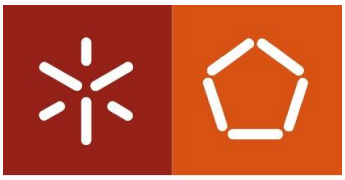

Universidade Minho

Departamento de Informática

Arquitetura e Cálculo
Modelação e análise de sistemas de tempo real

Eduardo Jorge Barbosa A83344
Márcio Sousa A82400

12 de Maio de 2020

Conteúdo

1 Introdução

O trabalho desenvolvido teve como objetivo modelar e analisar um sistema de tempo-real. Quatro aventureiros, com graus de destreza diferentes, têm que atravessar uma ponte frágil, sendo que apenas duas pessoas podem atravessar ao mesmo tempo. O tempo da travessia corresponde ao tempo da pessoa mais lenta. Como restrição adicional, pelo menos um aventureiro terá que levar a lanterna. O problema pode ser visto como um problema de exclusão mútua, onde apenas duas pessoas podem estar na ponte ao mesmo tempo. Neste relatório mostra-se o trabalho desenvolvido com recurso à ferramenta de modelação UPPAAL, baseada em autómatos temporais.

2 Arquitetura da solução

O modelo elaborado consiste em dois autómatos, *Person* e *Flashlight*. Em tempo de execução existem quatro processos do autómato *Person* a correr em paralelo com um processo do autómato *Flashlight*.

2.1 Declarações

O UPPAAL possui declarações de forma a auxiliar a modelação. Uma declaração pode ser um relógio, um *array*, uma constante, ou um valor inteiro limitado. Além destes tipos existem os *channels* que servem para sincronizar dois processos.

2.2 Declarações Globais

2.2.1 Channels

- **Take:** Pegar na lanterna;
- **Release:** Atravessar a ponte;
- **TakeHand:** Juntar-se à pessoa com a lanterna;
- **ReleaseHand:** A pessoa atravessa a ponte sendo iluminada por uma segunda pessoa com a lanterna, na ponte.

2.2.2 Clocks

- **time:** Tempo global do sistema

2.2.3 Variáveis e funções auxiliares

- **side:** *Bool* que indica a localização da lanterna. *False* indica que se encontra no lado esquerdo;
- **buddy:** *Bool* que indica se duas pessoas se encontram a atravessar a ponte;
- **hasFlashlight:** *Bool* que indica se alguma pessoa pegou na lanterna;
- **swap:** Função que troca o valor de um *bool*.

2.3 Declarações da Person

2.3.1 Clocks

- **time**: Tempo local de cada *Person*.

2.3.2 Variáveis e funções auxiliares

- **isOnTheLeft:** Função que verifica se quem tem a lanterna está do lado esquerdo;
- **isOnTheRight:** Função que verifica se quem tem a lanterna está do lado direito;
- **someone:** Função que verifica se o booleano passado é *True* para alguém.

3 Modelo

Por motivos de simplificação e legibilidade, o problema não foi modelado todo apenas num autómato. Em vez disso, optou-se pela elaboração de um autómato para cada uma das entidades presentes, sendo estas cada um dos quatro aventureiros e a lanterna. Numa fase inicial, também se modulou um autómato denominado de *BuddySystem* que representava a interação de duas pessoas atravessarem a ponte. Para simplificar o resultado final foi decidido juntar este autómato com o *Flashlight* (esta decisão é discutida mais à frente).

3.1 Flashlight

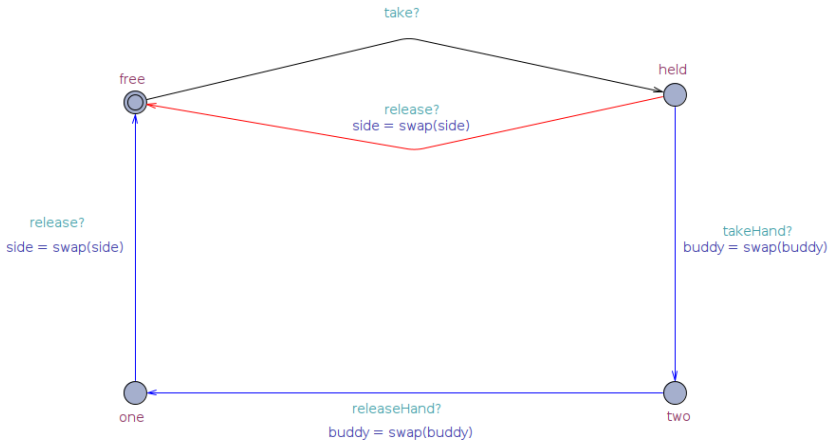

Figura 1: Autómato Flashlight

O autómato que representa a lanterna (Flashlight) possui 4 localizações, nomeadamente, **Free** que representa o estado em que a lanterna não está a ser segurada por ninguém, **Held**, referente a quando alguém está a utilizá-la. **Two** tal como o nome indica refere-se a quando a pessoa que tem a lanterna está num par. Finalmente, a localização **One** representa quando a pessoa que tem a lanterna larga a mão do seu par.

3.2 Person

Neste problema fala-se em quatro aventureiros, cada um deles com capacidades diferentes, o que se demonstra pelo facto de cada um demorar tempos diferentes a atravessar a ponte. Para tornar o problema mais interessante e não limitar os nossos aventureiros a serem apenas números, foi atribuído um nome a cada um deles. O primeiro, "TheDoctor", é aquele que demora apenas 1 minuto a fazer a travessia. Segue-se "RiverSong", a pessoa que demora 2 minutos, "Donna-Noble"com os seus 5 minutos de travessia e, por fim, "RoseTaylor"que com uma travessia de 10 minutos é a aventureira mais lenta. Cada pessoa tem também um *clock* associado.

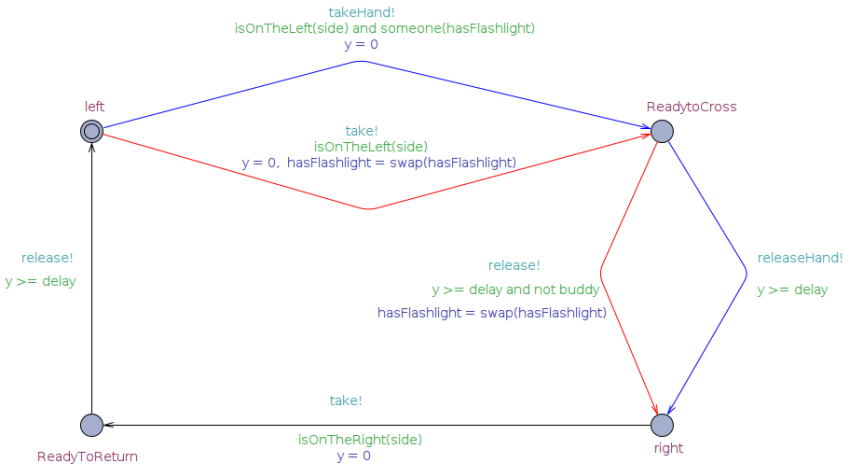

Figura 2: Autómato Person

3.2.1 Localizações

O autómato de cada pessoa (Person) é caracterizado por ter 4 localizações, sendo elas **Left**, que se refere à localização inicial dos aventureiros, **ReadytoCross** que indica que a pessoa se encontra pronta para atravessar a ponte , **Right** que diz respeito ao lado direito da ravina, ou seja, onde se encontram os aventureiros que já atravessaram, e por fim **ReadyToReturn**, que representa o estado de quem está pronto a atravessar de volta para o lado esquerdo da ravina.

3.2.2 Sincronizações, guardas e atualizações

- **Left** → **ReadytoCross**: Existem duas possibilidades de sincronização, que são **take** e **takeHand**. Ambas requerem que a pessoa selecionada esteja do lado esquerdo da ravina, no entanto, **takeHand** requer ainda que já exista uma pessoa que já tenha a lanterna e esteja pronta a atravessar;
- **ReadytoCross** → **Right**: Há aqui também dois eixos marcando as duas opções possíveis. A sincronização **release**, que requer que a pessoa que vai atravessar vá sozinha. A sincronização **releaseHand**, que simboliza atravessar a ponte iluminada pela lanterna de outro aventureiro. É importante realçar que os aventureiros podem "trocar a lanterna de mãos", ou seja quem atravessa via **releaseHand** não tem que ser a mesma pessoa que sincronizou via **take**;
- **Right** → **ReadyToReturn**: Um dos aventureiros que estão do lado direito da ravina pega na lanterna, ficando pronto a atravessar;
- **ReadyToReturn** → **Left**: Representa a travessia de volta ao lado esquerdo e o ato de largar a lanterna no lado esquerdo.

4 Expressões CTL

Verifica-se que a travessia de todos pode ser concluída em 17 minutos mas não menos do que isso. Além destas propriedades outras foram especificadas:

- **Safety:** A[] not deadlock
- **Liveness e Objetivo impossivel:** E<> TheDoctor.right and RiverSong.right and DonnaNoble.right and RoseTaylor.right and time<17

- **Liveness e Objetivo provado:** E<> TheDoctor.right and RiverSong.right and DonnaNoble.right and RoseTaylor.right and time==17
- **Cada aventureiro demora no mínimo o seu tempo a atravessar:**: A[] not (Person.right and time < <theirtime>)
- **Só podem estar duas pessoas na ponte ao mesmo tempo**: *Ver anexos*

5 Conclusões

Este problema é em todo semelhante a um problema de exclusão mútua, sendo que apenas duas pessoas podem estar na ponte ao mesmo tempo. Um problema muito comum em todos os modelos que fazemos foi o *bug* de um aventureiro ficar no "meio da ponte"com a lanterna, ajudando as pessoas a atravessar sem nunca concluir a sua própria travessia. Ultrapassado esse *bug* com a intrdução da variavel *buddy*, o projeto ficou simples. Outras soluções podem existir, como não fazer a distinção de *take* e *takeHand* o grupo ficou satisfeito com o modelo final apresentado.

Relativamente ao UPPAAL, a ferramenta utilizada, temos poucas críticas a apresentar. Não tivemos nenhum problema em testar os nossos modelos nem a verificar as formulas, sendo a ferramenta bastante rápida. Admitimos que o nosso espaço de procura pode ser bastante pequeno. Pelo que vimos na documentação é fácil simplificar os modelos sendo que o UPPAAL oferece *features* como localizações urgentes, *channels* urgentes e localizações *commited*. O maior entrave que encontramos foi enumerar propriedades sobre localizações, como o caso de "apenas duas pessoas encontram-se na ponte ao mesmo tempo". Foi preciso enumerar explicitamente cada processo do tipo **Person**.

6 Refinamento do modelo

Por motivos de limite de páginas esta secção foi colocada após as ***conclusões****, como se fosse um anexo.*

Tal como indicado anteriormente este modelo apresentado é o produto de um processo iterativo de refinamento.

6.1 Modelo inicial

Inicialmente foram considerandos 3 autómatos:

- Person;
- Flashlight;
- BuddySystem.

Segundo a nossa interpretação do problemas, decidimos separar a ideia de "segurar a lanterna"de "atravessar com outro aventureiro". Este baixo nível de abstração deu origem a dois autómatos.

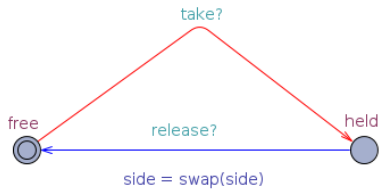

Figura 3: Autómato Flashlight inicial

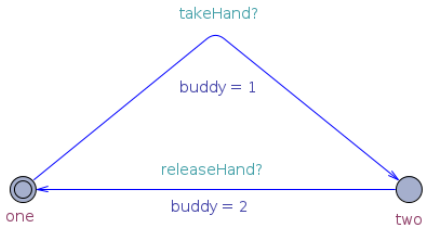

Figura 4: Autómato BuddySystem inicial

É evidente que no autómato **BuddySystem** a transação **takeHand!** não pode ocorrer antes da transação **take?** do autómato **Flashlight**. Não faz sentido, segundo a nossa interpretação, o companheiro juntar-se ao aventureiro com a lanterna antes de um aventureiro pegar na lanterna.

Está lógica introduz alguma complexidade ao autómato **Person**.

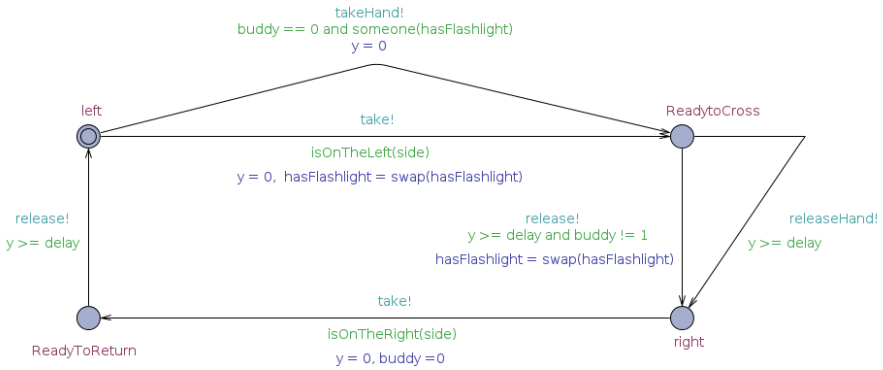

Figura 5: Autómato Person inicial

A grande diferença, comparativamente ao modelo final apresentado, é a variável *buddy* ser do tipo *int [0, 2]*.

- *buddy == 0* Indica que não existe companheiro;
- *buddy == 1* Indica que o companheiro encontra-se pronto a atravessar;
- *buddy == 2* Indica que o companheiro já atravessou.

É necessário saber que o companheiro já atravessou visto que pretendemos que a pessoa com a lanterna, seja ela qual for, seja a última a atravessar. Tendo dito isto, nada impede que os dois aventureiros troquem a lanterna entre eles.

6.2 Modelo Intermédio

Analisando o primeiro modelo alguma complexidade advém de existirem dois autómatos, **Flashlight** e **BuddySystem**, que precisam se estar "sincronizados entre si". Mais uma vez, a transação **takeHand!** no autómato **BuddySystem** não pode ocorrer antes da transação **take?** do autómato **Flashlight**.

Intuitivamente, isto significa que a localização *held* do autómato **Flashlight** corresponde à localização *one* do **BuddySystem**.

Juntando estes dois autómatos, surge:

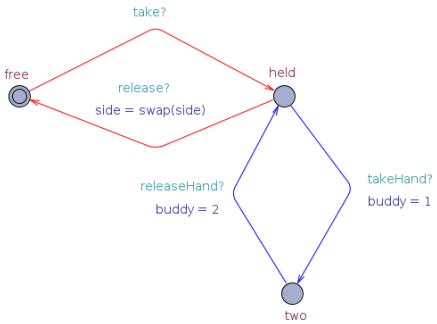

Figura 6: Autómato Flashlight intermédio

Apesar de termos aumentado o nível de abstração e tornado mais explicita a sequencia de transações corretas, a complexidade do autómato **Person** não diminui. Estudando melhor o novo **Flashlight** é possível ver que um traço poderá ser *take?, takeHand?, releaseHand?, takeHand?*

Isto corresponde a atravessar a ponte com um companheiro, deixar o companheiro passar para o lado direito, e **sem terminar a própria passagem e voltar para trás**, ajudar outro aventureiro a atravessar.

6.3 Modelo final

Tendo em conta o problema de um aventureiro ficar no meio da ponte, decidimos tornar esse traço impossível de ocorrer, de forma explicita no autómato.

Surgiu então o modelo final, já apresentado (2 e 1).

6.4 Equivalência dos modelos

Tendo estes três modelos poderá surgir a questão se realmente eles são equivalentes. Uma opção para provar que realmente têm um comportamento equivalente seria provar que são bissimilares. No entanto, o modelo não é de todo trivial e a prova seria enorme. Decidiu-se então desenhar os grafos das transações possíveis, quando os autómatos são compostos em paralelo. Caso os grafos sejam iguais, podemos concluir que os nossos modelos são equivalentes (apesar da igualdade ser uma restrição mais forte do que a necessária). Nestes grafos abstraiu-se a passagem do tempo visto que as restrições relativas aos relógios são iguais em todos os modelos. Mais uma vez, fazer isto para quatro aventureiros seria moroso, decidiu-se então fazer apenas para 2. No entanto, é possível de verificar que os grafos gerados são bastante simétricos, o que nós da um grau de confiança que sobre a equivalência dos modelos mesmo para 4 aventureiros. Os grafos podem ser consultados em anexo 7, 8 e 9.

7 Anexos

7.1 CTL - Só se encontram duas pessoas na ponte ao mesmo tempo

7.1.1 A atravessar para o lado esquerdo só podem estar duas pessoas na ponte ao mesmo tempo

A[] not ((TheDoctor.ReadytoCross and RiverSong.ReadytoCross and DonnaNoble.ReadytoCross and RoseTaylor.ReadytoCross) or (TheDoctor.ReadytoCross and RiverSong.ReadytoCross and DonnaNoble.ReadytoCross) or (TheDoctor.ReadytoCross and RiverSong.ReadytoCross and RoseTaylor.ReadytoCross) or (RiverSong.ReadytoCross and DonnaNoble.ReadytoCross and RoseTaylor.ReadytoCross))

7.1.2 A atravessar para o lado esquerdo só podem estar duas pessoas na ponte ao mesmo tempo

A[] not ((TheDoctor.ReadyToReturn and RiverSong.ReadyToReturn and DonnaNoble.ReadyToReturn and RoseTaylor.ReadyToReturn) or (TheDoctor.ReadyToReturn and RiverSong.ReadyToReturn and DonnaNoble.ReadyToReturn) or (TheDoctor.ReadyToReturn and RiverSong.ReadyToReturn and RoseTaylor.ReadyToReturn) or (RiverSong.ReadyToReturn and DonnaNoble.ReadyToReturn and RoseTaylor.ReadyToReturn))

7.2 Equivalência dos modelos

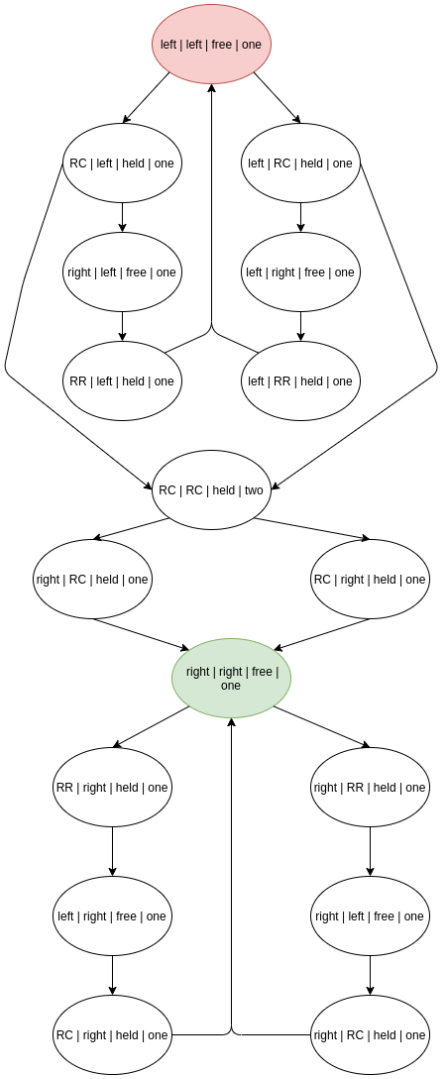

Figura 7: Grafo do modelo inicial

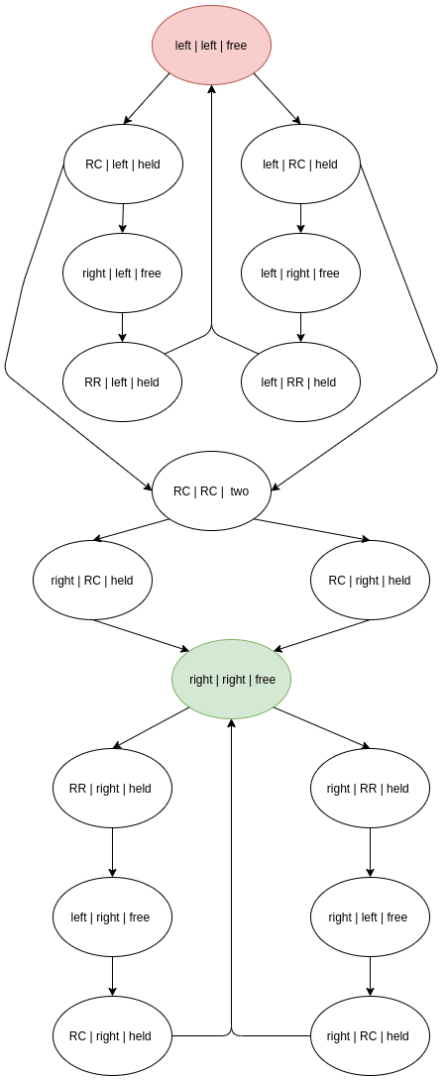

Figura 8: Grafo do modelo intermédio

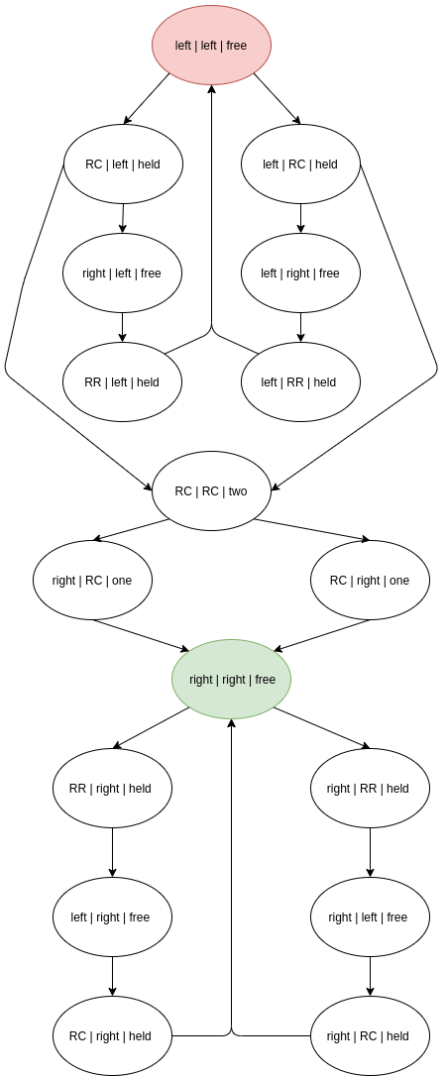

Figura 9: Grafo do modelo final