Analisi della starvation mediante ASM

Gennaro (Rino) Vessio
gennaro.vessio@uniba.it

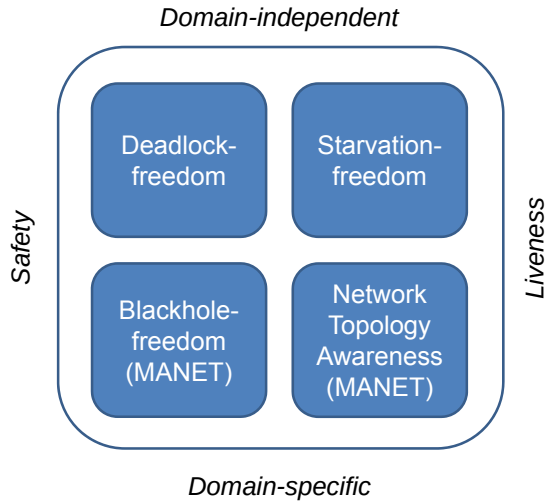
Verifica formale

La **verifica** è l'attività di dimostrare che il sistema in analisi soddisfa determinate proprietà

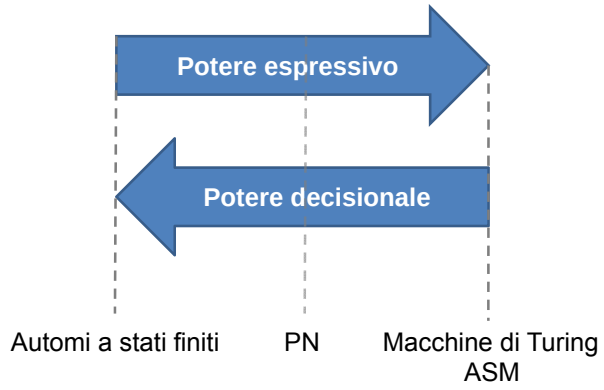
Una **proprietà** esprime un comportamento desiderato del sistema

L'approccio formale propone di condurre tale verifica su un **modello formale** del sistema

Classi di proprietà



Un dilemma annoso



Analisi mediante ASM

Allo stato dell'arte, l'analisi di proprietà mediante ASM fa ricorso al **model checking**

Limitazioni:

- perdita di potere espressivo
- difficile uso di logiche temporali

Idea

Dotare le ASM di definizioni **operazionali** di proprietà (come le reti di Petri)

Pro:

- no perdita di potere espressivo
- no logiche temporali

Contro:

- l'analisi non è completamente automatizzabile

Starvation

In letteratura **non** ne **esiste** una definizione univoca

In generale è l'**impossibilità** di un processo distribuito di compiere progressi computazionali

Alcune definizioni

Dijkstra: «*although all individual eating actions take only a finite period of time, a person may kept hungry for the rest of his days*»

Lamport: «*a process eventually receives service*»

Alpern & Schneider: «*a process makes progress infinitely often*»

Alcune definizioni

Tanenbaum: «*some policy is needed to make a decision about who gets which resource when. This policy, although seemingly reasonable, may lead to some processes never getting service even though they are not deadlocked*»

Pnueli: «*the starvation-freedom property relies on justice assumptions that none of the processes idles forever in some location*»

Baier & Katoen: «*each waiting process will eventually enter its critical section*»

La nostra definizione

«*One of the processes is unable to make computational progress because it is idle during the waiting for a desired external service*»

Condizioni necessarie

Condizioni necessarie alla starvation:

1. Esiste un insieme di processi distribuiti {p_1, …, p_n}
2. L'esecuzione di un processo p_i può essere completata solo quando un servizio offerto da un processo p_j è disponibile
3. L'esecuzione di p_i è forzata ad attendere

Condizioni necessarie (ASM)

Condizioni necessarie alla starvation:

1. Esiste una DASM composta da un insieme di agenti distribuiti {a_1, …, a_n}
2. L'esecuzione di un agente a_i dipende da un altro agente a_j
3. L'esecuzione di a_i è forzata ad attendere

Seconda condizione

– L'esecuzione di un agente a_i dipende da un altro agente a_j

Esiste almeno una regola la cui condizione riguarda funzioni che esprimono la **dipendenza** da a_j

Terza condizione

– L'esecuzione di a_i è forzata ad attendere

Tale regola genera una computazione che non cambia il predicato sugli stati che rappresenta la condizione di **attesa**

Predicati sugli stati

Un predicato ϕ su uno stato s è una formula del prim'ordine definita sulle location in s, tale che $s \vDash \phi$ [BPV15]

È l'equivalente di una funzione di etichettatura: ci aiuta a esprimere la **semantica** degli stati

Cinque filosofi a cena

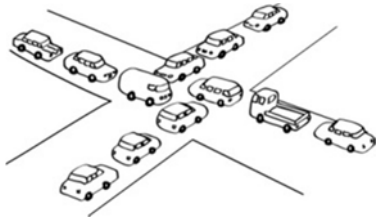
Cinque filosofi siedono a cena di fronte a un piatto di spaghetti

Ciascun filosofo, nella sua vita, pensa o mangia, rigorosamente con due forchette

Cinque filosofi a cena

Se non neghiamo la resource holding abbiamo un **deadlock**

Tutti i filosofi impugnano la propria forchetta e attendono indefinitamente l'altra



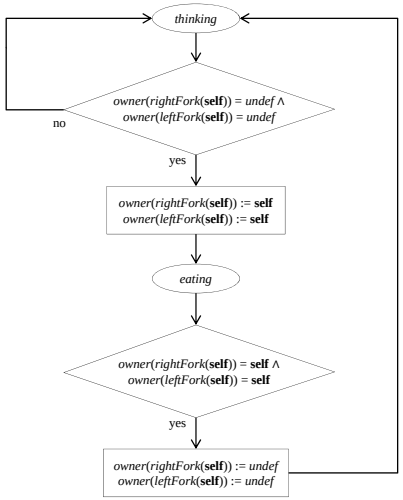
Cinque filosofi a cena

Neghiamo la resource holding: obblighiamo ciascun filosofo a impugnare le forchette solo quando entrambe libere

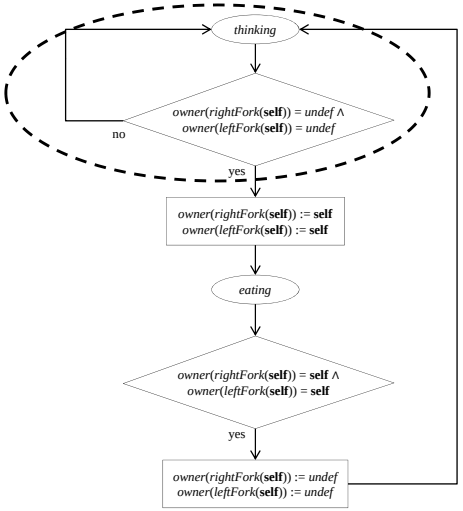
Sorge un nuovo problema: **starvation**

Le forchette potrebbero non essere mai libere

Modelliamo in ASM



Modelliamo in ASM

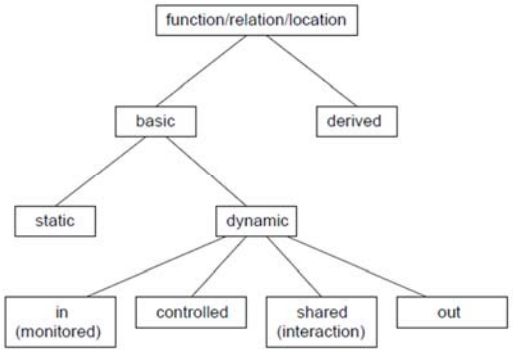


Vulnerable rule

È una regola nella forma **`if`** `cond` **`then`** `update_true` **`else`** `update_false` tale che:

1. il valore di verità di `cond` dipende da una o più funzioni di rischio
2. un update tra `update_true` e `update_false` genera una computazione per cui il predicato sugli stati di rischio è soddisfatto
3. la computazione evolve attraverso l'altro update verso uno stato in cui il predicato di rischio non è più soddisfatto

La natura delle funzioni di rischio

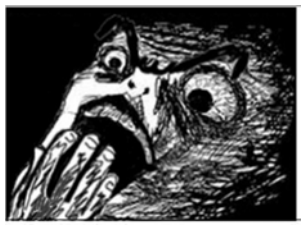


Semi-decidibilità

La vulnerable rule **cattura** le condizioni necessarie alla starvation

L'**assenza** di vulnerable rule implica la starvation-freedom del modello

Il viceversa **no**



Riferimenti

[BPV15] Bianchi A., Pizzutilo S., Vessio G. Applying Predicate Abstraction to Abstract State Machines. In: 20th International Conference on Exploring Modelling Methods for Systems Analysis and Design (EMMSAD 2015), Stockholm, Sweden, 283-292, Springer (2015)