

Algorithm 1

*x, y: shared registers,
initially y = FALSE*

```
repeat forever
0:   remainder exiti
1:   x := i;
2:   if y ≠ FALSE then goto 1;
3:   y := TRUE;
4:   if x ≠ i then goto 1;
5:   critical entryi
6:   critical exiti
7:   y := FALSE;
8:   remainder entryi
end repeat
```

(1)

Algorithm 2

f: local register

```
repeat forever
0:   remainder exiti
1:   f[i] := FALSE;
2:   if ∃j > i f[j] = TRUE then goto 1;
3:   f[i] := TRUE;
4:   if ∃j > i f[j] = TRUE then goto 1;
5:   if ∃j < i f[j] = TRUE then goto 5;
6:   critical entryi
7:   critical exiti
8:   f[i] := FALSE;
9:   remainder entryi
end repeat
```

(2)

Fig. 1 Due Algoritmi per la Mutua Esclusione (viene riportato lo pseudo-codice per il processo *i*): (1) Lamport's Style Mutual Exclusion; (2) Burns's Mutual Exclusion.

Esercitazioni del 15/12/2010

Corso di Logica 2

Per ciascuno dei due algoritmi proposti verificare automaticamente con un model checker se valgono la mutua esclusione e l'assenza di deadlock¹ (nel caso si usi un model checker a stati finiti, limitare il numero dei processi a due o a tre). Nel caso in cui tali proprietà non valgano, fornire le relative tracce di errore.

¹ Quest'ultima proprietà si formula dicendo che, se c'è un processo nella regione 'trying' e nessun processo in regione critica, allora in ogni esecuzione possibile prima o poi qualche processo entrerà in sezione critica.