Chandy-Misra-Haas Algorithm for the OR Model

Chandy-Misra-Haas distributed deadlock detection algorithm for OR model is based on the approach of diffusion-computation.

- A blocked process determines if it is deadlocked by initiating a diffusion computation.
- Two types of messages are used in a diffusion computation:
- query(i, j, k) and reply(i, j, k), denoting that they belong to a diffusion computation initiated by a process P_i and are being sent from process P_j to process P_k.

- A blocked process initiates deadlock detection by sending query messages to all processes in its dependent set.
- If an active process receives a query or reply message, it discards it.
- When a blocked process P_k receives a query(i, j, k) message, it takes the following actions:
  1. If this is the first query message received by P_k for the deadlock detection initiated by P_i (called the *engaging query*), then it propagates the query to all the processes in its dependent set and sets a local variable num_k(i) to the number of query messages sent.
  2. If this is not the engaging query, then P_k returns a reply message to it immediately provided P_k has been continuously blocked since it received the corresponding engaging query. Otherwise, it discards the query.

- Process P_k maintains a boolean variable $wait_k$(i) that denotes the fact that it has been continuously blocked since it received the last engaging query from process P_i.
- When a blocked process P_k receives a reply(i, j, k) message, it decrements num_k(i) only if $wait_k$(i) holds.
- A process sends a reply message in response to an engaging query only after it has received a reply to every query message it had sent out for this engaging query.
- The initiator process detects a deadlock when it receives reply messages to all the query messages it had sent out.

Algorithm

The algorithm works as follows:

Initiate a diffusion computation for a blocked process P_i:

send query(i, i, j) to all processes P_j in the dependent set DS_i of P_i;

num_i(i):= $|DS_i|$; $wait_i$(i):= true;

When a blocked process P_k receives a query(i, j, k):

if this is the engaging query for process P_i

then send query(i, k, m) to all P_m in its dependent set DS_k;

num_k(i): = $|DS_k|$; $wait_k$(i):= true

else if $wait_k$(i) then send a *reply*(i, k, j) to P_j.

When a process P_k receives a reply(i, j, k):

if $wait_k$(i)

then begin

num_k(i):= num_k(i) $-$ 1;

if num_k(i)= 0

then if i=k then **declare a deadlock**

else send reply(i, k, m) to the process P_m which sent the engaging query.

- In practice, several diffusion computations may be initiated for a process (A diffusion computation is initiated every time the process gets blocked), but, at any time only one diffusion computation is current for any process.
- However, messages for outdated diffusion computations may still be in transit.
- The current diffusion computation can be distinguished from outdated ones by using sequence numbers.

Performance Analysis

For every deadlock detection, the algorithm exchanges e query messages and e reply messages, where e=n(n-1) is the number of edges.