere, $oldGroup$))$_{ji}$: Get request from somebody testing its coordinator.

Effect:

if ($group = oldGroup$ and $coordinator = i$)

Add (AreYouThereAnswer, True) to $send_j$.

else

Add (AreYouThereAnswer, False) to $send_j$.

9. **Receive**((AreYouThereAnswer, $answer$))$_{ji}$: Get response from coordinator whether or not it is still alive or whether something has changed.

Effect:

if $coordinator \neq i$

$isCoordinator := answer$

10. **Receive**$((\mathsf{Invitation}, newCoordinator, newGroup))_{ji}$: Get an invitation to join another group.

Effect:

if $status = \mathsf{NORMAL}$

$status = \mathsf{ELECTION}$

if $coordinator = i$

for all p in up //Forward invitation

Add

$(\mathsf{Invitation}, newCoordinator, newGroup)$

to $send_p$.

$coordinator := newCoordinator$

$group := newGroup$

Add $(\mathsf{Accept}, group)$ to $send_{coordinator}$

$accepted := \mathsf{False}$

WAIT for an answer

if $accepted = \mathsf{True}$

$status = \mathsf{REORGANIZATION}$

else

$status = \mathsf{RECOVERY}$

11. **Receive**$((\mathsf{Accept}, newGroup))_{ji}$: Get message from new candidate of a group that it wants to join the group.

Effect:

if $(status = \mathsf{ELECTION}$ and $coordinator = i$ and $group = newGroup)$

$up := up \cup \{j\}$

Add $(\mathsf{AcceptAnswer}, \mathsf{True})$ to $send_j$.

else

Add $(\mathsf{AcceptAnswer}, \mathsf{False})$ to $send_j$.

12. **Receive**$((\mathsf{AcceptAnswer}, answer))_{ji}$: Get response from new coordinator whether or not it accepted joining its group.

Effect:

if $(status = \mathsf{ELECTION}$ and $coordinator = j)$

$accepted := answer$

13. **Receive**$((\mathsf{Ready}, newGroup, newDefinition))_{ji}$: Get the definitions for the computation.

Effect:

if $(status = \mathsf{REORGANIZATION}$ and $group = newGroup)$

$definition := newDefinition$

$status := \mathsf{NORMAL}$

Add $(\mathsf{ReadyAnswer}, \mathsf{True}, group)$ to $send_j$

else

Add $(\mathsf{ReadyAnswer}, \mathsf{False}, group)$ to $send_j$

14. **Receive**$((\mathsf{ReadyAnswer}, inGroup, newGroup))_{ji}$: Get response from group canditates whether or not they are ready to start the computation.

Effect:

if $inGroup = \mathsf{True}$ and $group = newGroup$

$numAnswer := numAnswer + 1$

Tasks

$\{\mathbf{Check}_i, \mathbf{Merge}_i, \mathbf{Recovery}_i, \mathbf{Timeout}_i\}$

for all j in *neighbors*:

$\{\mathbf{Send}(m)_{ij} \mid m \in M\}$

Correction Assertions

1. For any pair of nodes p_i and p_j, if

   $status_i \in \{\mathsf{NORMAL}, \mathsf{REORGANIZATION}\}$,

   $status_j \in \{\mathsf{NORMAL}, \mathsf{REORGANIZATION}\}$,

   $group_i = group_j$

   then

   $coordinator_i = coordinator_j$.

2. For any pair of nodes p_i and p_j, if

   $status_i = \mathsf{NORMAL}$,

   $status_j = \mathsf{NORMAL}$,

   $group_i = group_j$

   then

   $definition_i = definition_j$.

Let R a maximal connected set of nodes that can communicate in a state s_0. The following two conditions are eventually true in any run starting at s_0 so that R remains connected, and no failures occur:

3. There is a node $p_i \in R$ so that

   $status_i = \mathsf{NORMAL}$ and

   $coordinator_i = p_i$.

4. For every other node $p_j \in R$,

   $status_i = \mathsf{NORMAL}$ and

   $coordinator_j = p_i$.