# Algoritmi e Principi dell'Informatica
# Prima Prova in Itinere

20 Novembre 2012

Il tempo a disposizione è di **2 ore**

## Esercizio 1 (punti 7/15-esimi)

Il funzionamento delle reti di Petri (PN) è normalmente definito attraverso la cosiddetta "semantica interleaving" in cui si assume che una sola transizione per volta, tra quelle abilitate in una certa marcatura, possa scattare. La letteratura però ha proposto anche definizioni diverse della semantica delle PN. Tra queste la semantica "true concurrency", con lo scopo di evidenziare che *alcune* attività modellate dalle PN possono avvenire in parallelo, ammette lo scatto contemporaneo di più transizioni, tra quelle abilitate (purché ciò non violi proprietà di mutua esclusione). Un caso particolare di tale semantica è quella a "parallelismo massimo", che assume che *tutte* le transizioni abilitate in una certa marcatura scattino contemporaneamente, sempre rispettando eventuali vincoli di mutua esclusione.

***Esercizio 1.1***

Si forniscano definizioni precise, preferibilmente ricorrendo a notazioni formali (abilitazione, scatto delle transizioni, ecc.) della semantica "true concurrency" generale e di quella a "parallelismo massimo". Per semplicità si assuma che ogni transizione possa scattare una sola volta ad ogni passo, ossia che non siano ammessi scatti multipli e contemporanei della stessa transizione, anche se il numero di token nei posti di ingresso lo permetterebbe.

***Esercizio 1.2***

Si dica, giustificando brevemente la risposta, se le affermazioni seguenti sono vere o false:

a) Nella semantica "true concurrency" il comportamento della rete è deterministico

b) Nella semantica "a parallelismo massimo" il comportamento della rete è deterministico

c) Se una marcatura è raggiungibile da una marcatura iniziale secondo la semantica "interleaving" lo è anche secondo la semantica "true concurrency" e viceversa

d) Se una marcatura è raggiungibile da una marcatura iniziale secondo la semantica "interleaving" lo è anche secondo la semantica a "parallelismo massimo" e viceversa.

## Esercizio 2 (punti 5/15-esimi)

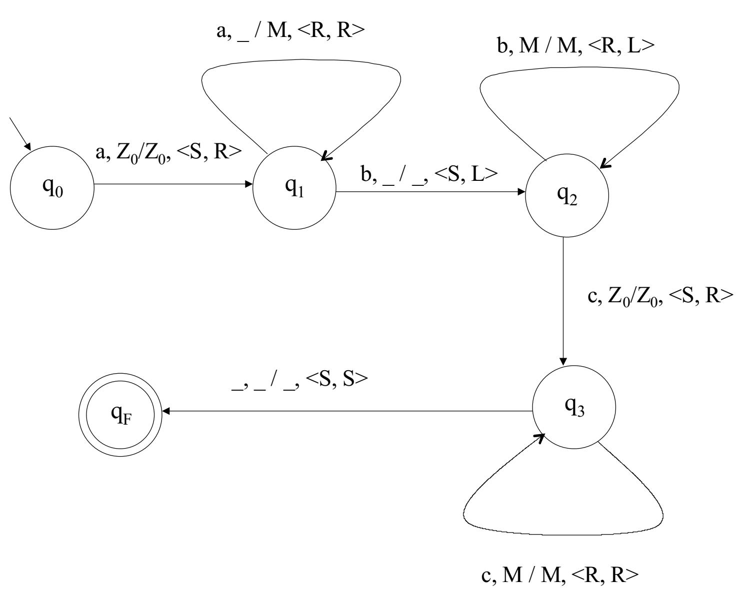
Si forniscano opportune pre- e post-condizioni, codificate mediante formule del prim'ordine, per specificare il comportamento desiderato di un (frammento di) programma che, avendo in memoria due array non vuoti di caratteri, A e B di lunghezza rispettivamente $n$ e $m$, con $m < n$, stabilisca se tutti i caratteri presenti in B compaiono anche in A con lo stesso numero di ripetizioni.

## Esercizio 3 (punti 5/15-esimi)

Data una classe C di automi, un automa A in C viene detto *minimo* nella classe C se non esistono altri automi in C che riconoscano il linguaggio L(A) riconosciuto da A e che abbiano un numero inferiore di stati rispetto ad A.

Rispondere ai quesiti seguenti fornendo un'opportuna giustificazione:

1. E' decidibile il problema di stabilire se un generico automa a stati finiti è minimo nella classe degli automi a stati finiti?
2. E' decidibile il problema di stabilire se, *fissati l'alfabeto* Γ *della pila e la lunghezza massima della stringa che ad ogni mossa l'automa può mettere in cima alla pila*, un generico automa a pila deterministico è minimo nella classe degli automi a pila deterministici?
   **Suggerimento**: può essere di aiuto tenere presente che il problema dell'equivalenza tra automi a pila deterministici è stato recentemente dimostrato decidibile (si veda la Tabella 8.1 a pagina 352 del testo)
3. E' decidibile il problema di stabilire se la macchina di Turing mostrata sotto è minima nella classe delle macchine di Turing?



**Tracce di Soluzioni**

**Esercizio 1.1**

In primo luogo si può estendere la definizione di transizione abilitata a un insieme di transizioni abilitate. Escludendo il caso più generale in cui una transizione possa scattare diverse volte in un colpo solo, si definisce, per una marcatura M:

Enab(M) = $\{t_1, t_2, \dots t_k\}$ se per tutte e sole le transizioni $t_1, t_2, \dots t_k$ i corrispondenti posti in ingresso (Preset) sono marcati.
Successivamente, la true concurrency definisce lo scatto multiplo e nondeterministico di un set di transizioni abilitate nel modo seguente

M |---- $\{t_{i1} \dots t_{ir}\}$ M' se e solo se $\{t_{i1} \dots t_{ir}\} \subseteq$ Enab(M)

$\wedge$

*token_sufficienti*($t_{i1} \dots t_{ir}$)

/* ossia P contiene abbastanza token da far scattare tutte le transizioni in $\{t_{i1} \dots t_{ir}\}$ da esso uscenti; ciò evidentemente implica una scelta nondeterministica ma vincolata su quali transizioni possano scattare concorrentemente tra quelle abilitate. */

$\wedge$

/* la marcatura di M' è definita nel modo tradizionale sommando gli effetti di eventuali transizioni concorrenti*/

dove

*token_sufficienti*($t_{i1} \dots t_{ir}$)

è definita come

$\forall$ P $\in \cup$ Preset $\{t_{i1} \dots t_{ir}\}$

$M(P) \geq |\{ t_{j1} \dots t_{jk}\}|$ dove $\{ t_{j1} \dots t_{jk}\} = \{t_{i1} \dots t_{ir}\} \cap$ Postset(P)

dove Postset(P) indica l'insieme di transizioni uscenti da P.

La semantica a parallelismo massimo è un caso particolare di quella true concurrency, per ottenere la quale basta aggiungere la clausola

$\neg\exists$ $\{t_{i1}' \dots t_{ir}'\}$ tale che

$(\{t_{i1} \dots t_{ir}\} \subset \{t_{i1}' \dots t_{ir}'\} \subseteq$ Enab(M)

$\wedge$

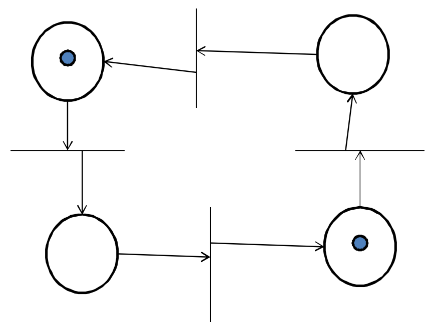
*token_sufficienti*($t_{i1}' \dots t_{ir}'$))

Si noti che anche con questa restrizione il comportamento della rete rimane nondeterministico e sarebbe anzi più corretto parlare di *parallelismo massimale piuttosto che massimo.*

**Esercizio 1.2**

Da quanto sopra le affermazioni a) e b) sono false.
La c) è vera perché la semantica interleaving non fa che mettere in sequenza scatti di transizioni "impacchettati" in un unico scatto globale dalla semantica true concurrency.
La d) invece è falsa: la semantica interleaving (vedi anche punto precedente) può "sviluppare" qualsiasi sequenza di scatti multipli, ma obbligando a far scattare

sempre in parallelo tutte le transizioni abilitate non è detto che si possano produrre tutte le marcature ottenibili con la semantica interleaving, come mostrato dal controesempio seguente:



Nella semantica a parallelismo massimo i due token continuano a "rincorrersi" in modo sincrono senza mai raggiungersi né superarsi; cosa che non avviene nella semantica interleaving.

**Esercizio 2**

Usando la notazione del metodo di Hoare la specifica del programma desiderato può essere espressa nel modo seguente:

$\{n > m > 0\}$
P
$\{OK \Leftrightarrow \forall\ x\ (Carattere(x) \wedge \exists i\ (0 < i \leq m \wedge B[i] = x) \Rightarrow\ CountB(x) = CountA(x))\}$

dove le funzioni CountA(x) e CountB(x) sono definite nel modo seguente:
CountA(x) = Count_Aux_B(x, n) e
CountB(x) = Count_Aux_B(x, m).

Count_Aux_B(x, y) è a sua volta definita da:
Count_Aux_B(x, 0) = 0
$\wedge$
$\forall y(0 < y \leq m \Rightarrow (\ B[y] = x \Rightarrow Count\_Aux\_B(x, y) = Count\_Aux\_B(x, y - 1) + 1)$
$\wedge$
$(\neg\ B[y] = x \Rightarrow Count\_Aux\_B(x, y) = Count\_Aux\_B(x, y - 1)))$
La definizione di Count_Aux_A(x, y) è analoga.

**Esercizio 3**

1. Il primo problema è decidibile, in quanto è sufficiente enumerare tutti gli automi a stati finiti (costruibili sull'alfabeto di simboli di A), in ordine crescente rispetto al numero di stati, fino al primo automa (escluso) con un numero di stati pari al numero di stati in A. Questo tipo di enumerazione è certamente possibile ed è simile a quello utilizzato per enumerare le macchine di Turing. A questo punto, per ciascun automa in questa enumerazione è sufficiente verificarne l'equivalenza con A, proprietà decidibile grazie alle proprietà di chiusura della famiglia e della decidibilità della *emptiness* di un linguaggio regolare: se l'equivalenza sussiste, A non è minimo, altrimenti lo è.
   **NB**: In alternativa e in maniera più immediata il problema può essere dimostrato decidibile facendo ricorso alla procedura di minimizzazione per gli automi a stati finiti e quindi semplicemente confrontando il numero di stati dell'automa di partenza e dell'automa minimo corrispondente.

2. Con un procedimento del tutto simile al precedente (e mantenendo *fissato l'alfabeto di pila* dell'automa) risulta decidibile anche il secondo problema (tenendo conto, come stabilito nella tabella 8.1, che è decidibile anche l'equivalenza tra automi a pila deterministici e che limitando la lunghezza della stringa depositabile in pila esiste solo un numero finito di automi con un determinato numero di stati, a meno di una irrilevante ridenominazione degli stati).
   Si noti che se l'alfabeto di pila non fosse fissato sarebbe sempre possibile ridurre il numero di stati (fino a due) memorizzando l'informazione in essi contenuta in opportuni simboli depositati sulla pila.
   Si noti inoltre che ogni automa a pila può essere trasformato in uno equivalente che non depositi in pila più di k caratteri per ogni mossa, purché k sia > 1 e al prezzo di un aumento del numero di stati.

3. Il quesito del terzo problema è una domanda chiusa, che non dipende da alcun parametro, pertanto il problema è certamente decidibile.