Communicating Sequential Processes

Communicating Sequential Processes

- CSP è una notazione per descrivere sistemi in cui agenti operano in parallelo e comunicano scambiandosi messaggi
- Ottimo strumento per studiare la concorrenza
  - In questo corso CSP sarà presentato come strumento per la modellizzazione
- Poco adatto per comunicare con parti interessate che NON siano informatici

Scopo

- Notazione per la descrizione delle interazioni tra agenti
- Particolarmente utile nel caso di sistemi in cui alcune componenti svolgono attività computazionali influenzate dalle attività computazionali svolte da altre componenti
  - Adatto per modellizzare componenti di sistemi distrubuiti

Applicazioni

- Modellizzazione di
  - interazione utente-sistema
  - protocolli (ad es. sicurezza)
  - sistemi di controllo safety-critical
- Specifica di sistemi

Interazione = Comunicazione

- Presupposto: l'interazione tra due componenti di un sistema, o di un sistema con l'esterno avviene mediante un'opportuna comunicazione
- Nell'ambito di CSP la comunicazione prende la forma di eventi o azioni visibili
  - All'interno di un processo vengono anche svolte attività interne, invisibili, che hanno effetti indiretti sul mondo esterno

Nozioni di base (1)

- CSP fornisce una modalità
  - per descrivere gli stati attraversati dal sistema
  - quali azioni sono eseguite
- Sia
  - Σ l'insieme di tutti gli eventi visibili di un processo
  - τ l'azione svolta internamente
    - per il momento è sufficiente l'astrazione secondo cui il processi svolge un'unica azione

Nozioni di base (2)

- Nel seguito useremo indifferentemente le espressioni
  - processo / programma
  - comunicazione / azione visibile

Processi equivalenti

- Chiamiamo equivalenti due diversi processi ciascuno dei quali produce un proprio pattern di azioni visibili e tali pattern non possono essere distinti l'uno dall'altro
- Di conseguenza, la caratteristica di uno specifico processo è data dalle sue forme di comunicazione

Processo Stop

- Il processo più semplice è Stop, che non svolge alcuna azione
  - nessuna azione visibile
  - nessuna azione interna
- E' il più semplice processo equivalente ad ogni processo che non comunica con altri
- E' una modalità per esprimere il deadlock

Prefixing

- Dato un processo P e una azione a, allora il processo a -> P (a then P) è il programma
  - che svolge l'azione a
  - e poi si comporta come P
- Se in e out sono due azioni in Σ, allora il processo in -> out -> Stop svolge le azioni in successione e poi si ferma
  - In realtà l'ambiente del processo potrebbe scegliere di non accettare un'azione

Ricorsione (1)

- L'operatore di prefixing realizza il concetto di sequenza tra processi
- La ricorsione può essere realizzata componendo adeguatamente chiamate sequenziali allo stesso processo
- Ad es:
  - Alt = to -> fro -> Alt
  - Dalt = to -> fro -> to -> fro -> Dalt
  - Malt1 = to -> Malt2
  - Malt2 = fro -> Malt1
  - Nalt = to -> from -> Dalt

Ricorsione (2)

- Alt è la ricorsione più semplice; Dalt è più complessa, ma ottiene lo stesso risultato
- Malt1, Malt2 implementano una mutua ricorsione
- Nalt non è ricorsivo su sé stesso, ma rimanda alla ricorsione di Dalt

Forma più compatta

- È possibile rappresentare in forma più compatta il processo Alt precedente secondo la seguente notazione

  μP.to -> fro -> P

Scelta di azioni (1)

- Il comportamento di un processo può essere conseguenza delle azioni svolte
- Sia A un sottoinsieme di Σ, allora il processo

  ?x : A -> P(x)

  (scegli un x in A e P prosegue di conseguenza)

  permette all'ambiente di scegliere una tra tutte le possibili azioni visibili in A

Scelta di azioni (2)

- Un caso particolare di scelta è

  RUN_A = ?x:A-> RUN_A

  – lo stato RUN_A risultante da ogni scelta è sempre lo stesso
  – Si tratta di un processo a comportamento costante

Parametrizzazione

- Scegliendo un a ∈ A allora il processo prosegue comportandosi come P(a)
  – rappresenta la parametrizzazione del processo rispetto all'azione a
- a si comporta come un parametro per P(a)
- Strumento per la parametrizzazione del comportamento del processo
- Può essere usato per decidere come comportarsi a seguito dei possibili casi scelti

Scelta tra due processi (1)

- L'operatore □ permette di scegliere di continuare l'esecuzione secondo quanto previsto da due processi distinti

  (?x : B->P(x)) □ (?x : C->P(x))

- Se A = B ∪ C allora la seguente indica che i due processi si comportano in modo equivalente

?x : A->P(x) = (?x : B->P(x)) □ (?x : C->P(x))

Scelta tra due processi (2)

- Se B = Ø allora, dal momento che non viene fornita alcuna possibilità all'ambiente, si ottiene ?x : B->P(x) = Stop
- Quindi, essendo A = A ∪ Ø, si ottiene

  (?x : A->P(x)) = (?x : A->P(x)) □ Stop

- In generale, P □ Stop è equivalente a Stop

Scelta tra due processi (3)

- Il processo risultante da

  (?x : B->P(x)) □ (?x : C->P(x))

- non genera mai ambiguità
- Infatti, se B e C
  - sono disgiunti allora la loro combinazione fornisce una scelta senza ambiguità
  - non sono disgiunti, il comportamento prosegue ancora senza ambiguità perché è lo stesso in entrambi i casi

Scelta non deterministica

P ⊓ Q

Si comporta come P oppure come Q, indipendentemente dalla volontà dell'utente

- La scelta è determinata dalle azioni interne
  - Si può supporre che esista più di una azione interna τ_1, τ_2, …, τ_n

Confronto tra operatori di scelta

- L'operatore □ rappresenta una scelta compiuta dall'ambiente che interagisce con il processo in run time
- L'operatore ⊓ rappresenta una scelta compiuta dall'implementazione del processo

Operatori di I/O (1)

- Gli operatori di scelta permettono di caratterizzare le operazioni di input e output
  - l'input è espresso da ?
  - l'output è espresso da !

Operatori di I/O (2)

- L'operatore ? permette di indicare che l'ambiente fornisce elementi di input sotto forma di eventi che intervengono nell'esecuzione del processo
  - in?x ->P indica che l'evento x, che assume valori nell'insieme definito da in, è posto in prefixing rispetto a P
  - in rappresenta il channel rispetto a cui avviene l'input

Operatori di I/O (3)

- L'operatore ! permette di indicare che il processo produce elementi di output sotto forma di eventi che saranno poi consumati dall'ambiente
  - P->out!x -> Q indica che l'evento x, che assume valori nell'insieme definito da out, è un evento prodotto da P, ed è posto in prefixing rispetto a Q
  - out rappresenta il channel rispetto a cui avviene l'output

CSP e FSM Deterministiche (1)

- È possibile specificare una FSM in CSP
  - In corrispondenza di ogni stato s_i della FSM è associato un insieme di possibili azioni A_i
    - che determinano le transizioni in uscita dallo stato
    - tali transizioni determinano l'esecuzione di un processo
  - se $x \in A_i$ allora $P'_i(x)$ è il risultato corrispondente all'esecuzione di x nello stato s_i
  - Quindi la FSM è descritta da

$$P_i = ?x : A_i \rightarrow P'_i(x)$$

CSP e FSM Deterministiche (2)

- Le scelte tra le possibili azioni in ogni stato sono presentate come un'opportuna combinazione di input, output e □

Trace (1)

- Una traccia (trace) di un processo è la sequenza finita di simboli che mostra gli eventi in cui il processo è stato coinvolto in un determinato periodo di tempo
- Rappresenta la sequenza delle comunicazioni visibili svolte dal processo

Trace (2)

- I due processi

$$P \Box Q \text{ e } P \sqcap Q$$

  mostrano le stesse tracce
- Se analizzati rispetto alle tracce, risultano indistinguibili
  - ma in sostanza possono essere molto differenti

Trace (3)

- Esempio: consideriamo

  (a ->P) ⊓ Stop e (a ->P)□ Stop

- Il deadlock potrebbe avvenire

  immediatamente, oppure successivamente

Time

- Un ulteriore operatore di scelta è time
- P ▷t Q rappresenta che viene svolto ciò che offre P per t unità di tempo e se nulla viene scelto si prosegue come Q
- P ▷t Q determina le stesse tracce che si otterrebbero applicando i due precedenti operatori di scelta

if – then – else (1)

- Si tratta della scelta effettuata sulla base dello stato interno a un processo
- Esempio processo firewall

FW(s)=in?x ->

(if valid(x, s) then out!x -> FW(newstate(s, x))

else FW(newstate(s, x)))

- Nota:
  - in?x indica l'input di un valore x scelto nell'insieme in
  - out!x è l'output del valore x

if – then – else (2)

- Notazione alternativa
- P◁b▷Q è equivalente a if b then P else Q