PCD Module 2.6

Formalismi visuali

Formalismi visuali

Servono per descrivere rigorosamente modelli di sistemi concorrenti per mezzo di un qualche tipo di diagramma visuale. Possono rappresentare, ad esempio, le dinamiche di un sistema o la struttura di task e dipendenze.

Sono utili sia per la specifica dei requisiti che per l'analisi e la progettazione. L'analisi è formale quando è formalmente specificata.

In questo capitolo:

- Reti di Petri
- Diagrammi a stati
- Diagrammi di attività

Reti di Petri

- Modello formale e astratto del flusso di informazioni; descrive ed analizza i flussi di informazioni e controllo nel sistema, in particolare nei sistemi che esibiscono attività concorrenti e asincrone.
- Utilizzo principale: modellazione di sistemi ad eventi in cui è possibile che alcuni eventi possano occorrere concorrentemente, ma sono presenti vincoli sulla concorrenza, precedenza o frequenza di queste occorrenze.

Grafo delle reti di Petri

- Grafo bipartito che rappresenta una rete di Petri,
  - due tipi di nodi: **posti** (cerchi), **transizioni** (barre)
  - connnessi da archi direzionati

- Modella le proprietà statiche del sistema

Token e marcatori

- Una rete di Petri ha proprietà dinamiche che risultano dalla sua esecuzione
  - l'esecuzione di una rete è controllata dalla posizione e dal movimento di marcatori chiamati

token, indicati da punti neri.
    - una rete con token è chiamata **rete di Petri contrassegnata**

Regole di esecuzione

- I token sono mossi dall'esecuzione di transizioni nella rete
- Una transizione deve essere *abilitata* affinchè venga eseguita
- Una transizione è abilitata quando tutti i suoi *posti di input* hanno un token all'interno
- Le transizioni vengono eseguite rimuovendo i token di abilitazione dai relativi posti di input e generando nuovi token che sono depositati nel posto di output della transizione. (vedi figure, si capisce di piu)

Marcatori

La distribuzione dei token in una rete di Petri marcata definisce lo stato della rete ed è chiamata marcatore. Il marcatore può cambiare come risultato dell'esecuzione di una transizione. Transizioni diverse possono essere eseguite con risultati di marcatori diversi (non-determinismo inerente).

(vedi figure)

Transizioni di demarcazione

- transizione **sorgente**: senza alcun posto di input, produce solo token ($|| \rightarrow \circle$)
- transizione **rubinetto**: senza alcun posto di output, consuma solamente token ($\circle \rightarrow ||$)

Archi pesati

- Una variante consiste nell'uso di archi pesati
    - una transizione è abilitata se ogni posto di input p di t è marcato con almeno $w(p,t)$ token, ovvero con il peso dell'arco da p a t.
    - l'esecuzione di una transizione abilitata t rimuove $w(p,t)$ token da ogni posto di input p di t e aggiunge w token ad ogni posto di output p di t, con $w(t,p)$ peso dell'arco da t a p.

Esempio: reazione H2O

Modellazione con reti di Petri

Le RdP possono essere usate per modellare naturalmente sistemi concorrenti in termini di **eventi**, **condizioni** e le **relazioni** tra di essi. Ovvero: in un sistema, in un qualsiasi momento, ci saranno certe condizioni. Il fatto che tali condizioni siano mantenute può causare l'occorrenza di certi eventi. Le occorrenze possono cambiare lo stato del sistema, causando alcune delle condizioni precedenti per cessare il mantenimento e causando altre condizioni ad essere mantenute ???. Esecuzione di una transizione = occorrenza di un evento, atomico.

Modellazione di concorrenza e parallelismo

Le RdP sono ideali per modellare sistemi a controllo distribuito con multipli processi che occorrono concorrentemente. **Vedi img su slide**

Modellare conflitti ed eventi concorrenti

- **eventi di conflitto**: due eventi sono in conflitto se uno dei due può accadere ma non entrambi.
- **eventi concorrenti**: due eventi sono concorrenti se entrambi gli eventi possono accadere in qualsiasi ordine senza conflitti.
- Una situazione dove conflitto e concorrenza sono mischiati è chiamata **confusione**.

Asincronia e località

In una RdP non ci sono inerenti misure di tempo o di flusso di tempo. L'unica importante proprietà del tempo, da un punto di vista logico, è definire un **ordinamento parziale** di occorrenze degli eventi. Eventi che hanno bisogno di non essere vincolati in termini di ordine di occorrenza non sono vincolati.

Località: in un sistema complesso composto da sottoparti operative asincronicamente e indipendente, ogni parte può essere modellata tramite RdP. L'abilitazione e l'esecuzione di transizioni sono affette da, e affliggono solo, cambiamenti locali nella marcatura di una RdP.

Non-determinismo

Una RdP è vista come una sequenza di eventi discreti il cui ordine di occorrenza è uno dei possibili consentiti dalla struttura base. Se in un qualsiasi momento più di una transizione è abilitata, allora qualsiasi delle diverse transizioni abilitate può essere eseguita. La scelta di quale transizione venga eseguita è fatta in maniera non deterministica (random).

Eventi atomici vs nonatomici

L'occorrenza di eventi primitivi è istantanea: eventi non primitivi (con una durata) devono essere modellati da multipli eventi, ad esempio attività con eventi di inizio e di fine.

Gerarchie

- Supporto naturale alla modellazione di gerarchie:
  - **astrazione** un'intera rete può essere rimpiazzata da un singolo posto o transizione per modellare ad un livello più astratto
  - **rifinimento**: posti e transizioni possono essere rimpiazzati da sottoreti per fornire una modellazione più dettagliata.

Applicazioni alla progettazione concorrente e programmazione

Le RdP possono essere utilizzate per modellare sistemi software, in particolare quelli concorrenti, per:

- Rappresentare problemi
  - problemi di CS e Mutex
  - sincronizzazione
- Rappresentare il comportamento di meccanismi
  - semafori
  - sincronizzatori
- Rappresentare interi problemi
  - produttore / consumatore
  - lettori / scrittori
  - filosofi a cena

Problemi di CS e mutex

Vedi slide

Sincronizzazione

Si impone un ordine tra le azioni dei processi, rappresentate tramite transizioni e relazionando le azioni (transizioni) tramite le condizioni (posti)

Vedi slide

Semafori

- Semafori modellati come risorsa condivisa (posti)
  - **wait(P)** modellato come transizione dove il risultato del semaforo è il posto in input
  - **signal(Q)** modellato come transizione dove il risultato del semaforo è il posto in output

Produttori e consumatori

Vedi slide

Lettori e scrittori

Gli n token in p_1 rappresentano n processi che possono voler leggere o scrivere in una memoria condivisa rappresentata da p_3.

Vedi slide

Filosofi a cena

Le forchette sono rappresentate dai posti a_i. I filosofi pensano nei posti A_t, B_t, C_t, D_t, E_t e mangiano nei posti A_e, B_e, C_e, D_e, E_e.

Reti di Petri estese con archi inibitori

Estensione zero-testing: si estendono le RdP base con la possibilità di eseguire una transizione solo se certi posti hanno zero token. Gli **archi inibitori** sono rappresentati da archi con un piccolo cerchio alla fine.

RdP + Archi inibitori = Turing-equivalente: problemi di espressività e indecidibili.

Descrizioni formali di Reti di Petri

Le RdP possono essere formalmente descritte per abilitare un'analisi rigorosa delle proprietà e dei problemi del sistema modellato:

- **proprietà strutturali**, indipendenti dalle marcature iniziali
- **proprietà comportamentali**, dipendente dalle marcature

Mappando la correttezza del sistema su proprietà strutturali/comportamentali - proprietà di sicurezza e vitalità.

Struttura delle Reti di Petri

- Può essere descritta dalla tupla $C = (P, T, I, O)$.
    - P = insieme di posti
    - T = insieme di transizioni
    - I = funzione di input, definisce l'insieme di posti di input per ogni transizione t_j
    - O = funzione di output, definisce l'insieme di posti di output per ogni transizione t_j

Esempio

```
P = { p1, p2, p3, p4, p5 }
T = { t1, t2, t3, t4 }

I(t1) = {p1}
I(t2) = {p2,p3,p5}
I(t3) = {p3}
I(t4) = {p4}

O(t1) = {p2,p3,p5}
O(t2) = {p5}
O(t3) = {p4}
O(t4) = {p2,p3}
```

Marcatori

Un marcatore è un'assegnazione di token ai posti della rete. Può essere rappresentata formalmente come:

- un vettore di N elementi, uno per posto, rappresentanti il numero di token per ogni posto
- una funzione $\mu : P \to N$, $\mu(p_i)$ = numero di token nel posto p_i

Una RdP marcata è rappresentata dalla 5-tupla $M = (P, T, I, O, \mu)$

Semantica di esecuzione delle regole

- Lo stato di una RdP è definito dai suoi marcatori. L'esecuzione di una transizione rappresenta un cambiamento in uno stato della rete
- **Funzione parziale prossimo-stato** $\delta(\mu, t_j)$: la funzione è indefinita se la transizione non è abilitata alla marcatura. Se t_j è abilitata, $\mu' = \delta(\mu, t_j)$ è la marcatura che risulta dalla rimozione dei token dall'input di t_j e aggiungendo token all'output di t_j.
- Data una RdP e un marcatore iniziale, possiamo eseguire la RdP tramite successive esecuzioni di transizioni
  - Eseguire una transizione t_j in un marcatore iniziale produce un nuovo marcatore $\mu^1 = \delta(\mu^0, t_j)$
  - In questo nuovo marcatore possiamo eseguire qualsiasi nuova transizione abilitata, chiamiamola t_k, risultante in un nuovo marcatore $\mu^2 = \delta(\mu^1, t_k)$
  - Ciò può continuare finchè c'è almeno una transizione abilitata in ogni marcatore
  - Se raggiungiamo un marcatore dove nessuna transizione è abilitata, allora nessuna transizione può essere eseguita e la RdP deve fermarsi.

- **Nondeterminismo**

  - Sequenze multiple di marcatori $(\mu^0, \mu^1, \mu^2 \dots)$ e le relative transizioni $(t_j^0, t_j^1, t_j^2 \dots)$ possono risultare dall'esecuzione di una RdP

Insieme raggiungibile

- **Marcatori immediatamente raggiungibili**: un marcatore m' è immediatamente raggiungibile da m se possiamo eseguire qualche transizione abilitata in m risultante in m'.
- **Marcatori raggiungibili**: un marcatore m' è raggiungibile da m se è immediatamente raggiungibile da m o è raggiungibile da qualsiasi marcatore m'' che è immediatamente raggiungibile da m'.
- **Insieme raggiungibile di una RdP**: insieme di tutti gli stati in cui la RdP può entrare in una qualsiasi esecuzione

Rappresentazone matriciale

- I componenti di una RdP possono essere convenientemente rappresentati come matrici:
  - matrice $m \times n$, m righe = transizioni, n colonne = posti
  - **input**: matrice $D-$. Elemento $D - [i, j]$ = peso dell'arco che connette p_j con la transizione t_i.
  - **output**: matrice $D+$. Elemento $D + [i, j]$ = peso dell'arco che connette t_i con la transizione p_j.
  - **incidente**: matrice $D = D + -D-$
  - **transizione**: matrice T di dimensione $(1 \times m)$ rappresentante l'esecuzione della RdP
  - **marcatore corrente**: matrice M di dimensione $(1 \times n)$ rappresentante il numero corrente di token a posto

- Per calcolare l'evoluzione di una RdP: $M' = T \times D + M$

Analisi dei modelli di RdP

- **reti sicure**: reti di Petri in cui non più di un token può mai essere in un qualsiasi posto nella rete allo stesso tempo. Giustificazioni basate sulla definizione originale di eventi e condizioni.
- **reti delimitate o k-reti delimitate**: reti in cui il numero di token in qualsiasi posto è limitato da k. Le reti sicure sono 1-rete delimitata. La delimitazione è una proprietà pratica molto importante.
- **reti conservative**: una RdP è conservativa se il numero di token nella rete è conservato.

Vitalità

- Basata su **analisi delle transizioni**

  - **transizioni morte** in una marcatura, se non c'è sequenza di transizioni che possano abilitarla (deadlock)
  - **potenzialmente eseguibile**, se non esiste qualche sequenza che la abilita (starvation)
  - **transizione viva** eseguibile in tutti i marcatori raggiungibili

- Per la vitalità, è importante non solo che una transizione sia eseguibile in un dato marcatore, ma rimangono potenzialmente eseguibili in tutti i marcatori raggiungibili per quel marcatore

Estensioni di RdP

- **Reti temporizzate**
    - reti temporizzate deterministiche
    - **Reti stocastiche**
- **Reti ad alto livello**
    - **Reti colorate**

Statecharts

- Per modellare **sistemi reattivi complessi**, ora parte di UML
- **Sistemi reattivi**: sistemi guidati dagli eventi, devono continuamente reagire a stimoli interni e esterni (automobili, reti di comunicazione, OS, ecc...). In contrasto con sistemi trasformazionali (I/O, processazione dati)
- *Obiettivo*: introdure un modo per descrivere comportamenti reattivi che sono chiari e realistici, e allo stesso momento formali e rigorosi, così da essere simulati e analizzati.

Oltre i diagrammi a stati di base

- **Stati ed eventi** sono un mezzo naturale per descrivere il comportamento dinamico di un sistema complesso.
- **Transizione di stato**: quando un evento accade in A, se la condizione C è vera in quel momento, il sistema passa allo stato B
- FSM e transizioni non scalano con la complessità: ingestibili, crescono esponenzialmente gli stati, i quali devono essere gestiti in maniera "piatta". Portano inoltre a diagrammi di stato caotici, non strutturati e irrealistici.
- Per essere utile, un approccio stato/evento deve essere *modulare, gerarchico e ben strutturato*. Deve risolvere il problema di ingrandimento esponenziale, rilassando i requisiti per cui tutte le combinazioni di stati devono essere rappresentate esplicitamente.
- **Statecharts**: estensione dei diagrammi di stato convenzionali tramite meccanisimi che migliorano la potenza descrittiva.

Formalismo

- Formalismo formale per descrivere stati e transizioni in modo modulare:
    - **gerarchia**: raggruppamento, raffinamento, promozione di capacità di "zoom" per muoversi facilmente tra i vari livelli di astrazione
    - **ortogonalità**: indipendenza/concorrenza dei sottostati, sincronizzazione tra sottostati

Eventi e stati negli statecharts

- rettangoli arrotondati = stati

- frecce etichettate = eventi (opzionalmente con condizioni tra parentesi e azioni)
- diversi livelli di stati (gerarchia): l'incapsulamento esprime la gerarchia, le frecce possono partire e terminare in qualunque livello.

Gerarchia

- Decomposizione XOR

Raggruppamento (clustering)

- Vedi immagine: dal momento che β porta il sistema verso B da entrambi gli stati A o C, si possono raggruppare questi ultimi all'interno di un super-stato D e rimpiazzare le due frecce con una.
- La semantica di D è uno XOR tra A e C: per essere nello stato D uno deve essere in A o C, e non in entrambe. D è un'astrazione di A e C.
- Approccio bottom-up

Raffinamenti

- Direzione opposta, rifinimento degli stati
- Approccio top-down
- Supporto a zoom-in (guardando dentro lo stato) e zoom-out (astraendo)

Stati di default

Freccie speciali che rappresentano esplicitamente lo stato di entrata di default, a qualsiasi livello.

Ortogonalità

- **Decomposizione AND**
- Cattura la proprietà per cui, in uno stato, il sistema possa essere in tutti i suoi componenti AND
- La notazione è quella di suddividere un box in due componenti tramite tratteggiatura
- (Vedi figura) Lo stato Y consiste nell'AND dei componenti A e D, con la proprietà che essere in Y significa essere in una qualche combinazione di B o C con E, F o G. Y è il prodotto ortogonale di A e D.
- Indipendenza e/o concorrenza

Indipendenza

- (Vedi immagine) Mi accade a (B,F) e affligge solo il componente D, risultante in (B,E).
- Ciò illustra un certo tipo di indipendenza: la transizione è la stessa sia che il sistema sia in B o C nella sua componente A.

Equivalente senza AND

- L'equivalente senza AND ha il prodotto degli stati (vedi immagine)
- Se abbiamo due componenti con 1000 stati, avremo un milione di stati nel prodotto. 3 componenti: 10^9.

Sincronizzazione

- (Vedi immagine) se accade un evento α, trasferisce B a C e F a G simultaneamente, risultando in uno stato combinato (C,G)
- **Sincronizzazione**: un singolo evento causa eventi simultanei

Notazioni per la decomposizione AND, esempi

Vedi slide

Azioni

Le **azioni** rappresentano la capacità dei diagrammi di stati di generare eventi e di cambiare il valore delle condizioni, influenzando altri componenti e l'ambiente del sistema.

Espresse dalla notazione ".../W" che può essere attaccata all'etichetta della transizione. W è una azione portata avanti dal sistema. Le azioni hanno occorrenze istantanee che impiegano tempo zero (idealmente).

Le azioni possono essere riprodotte anche mentre si entra o esce da uno stato.

Vedi immagini.

Rappresentare le attività

Le attività impiegano sempre un tempo diverso da zero (beep, displaying, ecc). Negli statechart, le attività sono associate con gli stati, esplodendo le azioni di entrata e uscita.

Statemate

Strumento di modellazione e simulazione grafica per lo sviluppo di sistemi embedded complessi basati su statechart. Fornisce un collegamento formale e diretto tra requisiti dell'utente e l'implementazione software consentendo all'utente dio creare specifiche complesse ed eseguibili.

Diagrammi di attività

Sono uno degli schemi adottati in UML per rappresentare il business e i flussi operativi di un sistema software. Un diagramma di attività è uno schema dinamico che mostra l'attività e l'evento che causa

l'oggetto deve essere in questo stato

- Diagrammi di attività vs di stato
  - un diagramma di stato mostra i diversi Stati di un oggetto durante il ciclo di vita della sua esistenza nel sistema e le transizioni degli Stati degli oggetti. Queste transizioni raffigurano le attività che causano queste transizioni, indicate dalle frecce
  - un diagramma di attività parla di più su queste transizioni e le attività che causano i cambiamenti negli Stati oggetto

Panoramica

Mostrando il flusso delle attività attraverso il sistema, i diagrammi sono letti dall'alto verso il basso e hanno rami e forche per descrivere condizioni e attività parallele. Una forchetta viene utilizzata quando più attività si verificano allo stesso tempo.

Il diagramma seguente illustra una forchetta dopo activity1. Questo indica che sia activity2 e attività3 si verificano allo stesso tempo. Dopo activity2 c'è un ramo. Il ramo viene descritto quali attività si svolgeranno basato su un insieme di condizioni. Tutti i rami a un certo punto sono seguiti da un'Unione per indicare la fine del comportamento condizionale iniziato da quel ramo.

Dopo l'Unione tutte le attività parallele devono essere combinate da un join prima di passare allo stato di attività finale.

Esempio: gestione ordini

Il diagramma mostra il flusso delle azioni del flusso di lavoro del sistema

- una volta ricevuto l'ordine le attività divise in due serie parallele di attività
- un lato riempie e invia l'ordine mentre altre la gestisce la fatturazione
- sul lato ordine di riempire, il metodo di consegna viene deciso in modo condizionale.
- a seconda della condizione viene eseguita sia la consegna durante la notte o l'attività di consegna regolari.
- Infine le attività parallele si combinano per chiudere l'ordine.

Corsie

Un corsia è un modo per attività di gruppo svolte dall'attore stesso su un diagramma di attività o per attività di gruppo in un singolo thread