

Interação e Concorrência

Exercicios Propostos

A98980 Eduardo Cunha

2024

**Problema 1**

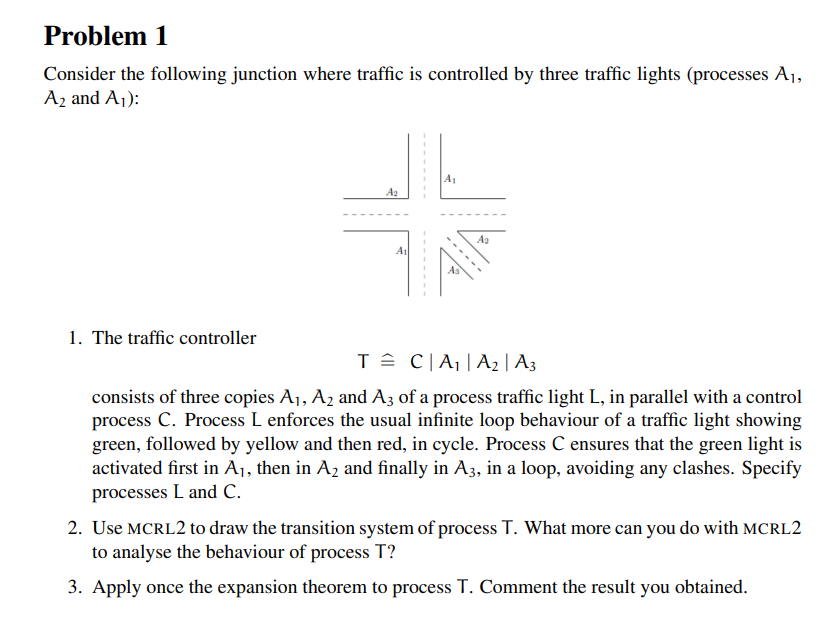
****

Figura 1 - Problema 1

**1.1**

O processo L, considerando o enunciado, visa garantir um ciclo infinito com a troca de luzes correta de um semáforo. Ele consistiria num processo da seguinte maneira: dado uma cor, o sistema altera para a cor seguinte e então invoca novamente o processo L com essa nova cor, assegurando a formação de um ciclo infinito.

Ou seja:

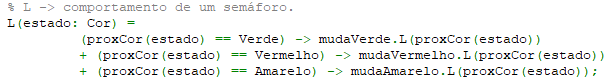


Figura 2 - Processo L

O processo C, conforme descrito no enunciado, representa um procedimento de controle que concede permissão para o semáforo 1 exibir luz verde, seguido pelo semáforo 2 e depois pelo semáforo 3, sucessivamente, de forma a evitar conflitos.

Ou seja:

Uma imagem com texto, Tipo de letra, file, captura de ecrã

Descrição gerada automaticamente

Figura 3 - Processo C

**1.2**

**Uma imagem com texto, Tipo de letra, captura de ecrã

Descrição gerada automaticamente**

**Uma imagem com texto, captura de ecrã, Tipo de letra, número

Descrição gerada automaticamente**

Figura 4 - Código mCLR2 criado

Correndo este código obtemos o seguinte resultado:

Uma imagem com diagrama

Descrição gerada automaticamente

Figura 5 - LTS gerado

Além de elaborar o diagrama de transição do processo T, é possível realizar diversas análises para compreender melhor o comportamento desse processo. Algumas dessas análises incluem:

* Verificação de propriedades, como a deteção de “deadlock” ou de estados inalcançáveis;
* “Model Checking”;
* Simulação, que permite avaliar o comportamento do sistema sob diferentes condições e entradas, proporcionando compreensão sobre seu funcionamento em variados cenários;
* Geração de testes, utilizando o *mCRL2* para automaticamente criar casos de teste;
* Análise de desempenho, possibilitada pelo *mCRL2*, que permite medir o tempo de resposta do sistema e a utilização de recursos, entre outros aspetos.

**1.3**

Conforme estudado nas aulas, o teorema da expansão afirma que todo processo é equivalente à soma de seus derivados. Vamos aplicar esse teorema ao processo "T":

Dado que "T" é composto por vários subprocessos e comunicações, podemos representar T como a soma desses subprocessos:

**T = C(0) + L(Vermelho) + L(Vermelho) + L(Vermelho)**

Agora, expandindo cada um desses subprocessos de acordo com suas definições, obtemos:

C(0) representa o ciclo de controle do semáforo começando pelo estado inicial. Ele faz transições de acordo com as regras definidas no processo C.

Pode ser expandido como:

**C(0) = (0 < 2) -> bloqVerde.desbloqVerde.C(0+1)**

**+ (0 == 2) -> bloqVerde.desbloqVerde.C(0)**

L(Vermelho) representa o comportamento do semáforo quando está no estado vermelho. Ele faz transições para o próximo estado com base nas regras definidas no processo L. Portanto, pode ser expandido como:

**L(C = Vermelho) = (proxCor(C) == Verde) −> mudaVerde.L(proxCor(C))**

**+ (proxCor(C) == Vermelho) −> mudaVermelho.L(proxCor(C))**

**+ (proxCor(C) == Amarelo) −> mudaAmarelo.L(proxCor(C))**

Por fim, "T" é representado como a soma desses subprocessos expandidos, levando em consideração que o processo L(Vermelho) ocorre três vezes consecutivas. Ao expandir T, obtemos uma compreensão mais clara de como o sistema opera e faz transições entre diferentes estados. Em geral, o resultado obtido através do teorema da expansão permite uma análise mais detalhada do comportamento do sistema e proporciona uma compreensão mais profunda de sua dinâmica.

**Problema 2**

**Uma imagem com texto, captura de ecrã, Tipo de letra, documento

Descrição gerada automaticamente**

Figura 6 - Problema 2

**2.1**

**⟨⟨fdee⟩⟩ true:**

Esta expressão representa que após a realização de zero ou mais transições não observáveis, é necessário que ocorra uma transição "fdee". Posto isso, pode haver (ou não) a ocorrência de transições não observáveis que levem a um estado de verificação, após a transição "fdee".

Exemplos:

Seja "t" equivalente ao "tau" (letra grega), que representa as transições não observáveis, temos:

-t.t.fdee

-fdee

**[[-]]false:**

Esta expressão enfatiza que, após zero ou mais transições não observáveis, qualquer transição do conjunto (-), não levará à verificação da propriedade. Pode haver zero ou mais transições observáveis após a transição mencionada, mas isso ainda resultará num estado de não verificação.

Em suma, isso implica que, em qualquer trajetória possível do sistema, se uma ação observável ocorrer a mesma irá ser conduzida a um estado de não verificação.

Exemplos:

Seja "t" equivalente ao "tau" (letra grega), que representa as transições não observáveis e "e" o equivalente a " epsilon" (letra grega) que representa a ausência de transição, temos

-e

-t.t.t.t.t

**2.2**

Nenhuma das opções anteriores expressa corretamente a ideia de “inevitabilidade” da ação “a”.

Na alínea a), embora pareça correta, falha num aspeto crucial: a possibilidade da existência infinita de “taus” (sendo “taus” as transições não observáveis). A formula “a)”, sucintamente, diz que através de zero ou mais ações não observáveis, chegaremos a um “estado” em que existe uma ação observável que leva a um “estado” válido e ao mesmo tempo que qualquer ação observável que não “a”, conduzirá a um “estado” inválido. No entanto, não há garantia de que as ações não observáveis eventualmente parem.

Na alínea “b)”, encontramos uma situação semelhante. Significa essencialmente que, através de transições não observáveis, alcançamos um “estado” (vamos chamá-lo de “F”) no qual qualquer transição observável verifica a fórmula, e ao mesmo tempo qualquer ação visível diferente de “a” não verifica a fórmula. No entanto, assim como na situação anterior, não há garantia de que as transições não observáveis que levam ao “estado F” eventualmente terminem, nem de que as ações invisíveis após este estado terminem também.

Uma possível solução seria implementar a ideia da alínea “a)”, mas com alguma garantia de que as transições não observáveis não são infinitas, algo do género:

**<<->>true /\ [[-a]] false /\ taus finitos**

**2.3**

Bissimilaridade e Equivalência Modal têm uma relação próxima, mas diferem em sua abordagem.

Em essência, dois sistemas são considerados modalmente equivalentes se satisfazem exatamente o mesmo conjunto de propriedades modais em todas as situações possíveis (considerando comportamentos observaveis e não observáveis). Isso significa que, mesmo que os sistemas possam ter estruturas ou implementações diferentes, eles são considerados equivalentes se se comportarem de maneira idêntica em relação às propriedades modais especificadas.

Por outro lado, a bisimilaridade vai além, comparando o comportamento interno dos sistemas. Na prática, dois sistemas são bisimilares se podem imitar um ao outro em todas as situações possíveis. Isso implica que, se um sistema pode executar uma ação em resposta a um estímulo, então o outro sistema também pode realizar uma ação semelhante em uma situação correspondente.

Posto isto é notório a divergencia entre estas relações, visto que dois programas podem facilmente ser equivalentes modalmente e não bissimilares por possuirem uma estrutura interna diferente.

Já, a Equivalência Observacional compara o comportamento externo de dois sistemas, focando apenas nas ações visíveis, ignorando as diferenças internas. Noutras palavras, dois sistemas são considerados observacionalmente equivalentes se se comportarem da mesma maneira, mesmo que possuam estruturas internas ou métodos de execução diferentes.

Conclui-o assim que a Equivalência Observacional e a Equivalência Modal compartilham a ideia de considerar apenas os resultados visíveis dos sistemas, porém diferem na gama de comportamentos que consideram, sendo a Equivalência Modal mais abrangente por também levar em conta comportamentos não observáveis. Desta forma se restringirmos comportamentos não observaveis temos uma convergencia nas definições de Equivalência Observacional e Equivalência Modal.