

Número: _____ Nome: _____

1. TLA+ (10 pontos)

Considere o seguinte algoritmo distribuído cujo objectivo é calcular o máximo identificador de todos os nós envolvidos: os nós começam com um valor idêntico ao seu identificador único; a qualquer momento podem fazer *broadcast* desse valor para todos os outros nós; quando um nó recebe um valor de outro nó atualiza o seu próprio valor para conter o máximo do valor recebido e do valor anterior. Podemos especificar este protocolo em TLA+ da seguinte forma, onde N é o número de nós, id é uma função que devolve o identificador de cada nó, $value$ é uma função variável que armazena o valor de cada nó, $sent$ é uma função variável que indica se um nó já enviou o seu valor para todos os outros, e $inMsgs$ é uma função variável que contém as mensagens recebidas por cada nó.

----- MODULE maximum -----

EXTENDS Naturals, FiniteSets, TLC

CONSTANTS N, id

ASSUME $N > 0 \wedge id \in [0..N-1 \rightarrow \text{Nat}] \wedge \forall i, j \in 0..N-1 : i \neq j \Rightarrow id[i] \neq id[j]$

VARIABLES $value, sent, inbox$

TypeOK == $\wedge value \in [0..N-1 \rightarrow \text{Nat}]$
 $\wedge sent \in [0..N-1 \rightarrow \text{BOOLEAN}]$
 $\wedge inMsgs \in [0..N-1 \rightarrow \text{SUBSET}(\text{Nat})]$

Init == ...

Send(i) == ...

Rcv(i) == ...

Next == $\vee i \in 0..N-1 : \text{Send}(i) \vee \text{Rcv}(i)$

vars == $\langle\langle value, sent, inMsgs \rangle\rangle$

Spec == Init $\wedge [][\text{Next}]_{\text{vars}} \wedge \text{WF}_{\text{vars}}(\text{Next})$

=====

- Especifique o predicado Init , que caracteriza o estado inicial do algoritmo (1 ponto).
- Especifique a ação Send , onde o nó i faz broadcast do seu valor para todos os outros (2 pontos).
- Especifique a ação Rcv , onde o nó i recebe um valor (2 pontos).
- Especifique as seguintes propriedades expectáveis neste algoritmo (3 pontos):
 - Uma vez verdadeiro, o $sent$ de cada nó fica verdadeiro para sempre.
 - O valor de cada nó nunca decresce.
 - Quando o algoritmo termina (i.e. todos os broadcasts foram efectuados e todas as mensagens lidas) o valor de todos os nós é igual ao identificador máximo.
- Especifique propriedades (falsas) cujos contra-exemplos correspondam a possíveis traços onde se verifica o seguinte (2 pontos):
 - Os nós fazem broadcast dos seus ids por ordem (começando no nó 0 até ao $N-1$).
 - Todos os nós fazem broadcast dos seus ids antes de qualquer valor ter sido recebido.

Número: _____ Nome: _____

2. Why3 (10 pontos)

O foco deste exercício é o mesmo algoritmo distribuído considerado no Exercício 1.

Consideramos o seguinte modelo em WhyML de um sistema de n_nodes nós, tendo cada um uma fila de espera de mensagens (note que na especificação em TLA+ se usava um conjunto para armazenar as mensagens recebidas, mas aqui será usada uma lista) e duas variáveis locais `sent` e `value`. Cada nó tem associado um identificador único, e as mensagens trocadas são precisamente estes identificadores.

```
type node = int
val constant n_nodes : int
axiom n_nodes_ax : 2 <= n_nodes

type id = int
val function id node : id
axiom uniqueIds : forall i j :node. id i = id j <-> i=j

type msg = id
type world = { value : map node id ;
               sent : map node bool ;
               inMsgs : map node (list msg) }

predicate initWorld (w:world) = . . .
predicate inv (w:world) = . . .

let rec ghost function broadcast (w:world) (n:int) (sndr:node) : map node (list msg)
  requires { 0<=n<=n_nodes /\ 0<=sndr<n_nodes }
  ensures { ... }
  variant { n }
= if n=0 then w.inMsgs
  else let inb = broadcast w (n-1) sndr
       in if sndr = n-1 then inb else inb[n-1 <- inb[n-1] ++ Cons (id sndr) Nil]

predicate send_enbld (w:world) (h:node) = 0<=h<n_nodes /\ not w.sent[h]
let ghost function send (w:world) (h:node) : world
  requires { send_enbld w h }
  ensures { inv w -> inv result }
= { value = w.value ;
    sent = w.sent[h<-true] ;
    inMsgs = broadcast w n_nodes h }

predicate rcv_enbld (w:world) (h:node) = . . .
let ghost function rcv (w:world) (h:node) : world
  requires { rcv_enbld w h }
  ensures { inv w -> inv result }
= . . .

goal correctness : forall w :world. reachable w -> . . .
```

- a) Complete a definição do predicado `initWorld` que identifica as configurações iniciais do sistema (2 pontos).
- b) A função `broadcast` é auxiliar para a definição da ação `send`. Constrói recursivamente o map contendo as filas de mensagens dos diferentes nós depois de um broadcast efectuado pelo nó `sndr`, num sistema com configuração `w`.
Acrescente a esta função um conjunto de pós-condições que descrevam em termos lógicos o que a função faz (utilize a palavra reservada `result` para se referir ao resultado da função) (2 pontos).
- c) Complete a definição do predicado `rcv_enblId` e da ação `rcv` (2 pontos).
- d) Complete a definição do goal `correctness` que exprime a seguinte propriedade: quando o algoritmo termina (i.e. todos os broadcasts foram efectuados e todas as mensagens lidas), o valor de todos os nós é igual ao identificador máximo. Considere que está definida a constante `maxId_global`, tal como na formalização do algoritmo de Chang-Roberts (2 pontos).
- e) Finalmente, defina um invariante indutivo que permita a prova da propriedade anterior.
Para isso, considere o que pode ser dito sobre o `value` de um nó `k` e sobre as suas mensagens, tendo em conta os broadcasts que tenham já sido feitos. O invariante deverá incluir também informação sobre o limite superior (`maxId_global`) para os valores de todas as mensagens em espera, bem como dos valores dos `values` dos diferentes nós (2 pontos).