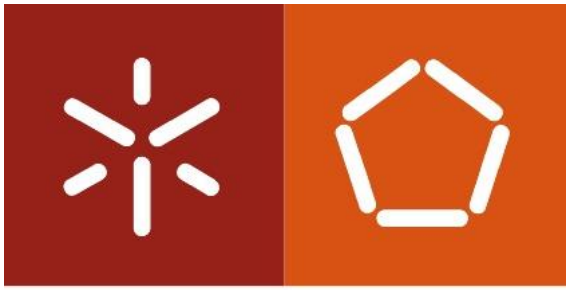


Especificação e Modelação

Mestrado Integrado em Engenharia Informática 2020/2021

Universidade do Minho

14 de janeiro de 2021

ENUNCIADO II - TERMINATION DETECTION IN A RING

Grupo 5

Ana Almeida, A83916

Jorge Cerqueira, A85573

1 Breve Introdução

No âmbito da unidade curricular de Especificação e Modelação foi proposto aos alunos que efetuassem a tradução de uma especificação feita em `TLA+` para `Electrum`. Assim sendo, decidimos optar pelo enunciado II - *Termination detection in a ring*, que consiste numa especificação, denominada por `EWD 840`, do algoritmo de `Dijkstra` para a problemática de *termination detection* (detetar quando a computação/execução está completa, ou seja, todos os processos inativos e nenhuma mensagem em trânsito) *in a ring* (topologia de rede em que cada nodo se coneta exatamente a dois outros nodos, formando um único caminho contínuo entre cada nodo - um anel).
O módulo `EWD840.tla`, encontrado na página de *github* fornecida, contém a especificação do algoritmo em `TLA+`. Assim, criámos os módulos `EWD840.ele` e `EWD840.thm` (em anexo) que contêm a especificação e o *theme* do algoritmo em `Electrum`, respetivamente.

2 Principais decisões e Dificuldades sentidas

Ao longo da realização desta tradução foram tomadas decisões para facilitar a execução da especificação, tais como:

- No estado inicial (`Init`) foi assumido que o `probe` ainda não tinha começado, o que levou a colocarmos `no Token` e a tratar das cores no predicado `startProbe`:

```
(Token.token_color)' = White
(Token.position)' = min[Node].next_node;
```

- Adicionar *signatures* relativas à transição que ocorreu, de maneira a facilitar a visualização das instâncias e, consequentemente, a correção dos modelos;
- Permitir que mais do que um nodo se possa tornar `Passivo` na transição para o próximo estado, o que leva a que o sistema possa terminar mais rapidamente.

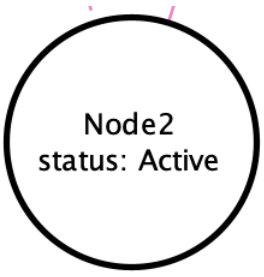
Todavia também foram encontradas algumas dificuldades/desafios, tais como:

- Após uma primeira análise, considerámos que seria necessário ter a noção da numeração (o que seria um desafio), todavia, após uma leitura atenta em `https://www.cs.utexas.edu/users/EWD/ewd08xx/EWD840.PDF`, apercebemo-nos que, nas regras, apenas era usada a relação de *maior* e *menor*, por isso, considerámos que ter uma ordem nos nodos seria suficiente - `open util/ordering[Node]`;
- Inicialmente modelámos o sistema de forma a que, em cada transição, apenas pudesse ser enviada uma mensagem, no entanto, eventualmente, percebemos que isto seria um problema sendo que, em algumas ocasiões, há múltiplos envios singulares de mensagens, o que deixaria o sistema no mesmo estado.

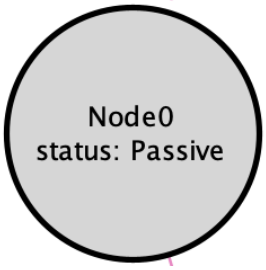
3 Ilustração do *theme* criado

No `Electrum`, as instâncias são representadas de forma gráfica. Existe uma versão *standard*, todavia optámos por personalizar a nossa recorrendo a um *theme*, alterando as formas e cores dos nodos e arestas de acordo com as várias operações. É de ressalvar que os *themes* são bastante úteis quando se lida com sistemas complexos. Assim, no nosso caso, optámos por representar:

- `Nodos` - círculo branco, caso a sua cor seja `Branco` ou com um círculo cinzento (para facilitar a visão), caso a sua cor seja `Preto`;
- `Relações` - relação de `posição` com uma seta tracejada laranja e relação `próximo nodo` com uma seta a cheio rosa;
- `Tokens` - retângulo branco tracejado, caso ainda `não tenha` existido trocas de mensagens ou com um retângulo preto tracejado, caso já `tenha` existido trocas de mensagens;
- `Transições` - hexágonos amarelos com o nome da respetiva transição.

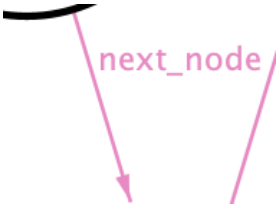


(a) Nodo `Branco` com *status* `Ativo`



(b) Nodo `Preto` com *status* `Passivo`

(c) Relação `posição`



(d) Relação `próximo nodo`

(e) Token antes da existência de `mensagem`

(f) Token após existência de `mensagem`

(g) Transição que mantém o estado `inalterado`

(h) Transição que indica que um `Token` foi passado para o próximo nodo

(i) Transição que `envia` mensagem

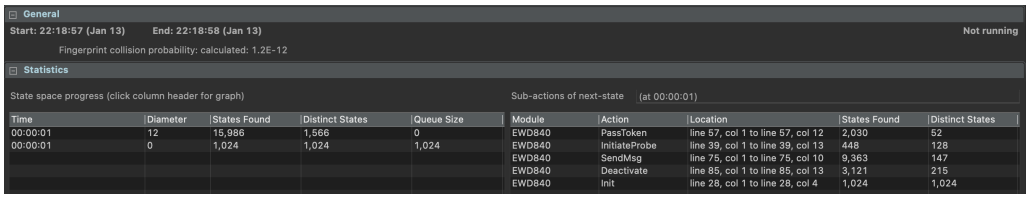
(j) Transição que passa o estado de um ou mais nodos para `Passivo`

(k) Transição que inicia o `probe`

Figura 1: Componentes do *theme* criado

4 Análise de *performance*

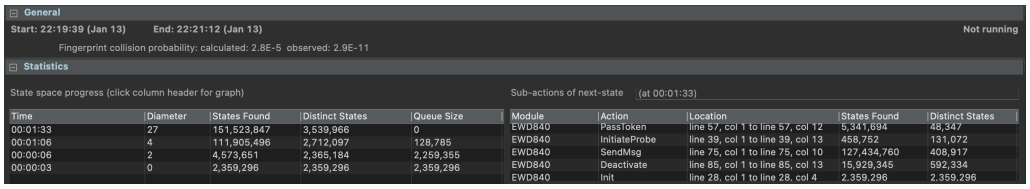
Nesta secção é feita uma comparação entre a *performance* das duas especificações (`TLA+` e `Electrum`), para melhor compreendermos a potência destas ferramentas. De acordo com o ficheiro `EWD840_anim.cfg`, o número máximo de nodos é 9, caso contrário são demasiados estados iniciais para o `TLA+` conseguir tratar. Assim, efetuámos uma comparação entre `N` (`TLA+`) e `Node` (`Electrum`) iguais a 4 e 9[1].


General
Start: 22:18:57 (Jan 13) End: 22:18:58 (Jan 13) Not running
Fingerprint collision probability: calculated: 1.2E-12
Statistics
State space progress (click column header for graph)
Sub-actions of next-state (at 00:00:01)

Time	Diameter	States Found	Distinct States	Queue Size
00:00:01	12	15,986	1,566	0
00:00:01	0	1,024	1,024	1,024

Module	Action	Location	States Found	Distinct States
EWD840	PassToken	line 57, col 1 to line 57, col 12	2,030	52
EWD840	InitiateProbe	line 39, col 1 to line 39, col 13	448	128
EWD840	SendMsg	line 75, col 1 to line 75, col 10	9,363	147
EWD840	Deactivate	line 85, col 1 to line 85, col 13	3,121	215
EWD840	Init	line 28, col 1 to line 28, col 4	1,024	1,024

Figura 2: `TLA+` para `Node` = 4 com *scope unbounded*


General
Start: 22:19:39 (Jan 13) End: 22:21:12 (Jan 13) Not running
Fingerprint collision probability: calculated: 2.8E-5 observed: 2.9E-11
Statistics
State space progress (click column header for graph)
Sub-actions of next-state (at 00:01:33)

Time	Diameter	States Found	Distinct States	Queue Size
00:01:33	27	151,523,847	3,539,966	0
00:01:06	4	111,905,496	2,712,097	128,785
00:00:06	2	4,573,651	2,365,184	2,259,355
00:00:03	0	2,359,296	2,359,296	2,359,296

Module	Action	Location	States Found	Distinct States
EWD840	PassToken	line 57, col 1 to line 57, col 12	5,341,694	48,347
EWD840	InitiateProbe	line 39, col 1 to line 39, col 13	458,752	131,072
EWD840	SendMsg	line 75, col 1 to line 75, col 10	127,434,760	408,917
EWD840	Deactivate	line 85, col 1 to line 85, col 13	15,929,345	592,334
EWD840	Init	line 28, col 1 to line 28, col 4	2,359,296	2,359,296

Figura 3: `TLA+` para `Node` = 9 com *scope unbounded*

```
Executing "Check liveness for 4 but 1..15 steps"
  Solver=sat4j Decomposed=disabled Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  1..15 steps. 111770 vars. 5760 primary vars. 301400 clauses. 35346ms.
  No counterexample found. Assertion may be valid. 7378ms.
```

Figura 4: `Electrum` para `Node` = 4 com *scope* de 15 *steps*

```
Executing "Check liveness for 9 but 1..15 steps"
  Solver=sat4j Decomposed=disabled Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
  1..15 steps. 334270 vars. 18735 primary vars. 1013555 clauses. 209819ms.
  No counterexample found. Assertion may be valid. 54367ms.
```

Figura 5: `Electrum` para `Node` = 9 com *scope* de 15 *steps*

Como se pode observar nas imagens, a ferramenta `TLA+` para `N` igual a 4 e 9 demorou 1 segundo e 1 minuto e 33 segundos, respetivamente. Já a ferramenta `Electrum` para `Node` igual a 4 e 9 demorou 42,7 segundos e 4 minutos e 40 segundos, respetivamente. Como se pode constatar, a ferramenta `Electrum` é mais lenta a executar do que a ferramenta `TLA+`, muito possivelmente por:

- A ferramenta `TLA+` ser de mais baixo nível;
- Os *operadores* extra que o `Electrum` possui;
- Ter de gerar variáveis iniciais, enquanto que em `TLA+` esse *setup* é feito manualmente.

É de ressalvar que a verificação em `TLA+` é sempre *unbounded* no horizonte temporal, ou seja, para um número arbitrário de *steps*[2]. Ainda assim, como a ferramenta `Electrum` tem um desempenho mais lento com um *scope* de 15 *steps*, é expectável que tenha um desempenho também mais lento com um *scope unbounded*.

[1] Para o `Electrum`, foi executada a cláusula `check liveness`, pois esta vai analisar os estados todos, tal como acontece na execução com `TLA+`.

[2] Para fazer uma comparação mais justa entre as duas ferramentas poderíamos usar um *scope unbounded* em `Electrum`, todavia isto obrigaria à instalação do *model checker* `nuXmv`.