Modello degli Attori

Walter Cazzola

Dipartimento di Informatica e Comunicazione
Università degli Studi di Milano

Outline

Modello degli Attori
Sistemi ad Attori: Nozioni Generali e Caratteristiche.

In un sistema computazionale basato sul modello degli attori la computazione è portata avanti in risposta alle comunicazioni inviate al sistema (computazione data-driven).

Le comunicazioni sono contenute in **task**.

Man mano che la computazione procede il sistema evolve includendo nuovi task e nuovi attori che sono creati nel processare task già presenti nel sistema.

Tutti i task già processati e gli attori non più utilizzati possono essere rimossi dal sistema (garbage collection).

Modello degli Attori
Task.

I task non ancora processati, in un sistema ad attori, rappresentano la forza che guida la computazione dell'intero sistema.

I task sono triple composte da:

- un'etichetta che identifica univocamente il task;
- l'*indirizzo di posta* del destinatario (*target*) del messaggio; e
- le informazioni che l'attore dovrà elaborare (una tupla di valori).

La comunicazione tra un attore α ed un altro attore β può avvenire solo se ne conosce l'indirizzo

- l'indirizzo di β può essere già noto all'attore α;
- β può essere il risultato della computazione di α; oppure
- α recupera l'indirizzo di β tramite l'ultimo messaggio $\bar{k}$ ricevuto.

Modello degli Attori
Attori: Nozioni Generali e Caratteristiche.

Gli *attori* sono agenti computazionali che mappano ogni messaggio che ricevono su una tripla composta da:

- un insieme di messaggi da inviare agli altri attori;
- un nuovo comportamento (usato per processare il messaggio successivo); e
- un insieme di nuovi attori.

Bisogna notare che:

- il comportamento di un attore può dipendere dalla storia;
- non c'è un ordine con cui un attore esegue delle azioni (le sue azioni sono funzione del suo comportamento corrente e del messaggio che si sta processando); e
- la creazione di nuovi attori è parte integrante del modello.

Un attore identificato da: un *indirizzo mail* e da un *comportamento*.

Modello degli Attori
Attori: Modello Computazionale e Comportamento di un Attore.

Tutta la computazione in un sistema ad attori è il risultato dell'elaborazione dei messaggi ricevuti.

Il modello computazionale è di tipo data-driven in contrasto con i sistemi distribuiti classici che o terminano o sono sempre attivi.

Gli attori accettano un messaggio quando processano un task contenente tale messaggio.

Un attore può processare solo quei task di cui è target.

Modello degli Attori
Attori: Modello Computazionale e Comportamento di un Attore (Segue).

Un attore può essere descritto specificando:

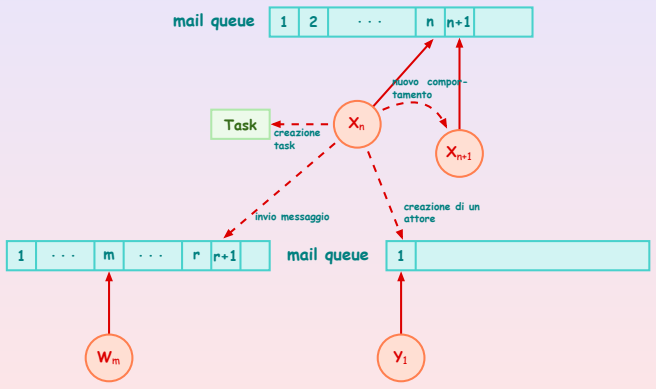
- il suo indirizzo di posta, a cui è associata una coda di dimensioni congrue per i messaggi non ancora processati; e
- il suo comportamento che è funzione delle comunicazioni accettate.

Quando un attore accetta un messaggio può creare nuovi attori e task e rielabora (cambiandolo) il proprio comportamento.

L'ordine d'arrivo dei messaggi indirizzati ad uno specifico attore è sequenziale. Deve essere presente un meccanismo (*mail queue*) di buffering e di arbitrazione dei messaggi in arrivo.

Quando un attore X_n accetta l'n-esimo messaggio presente nella sua mail queue, creerà l'attore X_{n+1} che lo rimpiazzerà proseguendo nell'elaborazione del messaggio successivo presente nella coda di X_n.

Modello degli Attori
Esempio di Transazione.



Modello degli Attori

Linguaggio di Programmazione Basato sugli Attori.

Un programma in un linguaggio basato sul modello degli attori consiste:

- nel definire i possibili comportamenti degli attori (si associa un'etichetta ad uno schema di comportamento, senza effettivamente creare un attore);
- nella creazione di nuovi attori;
- nella creazione di nuovi task, quindi, nell'invio di messaggi;
- nel dichiarare quali attori (detti *receptionist*) possono interagire con l'esterno; e
- nel dichiarare quali sono gli attori esterni al sistema con cui è possibile interagire.

Gli ultimi due punti servono ad aumentare la modularità del sistema.

Modello degli Attori

Definizione del Comportamento di un Attore.

Ogni comportamento deve prevedere un possibile comportamento che lo rimpiazzerà. Dà origine ad una definizione infinita.

I comportamenti sono definiti per induzione (o ricorsione).

Ogni comportamento è parametrizzato sia sulle informazioni caratterizzanti un attore (*acquaintance list*) sia sui messaggi che un attore può ricevere (*communication list*).

Es. il comportamento di un conto corrente dipende dal bilancio del conto stesso.

Si definisce il comportamento di ogni conto in termini del suo bilancio.

Quando si crea un nuovo conto o se ne rimpiazza il comportamento un valore per il bilancio deve essere specificato.

Modello degli Attori

Definizione del Comportamento di un Attore.

Rimpiazzare i Comportamenti

`become` *<espressione>*

la primitiva `become` permette di cambiare il comportamento di un attore.

Comportamento per difetto.

Ogni attore deve avere un comportamento sostitutivo, quando non è definito, per difetto, l'attore viene rimpiazzato con un altro attore con lo stesso comportamento.

Modello degli Attori

Esempio: Stack come lista linkata di Attori.

Lo stack è implementato come una lista linkata di attori con comportamento uniforme.

`forwarder` descrive un comportamento predefinito: il messaggio è girato all'attore specificato nell'acquaintance list.

```
def stack-node (content, link)   ← Acquaintance List
  [case operation of
     pop: (customer)
     push: (newcontent)          } Communication List
  end case]
if (operation = pop) ∧ (content ≠ nil) then
  become forwarder (link)
  send content to customer
fi
if operation = push then
  let P = new stack-node(content, link) {
    become stack-node (newcontent, P)
  }
fi end def
```

Il top dello stack è l'unico elemento che comunica con l'esterno e verrà creato dall'istruzione: `let p = new stack-node(nil,sink)`.

Modello degli Attori

Creare Attori e Task.

Creare Attori.

`new` *<nome del comportamento>*`(`*<lista di acquaintance>*`)`

Un attore viene creato tramite un'invocazione di `new`, come valore di ritorno abbiamo l'indirizzo del nuovo attore.
<nome comportamento> identifica quale comportamento iniziale dovrà avere il nuovo attore. Tra parentesi sono inseriti i valori attuali per le acquaintance dell'attore.

Creare Task.

`send` *<messaggio>* `to` *<destinatario>*

Un task viene creato specificando il destinatario ed il contenuto del messaggio.

Riferimenti Bibliografici

- Gul Agha.
  Actors: A Model of Concurrent Computation in Distributed Systems.
  MIT Press, Cambridge, 1986.