**Instituto Tecnológico de Aeronáutica**
**Engenharia Aeroespacial e Mecânica**

# CE-288
# Programação Distribuída

## Professor: Dr. Celso Massaki Hirata

## Resolução da Serie de Exercícios nº3

**Lucas Kriesel Sperotto**

**21 de Novembro de 2011**

# 3º Série Programação Distribuída

1) Liste os valores das variáveis dos processos do algoritmo de anel lógico (J. Misra) para detecção de termino com cinco processos (P0 ... P4). Todos os processos estão descansando e existe uma mensagem em trânsito de P0 a P3. Suponha a seguinte sequência de eventos:
   a) P0 inicia a detecção;
   b) A mensagem (de P0) chega em P3 e P3 fica ativo;
   c) P3 envia mensagem para P2;
   d) A mensagem chega em P2 e P2 descansa.
   Em caso de detecção de término, qual processo que detecta o término?

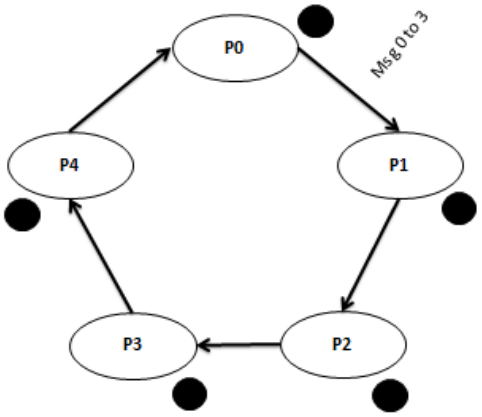



**Figura 1 - Estado Inicial**

| |
|---|
| Inicialmente todos os processos marcados como "**Preto**".<br>P0 envia msg de aplicação para P3;<br>P0 inicia detecção; m=0; P0 = Branco;<br>P1 m = 0; P1 = Branco;<br>P2 m = 0; P2 = Branco;<br>P3 recebe msg de aplicação de P0; P3 = Preto; P3 envia msg de aplicação para P2;<br>P3 m = 0; P3 = Branco;<br>P4 m = 0; P4 = Branco;<br>P0 m = 1;<br>P1 m = 2;<br>P2 recebe msg de aplicação de P3; P2 = Preto;<br>P2 m = 0; P2 = Branco;<br>P3 m = 1;<br>P4 m = 2;<br>P0 m = 3;<br>P1 m = 4;<br>P2 m = 5; **P2 detecta termino da computação**; |

**Quadro 1 – Eventos**

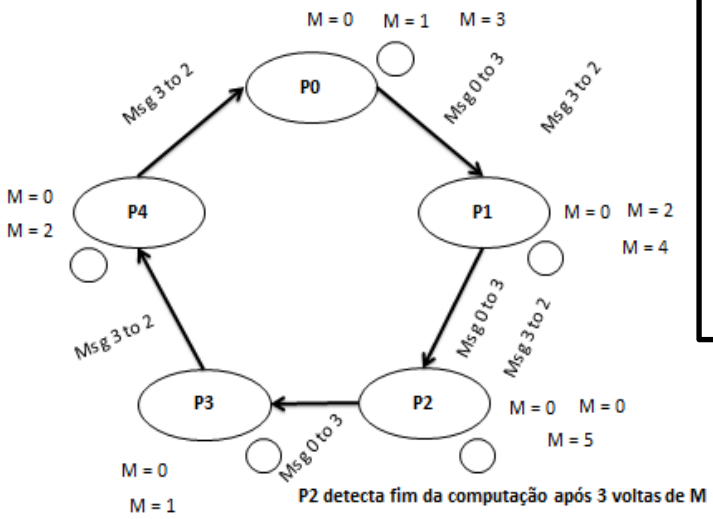



**Figura 2 - Estado Final**

Para a resolução do exercício, **assumi** que o processo **P3 descansa** após enviar mensagem de aplicação para P2. Caso o processo **P3 não descanse**, o token não é enviado por ele e **o termino não seria detectado**. Na figura 1 vemos o estado inicial da execução e na figura 2 o estado final. A descrição dos eventos pode ser visto no quadro 1. O processo que detecta término é P2, dado que ele é o ultimo a dormir.

2) Usando o algoritmo de Lomet (prevenção de deadlock de recurso centralizado), com **quatro** recursos e **quatro** transações e cada transação requer **um** recurso, mas potencialmente pode requerer no máximo **dois** recursos, mostre uma situação onde o recurso não é concedido apesar de disponível. Explique:

Recursos a, b, c, d;
T1: necessidade máxima de a, b;
T2: necessidade máxima de b, c;
T3: necessidade máxima de c, d;
T4: necessidade máxima de d, a;
Escalonamento: T1(a, b), T2(b, c), T3(c, d), T4(d, a), O(1, a), O(2, b), O(3, c), O(4, d);

| | |
|---|---|
| O(1, a): | T1 -> T4; |
| O(1, a), O(2, b): | T2 -> T1 -> T4; |
| O(1, a), O(2, b), O(3, c): | T3 -> T2 -> T1 -> T4; |
| O(1, a), O(2, b), O(3, c), O(4, d): | T4 -> T3 -> T2 -> T1 -> T4; |

Ocorrência de ciclo, o recurso **d** esta disponível para **T4**, entretanto ele adicionaria um arco de potencialmente bloqueando para T3, já que T3 pode precisar de d, o que geraria um ciclo e o recurso não é concedido.

3) Usando o algoritmo para prevenção de deadlock de recurso distribuído, com **quatro** recursos e **quatro** transações e cada transação requer **um** recurso, mas potencialmente pode requerer no máximo **dois** recursos:
a) mostre uma situação onde um recurso não é concedido apesar de disponível;
b) Qual deveria ser a ordem de execução das transações para que ocorresse o maior número de concessões de recursos?

Recursos a, b, c, d;
T1: necessidade máxima de a, b;
T2: necessidade máxima de b, c;
T3: necessidade máxima de c, d;
T4: necessidade máxima de d, a;
Rotulo de Tempo: T1 => T2 => T3 => T4;
Escalonamento: T1(a, b), T2(b, c), T3(c, d), T4(d, a), O(1, a), O(2, b), O(3, c), O(4, d);

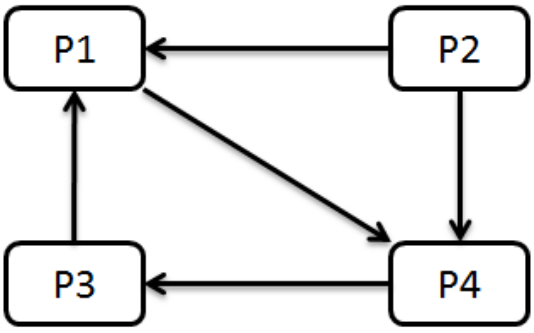
| | |
|---|---|
| Sa: T1 => T4 | T1 -> T4 por O(1, a); **concedido** |
| Sb: T1 => T2 | T2 -> T1 por O(2, b); **não concedido** |
| Sc: T2 => T3 | T3 -> T2 por O(3, c); **não concedido** |
| Sd: T3 => T4 | T4 -> T3 por O(4, d); **não concedido** |

Em **Sb** o recurso está disponível, mas ocorre um deadlock por rótulo de tempo.
Rotulo de Tempo: T1 => T4 => T3 => T2;
Escalonamento: T1(a, b), T2(b, c), T3(c, d), T4(d, a), O(1, a), O(4, d), O(3, c), O(2, b);

| | |
|---|---|
| Sa: T1 => T4 | T1 -> T4 por O(1, a); **concedido** |
| Sb: T1 => T2 | T2 -> T1 por O(2, b**); não concedido** |
| Sc: T3 => T2 | T3 -> T2 por O(3, c); **concedido** |
| Sd: T4 => T3 | T4 -> T3 por O(4, d); **concedido** |

Com essa ordem recurso não é concedido apenas em **Sb**.

4) Usando o algoritmo de detecção de deadlock de comunicação de Chandy, Misra & Haas, dê um exemplo de execução com os quatro processos P1, P2, P3 e P4 em deadlock de comunicação. Suponha que os processos P3 e P4 iniciam a detecção. Mostre os valores das variáveis de trabalho.




**Seta indica espera por mensagem.**

**Todos os processos estão descansando – aguardando por mensagem.**
**P1** espera por **P4;**
**P2** espera por **P1** e **P4;**
**P3** espera por **P1;**
**P4** espera por **P3;**
**P3** e **P4** iniciam detecção**;**

| Tempo | Ação |
|---|---|
| 1 | P4 inicia sua primeira detecção |
| | P3 inicia sua primeira detecção |
| | P4 envia query (4,1,4,3) |
| | P3 envia query (3,1,3,1) |
| 2 | P3 recebe query(4,1,4,3) |
| | P3 envia query (4,1,3,1) |
| | P1 recebe query(3,1,3,1) |
| | P1 envia query(3,1,1,4) |
| 3 | P4 recebe query (3,1,1,4) |
| | P4 envia query (3,1,4,3) |
| | P1 recebe query(4,1,3,1) |
| | P1 envia query(4,1,1,4) |
| 4 | P3 recebe query(3,1,4,3) |
| | P3 envia reply(3,1,3,4) |
| | P4 recebe query (4,1,1,4) |
| | P4 envia reply(4,1,4,1) |
| 5 | P4 recebe reply(3,1,3,4) |
| | P4 envia reply(3,1,4,1) |
| | P1 recebe reply(4,1,4,1) |
| | P1 envia reply(4,1,1,3) |
| 6 | P1 recebe reply(3,1,4,1) |
| | P1 envia reply(3,1,1,3) |
| | P3 recebe reply(4,1,1,3) |
| | P3 envia reply(4,1,3,4) |
| 7 | P3 recebe reply(3,1,1,3) |
| | **P3 declara Deadlock** |
| | P4 recebe reply(4,1,3,4) |
| | **P4 declara Deadlock** |

Temos na tabela ao lado as trocas de mensagens entre os processos até a detecção de Deadlock.

Na figura abaixo, temos os valores nas variáveis em cada um dos processos para cada instante de tempo. No tempo 4 não houve mudança nas variáveis e não foi mostrado.

**Tempo 0**

| | P1 | | | | P2 | | | | P3 | | | | P4 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Latest(i) | engager(i) | num(i) | wait(i) | Latest(i) | engager(i) | num(i) | wait(i) | Latest(i) | engager(i) | num(i) | wait(i) | Latest(i) | engager(i) | num(i) | wait(i) |
| 1 | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F |
| 2 | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F |
| 3 | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F |
| 4 | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F |

**Tempo 1**

| | P1 | | | | P2 | | | | P3 | | | | P4 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Latest(i) | engager(i) | num(i) | wait(i) | Latest(i) | engager(i) | num(i) | wait(i) | Latest(i) | engager(i) | num(i) | wait(i) | Latest(i) | engager(i) | num(i) | wait(i) |
| 1 | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F |
| 2 | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F |
| 3 | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 1 | 0 | 1 | T | 0 | 0 | 0 | F |
| 4 | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 1 | 0 | 2 | T |

**Tempo 2**

| | P1 | | | | P2 | | | | P3 | | | | P4 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Latest(i) | engager(i) | num(i) | wait(i) | Latest(i) | engager(i) | num(i) | wait(i) | Latest(i) | engager(i) | num(i) | wait(i) | Latest(i) | engager(i) | num(i) | wait(i) |
| 1 | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F |
| 2 | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F |
| 3 | 1 | 3 | 0 | T | 0 | 0 | 0 | F | 1 | 0 | 1 | T | 0 | 0 | 0 | F |
| 4 | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 1 | 4 | 1 | T | 1 | 0 | 1 | T |

**Tempo 3**

| | P1 | | | | P2 | | | | P3 | | | | P4 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Latest(i) | engager(i) | num(i) | wait(i) | Latest(i) | engager(i) | num(i) | wait(i) | Latest(i) | engager(i) | num(i) | wait(i) | Latest(i) | engager(i) | num(i) | wait(i) |
| 1 | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F |
| 2 | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F |
| 3 | 1 | 3 | 1 | T | 0 | 0 | 0 | F | 1 | 0 | 1 | T | 1 | 1 | 1 | T |
| 4 | 1 | 3 | 1 | T | 0 | 0 | 0 | F | 1 | 4 | 1 | T | 1 | 0 | 1 | T |

**Tempo 5**

| | P1 | | | | P2 | | | | P3 | | | | P4 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Latest(i) | engager(i) | num(i) | wait(i) | Latest(i) | engager(i) | num(i) | wait(i) | Latest(i) | engager(i) | num(i) | wait(i) | Latest(i) | engager(i) | num(i) | wait(i) |
| 1 | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F |
| 2 | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F |
| 3 | 1 | 3 | 1 | T | 0 | 0 | 0 | F | 1 | 0 | 1 | T | 1 | 1 | 0 | T |
| 4 | 1 | 3 | 0 | T | 0 | 0 | 0 | F | 1 | 4 | 1 | T | 1 | 0 | 1 | T |

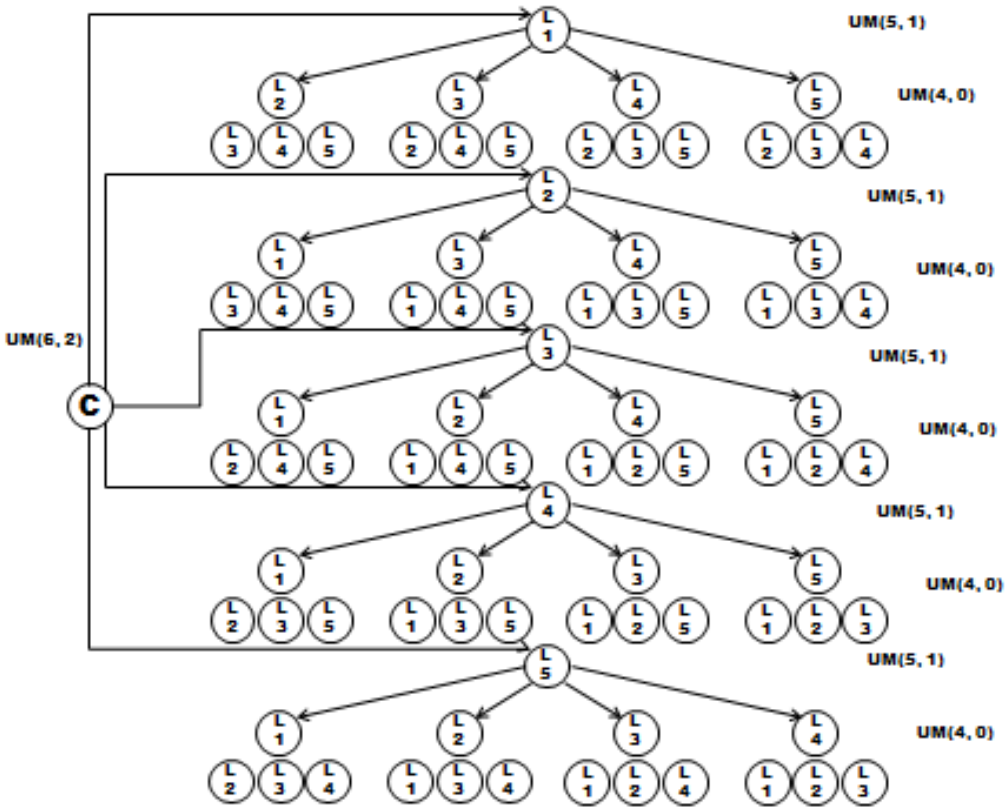
**Tempo 6**

| | P1 | | | | P2 | | | | P3 | | | | P4 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Latest(i) | engager(i) | num(i) | wait(i) | Latest(i) | engager(i) | num(i) | wait(i) | Latest(i) | engager(i) | num(i) | wait(i) | Latest(i) | engager(i) | num(i) | wait(i) |
| 1 | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F |
| 2 | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F |
| 3 | 1 | 3 | 0 | T | 0 | 0 | 0 | F | 1 | 0 | 1 | T | 1 | 1 | 0 | T |
| 4 | 1 | 3 | 0 | T | 0 | 0 | 0 | F | 1 | 4 | 0 | T | 1 | 0 | 1 | T |

**Tempo 7**

| | P1 | | | | P2 | | | | P3 | | | | P4 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Latest(i) | engager(i) | num(i) | wait(i) | Latest(i) | engager(i) | num(i) | wait(i) | Latest(i) | engager(i) | num(i) | wait(i) | Latest(i) | engager(i) | num(i) | wait(i) |
| 1 | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F |
| 2 | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F | 0 | 0 | 0 | F |
| 3 | 1 | 3 | 0 | T | 0 | 0 | 0 | F | 1 | 0 | 0 | T | 1 | 1 | 0 | T |
| 4 | 1 | 3 | 0 | T | 0 | 0 | 0 | F | 1 | 4 | 0 | T | 1 | 0 | 0 | T |

5) Mostre os valores para UM(6,2) com general comandante e general tenente L1 traidores. O general comandante envia ordens diferentes para cada um de seus generais tenentes (a, b, c, d, e) e o general tenente traidor sempre envia a ordem (f). Qual é o número de mensagens que cada general tenente recebe? Qual é a configuração final de mensagens recebidas de L5?

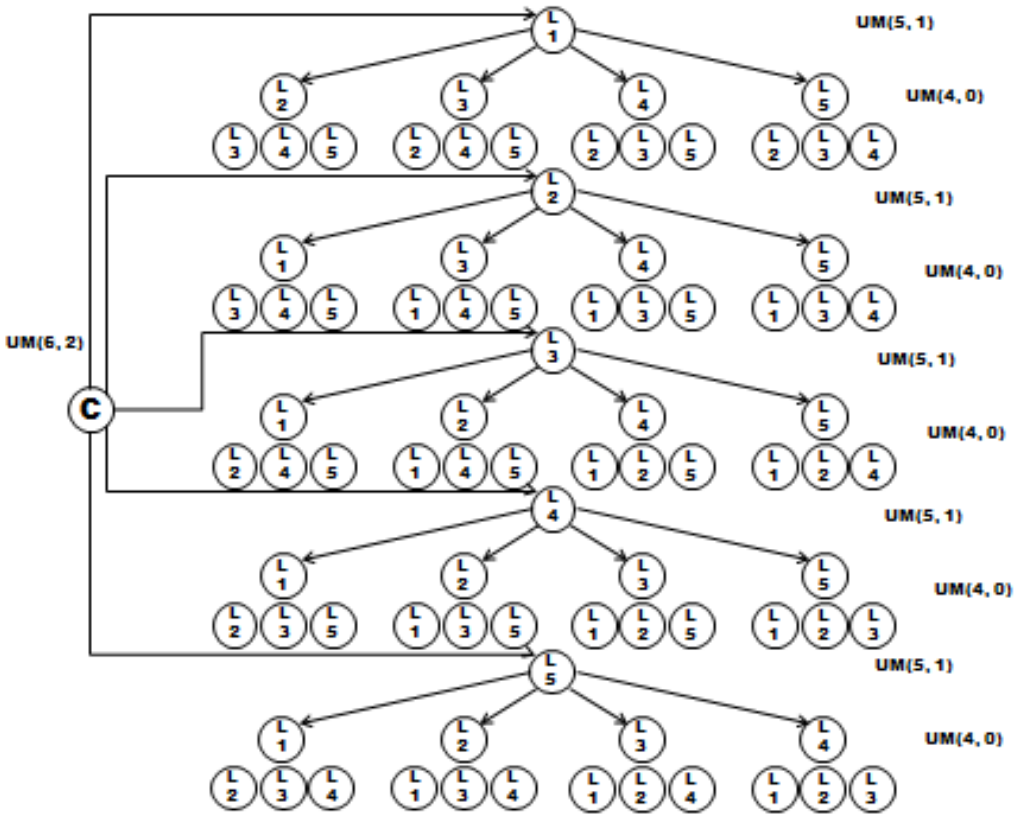



Na figura acime, vemos o grafo gerado para as trocas de mensagens. Cada General Tenente recebe **17 mensagens**.

| | 1 | 2 | | | | 3 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| L1 | a | b | c | d | e | b | b | b | c | c | c | d | d | d | e | e | e |
| L2 | b | f | c | d | e | f | f | f | f | c | c | f | d | d | f | e | e |
| L3 | c | f | b | d | e | f | f | f | f | b | b | f | d | d | f | e | e |
| L4 | d | f | b | c | e | f | f | f | f | b | b | f | c | c | f | e | e |
| L5 | e | f | b | c | d | f | f | f | f | b | b | f | c | c | f | d | d |

**L5** = {e, f, b, c, d, f, f, f, f, b, b, f, c, c, f, d, d}.

Dada à definição do problema, as mensagens deve ser ataque ou recue (0 ou 1). E um tenente traidor envia aleatoriamente ataque ou recue. Apresento a seguir uma segunda solução para o problema baseado nas afirmações descritas. Claro que poderíamos considerar uma das mensagens (a) como ataque e as demais como falha no envio, entretanto para isso deveríamos considerar um Vdef.

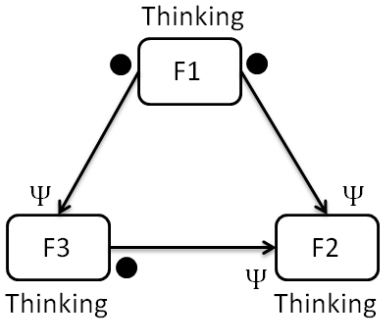



| | 1 | 2 | | | | 3 | | | | | | | | | | | | Decisão |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| L1 | a | r | a | r | a | r | r | r | a | a | a | r | r | r | a | a | a | **Traidor** |
| L2 | r | a | a | r | a | r | a | r | a | a | a | a | r | r | a | a | a | **Ataque** |
| L3 | a | r | r | r | a | a | a | r | a | r | r | r | r | r | r | a | a | **Recue** |
| L4 | r | a | r | a | a | r | r | r | r | r | r | r | a | a | a | a | a | **Recue** |
| L5 | a | r | r | a | r | a | r | a | a | r | r | a | a | a | a | r | r | **Ataque** |

As decisões mostradas na tabela confirmam que para essa configuração o algoritmo não consegue gerar uma decisão unânime para todos os generais. L1 como sendo traidor não toma decisão. Configurando para UM(7, 2) todos tomam a mesma decisão, sendo este o propósito do algoritmo.

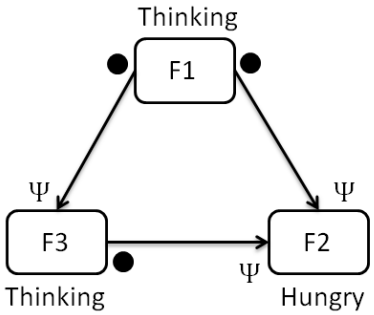
6) Explique a Solução Distribuída de Chandy e Misra (grafos de precedência) para o problema dos filósofos que jantam com 3 filósofos. Use o seguinte roteiro:
• Inicie adequadamente os processos;

• Explique o processamento de mensagens até que um dos filósofos que esteja com fome consiga comer.
• Liste quais regras (1 a 7) que estão sendo usadas a cada transição.




Ψ: Garfo
●: Token

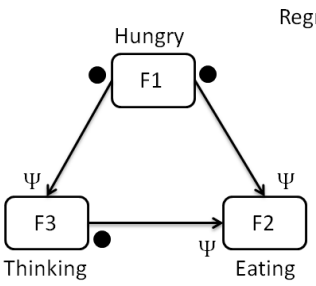
Para a inicialização, o grafo deve ser acíclico. Na primeira figura temos uma possível inicialização com um grafo acíclico.
Como F2 esta com fome e possui os garfos ele pode comer.
F1 precisa enviar request para F2 e F3.
F2 e F3 enviam forktoken para F1.
F1 Recebendo os garfos ele pode comer.




Regra:7

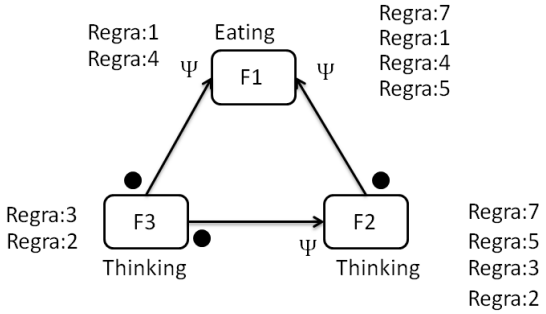
Ψ: Garfo
●: Token

**Ordem da aplicação das regras nos filósofos:**
**F2: Regra 7;**
**F2: Regra 5;**
**F1: Regra 7;**
**F1: Regra 1 para F2;**
**F1: Regra 1 para F3;**
**F3: Regra 2;**
**F3: Regra 3;**
**F2: Regra 2;**
**F2: Regra 3;**
**F1: Regra 4 para F2;**
**F1: Regra 4 para F3;**
**F1: Regra 5;**




Regra:7
Regra:5

Ψ: Garfo
●: Token




Ψ: Garfo
●: Token