

Interessanti Proprietà dei Sistemi Distribuiti Cooperativi

Proprietà

1

Analisi di Sistemi Distribuiti Cooperativi (1)

- Per sistemi complessi è necessario analizzare caratteristiche specifiche, che possono dare origine a problemi
- Ad es.
 - deadlock
 - transizioni non attivabili
 - mutua esclusione
 - cooperazione
 - ...

Proprietà

2

Analisi di Sistemi Distribuiti Cooperativi (2)

- L'analisi delle proprietà computazionalmente interessanti di sistemi distribuiti cooperativi è spesso ardua
- La tesi di fondo del corso è che **l'analisi delle proprietà può essere grandemente semplificata, se svolta su un opportuno modello del sistema**

Proprietà

3

Classificazione delle Proprietà (1)

- È possibile classificare le proprietà computazionalmente interessanti in due categorie
 - **Domain Independent** properties
 - **Domain Specific** properties

Proprietà

4

Classificazione delle Proprietà (2)

- Per entrambe le precedenti categorie è possibile inoltre individuare due classi
 - **Safety** properties: something bad never happens
 - il sistema non entra mai in stati indesiderati
 - **Liveness** properties: something good eventually does happen
 - il sistema prima o poi sicuramente entra in uno stato desiderato

Esempi (1)

- Tipiche domain independent safety properties sono
 - Correttezza parziale
 - Assenza di deadlock
 - In un sistema di mutua esclusione la “bad thing” è la simultanea entrata di due processi nella regione critica
 - ...

Esempi (2)

- Tipiche domain independent liveness properties sono
 - Raggiungibilità di uno stato (terminazione)
 - Assenza di starvation
 - In un sistema di mutua esclusione la “good thing” è il fatto che prima o poi ogni processo entri nella regione critica

Raggiungibilità

- Supponendo l'esecuzione di un sistema come l'attraversamento di un insieme di stati, possiamo immaginare ogni stato come la condizione per l'esecuzione di un'attività computazionale
- È utile sapere se ogni stato del sistema può **essere raggiunto** a partire dallo stato iniziale
 - Se uno stato non fosse raggiungibile, allora le attività computazionali ad esso associate non potrebbero mai essere eseguite
 - Quello stato è **inutile**

(II) Limitatezza delle Risorse

- È necessario conoscere **prima** della esecuzione
 - quali e quante risorse sono necessarie - (un)boundness
 - se il numero di risorse è costante o variabile - conservativeness
- Si dovrà poi definire un'opportuna politica per l'acquisizione/rilascio delle risorse necessarie

Vitalità (1)

- **Quasi-vitalità (quasi-liveness)** di una componente logica
 - È la possibilità di far eseguire almeno una volta la componente a partire dallo stato iniziale
 - Quando uno stato perde questa proprietà, l'evento associato è inessenziale ai fini del funzionamento del sistema
- Un sistema è quasi-vitale se lo sono tutte le sue componenti

Vitalità (2)

- **Vitalità (liveness)** di una componente logica
 - È la possibilità di far eseguire la componente a partire dallo stato iniziale
- Un sistema è vitale se lo sono tutte le sue componenti
- La vitalità garantisce l'assenza di deadlock

Reversibilità (1)

- Un sistema ha comportamento reversibile se per ogni stato S_K raggiungibile dallo stato iniziale S_0 , allora S_0 è raggiungibile da S_K
- La reversibilità indica la possibilità di comportamento ciclico del sistema
 - È particolarmente utile per il recovery in caso di guasti

Completezza

- Un sistema è completo se, a partire dallo stato iniziale, è possibile giungere a ogni stato finale
- Può essere valutata mediante la raggiungibilità
- Indica la capacità del sistema di rappresentare tutte le possibili configurazioni volute

Multimodalità

- Indica la capacità del sistema di raggiungere lo stesso stato in modi diversi

Complessità

- È il numero massimo di percorsi distinti che permettono di raggiungere gli stati finali a partire da quello iniziale