Faculdade de Ciências Da Universidade do Porto

Licenciatura em Ciência de Computadores

Relatório trabalho prático de Programação Concorrente

Trabalho realizado por:
Bianca Oliveira up202000139
Marta Pereira up202105713

Unidade curricular: Programação Concorrente
Ano letivo: 2023/2024
2º Semestre

Exercício 1

Contexto:
Exclusão mútua na sincronização é um conceito fundamental usado para garantir que apenas um processo ou thread possa aceder a um recurso compartilhado ou uma secção crítica num determinado momento. Ela impede que múltiplas entidades modifiquem ou leiam o recurso compartilhado simultaneamente, evitando assim a corrupção de dados e outros problemas de sincronização.

O objetivo da exclusão mútua é manter a integridade dos dados e preservar a consistência dos recursos compartilhados em ambientes multi-thread ou multi-processo. Quando múltiplos processos ou threads tentam aceder o mesmo recurso simultaneamente, há o risco de operações conflitantes que levam a resultados incorretos ou comportamento imprevisível. A exclusão mútua fornece um mecanismo para controlar o acesso ao recurso compartilhado, garantindo que apenas uma entidade possa operá-lo por vez.

Para alcançar a exclusão mútua, são utilizados mecanismos de sincronização como locks, semáforos e seções críticas. Quando um processo ou thread deseja aceder o recurso compartilhado, ele deve adquirir o lock ou semáforo correspondente, garantindo acesso exclusivo. Uma vez que a tarefa é concluída, o lock é libertado, permitindo que outros processos ou threads o adquiram por sua vez.

O seguinte algoritmo, Hyman, é um exemplo de algoritmo MUTEX que garante que múltiplos processos ou threads não acedem simultaneamente a uma seção crítica do código.

$$P_1 = \left[\begin{array}{l} \texttt{while true do} \\ \quad \texttt{noncricital actions} \\ \quad \texttt{b}_1 \leftarrow \texttt{true;} \\ \quad \texttt{while k} \neq 1 \texttt{ do} \\ \quad\quad \texttt{while b}_2 \texttt{ do} \\ \quad\quad\quad \texttt{skip;} \\ \quad\quad \texttt{k} \leftarrow 1 \\ \quad \texttt{critical actions} \\ \quad \texttt{b}_1 \leftarrow \texttt{false} \end{array}\right] \qquad P_2 = \left[\begin{array}{l} \texttt{while true do} \\ \quad \texttt{noncricital actions} \\ \quad \texttt{b}_2 \leftarrow \texttt{true;} \\ \quad \texttt{while k} \neq 2 \texttt{ do} \\ \quad\quad \texttt{while b}_1 \texttt{ do} \\ \quad\quad\quad \texttt{skip;} \\ \quad\quad \texttt{k} \leftarrow 2 \\ \quad \texttt{critical actions} \\ \quad \texttt{b}_2 \leftarrow \texttt{false} \end{array}\right]$$

Exercício 1.1 - Implementação do algoritmo em CSS (Calculus of Sequential Systems) para dois processos, P1 e P2.

Há dois processos, duas variáveis b1 e b2 do tipo booleano com valor inicial a false e uma variável k que pode assumir os valores de 1 ou 2, inicialmente toma o valor de 1 como pedido no enunciado.

- Para b1 temos o estado B1t se o valor for true e o estado B1f se for false;
- Para b2 temos o estado B2t se o valor for true e o estado B2f se for false;
- Para um processo ler true em b1 ou b2 sincroniza com esse processo por uma ação (canal) b1rt ou b2rt;
- Para um processo escrever false em b1 ou b2 sincroniza com esse processo por uma ação (canal) b1wf ou b2wf;
- Analogamente se define o comportamento de k que pode tomar os valores 1 e 2.

Processos para as variáveis:

```
B1f := b1wf?.B1f + b1rf!.B1f + b1wt?.B1t
B1t := b1wt?.B1t + b1wf?.B1f + b1rt!.B1t

B2f := b2wf?.B2f + b2rf!.B2f + b2wt?.B2t
B2t := b2wt?.B2t + b2wf?.B2f + b2rt!.B2t

K1 := kw1?.K1 + kw2?.K2 + kr1!.K1
K2 := kw2?.K2 + kw1?.K1 + kr2!.K2
```

Processos para P1 e P2:

- Apenas modelar a entrada e saída da zona crítica;
- Fazer a inicialização das variáveis bi e k;
- Supomos que não podem terminar na zona crítica ou ficar lá para sempre.

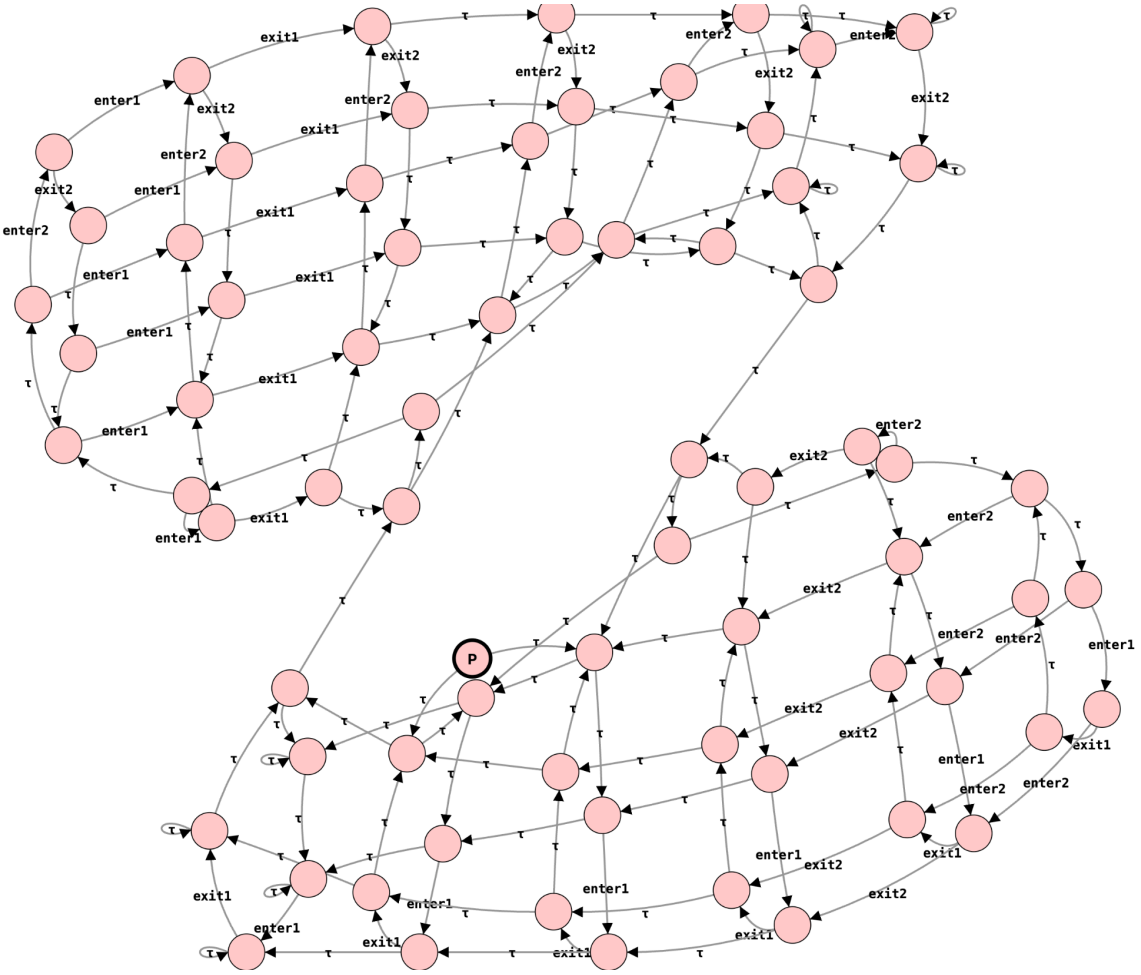
```
P1 := b1wt!.P11
P11 := kr1?.P14 + kr2?.P12
P12 := b2rf?.P13 + b2rt?.P12
P13 := kw1!.P11
P14 := enter1.exit1.b1wf!.P1

P2 := b2wt!.P21
P21 := kr2?.P24 + kr1?.P22
P22 := b1rf?.P23 + b1rt?.P22
P23 := kw2!.P21
P24 := enter2.exit2.b2wf!.P2
```

Se k = 1 no início:

```
P := (P1|P2|K1|B1f|B2f) \ {b1rf, b2rf, b1rt, b2rt, b1wf, b2wf, b1wt, b2wt, kr1, kr2, kw1, kw2}
```

A zona não crítica de cada processo é omitida na representação, pois estamos preocupados com o problema de exclusão mútua. Já a zona crítica de cada processo é representada como um par de ações, ‘enter’ e 'exit', representando entrada e saída da zona crítica, respectivamente.

Resultado:



Exercício 1.2 - Implementar este algoritmo em Scala usando uma thread para cada processo.

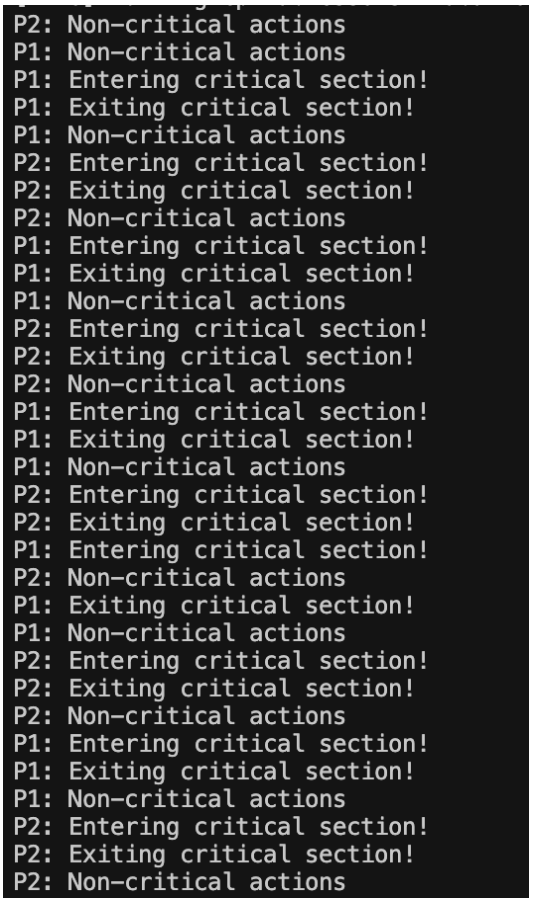
Uma thread é a menor unidade de processamento que pode ser gerida por um sistema operacional. Em termos simples, uma thread é uma sequência de instruções que pode ser executada independentemente por um processador, enquanto compartilha recursos, como memória, com outras threads pertencentes ao mesmo processo.

O algoritmo de Hyman's utiliza variáveis de controlo e sinalização para coordenar as threads. No exemplo dado, duas threads tentam aceder à seção crítica e utilizam variáveis booleanas (b1 e b2) e uma variável inteira (k) para garantir a exclusão mútua. Estas são definidas como `@volatile` para garantir que sejam sempre visíveis e atualizadas corretamente entre as threads, o que é crucial num ambiente de memória compartilhada para evitar problemas de cache e garantir a consistência dos dados.

Cada processo é implementado como uma thread separada que executa um loop infinito. Dentro deste loop, o processo indica a sua intenção de entrar na seção crítica, verifica se é a sua vez (`k == id`) e, se não for, espera até que seja. A variável k é usada para alternar entre os processos, garantindo que apenas um processo entra na seção crítica, ele liberta a entrada ajustando a sua variável de controle (b1 ou b2) para `false`. Dentro dos loops de espera, utilizamos busy-waiting, o que é simples mas pode não ser eficiente em termos de consumo de CPU.

Exercício 1.3 - Testar a implementação da alínea anterior utilizando tarefas fictícias concretas. Explicar abordagem de teste, como executar e replicar os seus experimentos e como podemos concluir (ou não) que ambos os processos executam a zona crítica um de cada vez.

Para verificar quando é que cada processo entra na zona crítica, definimos uma função que simula ações críticas e não críticas que cada processo executa. Estas apenas imprimem uma mensagem a informar de que o processo x entrou e saiu da zona crítica e fazem o processo parar por 1 segundo. Assim, conseguimos identificar se, em algum momento da execução, os processos acedem à zona crítica simultaneamente. Se isso acontecer, o output terá uma um print em que um processo entrou na zona crítica e antes de sair, o outro processo também entrou.



Exercício 1.4 - Implementar uma alternativa ao algoritmo Hyman's usando Java locks para controlar a entrada e saída da zona crítica.

Assim como o Hyman's, também o algoritmo de Peterson é MUTEX, e por isso permite que dois processos compartilhem recursos num sistema concorrente, garantindo que apenas um deles aceda à seção crítica por vez.

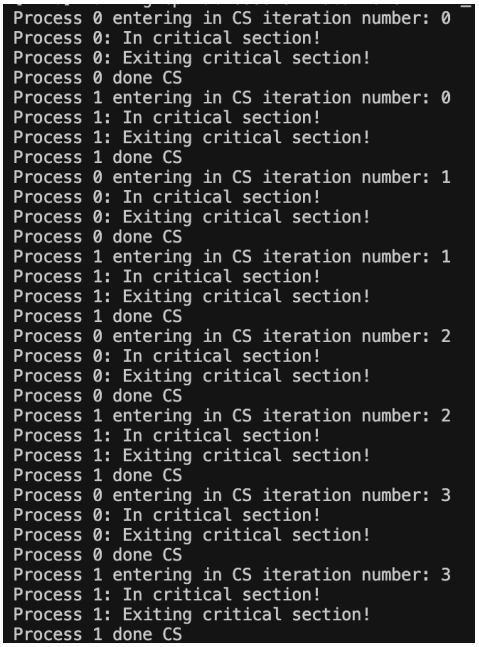
Este algoritmo utiliza variáveis compartilhadas para coordenar o acesso à seção crítica. Cada processo tem a sua própria flag de controlo, que indica se o processo deseja entrar na seção crítica. No algoritmo de Peterson, para dois processos (P0 e P1), há duas flags: `flag[0]` e `flag[1],` e estas são variáveis boolenas. Há uma variável, `turn`, que indica de quem é a vez de entrar na seção crítica. Quando um processo deseja entrar na seção crítica, ele primeiro define a sua própria flag para `true`, indicando que ele está interessado em entrar na seção crítica. Em seguida, ele atualiza a variável, `turn,` para o outro processo, indicando que é a vez do outro processo tentar entrar na seção crítica.

Após atualizar a variável de vez, `flag[1] = true`, o processo verifica se o outro processo também deseja entrar na seção crítica (`flag[1]` é verdadeiro). Se o outro processo não está interessado em entrar na seção crítica ou não é a sua vez (`turn != 1`), o processo pode entrar na seção crítica, onde o acesso é, sincronizado o acesso com `lock.synchronized`.

Caso contrário, o processo espera até que seja a sua vez (`turn == 1`) e o outro processo não esteja interessado em entrar na seção crítica. Quando um processo termina de executar na seção crítica, ele redefine a sua flag de controlo para `false`, indicando que não está mais interessado em entrar na seção crítica.

O uso de `synchronized` garante atomicidade e exclusão mútua para o bloco de código dentro dele, o que é crucial para garantir que os threads concorrentes não interfiram uns com os outros ao aceder recursos compartilhados.

Output:



Exercício 1.5 - Implementar uma alternativa ao algoritmo Hyman's usando uma abordagem sem bloqueio com uma variável atômica para controlar a entrada na zona crítica.

Para implementar uma alternativa ao algoritmo de Hyman usando uma abordagem lock-free para controlar a entrada na seção crítica, utilizamos a classe AtomicBoolean para garantir a exclusão mútua sem o uso de locks tradicionais. Este método visa evitar problemas de deadlock, oferecendo uma solução eficiente e segura para threads concorrentes.

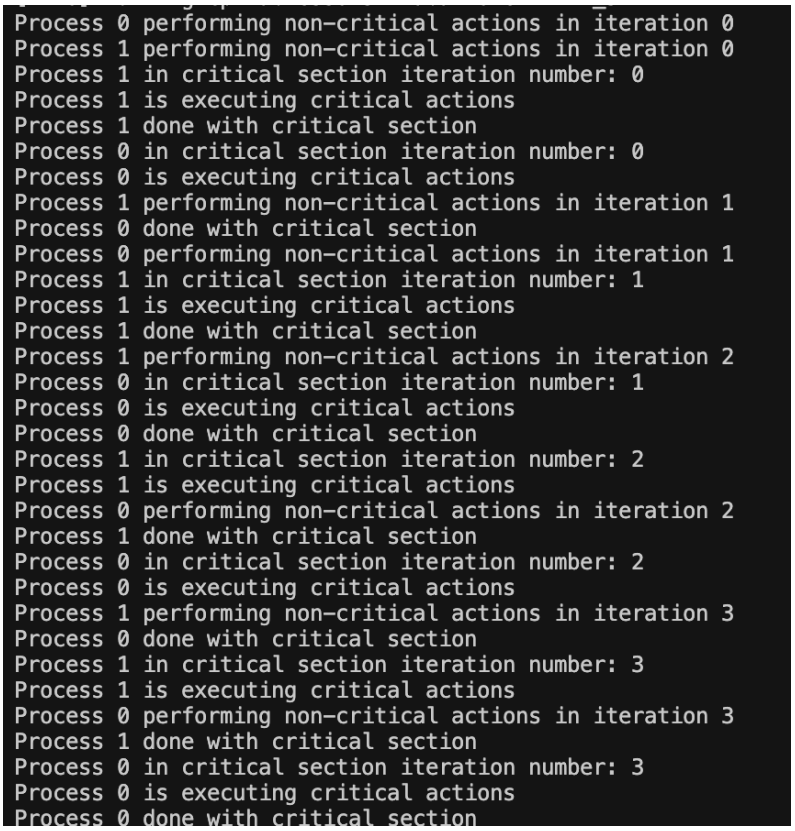
No início do algoritmo, é criada uma variável atómica `AtomicBoolean` denominada "lock". Esta variável é utilizada como um mecanismo de bloqueio para coordenar o acesso à secção crítica. O valor `false` da `AtomicBoolean` indica que o lock está livre, permitindo que um processo entre na secção crítica. Quando um processo deseja entrar na secção crítica, ele tenta adquirir o lock utilizando o método compareAndSet da `AtomicBoolean`.

O método compareAndSet verifica se o valor atual da `AtomicBoolean` é false (indicando que o lock está livre) e, se for o caso, define o valor como true (indicando que o lock está ocupado). Se o valor atual não for false, o método retorna false, indicando que o lock já está ocupado. Nesse caso, o processo entra em um loop de busy wait, continuamente tentando adquirir o lock até que esteja disponível.

Após adquirir o lock, o processo executa as suas ações críticas dentro da secção crítica. Ao terminar, o processo liberta o lock definindo o valor da AtomicBoolean como false, indicando que o lock está livre novamente e permitindo que outros processos possam aceder à secção crítica.

Essa abordagem lock-free do algoritmo, aliada ao uso do CAS (Compare And Set)) lecionado nas aulas, proporciona uma execução eficiente e segura, garantindo que apenas um processo acede a seção crítica por vez, sem a necessidade de bloqueios explícitos.

Resultado:



Exercício 2

A exclusão mútua pode ser implementada em diversas situações, um exemplo é o problema da refeição de sushi:

Um grupo de 5 amigos vão juntos comer sushi e sentam-se numa mesa redonda. Entre cada amigo existe um único pauzinho, compartilhado por dois amigos. Durante a refeição, cada amigo come com os pauzinhos esquerdo e direito ou conversa sem usar os pauzinhos, alternando constantemente entre esses dois estados.

Este problema é uma representação clássica de questões de concorrência onde os pauzinhos representam recursos compartilhados (por exemplo, impressoras, arquivos, etc.), e os amigos representam processos ou threads que necessitam de acesso exclusivo a esses recursos para realizar as suas tarefas.

Exercício 2.1 - Codifique 3 variações do problema dos comedores de sushi como fórmulas CCS com 5 processos em paralelo.

- Para fome1, que representa a fome do amigo 1, temos o estado F1t se o valor for true e o estado F1f se for false;
- Para fome2, que representa a fome do amigo 2, temos o estado F2t se o valor for true e o estado F2f se for false;
- Para fome3, que representa a fome do amigo 3, temos o estado F3t se o valor for true e o estado F3f se for false;
- Para fome4, que representa a fome do amigo 4, temos o estado F4t se o valor for true e o estado F4f se for false;
- Para fome5, que representa a fome do amigo 5, temos o estado F5t se o valor for true e o estado F5f se for false;
- Para um processo ler true em fomeX com X > 0 e X<6 sincroniza com esse processo por uma ação (canal) fome1rt;
- Para um processo escrever false em fomeX com X > 0 e X<6 sincroniza com esse processo por uma ação (canal) fome1wf;

- Analogamente se define o comportamento de kX com X > 0 e X<6 que pode tomar os valores 1 e 2.

Processos para as variáveis:

```
F1f := fome1wf?.F1f + fome1rf!.F1f + fome1wt?.F1t
F1t := fome1wt?.F1t + fome1wf?.F1f + fome1rt!.F1t

F2f := fome2wf?.F2f + fome2rf!.F2f + fome2wt?.F2t
F2t := fome2wt?.F2t + fome2wf?.F2f + fome2rt!.F2t

F3f := fome3wf?.F3f + fome3rf!.F3f + fome3wt?.F3t
F3t := fome3wt?.F3t + fome3wf?.F3f + fome3rt!.F3t

F4f := fome4wf?.F4f + fome4rf!.F4f + fome4wt?.F4t
F4t := fome4wt?.F4t + fome4wf?.F4f + fome4rt!.F4t

F5f := fome5wf?.F5f + fome5rf!.F5f + fome5wt?.F5t
F5t := fome5wt?.F5t + fome5wf?.F5f + fome5rt!.F5t
```

```
Ka1 := kaw1?.Ka1 + kaw2?.Ka2 + kar1!.Ka1
Ka2 := kaw2?.Ka2 + kaw1?.Ka1 + kar2!.Ka2

Kb2 := kbw2?.Kb2 + kbw3?.Kb3 + kbr2!.Kb2
Kb3 := kbw3?.Kb3 + kbw2?.Kb2 + kbr3!.Kb3

Kc3 := kcw3?.Kc3 + kcw4?.Kc4 + kcr3!.Kc3
Kc4 := kcw4?.Kc4 + kcw3?.Kc3 + kcr4!.Kc4

Kd4 := kdw4?.Kd4 + kdw5?.Kd5 + kdr4!.Kd4
Kd5 := kdw5?.Kd5 + kdw4?.Kd4 + kdr5!.Kd5

Ke5 := kew5?.Ke5 + kew1?.Ke1 + ker5!.Ke5
Ke1 := kew1?.Ke1 + kew5?.Ke5 + ker1!.Ke1
```

V1: Cada amigo tenta pegar o pauzinho esquerdo (se disponível) e depois o pauzinho direito (se disponível). Depois de comer um pouco, ela larga os pauzinhos até sentir fome novamente e repete as mesmas ações.

Processos para P1, P2, P3, P4 E P5:

- Todas as pessoas(processos) realizam as mesmas ações diferindo apenas em quais palitinho estão no seu lado esquerdo e no seu lado direito e quais amigos estão concorrendo com ele para pegar esses palitinhos.
- Para o exemplo de P1, fome1 = 1 indica que o processo deseja entrar na zona crítica.
- Após realizar a ação de pegar o palitinho da esquerda(take_sticka) e pegar o palitinho a direita(take_sticke), pode comer o sushi, então colocar a fome1 = 0.

```
P1 := conversation1.fome1wt!.P11
P11 := kar1?.P14 + kar2?.P12
P12 := fome2rf?.P13 + fome2rt?.P12
P13 := kaw1!.P11
P14 := take_sticka.P15
P15 := ker1?.P18 + ker2?.P16
P16 := fome5rf?.P17 + fome5rt?.P16
P17 := kew1!.P15
P18 := take_sticke.eat.fome1wf!.P1
```

V2: O mesmo de antes, mas um dos amigos é canhoto e tenta agarrar o pauzinho direito antes de pegar o pauzinho esquerdo.

Processos para P1, P2, P3, P4 E P5:

- O V2 segue os mesmos passos que o V1, mas um de seus processo é diferente, pois representa o amigo canhoto.
- No processo do amigo canhoto, primeiro ele pega o palitinho direito(take_sticke) e depois pega o palitinho esquerdo(take_sticka).

```
// processo para o amigo canhoto
P1 := conversation1.fome1wt!.P11
P11 := ker1?.P14 + ker2?.P12
P12 := fome5rf?.P13 + fome5rt?.P12
P13 := kew1!.P11
P14 := take_sticke.P15
P15 := kar1?.P18 + kar2?.P16
P16 := fome2rf?.P17 + fome2rt?.P16
P17 := kaw1!.P15
P18 := take_sticka.eat.fome1wf!.P1
```

V3: Antes de comer, cada amigo deve servir alguns pedaços de sushi. Mais especificamente, ela deve (1) pegar a bandeja de sushi, (2) pegar o pauzinho esquerdo, (3) pegar o pauzinho direito, (4) devolver a bandeja de sushi, (5) comer, (6) e finalmente devolver os dois pauzinhos.

Processos para P1, P2, P3, P4 E P5 e para a variável t:

- O V3 segue os mesmos passos que o V1, mas possui mais uma zona crítica que é partilhada por todos os amigos, que são as ações relacionadas com a tigela de sushi no centro da mesa.
- É adicionada mais uma variável t, que é igual a 0 quando está livre no centro da mesa e igual a 1 quando está vendo usada.

```
T0 := tw0?.T0 + tw1?.T1 + tr0!. T0
T1 := tw1?.T1 + tw0?.T0 + tr1!.T2

P1 := conversation1.P111
P111 := tr0?.P112 + tr1?.P111
P112 := tw1!.grab_sushi_tray.fome1wt!.P11
P11 := kar1?.P14 + kar2?.P12
P12 := fome2rf?.P13 + fome2rt?.P12
P13 := kaw1!.P11
P14 := take_sticka.P15
P15 := ker1?.P18 + ker2?.P16
P16 := fome5rf?.P17 + fome5rt?.P16
P17 := kew1!.P15
P18 := take_sticke.return_sushi_tray.P19
P19 := tw0!P110
P110 := eat.fome1wf!.P1
```

Exercício 2.2 - Para cada uma das 3 variações acima, explique se há deadlock e forneça um trace quando aplicável.

Para as 3 variações do problema da refeição de sushi, a única que possui deadlock é V1, pois se todos os amigos pegam o palitinho esquerdo primeiro, ninguém consegue pegar o palitinho direito e os processos entram em ciclo infinito esperando entrar na zona crítica do palitinho direito e nunca saem da zona crítica do palitinho esquerdo.

Trace para V1:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{take\_sticke.eat.fome1wf!.P1 \xrightarrow{take\_sticke} eat.fome1wf!.P1}\,\text{prefix} \qquad \Gamma(P18) = take\_sticke.eat.fome1wf!.P1}{P18 \xrightarrow{take\_sticke} eat.fome1wf!.P1}\,\text{rec}}{P18 \mid P2 \xrightarrow{take\_sticke} eat.fome1wf!.P1 \mid P2}\,\text{par-l}}{P18 \mid P2 \mid P3 \xrightarrow{take\_sticke} eat.fome1wf!.P1 \mid P2 \mid P3}\,\text{par-l}}{P18 \mid P2 \mid P3 \mid P4 \xrightarrow{take\_sticke} eat.fome1wf!.P1 \mid P2 \mid P3 \mid P4}\,\text{par-l}}{P18 \mid P2 \mid P3 \mid P4 \mid P5 \xrightarrow{take\_sticke} eat.fome1wf!.P1 \mid P2 \mid P3 \mid P4 \mid P5}\,\text{par-l}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{ker1?.P18 \xrightarrow{ker1?} P18}\,\text{prefix}}{ker1?.P18 + ker2?.P16 \xrightarrow{ker1?} P18}\,\text{choice-l} \qquad \Gamma(P15) = ker1?.P18 + ker2?.P16}{P15 \xrightarrow{ker1?} P18}\,\text{rec}}{P15 \mid P2 \xrightarrow{ker1?} P18 \mid P2}\,\text{par-l}}{P15 \mid P2 \mid P3 \xrightarrow{ker1?} P18 \mid P2 \mid P3}\,\text{par-l}}{P15 \mid P2 \mid P3 \mid P4 \xrightarrow{ker1?} P18 \mid P2 \mid P3 \mid P4}\,\text{par-l}}{P15 \mid P2 \mid P3 \mid P4 \mid P5 \xrightarrow{ker1?} P18 \mid P2 \mid P3 \mid P4 \mid P5}\,\text{par-l}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{take\_sticka.P15 \xrightarrow{take\_sticka} P15}\,\text{prefix} \qquad \Gamma(P14) = take\_sticka.P15}{P14 \xrightarrow{take\_sticka} P15}\,\text{rec}}{P14 \mid P2 \xrightarrow{take\_sticka} P15 \mid P2}\,\text{par-l}}{P14 \mid P2 \mid P3 \xrightarrow{take\_sticka} P15 \mid P2 \mid P3}\,\text{par-l}}{P14 \mid P2 \mid P3 \mid P4 \xrightarrow{take\_sticka} P15 \mid P2 \mid P3 \mid P4}\,\text{par-l}}{P14 \mid P2 \mid P3 \mid P4 \mid P5 \xrightarrow{take\_sticka} P15 \mid P2 \mid P3 \mid P4 \mid P5}\,\text{par-l}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{kar1?.P14 \xrightarrow{kar1?} P14}\,\text{prefix}}{kar1?.P14 + kar2?.P12 \xrightarrow{kar1?} P14}\,\text{choice-l} \qquad \Gamma(P11) = kar1?.P14 + kar2?.P12}{P11 \xrightarrow{kar1?} P14}\,\text{rec}}{P11 \mid P2 \xrightarrow{kar1?} P14 \mid P2}\,\text{par-l}}{P11 \mid P2 \mid P3 \xrightarrow{kar1?} P14 \mid P2 \mid P3}\,\text{par-l}}{P11 \mid P2 \mid P3 \mid P4 \xrightarrow{kar1?} P14 \mid P2 \mid P3 \mid P4}\,\text{par-l}}{P11 \mid P2 \mid P3 \mid P4 \mid P5 \xrightarrow{kar1?} P14 \mid P2 \mid P3 \mid P4 \mid P5}\,\text{par-l}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{fome1wt!.P11 \xrightarrow{fome1wt!} P11}\,\text{prefix}}{fome1wt!.P11 \mid P2 \xrightarrow{fome1wt!} P11 \mid P2}\,\text{par-l}}{fome1wt!.P11 \mid P2 \mid P3 \xrightarrow{fome1wt!} P11 \mid P2 \mid P3}\,\text{par-l}}{fome1wt!.P11 \mid P2 \mid P3 \mid P4 \xrightarrow{fome1wt!} P11 \mid P2 \mid P3 \mid P4}\,\text{par-l}}{fome1wt!.P11 \mid P2 \mid P3 \mid P4 \mid P5 \xrightarrow{fome1wt!} P11 \mid P2 \mid P3 \mid P4 \mid P5}\,\text{par-l}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{conversation1.fome1wt!.P11 \xrightarrow{conversation1} fome1wt!.P11}\,\text{prefix} \quad \Gamma(P1) = conversation1.fome1wt!.P11}{P1 \xrightarrow{conversation1} fome1wt!.P11}\,\text{rec}}{P1 \mid P2 \xrightarrow{conversation1} fome1wt!.P11 \mid P2}\,\text{par-l}}{P1 \mid P2 \mid P3 \xrightarrow{conversation1} fome1wt!.P11 \mid P2 \mid P3}\,\text{par-l}}{P1 \mid P2 \mid P3 \mid P4 \xrightarrow{conversation1} fome1wt!.P11 \mid P2 \mid P3 \mid P4}\,\text{par-l}}{P1 \mid P2 \mid P3 \mid P4 \mid P5 \xrightarrow{conversation1} fome1wt!.P11 \mid P2 \mid P3 \mid P4 \mid P5}\,\text{par-l} \quad \Gamma(P) = P1 \mid P2 \mid P3 \mid P4 \mid P5}{P \xrightarrow{conversation1} fome1wt!.P11 \mid P2 \mid P3 \mid P4 \mid P5}\,\text{rec}$$

Trace para V2:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{take\_sticka.eat.fome1wf!.P1 \xrightarrow{take\_sticka} eat.fome1wf!.P1}\,\text{prefix} \quad \Gamma(P18) = take\_sticka.eat.fome1wf!.P1}{P18 \xrightarrow{take\_sticka} eat.fome1wf!.P1}\,\text{rec}}{P18 \mid P2 \xrightarrow{take\_sticka} eat.fome1wf!.P1 \mid P2}\,\text{par-l}}{P18 \mid P2 \mid P3 \xrightarrow{take\_sticka} eat.fome1wf!.P1 \mid P2 \mid P3}\,\text{par-l}}{P18 \mid P2 \mid P3 \mid P4 \xrightarrow{take\_sticka} eat.fome1wf!.P1 \mid P2 \mid P3 \mid P4}\,\text{par-l}}{P18 \mid P2 \mid P3 \mid P4 \mid P5 \xrightarrow{take\_sticka} eat.fome1wf!.P1 \mid P2 \mid P3 \mid P4 \mid P5}\,\text{par-l}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{kar1?.P18 \xrightarrow{kar1?} P18}\,\text{prefix}}{kar1?.P18 + kar2?.P16 \xrightarrow{kar1?} P18}\,\text{choice-l} \quad \Gamma(P15) = kar1?.P18 + kar2?.P16}{P15 \xrightarrow{kar1?} P18}\,\text{rec}}{P15 \mid P2 \xrightarrow{kar1?} P18 \mid P2}\,\text{par-l}}{P15 \mid P2 \mid P3 \xrightarrow{kar1?} P18 \mid P2 \mid P3}\,\text{par-l}}{P15 \mid P2 \mid P3 \mid P4 \xrightarrow{kar1?} P18 \mid P2 \mid P3 \mid P4}\,\text{par-l}}{P15 \mid P2 \mid P3 \mid P4 \mid P5 \xrightarrow{kar1?} P18 \mid P2 \mid P3 \mid P4 \mid P5}\,\text{par-l}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{take\_sticke.P15 \xrightarrow{take\_sticke} P15}\,\text{prefix}}{P14 \xrightarrow{take\_sticke} P15}\,\text{rec}}{P14 \mid P2 \xrightarrow{take\_sticke} P15 \mid P2}\,\text{par-l}}{P14 \mid P2 \mid P3 \xrightarrow{take\_sticke} P15 \mid P2 \mid P3}\,\text{par-l}}{P14 \mid P2 \mid P3 \mid P4 \xrightarrow{take\_sticke} P15 \mid P2 \mid P3 \mid P4}\,\text{par-l}}{P14 \mid P2 \mid P3 \mid P4 \mid P5 \xrightarrow{take\_sticke} P15 \mid P2 \mid P3 \mid P4 \mid P5}\,\text{par-l}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{ker1?.P14 \xrightarrow{ker1?} P14}\,\text{prefix}}{ker1?.P14 + ker2?.P12 \xrightarrow{ker1?} P14}\,\text{choice-l}}{P11 \xrightarrow{ker1?} P14}\,\text{rec}}{P11 \mid P2 \xrightarrow{ker1?} P14 \mid P2}\,\text{par-l}}{P11 \mid P2 \mid P3 \xrightarrow{ker1?} P14 \mid P2 \mid P3}\,\text{par-l}}{P11 \mid P2 \mid P3 \mid P4 \xrightarrow{ker1?} P14 \mid P2 \mid P3 \mid P4}\,\text{par-l}}{P11 \mid P2 \mid P3 \mid P4 \mid P5 \xrightarrow{ker1?} P14 \mid P2 \mid P3 \mid P4 \mid P5}\,\text{par-l}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{fome1wt!.P11 \xrightarrow{fome1wt!} P11}\,\text{prefix}}{fome1wt!.P11 \mid P2 \xrightarrow{fome1wt!} P11 \mid P2}\,\text{par-l}}{fome1wt!.P11 \mid P2 \mid P3 \xrightarrow{fome1wt!} P11 \mid P2 \mid P3}\,\text{par-l}}{fome1wt!.P11 \mid P2 \mid P3 \mid P4 \xrightarrow{fome1wt!} P11 \mid P2 \mid P3 \mid P4}\,\text{par-l}}{fome1wt!.P11 \mid P2 \mid P3 \mid P4 \mid P5 \xrightarrow{fome1wt!} P11 \mid P2 \mid P3 \mid P4 \mid P5}\,\text{par-l}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{conversation1.fome1wt!.P11 \xrightarrow{conversation1} fome1wt!.P11}\,\text{prefix}}{P1 \xrightarrow{conversation1} fome1wt!.P11}\,\text{rec}}{P1 \mid P2 \xrightarrow{conversation1} fome1wt!.P11 \mid P2}\,\text{par-l}}{P1 \mid P2 \mid P3 \xrightarrow{conversation1} fome1wt!.P11 \mid P2 \mid P3}\,\text{par-l}}{P1 \mid P2 \mid P3 \mid P4 \xrightarrow{conversation1} fome1wt!.P11 \mid P2 \mid P3 \mid P4}\,\text{par-l}}{P1 \mid P2 \mid P3 \mid P4 \mid P5 \xrightarrow{conversation1} fome1wt!.P11 \mid P2 \mid P3 \mid P4 \mid P5}\,\text{par-l}}{P \xrightarrow{conversation1} fome1wt!.P11 \mid P2 \mid P3 \mid P4 \mid P5}\,\text{rec}$$

Trace para V3:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{take\_sticka.P15 \xrightarrow{take\_sticka} P15}\,\text{prefix} \quad \Gamma(P14) = take\_sticka.P15}{P14 \xrightarrow{take\_sticka} P15}\,\text{rec}}{P14 \mid P2 \xrightarrow{take\_sticka} P15 \mid P2}\,\text{par-l}}{P14 \mid P2 \mid P3 \xrightarrow{take\_sticka} P15 \mid P2 \mid P3}\,\text{par-l}}{P14 \mid P2 \mid P3 \mid P4 \xrightarrow{take\_sticka} P15 \mid P2 \mid P3 \mid P4}\,\text{par-l}}{P14 \mid P2 \mid P3 \mid P4 \mid P5 \xrightarrow{take\_sticka} P15 \mid P2 \mid P3 \mid P4 \mid P5}\,\text{par-l}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{kar1?.P14 \xrightarrow{kar1?} P14}\,\text{prefix}}{kar1?.P14 + kar2?.P12 \xrightarrow{kar1?} P14}\,\text{choice-l} \quad \Gamma(P11) = kar1?.P14 + kar2?.P12}{P11 \xrightarrow{kar1?} P14}\,\text{rec}}{P11 \mid P2 \xrightarrow{kar1?} P14 \mid P2}\,\text{par-l}}{P11 \mid P2 \mid P3 \xrightarrow{kar1?} P14 \mid P2 \mid P3}\,\text{par-l}}{P11 \mid P2 \mid P3 \mid P4 \xrightarrow{kar1?} P14 \mid P2 \mid P3 \mid P4}\,\text{par-l}}{P11 \mid P2 \mid P3 \mid P4 \mid P5 \xrightarrow{kar1?} P14 \mid P2 \mid P3 \mid P4 \mid P5}\,\text{par-l}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{fome1wt!.P11 \xrightarrow{fome1wt!} P11}\,\text{prefix}}{fome1wt!.P11 \mid P2 \xrightarrow{fome1wt!} P11 \mid P2}\,\text{par-l}}{fome1wt!.P11 \mid P2 \mid P3 \xrightarrow{fome1wt!} P11 \mid P2 \mid P3}\,\text{par-l}}{fome1wt!.P11 \mid P2 \mid P3 \mid P4 \xrightarrow{fome1wt!} P11 \mid P2 \mid P3 \mid P4}\,\text{par-l}}{fome1wt!.P11 \mid P2 \mid P3 \mid P4 \mid P5 \xrightarrow{fome1wt!} P11 \mid P2 \mid P3 \mid P4 \mid P5}\,\text{par-l}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{grab\_sushi\_tray.fome1wt!.P11 \xrightarrow{grab\_sushi\_tray} fome1wt!.P11}\,\text{prefix}}{grab\_sushi\_tray.fome1wt!.P11 \mid P2 \xrightarrow{grab\_sushi\_tray} fome1wt!.P11 \mid P2}\,\text{par-l}}{grab\_sushi\_tray.fome1wt!.P11 \mid P2 \mid P3 \xrightarrow{grab\_sushi\_tray} fome1wt!.P11 \mid P2 \mid P3}\,\text{par-l}}{grab\_sushi\_tray.fome1wt!.P11 \mid P2 \mid P3 \mid P4 \xrightarrow{grab\_sushi\_tray} fome1wt!.P11 \mid P2 \mid P3 \mid P4}\,\text{par-l}}{grab\_sushi\_tray.fome1wt!.P11 \mid P2 \mid P3 \mid P4 \mid P5 \xrightarrow{grab\_sushi\_tray} fome1wt!.P11 \mid P2 \mid P3 \mid P4 \mid P5}\,\text{par-l}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{tw1!.grab\_sushi\_tray.fome1wt!.P11 \xrightarrow{tw1!} grab\_sushi\_tray.fome1wt!.P11}\,\text{prefix} \quad \Gamma(P112) = tw1!.grab\_sushi\_tray.fome1wt!.P11}{P112 \xrightarrow{tw1!} grab\_sushi\_tray.fome1wt!.P11}\,\text{rec}}{P112 \mid P2 \xrightarrow{tw1!} grab\_sushi\_tray.fome1wt!.P11 \mid P2}\,\text{par-l}}{P112 \mid P2 \mid P3 \xrightarrow{tw1!} grab\_sushi\_tray.fome1wt!.P11 \mid P2 \mid P3}\,\text{par-l}}{P112 \mid P2 \mid P3 \mid P4 \xrightarrow{tw1!} grab\_sushi\_tray.fome1wt!.P11 \mid P2 \mid P3 \mid P4}\,\text{par-l}}{P112 \mid P2 \mid P3 \mid P4 \mid P5 \xrightarrow{tw1!} grab\_sushi\_tray.fome1wt!.P11 \mid P2 \mid P3 \mid P4 \mid P5}\,\text{par-l}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{tr0?.P112 \xrightarrow{tr0?} P112}\,\text{prefix}}{tr0?.P112 + tr1?.P111 \xrightarrow{tr0?} P112}\,\text{choice-l} \quad \Gamma(P111) = tr0?.P112 + tr1?.P111}{P111 \xrightarrow{tr0?} P112}\,\text{rec}}{P111 \mid P2 \xrightarrow{tr0?} P112 \mid P2}\,\text{par-l}}{P111 \mid P2 \mid P3 \xrightarrow{tr0?} P112 \mid P2 \mid P3}\,\text{par-l}}{P111 \mid P2 \mid P3 \mid P4 \xrightarrow{tr0?} P112 \mid P2 \mid P3 \mid P4}\,\text{par-l}}{P111 \mid P2 \mid P3 \mid P4 \mid P5 \xrightarrow{tr0?} P112 \mid P2 \mid P3 \mid P4 \mid P5}\,\text{par-l}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{conversation1.P111 \xrightarrow{conversation1} P111}\,\text{prefix} \quad \Gamma(P1) = conversation1.P111}{P1 \xrightarrow{conversation1} P111}\,\text{rec}}{P1 \mid P2 \xrightarrow{conversation1} P111 \mid P2}\,\text{par-l}}{P1 \mid P2 \mid P3 \xrightarrow{conversation1} P111 \mid P2 \mid P3}\,\text{par-l}}{P1 \mid P2 \mid P3 \mid P4 \xrightarrow{conversation1} P111 \mid P2 \mid P3 \mid P4}\,\text{par-l}}{P1 \mid P2 \mid P3 \mid P4 \mid P5 \xrightarrow{conversation1} P111 \mid P2 \mid P3 \mid P4 \mid P5}\,\text{par-l} \quad \Gamma(P) = P1 \mid P2 \mid P3 \mid P4 \mid P5}{P \xrightarrow{conversation1} P111 \mid P2 \mid P3 \mid P4 \mid P5}\,\text{rec}$$

Exercício 2.3 - Implemente V3 usando threads simultâneos

Para a representação em Scala usando threads, cada amigo representa uma thread e conversar representa as ações não críticas. Já ações como pegar a bandeja de sushi, pegar os palitinhos, devolver a bandeja e comer representam as ações críticas, que apresentam concorrência ou um amigo ou todos os amigos(por exemplo, pegar e devolver a bandeja).

```
P1: conversation actions
P3: conversation actions
P5: conversation actions
P4: conversation actions
P2: conversation actions
P1: grab_sushi_tray
P1: return_sushi_tray
P1: eating..
P3: grab_sushi_tray
P3: return_sushi_tray
P3: eating..
P5: grab_sushi_tray
P5: return_sushi_tray
P4: grab_sushi_tray
P4: return_sushi_tray
P4: eating..
P5: eating..
P2: grab_sushi_tray
P2: return_sushi_tray
P2: eating..
```