Problema dos Filósofos

Sistemas Concorrentes - Lista 2

Sistemas Concorrentes - PEL - 2018.2
Anny Caroline Correa Chagas

O jantar dos filósofos é um problema clássico de exclusão mútua. Nele, cinco filósofos sentam-se ao redor de uma mesa circular que possui cinco pratos e cinco talheres.

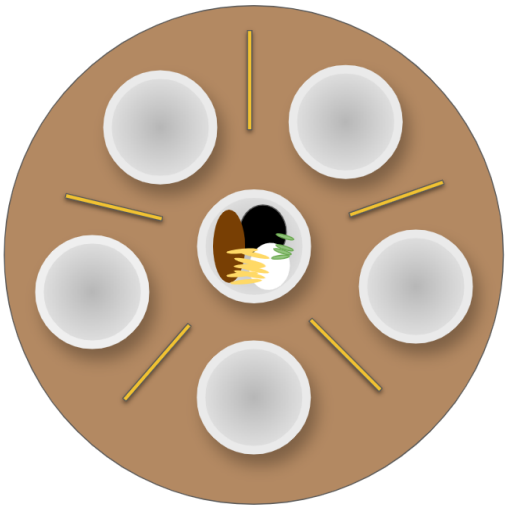

Figura 1. Jantar dos Filósofos

Os filósofos pensam e comem por um tempo arbitrário. Para comer, um filósofo deve, antes, obter os talheres da direita e da esquerda. O pseudo-código que ilustra a rotina de um filósofo é mostrado abaixo:

```
while(1){
    pensa();
    getTalherDireita();
    getTalherEsquerda();
    come();
}
```

Rotina de um Filósofo

A Rede de Petri (RP) correspondente é mostrada na Figura 2. A análise da ferramenta JSARP[1] informa que:

- **A rede não é viva:** uma RP é viva se para todos seus estados sempre existir pelo menos uma transição habilitada para disparo. Portanto, uma rede não viva como a da figura possui pelo menos um estado em que não exista transições habilitadas, caracterizando um *deadlock*. Para se obter um *deadlock* basta fazer com que todos os filósofos retirem o garfo da direita;
- **A rede é limitada, e que seu limite é igual a 1:** o que significa que em todos os seus possíveis estados nunca existe um local com mais de uma ficha;

[1] https://github.com/felipelino/jsarp

- **A rede não é conservativa:** ou seja, o total de fichas em uma rede não é constante para todos os seus estados.

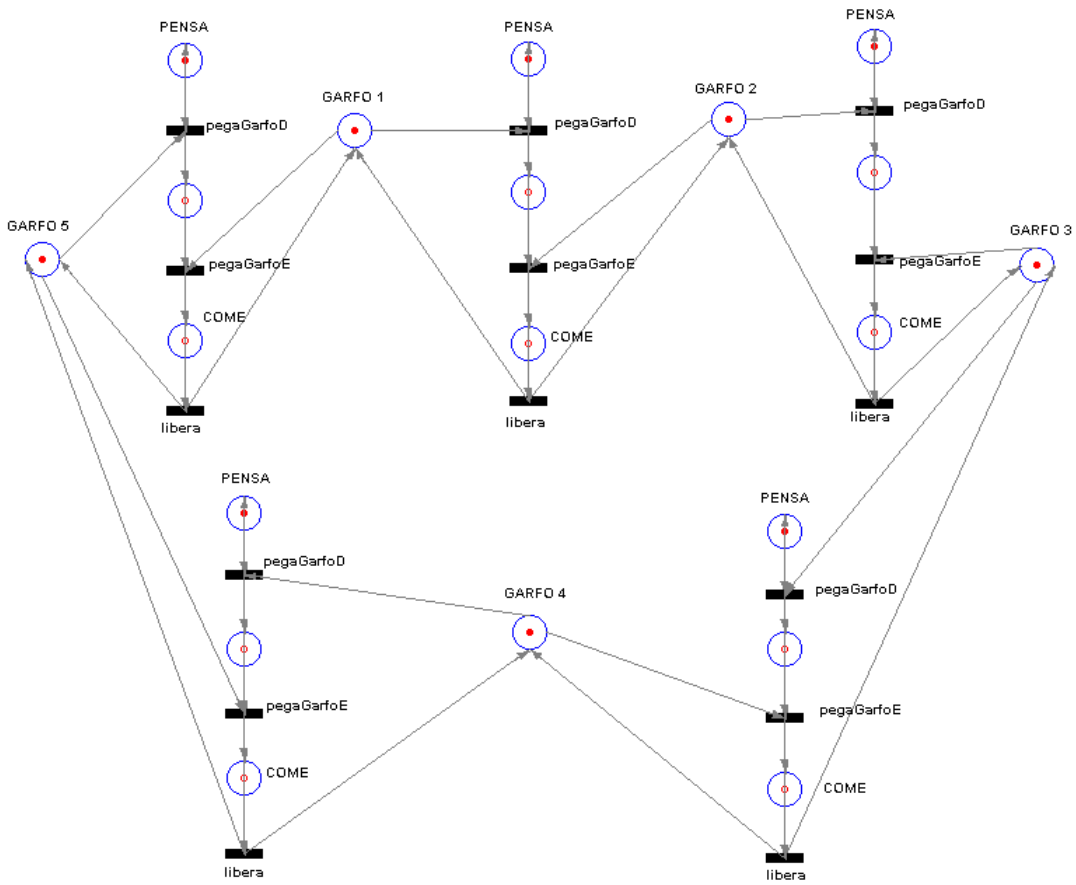

Figura 2. Deadlock

Observando o comportamento da rede da Figura 2, pode-se perceber que o *deadlock* ocorre no momento da retirada dos talheres. Mais especificamente, o problema ocorre pois os filósofos retiram um talher de cada vez, exigindo que o talher da direita fique reservado até que o da esquerda seja obtido. Uma solução seria garantir que os talheres fossem obtidos de uma só vez, de maneira atômica.

A nova rotina dos filósofos é apresentada abaixo, já a RP, na Figura 3. A rede continua 1-limitada (limitada com limite igual a 1) e não conservativa. Mas, como era esperado, não apresenta mais *deadlock* (é viva).

```
while(1){
    pensa();
    getTalheres();
    come();
}
```

Filósofo obtendo os talheres atomicamente

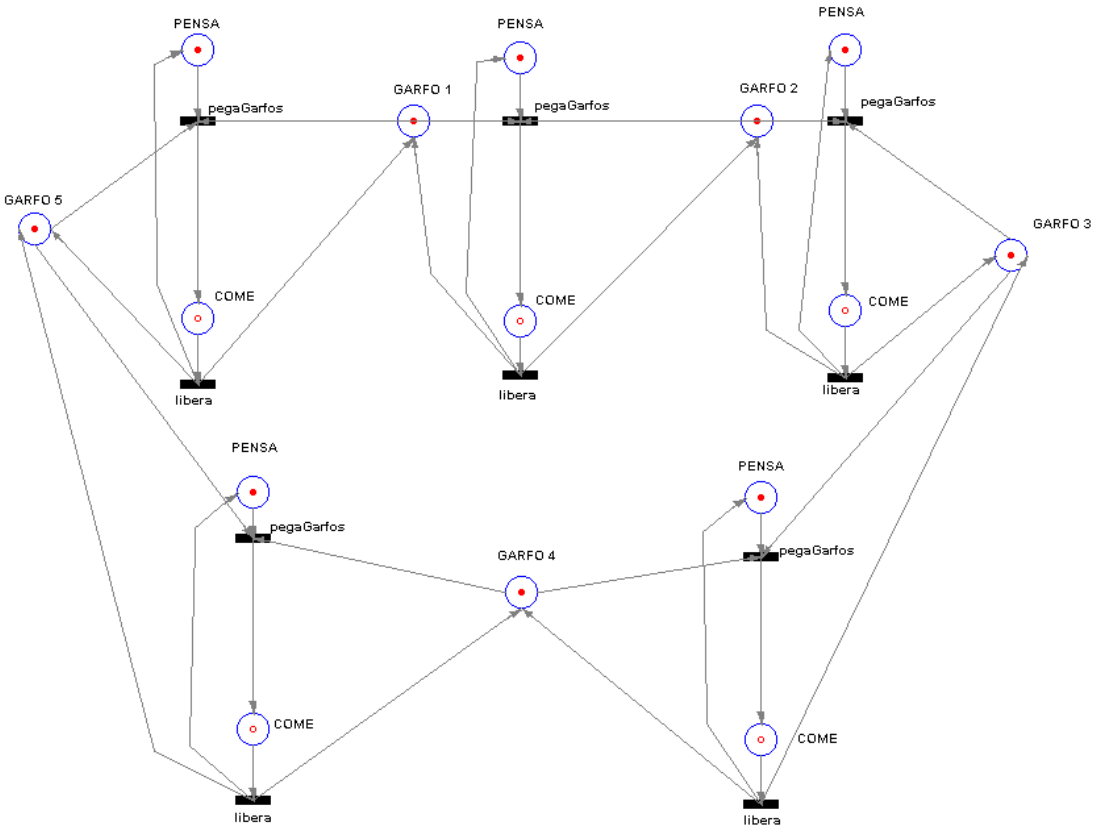

Figura 3. Jantar dos filósofo com talheres sendo obtidos atomicamente

Apesar de resolver o *deadlock*, durante a implementação recursos compartilhados não podem ser obtidos de maneira atômica tão simplesmente quanto em Redes de Petri. Uma primeira estratégia para aproximar a modelagem de uma implementação real seria utilizar um semáforo para garantir que um filósofo obtenha os talheres sem a interferência de outros. Cada filósofo passa a consumir uma ficha do semáforo antes de obter os talheres da direita e da esquerda. Após pegar os talheres, o semáforo é liberado.

```
while(1){
    pensa();
    wait(semaforo);

    getTalherDireita();
    getTalherEsquerda();

    signal(semaforo);
    come();
}
```

Filósofo com o uso de um semáforo

Entretanto, utilizar um semáforo com uma única ficha diminui desnecessariamente o grau de concorrência. Agora, dois filósofos não podem obter os talheres ao mesmo tempo, mesmo que não sejam vizinhos. Com **até quatro fichas** a rede continua 1-limitada, não conservativa e viva.

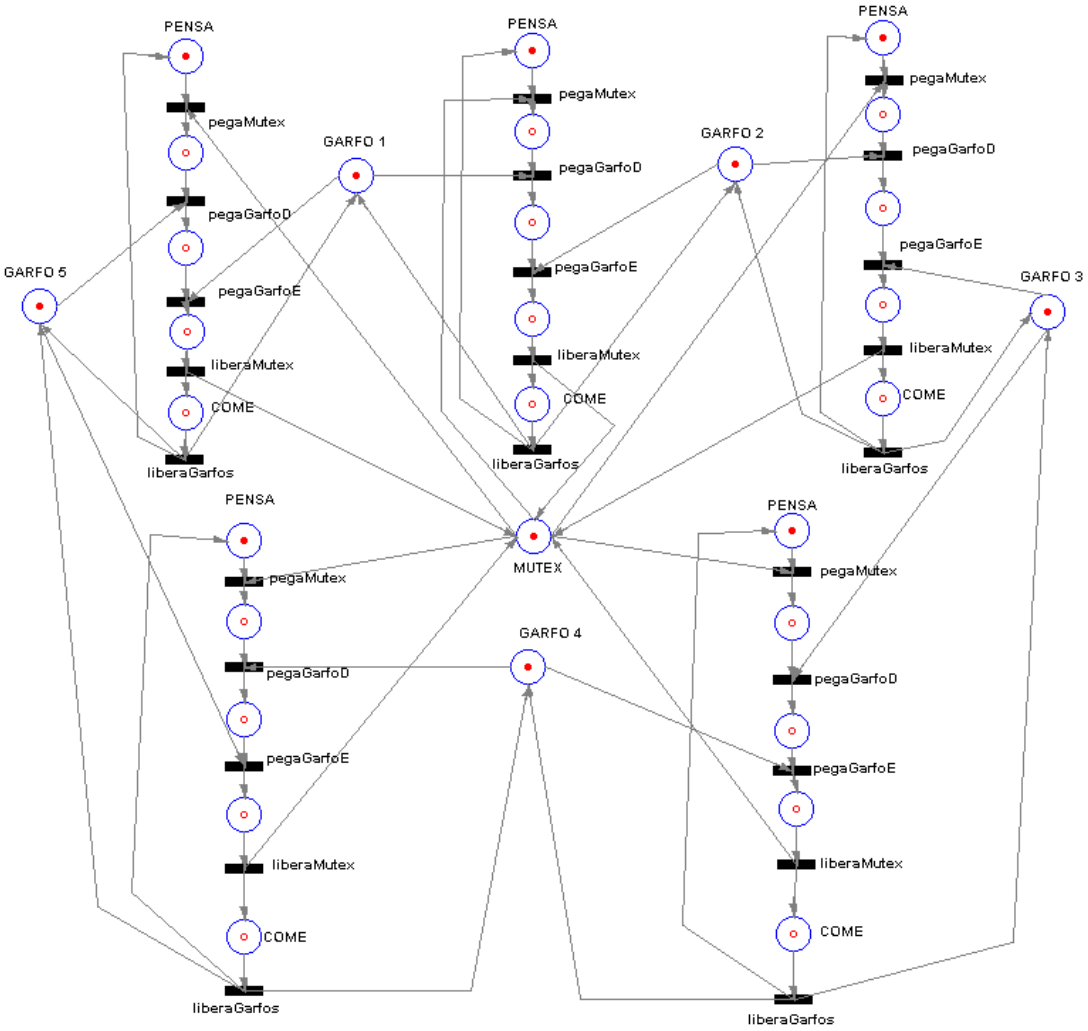

Figura 4. Uso do semáforo

Starvation

Com o problema de *deadlock* resolvido, podemos nos concentrar nas possibilidades de *starvation* das redes apresentadas. Quando o filósofo quer comer, ele deve consultar o semáforo compartilhado e aguardar até sua liberação. Entretanto, não há garantias que esse filósofo poderá comer assim que o semáforo for liberado. Há a possibilidade de outro filósofo ganhar o acesso aos talheres, mesmo que tenha tentado comer depois. A inanição de um filósofo acontece já que a liberação não segue nenhuma política.

A utilização de um semáforo com uma política FIFO (*First in First out*) garantiria a justiça entre os filósofos e resolveria problemas de *starvation*. Com isso, nenhum filósofo ficaria aguardando indeterminadamente para comer. Entretanto, não é possível representar semáforos FIFO através de RPs tradicionais.

Solução para *starvation* com o uso de extensões

Utilizando **Redes de Petri Coloridas** é possível implementar um semáforo FIFO. Essa extensão permite que fichas sejam associadas a valores (chamados de cores), e introduz a ideia de conjuntos de cores e listas de conjuntos.

A RP da Figura 5, modelada por meio da ferramenta CPN Tools[2], apresenta uma simplificação do problema do jantar dos filósofos utilizando um semáforo FIFO. Nela são representados apenas dois filósofos, mas a rede pode ser facilmente expandida. Os locais onde se obtêm os talheres e se come foram agrupadas em um único, o *Rotina*.

Nessa solução, antes de obterem o semáforo, os filósofos devem registrar sua intenção em um fila, representada pelo local *Intencao*. Esse local começa com uma ficha, associada a uma fila vazia (`1`[]`). Ao ativar uma transição *Registra Intencao*, uma ficha é concatenada ao fim da fila de intenções (`fl^^[f2]`, onde `fl` representa a fila e `f2`, a ficha a ser concatenada — nesse caso, proveniente do filósofo 2).

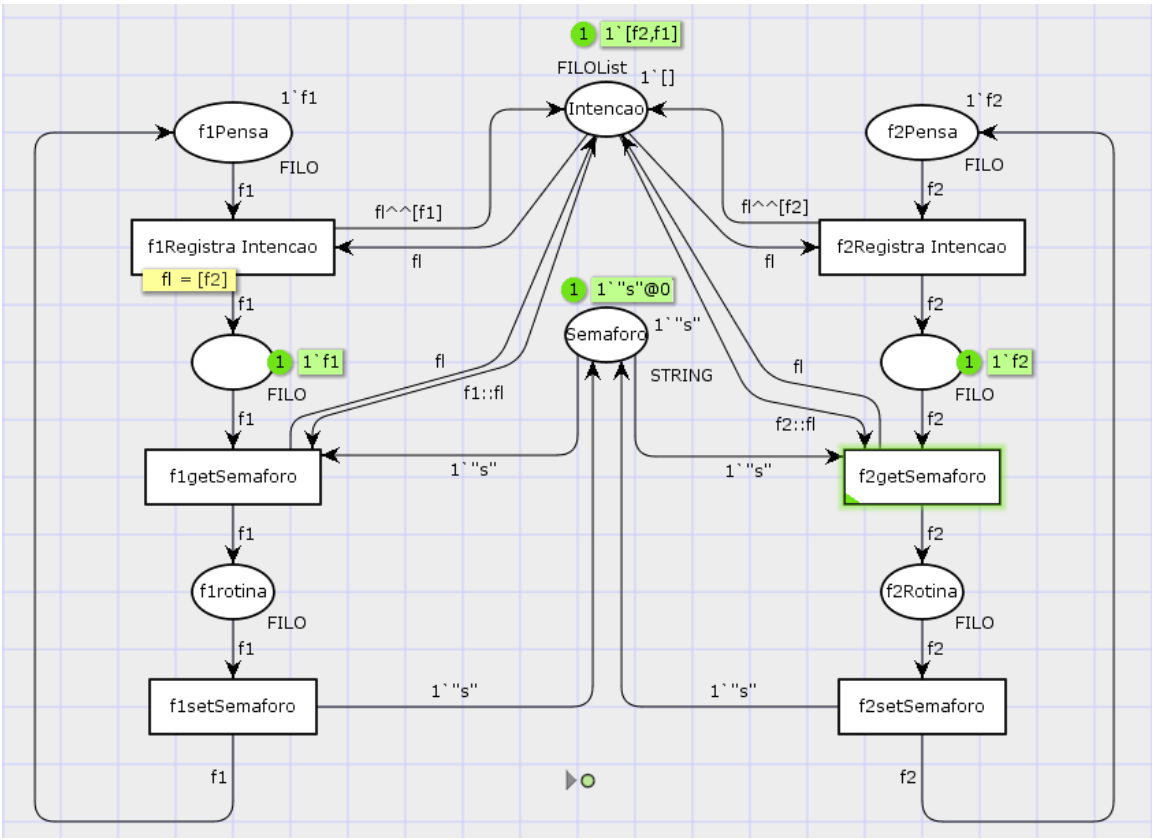

Figura 5. Solução utilizando Redes de Petri Coloridas

Para se obter o semáforo, é necessário que o primeiro elemento da fila corresponda ao filósofo. Por exemplo, caso a fila seja `1`[f1,f2]`, a transição *f2getSemaforo* não poderá ser ativada, somente a *f1getSemaforo*. O operador `::` é responsável por separar a lista em duas partes: a primeira se refere ao primeiro elemento, que passa a ser

[2] `http://cpntools.org/`

armazenado no primeiro operando (neste caso, `f1` ou `f2`). Já a segunda parte é justamente o complemento, ou seja, toda a lista menos o primeiro elemento. Como a fila sem seu primeiro elemento é armazenada na variável (`fl`) que antes armazenava a lista completa, o comando `f1::fl` pode ser entendido como uma operação de *pop*.

A rede apresentada poderia ser simplificada com o uso da extensão **FIFO-nets**. Em FIFO-nets, a primeira transição a entrar em um local é a primeira a sair. Em outra palavras, locais se comportam como filas FIFO, e não como contadores [3]. Portanto, no modelo apresentado na Figura 5 não seria mais necessário modelar as operações de inserção e concatenação na fila, resultando na RP da Figura 6. Vale citar que a ferramenta CPN Tools não oferece suporte à FIFO-nets [1]. A rede foi representada nesta ferramenta somente a fim de ilustrar a simplificação.

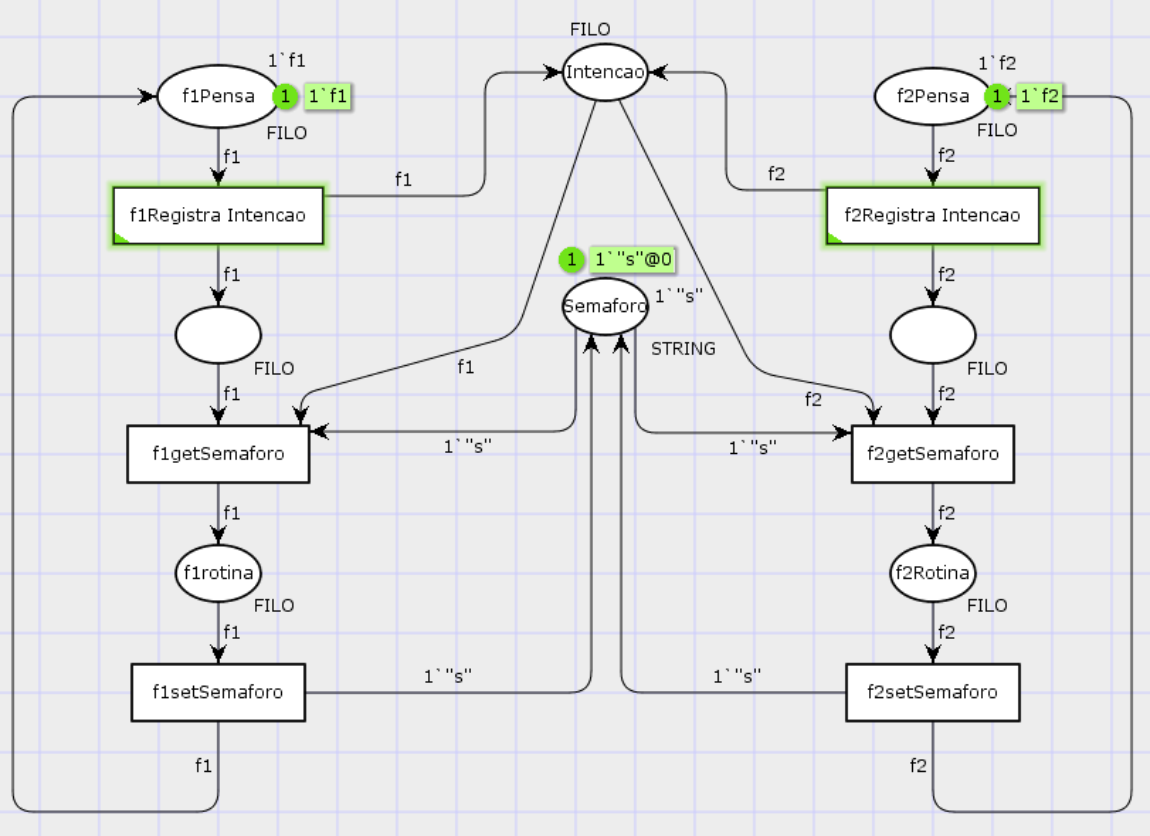

Figura 6. Solução utilizando Redes de Petri coloridas e FIFO-net.

Solução para *starvation* sem o uso de extensões

Uma proposta para resolver o problema de *starvation* utilizando somente RPs tradicionais é definir um número máximo de vezes que um filósofo pode comer (por exemplo, três vezes). Depois de todos terem comido três vezes, a contagem reinicia.

```
while(1){
    pensa();

    if (cont[i] > 0){
        cont[i]--;

        wait(semaforo);
        getTalherDireita();
        getTalherEsquerda();
        signal(semaforo);

        come();
    }
}
```

Filósofo i com o uso de contadores

```
while(1){
    int c=0;
    int n_filosofos = 5;
    for (i=0; i<n_filosofos; i++){
        if(cont[i]==0){
            c++;
        }
    }
    if(c==n_filosofos){
        for (i=0; i<n_filosofos; i++){
            cont[i]=3;
        }
    }
}
```

Processo responsável por resetar a região crítica

Para facilitar a visualização, a Figura 7 apresenta uma simplificação do jantar dos filósofos com a ideia discutida. Nela são apresentados somente dois filósofos. Os locais *Cont1* e *Cont2* representam os contadores responsáveis por definir a quantidade de vezes que um determinado filósofo pode comer: a cada vez que o filósofo come, uma ficha é retirada.

Após comer, uma ficha é armazenada nos locais acumuladores (*Acu1* e *Acu2*). Somente quando todos os estados acumuladores são maiores ou iguais a três (no caso dessa rede eles nunca serão maiores que três) é que a transição *reset* estará habilitada. Para habilitar a transição *reset* somente quando os acumuladores fossem maiores ou iguais a três, os arcos receberam peso[3] 3. A solução completa é apresentada na Figura 8.

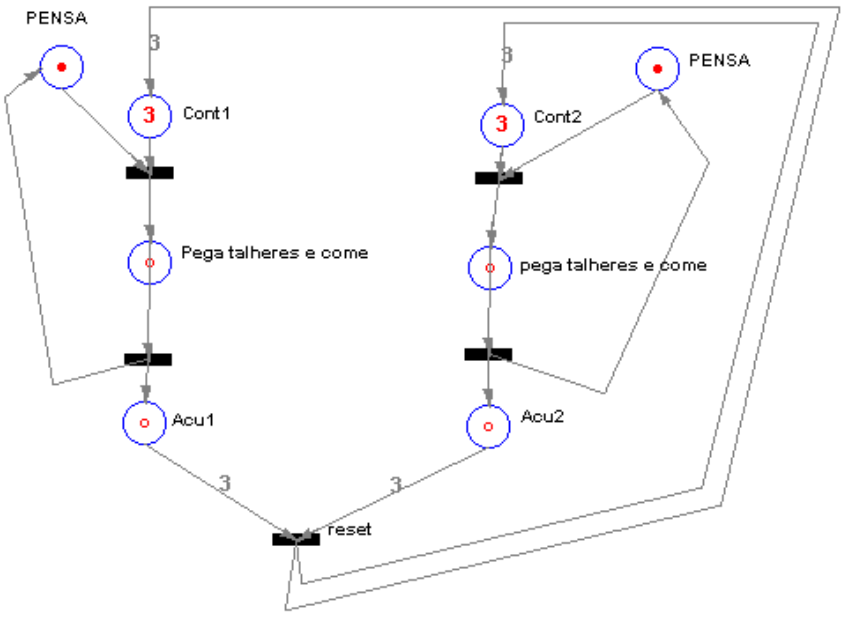

Figura 7. RP simplificada para solução de starvation

[3]Uma transição está habilitada para disparar se todo lugar de entrada, que possuir arco para a transição, possuir um número de marcas maior ou igual ao peso do arco [2]. O peso padrão para um arco é 1.

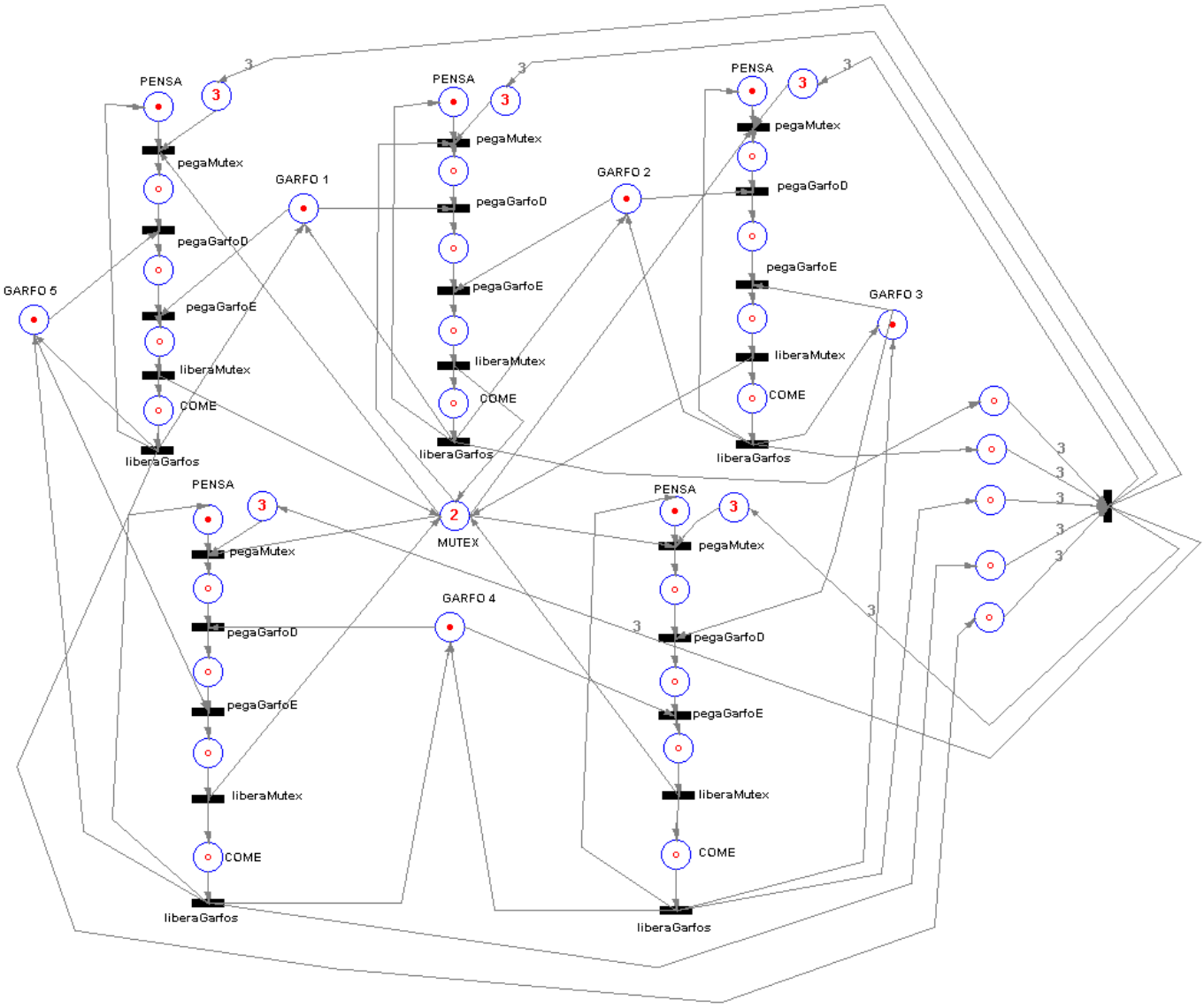

Figura 8. RP completa para a solução de starvation

Referências

[1] Cpn tools - queues and stacks. http://cpntools.org/2018/01/11/queues-and-stacks/. Data de acesso: 23 de novembro de 2018.

[2] Usp - aula sobre redes de petri. https://edisciplinas.usp.br/pluginfile.php/1881413/mod_resource/content/0/Aula_Redes%20de%20Petri.pdf. Data de acesso: 23 de novembro de 2018.

[3] Brauer, W., Reisig, W., and Rozenberg, G. *Petri nets: central models and their properties: advances in petri nets 1986, part I proceedings of an advanced course bad honnef, 8.–19. September 1986*, vol. 254. Springer, 2006.