Elaborato di Fondamenti di Informatica 2007/08

Claudio Guidi
cguidi@cs.unibo.it

Perché e quando.

- Consente di comprendere come i risultati teorici della teoria dei linguaggi formali possano avere un impatto importante dal punto di vista implementativo dei compilatori/interpreti di linguaggi.
- Obbligatorio per superare l'esame
- Consegna libera durante la sessione. La scadenza per ciascuna sessione verrà esposta sulla homepage: http://www.cs.unibo.it/~cguidi
- Gli elaborati verranno corretti solo dopo la scadenza della consegna. In casi eccezionali può essere richiesto, previo accordo via e-mail, di effettuare la correzione prima di tale data.

Consegna.

- L’elaborato deve essere consegnato via email all’indirizzo cguidi@cs.unibo.it con oggetto “IT - consegna progetto 2007/08 ”
- Il testo della mail deve contenere nome, cognome e matricola dello studente.
- In allegato il file .zip (altri formati non verranno accettati) del progetto con:
  - Documentazione
  - Sorgenti progetto
  - File di esempio

Come e dove.

- Ogni studente dovrà sviluppare un elaborato
- Non sono consentiti elaborati di gruppo
- L'elaborato deve essere sviluppato in linguaggio Java.
- L'elaborato verrà corretto in laboratorio
  - L'elaborato deve essere compatibile con la versione della JVM correntemente installata in laboratorio
- L'elaborato deve essere funzionante
- Il giorno dell'elaborato deve essere consegnata una stampa cartacea della documentazione.

Il progetto.

- Il progetto consiste nello sviluppo di un linguaggio di coreografia le cui specifiche verranno illustrate di seguito
- Di tale linguaggio si deve sviluppare uno scanner ed un parser che ne riconoscano la grammatica
- Si deve sviluppare un interprete in grado di produrre un file contenente il transition system generato a partire dalla coreografia in input
- Sul transition system deve essere effettuata la verifica di alcune proprietà
- Con il linguaggio sviluppato si deve dare rappresentazione ad un esempio fornito con le specifiche

Linguaggio di coreografia: intro

- Un linguaggio di coreografia è un linguaggio di alto livello che consente di **descrivere** un sistema orientato ai servizi da un punto di vista **globale**. Il linguaggio qui proposto è ispirato alla specifica per Web Services denominata WS-CDL.
  - (http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217/)
- Con una coreografia **non si implementa** effettivamente un sistema ma lo si descrive.
- Le coreografie possono essere utilizzate:
  - come strumento di progettazione da cui derivare in modo automatico gli scheletri dei diversi servizi coinvolti
  - per verificare che alcune proprietà siano rispettate dal sistema
  - come insieme di vincoli che un determinato sistema deve rispettare.

Linguaggio di coreografia: elementi

- Una coreografia si compone essenzialmente di tre parti:
  - Dichiarazione delle informazioni
  - Dichiarazione dei ruoli
  - Dichiarazione dell'evoluzione della coreografia
- Le informazioni sono quelle che i diversi ruoli possono scambiarsi durante l'**evoluzione** della coreografia
- I ruoli descrivono i partecipanti coinvolti in una coreografia
- Le interazioni sono gli effettivi scambi di messaggi tra ruoli che possono essere performati all'interno di un sistema.

Dichiarazione delle informazioni

- In una coreografia le informazioni vengono descritte in modo astratto e sono semplicemente identificate da nomi.
- Tutte le informazioni utilizzate all'interno di una coreografia devono essere definite all'interno della *dichiarazione delle informazioni*.

Dichiarazione dei ruoli

- In una coreografia un ruolo descrive in modo astratto un partecipante del sistema. Ogni ruolo è rappresentato da:
  - Un nome
- Ogni ruolo contiene:
  - Una conoscenza iniziale rappresentata da un insieme di informazioni.

Dichiarazione dell'evoluzione

- L'evoluzione di una coreografia si progetta componendo tra di loro due diverse primitive:
  - Interazione
  - Assegnamento
- Le primitive possono essere composte in:
  - Sequenza
  - Parallelo
  - Scelta deterministica
- Chiameremo **blocco** un insieme di primitive composte tra di loro.

Interazione

- Un’interazione esprime lo scambio di un’informazione tra un ruolo ed un altro.
- In un’interazione devono comparire i seguenti elementi:
  - Il ruolo che spedisce
  - Il ruolo che riceve
  - L’informazione spedita

Assegnamento

- L’assegnamento consente di dare un valore ad un’informazione.
- Un assegnamento avrà una forma riconducibile alla seguente:
  - *i :=(ruolo) expr*
  - Dove *i* è l’identificativo che rappresenta l’informazione
  - (ruolo) rappresenta il ruolo all’interno del quale viene performato l’assegnamento.

Composizione in sequenza

- Un’interazione, un assegnamento o una qualsiasi loro composizione possono essere composte in sequenza. In una sequenza il primo blocco viene processato prima del secondo.
- Supponendo che il simbolo ; rappresenti l’operatore di sequenza allora:
  - A;B significa che il generico blocco A verrà eseguito prima del blocco B.

Composizione in parallelo

- In questo caso tutti i blocchi specificati verranno eseguiti in parallelo.
- Supponendo che il simbolo | rappresenti l'operatore di parallelo allora:
  - A | B significa che i blocchi A e B verranno eseguiti in parallelo.

Composizione in scelta deterministica

- E’ fondamentalmente l’analogo dell’istruzione di *If then else*
- La composizione in scelta deterministica deve contenere:
  - Una condizione da testare su almeno una informazione
  - Un blocco da eseguire in caso la condizione sia verificata
  - Un eventuale blocco da eseguire in caso la condizione non sia verificata
  - Il ruolo all’interno del quale viene valutata la condizione

Composizione

- Sequenza e parallelo sono operatori applicabili anche a blocchi di interazioni già composte tra di loro
- Si può avere per esempio che:
  - A;B;(C|D) ;E dove, per semplicità, A,B,C,D ed E rappresentano interazioni o assegnamenti. In tal caso A e B vengono eseguiti prima del parallelo tra C e D mentre E viene eseguito solamente quando il parallelo è terminato
  - (A;B;C)|(D;E) rappresenta il parallelo di due sequenze

Un po' di semantica

- Di seguito si riporta la semantica operazionale strutturata per le primitive e gli operatori di composizione. Dove si utilizzeranno i seguenti simboli:
  - $\mathcal{K}$: rappresenta la funzione che associa ad ogni ruolo l'insieme delle informazioni da esso conosciute
  - γ : rappresenta la funzione che ad ogni informazione associa il rispettivo valore
  - μ : generica etichetta

Sintassi essenziale

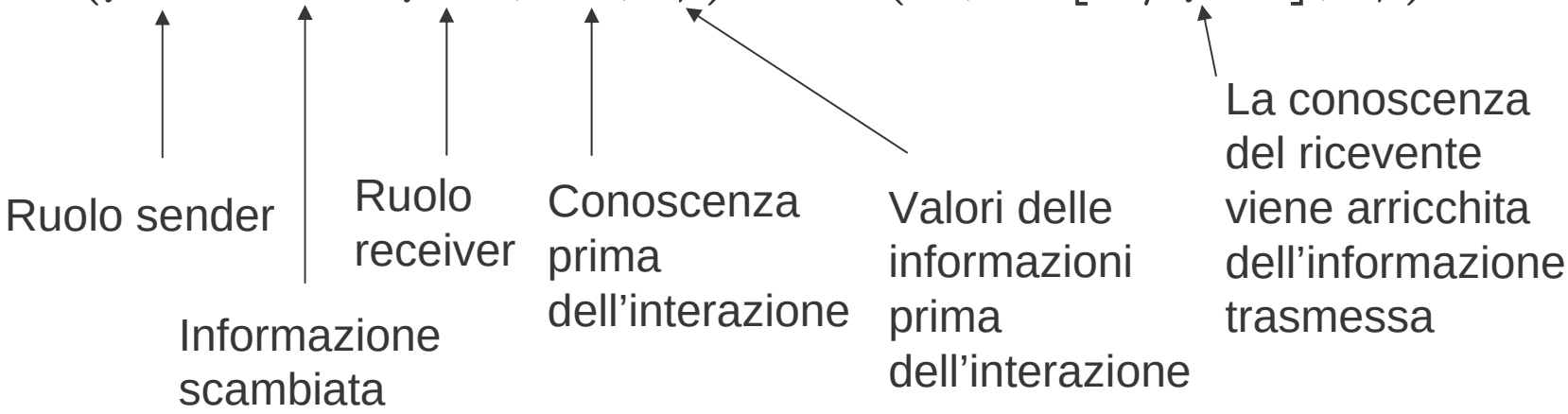
- Utilizzeremo la seguente sintassi essenziale:

$$
\begin{array}{lll}
C ::= & & \\
 & \mathbf{0} & \text{Processo nullo} \\
 & \rho_A : x : \rho_B & \text{Interazione} \\
 & i :=_\rho e & \text{Assegnamento} \\
 & C ; C & \text{Sequenza} \\
 & C | C & \text{Parallelo} \\
 & \texttt{if } \chi_\rho \texttt{ then } C \texttt{ else } C & \text{Scelta deterministica}
\end{array}
$$

Regola per la transizione

(Interazione)

$$(\rho_A : x : \rho_B, \mathcal{K}, \gamma) \xrightarrow{\mu} (\mathbf{0}, \mathcal{K}[x/\rho_B], \gamma)$$



L'interazione non altera i valori delle informazioni ma fa sì che il ricevente aumenti la propria conoscenza grazie all'informazione ricevuta

Regola per l'assegnamento

$$(\textsc{Assegnamento})\quad \frac{e \hookrightarrow_{\gamma} \delta}{(i :=_{\rho} e, \mathcal{K}, \gamma) \xrightarrow{\mu} (\mathbf{0}, \mathcal{K}, \gamma[\delta/i])}$$

L'espressione *e* viene valutata nel valore generico delta

Informazione assegnata

Ruolo in cui viene effettuato l'assegnamento

Il valore dell'informazione *i* viene aggiornato a delta

L'assegnamento altera solamente il valore dell'informazione assegnata

Regola per sequenza e parallelo

(Sequenza)

$$\frac{(C, \mathcal{K}, \gamma) \xrightarrow{\mu} (C', \mathcal{K}', \gamma')}{(C; D, \mathcal{K}, \gamma) \xrightarrow{\mu} (C'; D, \mathcal{K}', \gamma')}$$

(Parallelo)

$$\frac{(C, \mathcal{K}, \gamma) \xrightarrow{\mu} (C', \mathcal{K}', \gamma')}{(C \mid D, \mathcal{K}, \gamma) \xrightarrow{\mu} (C' \mid D, \mathcal{K}', \gamma')}$$

; e | rappresentano rispettivamente l'operatore di sequenza e quello di parallelo.

N.B. Il caso **0;C** si riduce a **C.** Inoltre si consideri che esiste anche la regola simmetrica per il parallelo in cui avanza D

Regola per la scelta deterministica

La condizione χ è soddisfatta nello stato γ

$$\text{(If then)}\quad \frac{\gamma \vdash \chi}{(\texttt{if}\ \chi_\rho\ \texttt{then}\ C\ \texttt{else}\ D, \mathcal{K}, \gamma) \xrightarrow{\mu} (C, \mathcal{K}, \gamma)}$$

La condizione non è soddisfatta

$$\text{(Else)}\quad \frac{\gamma \nvdash \chi}{(\texttt{if}\ \chi_\rho\ \texttt{then}\ C\ \texttt{else}\ D, \mathcal{K}, \gamma) \xrightarrow{\mu} (D, \mathcal{K}, \gamma)}$$

Interpretazione

- L'interpretazione consiste nella generazione del transition system e di alcune verifiche da effettuare su di esso.
- Il transition system deve essere prodotto su un file di tipo testuale con il seguente formato:
  - Nome file di origine della coreografia
  - Stati: *Elenco stati*
  - Transizioni: *Elenco di tutte le transizioni*
- Gli stati riportano tutte le informazioni relative alle conoscenze di ciascun ruolo ed al valore delle informazioni
- Se più comodo l'elenco degli stati e quello delle transizioni possono essere riportati su due file separati

Formato dell'elenco degli stati: elenco delle conoscenze

- Uno stato per ogni linea
- Ogni stato ha il seguente formato:
  - *<state_num>=num*
  - *<finale>=true|false*
  - *<state>=<state_num>: <elenco_conoscenza>, <elenco_val_inf>:<finale>*
  - Dove *<finale>* esplicita se si tratta di uno stato finale o meno.
- Elenco delle informazioni:
  - Rappresenta tutte le informazioni conosciute in ciascuno stato da ogni ruolo
  - L'elenco delle conoscenze ha il seguente formato:
    - *<nome_ruolo>= id*
    - *<inf>=id*
    - *<inf_list>=,<inf><inf_list>| ε*
    - *<con_ruolo>=<nome_ruolo>:<inf><inf_list>*
    - *<con_ruolo_list>=;<con_ruolo><con_ruolo_list>| ε*
    - *<elenco_conoscenza>=[<con_ruolo><con_ruolo_list>]*

Formato dell'elenco degli stati: elenco dei valori delle informazioni

- Elenco dei valori delle informazioni:
  - Rappresenta tutti i valori delle informazioni in un determinato stato
  - L'elenco dei valori delle informazioni ha il seguente formato:
    - $<val>=num$
    - $<inf\_val>=<inf>:<val>$
    - $<inf\_val\_list>=,<inf\_val><inf\_val\_list>|\ \varepsilon$
    - $<elenco\_val\_inf>=[<inf\_val><inf\_val\_list>]$

Formato dell'elenco transizioni

- Una transizione per ogni linea
- In caso di interazione la transizione deve avere il seguente formato:
  - *<start_state>=num*
  - *<end_state>=num*
  - *<da_ruolo>=<nome_ruolo>*
  - *<verso_ruolo>=<nome_ruolo>*
  - *<transizione>=<start_state>,<end_state>#<da_ruolo>,<verso_ruolo>,<inf>*
  - Dove *<inf>* rappresenta l'id dell'informazione scambiata
- In caso di assegnamento:
  - *<transizione>=<start_state>,<end_state>? <inf>,"<expr>",<nome_ruolo>*
  - Dove *<expr>* rappresenta esattamente l'espressione come è stata inserita dal programmatore e *<nome_ruolo>* indica il ruolo in cui è stato processato l'assegnamento
- In caso di scelta deterministica:
  - *<transizione>=<start_state>,<end_state>$ <cond>,<nome_ruolo>*
  - Dove *<cond>* rappresenta la consizione testata e *<nome_ruolo>* indica il ruolo in cui è stato processata la condizione

Verifiche sul transition system

- Una volta generato il transition system su di esso andranno verificate le seguenti proprietà
  - Consistenza rispetto alle interazioni
  - Consistenza rispetto agli assegnamenti
  - Consistenza rispetto alle scelte deterministiche
- **Facoltativo per la lode**:
  - Raggiungibilità di un ruolo

NB: se più comodo la verifica delle proprietà può essere realizzata contestualmente alla generazione del transition system

Consistenza rispetto alle interazioni

- Una interazione per essere consistente deve rispettare la seguente condizione:
  - L'informazione trasmessa deve essere conosciuta dal ruolo *sender* prima dell'interazione
- Dato un transition system esso è consistente rispetto alle interazioni se:
  - Tutte le interazioni sono consistenti

Consistenza rispetto agli assegnamenti

- Un assegnamento per essere consistente deve rispettare le seguenti condizioni:
  - L'informazione assegnata deve essere conosciuta SOLAMENTE dal ruolo specificato nell'assegnamento
  - Tutte le informazioni che eventualmente compaiono all'interno dell'espressione da assegnare devono essere conosciute dal ruolo specificato nell'assegnamento
- Dato un transition system esso è consistente rispetto agli assegnamenti se:
  - Tutti gli assegnamenti sono consistenti

Consistenza rispetto alle scelte deterministiche

- Una scelta deterministica per essere consistente deve rispettare la seguente condizione:
  - Tutte le informazioni presenti nella condizione da testare devono essere conosciute dal ruolo specificato
- Dato un transition system esso è consistente rispetto alle scelte deterministiche se:
  - Tutte le scelte deterministiche sono consistenti

Raggiungibilità di un ruolo (facoltativo)

- Con raggiungibilità di un ruolo si intende:
  - Data una informazione ed un ruolo in input
  - Stabilire se l'informazione raggiunge sempre il ruolo specificato per ogni traccia di esecuzione della coreografia.

Esempio (1)

- Con il linguaggio sviluppato si dia rappresentazione del seguente scenario:
  - In un sistema di commercio elettronico un cliente, tramite un servizio intermediario denominato *market,* può acquistare o dei cd o delle macchine fotografiche. Si supponga che i prodotti siano ordinabili presso due diversi fornitori. Una volta scelto il prodotto il cliente dovrà decidere se acquistarlo o meno. In caso affermativo il market dovrà provvedere a richiedere la transazione bancaria ad un apposito servizio bancario. Tale servizio richiede sia al cliente che al fornitore indicato le coordinate bancarie (numero carta di credito e tipo) e poi rigira le transazioni ai rispettivi servizi delle carte di credito (Visa e Master Card). Supponendo che tutto vada a buon fine, il cliente, il fornitore ed il market devono essere notificati dal servizio bancario

Esempio (2)

- Ruoli:
  - Cliente
  - Market
  - Banca
  - Visa
  - Master Card
  - Fornitore CD
  - Fornitore Macchine fotografiche

Esempio (3)

- Evoluzione del sistema:
  - Il cliente imposta un prodotto (due prodotti possibili: cd o macchina fotografica) e richiede il prezzo al market
  - Il market richiede il prezzo al fornitore giusto dipendentemente dalla categoria merceologica
  - Il fornitore risponde al market con il prezzo
  - Il market aggiunge un sovrapprezzo del 5% e reinoltra il nuovo prezzo al cliente
  - In base ad una propria soglia interna il cliente decide se acquistare o meno
  - Il cliente risponde al market la sua intenzione di acquistare o meno
  - Se decide di acquistare, il market inoltra la richiesta di transazione alla banca specificando di quale fornitore si tratta ed il prezzo da pagare
  - La banca chiede parallelamente sia al cliente che al fornitore il numero di carta di credito e il tipo di carta (Visa o Master Card)
  - In base alla tipologia di carta la banca manderà un inoltro transazione al servizio Visa o al servizio Master Card
  - I servizi Visa e Master Card risponderanno con un messaggio di avvenuta transazione
  - La banca invierà il messaggio di successo al cliente, al fornitore ed al servizio di market.

Documentazione.

- Deve essere fornita in *formato pdf*
  - **Il giorno della discussione dell'elaborato deve essere consegnata una stampa cartacea della documentazione**
- Deve riportare nome, cognome e matricola dello studente
- Deve essere suddivisa in quattro sezioni
  - Linguaggio
  - Scanner
  - Parser
  - Interprete

Documentazione: Linguaggio

- **NB: In grassetto sono riportati i titoli delle sottosezioni**
- **Introduzione:** descrizione generica del linguaggio sviluppato illustrando velocemente i vari costrutti linguistici utilizzati
- **La grammatica:** deve riportare le produzioni della grammatica in formato BNF. Commentare le produzioni della grammatica che si ritengono più significative.
- **Esempi:** riportare almeno tre esempi di codice. Nel complesso dei tre listati si deve fare utilizzo di tutti i costrutti presenti nella grammatica. Ogni esempio deve essere corredato da una breve descrizione che ne illustra il funzionamento.
- **Osservazioni** (non obbligatoria): qualora necessario, utilizzare questa sezione per eventuali commenti od osservazioni legate allo sviluppo del linguaggio.

Documentazione: Scanner

- **Introduzione:** descrizione generica del codice che implementa lo scanner
- **I token:** deve riportare le espressioni regolari dei token utilizzati
- **Il codice**: breve descrizione delle classi più importanti
- **Osservazioni** (non obbligatoria): qualora necessario, utilizzare questa sezione per eventuali commenti od osservazioni legate all'implementazione dello scanner

Documentazione: Parser

- **Introduzione:** descrizione generica del codice che implementa il parser, quale tipologia di parser si è scelto, motivazioni.
- **Il codice**: breve descrizione delle classi più importanti
- **Osservazioni** (non obbligatoria): qualora necessario, utilizzare questa sezione per eventuali commenti od osservazioni legate all'implementazione del parser

Documentazione: L'interprete

- **Introduzione:** descrizione generica del codice che implementa l'interprete
- **Architettura e strutture dati:** descrivere l'architettura dell'interprete e delle strutture dati (utilizzare una rappresentazione grafica qualora necessario)
- **Esecuzione:** descrivere tutti i passi necessari all'utente al fine di poter mettere in esecuzione l'interprete (linea di comando, eventuali parametri, ecc.)
- **Esempio:** Riportare l'esempio di specifica. La coregrafia che lo rappresenta deve ovviamente soddisfare le proprietà di consistenza
- **Il codice**: breve descrizione delle classi più importanti
- **Osservazioni** (non obbligatoria): qualora necessario…