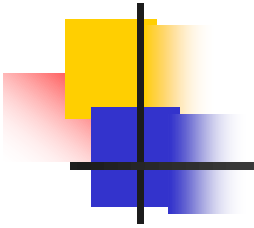
VPC 16-17
Esercizio di model-checking con GreatSPN e NUSMV e esercizio di confronto con algebra dei processi

Prof.ssa Susanna Donatelli
Universita' di Torino
www.di.unito.it
susi@di.unito.it



Visione di insieme

Valuteremo la correttezza di diverse soluzioni per la mutua esclusione proposte dal libro di testo di M. Ben-Ari "Principles of Concurrent and Distributed Programming", cap. 3

Tutti gli algoritmi devono essere implementi in NuSMV e in GreatSPN, le tre proprietà del prossimo lucido devono essere definite in LTL e CTL per NuSMV e in CTL per GreatSPN

I primi due algoritmi devono essere sviluppati anche in algebra dei processi e si chiede di confrontare la soluzione a rete di Petri con quella di algebra dei processi usando le nozioni di equivalenza viste a lezione

Particolare attenzione a: progresso, fairness, consistenza dei risultati ottenuti per lo stesso algoritmo nei vari formalismi

La mutua esclusione

Definizione del problema:

1. Ognuno degli N processi esegue un loop infinito di istruzioni divise in due gruppi: la sezione critica e la sezione non critica
2. La correttezza di un algoritmo di mutua esclusione è definita dalla congiunzione delle seguenti condizioni:
   - **1. Mutua esclusione**: le istruzioni delle sezioni critiche di due o più processi non possono essere eseguite in modo interfogliato
   - **2. Assenza di deadlock**: Se qualche processo cerca di accedere alla regione critica eventualmente un processo potrà farlo
   - **3. Assenza di starvation individuale**: Se un processo cerca di accedere alla regione critica eventualmente quel processo potra' farlo
3. Assumiamo che le variabili usate dal protocollo di accesso siano usate solo dal protocollo di accesso
4. C'è progresso nella regione critica (se un processo inizia l'esecuzione in regione critica alla fine terminerà tale esecuzione)
5. Non si richiede progresso da parte dei processi nelle istruzioni che non appartengono alla regione critica

La mutua esclusione (3.2)

Prima soluzione (testo del Ben-Ari): una singola variabile turn, quando turn vale 1 entra il processo 1, quando turn vale due entra il processo 2. Può essere più semplice assumere che la variabile turn possa valere p o q anzichè 1 o 2

Algorithm 3.2: First attempt

integer turn ← 1

p		q	
	loop forever		loop forever
p1:	non-critical section	q1:	non-critical section
p2:	await turn = 1	q2:	await turn = 2
p3:	critical section	q3:	critical section
p4:	turn ← 2	q4:	turn ← 1

La mutua esclusione (3.5)

Il Ben-Ari propone anche una versione semplificata, minimale.

Algorithm 3.5: First attempt (abbreviated)	
integer turn ← 1	
p	**q**
loop forever	loop forever
p1: await turn = 1	q1: await turn = 2
p2: turn ← 2	q2: turn ← 1

La mutua esclusione (3.6)

Il Ben-Ari propone questa ulteriore soluzione, basata su due variabili.

Algorithm 3.6: Second attempt

boolean wantp ← false, wantq ← false

p		q	
	loop forever		loop forever
p1:	non-critical section	q1:	non-critical section
p2:	await wantq = false	q2:	await wantp = false
p3:	wantp ← true	q3:	wantq ← true
p4:	critical section	q4:	critical section
p5:	wantp ← false	q5:	wantq ← false

La mutua esclusione (3.8)

Il Ben-Ari propone anche questa terza soluzione, sempre basata su due variabili. Quest soluzione inverte le istruzioni di setting di wantp e di attesa su wantq (e viceversa per l'altro processo).

Algorithm 3.8: Third attempt	
boolean wantp ← false, wantq ← false	
p	**q**
loop forever	loop forever
p1: non-critical section	q1: non-critical section
p2: wantp ← true	q2: wantq ← true
p3: await wantq = false	q3: await wantp = false
p4: critical section	q4: critical section
p5: wantp ← false	q5: wantq ← false

La mutua esclusione (3.9)

Il Ben-Ari propone anche questa quarta soluzione, sempre basata su due variabili. Questa soluzione evita l' "intestardimento" dei processi nel voler entrare in regione critica, settando e resettando la propria variabile "want" per permettere all'altro processo di passare.

Algorithm 3.9: Fourth attempt	
boolean wantp ← false, wantq ← false	
p	**q**
loop forever	loop forever
p1: non-critical section	q1: non-critical section
p2: wantp ← true	q2: wantq ← true
p3: while wantq	q3: while wantp
p4: wantp ← false	q4: wantq ← false
p5: wantp ← true	q5: wantq ← true
p6: critical section	q6: critical section
p7: wantp ← false	q7: wantq ← false

La mutua esclusione (3.10)

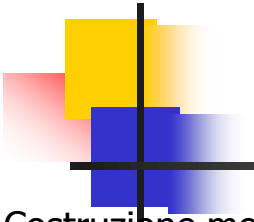
La combinazione del primo e del quarto tentativo portano all'algoritmo di mutua esclusione noto come "algoritmo di Dekker".

Algorithm 3.10: Dekker's algorithm

```
boolean wantp ← false, wantq ← false
integer turn ← 1
```

	p		**q**
	loop forever		loop forever
p1:	non-critical section	q1:	non-critical section
p2:	wantp ← true	q2:	wantq ← true
p3:	while wantq	q3:	while wantp
p4:	if turn = 2	q4:	if turn = 1
p5:	wantp ← false	q5:	wantq ← false
p6:	await turn = 1	q6:	await turn = 2
p7:	wantp ← true	q7:	wantq ← true
p8:	critical section	q8:	critical section
p9:	turn ← 2	q9:	turn ← 1
p10:	wantp ← false	q10:	wantq ← false

Cosa dovete fare



Costruzione modello nuSMV e analisi delle proprietà 1, 2 e 3 per tutti gli algoritmi dati. Potete definire anche ulteriori proprietà per assicurarvi che il modello rispetti effettivamente gli algoritmi dati – usare LTL e CTL. Confrontare e giustificare i risultati sulla base della verifica delle proprietà e degli eventuali contro-esempi e witnesses.

Costruzione modello rete di Petri e analisi delle proprietà 1, 2 e 3 (versione CTL) per tutti gli algoritmi dati. Potete definire anche ulteriori proprietà per assicurarvi che il modello rispetti effettivamente gli algoritmi dati. Usare RGMEDD e CTL. Confrontare e giustificare i risultati sulla base dellà verifica delle proprietà e degli eventuali contro-esempi e witnesses. Confrontare 3.2 con 3.5 e 3.6 con 3.8 usando tecniche di riduzione strutturale

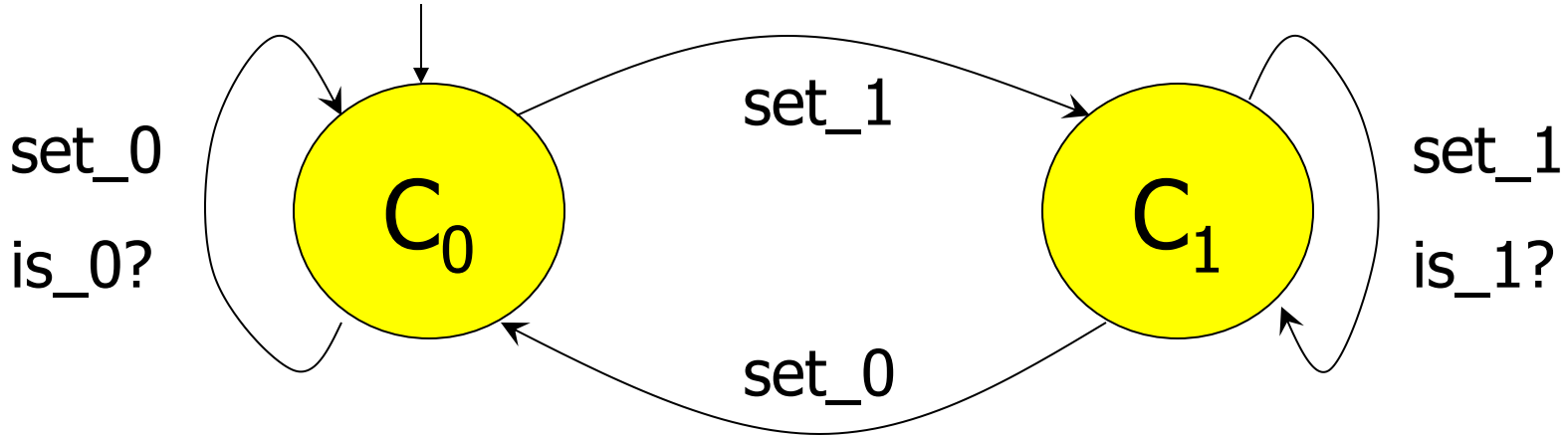
Costruzione modello ad algebra dei processi (CCS o CSP, a scelta) per i primi due algoritmi (3.2 e 3.5) e confronto con modello a rete di Petri calcolando le varie equivalenze fra il derivation graph e il reachability graph del 3.2 e fra il derivation graph e il reachability graph del 3.5. Confronto fra 3.2 e 3.5 in algebra dei processi usando le equivalenze: assumere che le azioni che non sono presenti in uno dei due algoritmi siano modellati da azioni non osservabili tau e, se serve, usare la versione di bisimulazione estesa a considerare la azioni tau.

Attenzione a fairness e alla corretta resa, nel modello, delle condizioni di progresso: c'è sicuramente progresso solo in regione critica e nel protocollo di accesso. Verificare, su almeno un modello, quali sono invece le conseguenze di richiedere il progresso in regione critica ma non nel protocollo di accesso.

Creare una tabella di confronto dei risulti ottenuti con NuSMV e con GreatSPN e commentare/giustificare eventuali discrepanze nei risultati o nei contro-esempi/witnesses

Modeling binary variable

Suggerimento per la modellazione delle variabili in algebra dei processi



$$C_0 = is\_0? \cdot C_0 + set\_1 \cdot C_1 + set\_0 \cdot C_0$$

$$C_1 = is\_1? \cdot C_1 + set\_0 \cdot C_0 + set\_1 \cdot C_1$$