A che serve un'equivalenza comportamentale?

Riepilogando,

- CCS, π-calculus, Dπ, e Mobile Ambients sono dei semplici linguaggi (o meglio modelli) per descrivere sistemi concorrenti e/o mobili.

- Un'equivalenza comportamentale tra sistemi (ovvero programmi scritti nei linguaggi menzionati prima) è un valido strumento per stabilire quando due sistemi P e Q si comportano nello stesso modo dal punto di vista di un osservatore esterno. Cosa che per adesso scriveremo
$$P \;\cong\; Q\;.$$

- **Domanda**: Per quale motivo dovremmo preoccuparci di trovare una nozione di equivalenza adeguata per i programmi scritti nei linguaggi concorrenti/mobili menzionati prima? Le ragioni sono molteplici.

- La prima è concettuale: in quanto non possiamo affermare di conoscere il comportamento di un programma se non siamo in grado di stabilire quando due programmi si comportano nello stesso modo.

- La seconda è più pragmatica. I linguaggi che vedremo vengono detti "*single language approach*" poichè possono essere utilizzati sia per fornire la *specifica* (Spec) che per realizzarne un'*implementazione* (Impl).

- La specifica ha il dovere di essere chiara e rigorosa nel mostrare le funzionalità richieste al sistema. Essa è spesso costituita da un programma monolitico e sequenziale. L'implementazione, invece, è un programma ben più complesso in cui si cerca di spezzare la computazione in più moduli da eseguire concorrentemente.

- Come facciamo allora a garantire che l'implementazione rispetti la specifica? Semplice, provando che

$$\text{Spec} \;\cong\; \text{Impl}\ .$$

- E come si fa a provare Spec $\cong$ Impl? Durante il corso mostreremo una serie di tecniche che applicheremo ad esempi ragionevolmente piccoli. Comunque, per sistemi di dimensioni reali esistono dei tool appropriati in grado di verificare automaticamente che l'implementazione rispetta la verifica.

- Si noti che una nozione appropriata di equivalenza comportamentale non è solo utile a livello di verifica ma anche per la realizzazione di *compilatori ottimizzanti* in grado di sostituire codice inefficiente con codice efficiente preservando il comportamento (semantica) del programma.

Verso un linguaggio concorrente

- Comunque, prima di definire un'equivalenza comportamentale dobbiamo fissare il nostro linguaggio concorrente. Più precisamente il modello semplificato attraverso cui vogliamo rappresentare i nostri sistemi reattivi/concorrenti.

- In prima approssimazione si potrebbe pensare di modellare un sistema reattivo attraverso una qualche forma di automa.
- Dopodichè, usando la teoria classica degli automi potremmo concludere che due sistemi reattivi sono equivalenti *se e solo se* i corrispondenti automi riconoscono lo stesso linguaggio.
- Mostreremo però che tale approccio non è appropriato per lo studio del comportamento di sistemi reattivi e/o concorrenti.

Cenni di teoria degli automi

Sia *Act* un insieme di **azioni**, anche chiamato **alfabeto**.

Definizione 1 *Un automa A con alfabeto Act ha quattro componenti:*

- *un insieme $Q = \{q_0, q_1, \ldots\}$ di* stati*;*
- *uno stato $q_0 \in Q$ chiamato* **stato iniziale***;*
- *un insieme F, dove $F \subseteq Q$, di* **stati finali***;*
- *una relazione $\mathcal{T}$, dove $\mathcal{T} \subseteq Q \times Act \times Q$, che rappresenta le* **transizioni**.

Una **transizione** (q, a, q') *la scriveremo come* $q \xrightarrow{a} q'$. *Un automa* A *è a* **stati finiti** *se l'insieme* Q *è finito, e* **deterministico** *se per ogni coppia* $(q, a) \in Q \times Act$ *esiste* **esattamente una** *transizione* $q \xrightarrow{a} q'$.

Un automa viene generalmente rappresentato come un **grafo orientato**, i cui nodi sono gli **stati** e gli **archi** le transizioni.

Definizione 2 (Linguaggio riconosciuto da un automa) *Sia* A *un automa con alfabeto* Act, *e* $s = a_1 \cdots a_n$ *una stringa di elementi in* Act. *Allora,* A **riconosce** s *se esiste un cammino in* A, *che parta dalla stato iniziale* q_0 *e raggiunga uno qualunque degli stati finali in* F, *i cui archi siano etichettati con i simboli* $a_1 \cdots a_n$. *Con* $\widehat{A}$ *denoteremo il linguaggio riconosciuto da* A, *ovvero l'insieme di stringhe riconosciute.*

Ad esempio, il linguaggio riconosciuto dall'automa a stati finiti A1, alla pag. 12 del vostro libro, è

$$\widehat{A1} = a((bc + c)a)^{\star}$$

(espresso attraverso espressioni regolari).

Definizione 3 (Equivalenza tra automi) *Due automi A e B vengono detti equivalenti se e solo se riconoscono lo stesso linguaggio, ovvero se e solo se $\widehat{A} = \widehat{B}$.*

Proposizione 4 (Determinismo vs non determinismo) *Per ogni automa a stati finiti non deterministico A ne esiste uno deterministico A' equivalente, cioè tale che $\widehat{A} = \widehat{A'}$.*

È facile mostrare che l'automa a stati finiti non deterministico A2 di pag. 12, riconosce lo stesso linguaggio di A1: $a((bc + c)a)^\star$.

Domanda: Da un punto di vista puramente reattivo, è appropriato considerare l'automa deterministico A1 e quello non deterministico A2 equivalenti?

Detta in altri termini, vogliamo veramente trascurare il non determinismo all'interno di in un sistema reattivo?

Automi visti come sistemi reattivi

- In un approccio puramente reattivo un automa può essere visto come una semplice *scatola nera* provvista di pulsanti utilizzabili dall'esterno.

- La scatola è provvista di un pulsante per ogni elemento dell'alfabeto Act dell'automa, cioè uno per ogni azione possibile dell'automa.

- L'unico modo per interagire con l'automa (o meglio la scatola) è premendo (se possibile) i pulsanti.

- A volte i pulsanti saranno bloccati, altre volte invece potranno essere premuti. Ciò dipende dallo stato interno dell'automa. Stato che, comunque, noi, dall'esterno, non possiamo osservare (proprio per questo diciamo di avere a che fare con una scatola nera).

- L'unico modo per notare una differenza tra scatole definite su uno stesso alfabeto (e quindi con gli stessi tasti) è attraverso la pressione dei tasti medesimi.

- Due scatole sono differenti se esiste una stringa $s = b_1 \cdots b_k$, per cui la sequenza di tasti $b_1 \cdots b_k$ può essere premuta in una scatola ma non nell'altra.

- Di conseguenza, di una stringa non ci interessa più se questa ci porta in uno stato finale (dall'esterno non è possibile sapere se l'automa è in uno stato finale); al contrario, siamo interessati ad *osservare* se l'automa è in grado di eseguire la stringa in maniera interattiva, cioè se consente la pressione in sequenza dei pulsanti associati ai simboli della stringa.

- In altre parole possiamo limitarci a considerare automi in cui tutti gli stati sono finali. Cioè per cui $F = Q$.

- Vediamo l'esempio della macchina distributrice di bevande calde a pag. 14 del vostro libro.

- In tale esempio si considera una semplice macchina che distribuisce tè oppure caffè. Il prezzo di un tè è 2p mentre quello di un caffè è 4p. La macchina accetta solo monete da 2p. Di tale macchina, come d'abitudine, non ne conosciamo la struttura interna ma interagiamo con essa attraverso la pressione di tasti, immettendo denaro, e prelevando la bevanda desiderata.

- Supponiamo allora che all'interno della macchina vi sia un meccanismo che funzioni come espresso dall'automa B1 del vostro libro. Il funzionamento è sicuramente molto intuitivo.

- Supponiamo poi di voler comprare tale macchina per il nostro ufficio, e di ricevere un'ottima offerta sotto costo da parte di un rivenditore locale. Ci informiamo sulle sue funzionalità e del perchè venga venduta sotto costo. Il rivenditore ci risponde che la vende ad un prezzo più basso solo perchè l'automa interno della macchina non è quello originale, B1, ma un altro, B2, **non deterministico,** ma completamente equivalente. E per suffragare la bontà della sua macchina ci mostra la garanzia, cioè che $\widehat{B1} = \widehat{B2}$.

- Che facciamo? La compriamo o no?

- Nel caso decidessimo di comprarla potremmo finire col passare delle mattinate "piuttosto nervose" nel tentativo di prendere un caffè o un tè.

– Ad esempio, supponiamo di aver voglia di un caffè, che costa 4p. Infiliamo dentro i primi 2p. Quando però tentiamo di di immettere i secondi 2p la macchina non ce lo consente. L'unica cosa che ci consente di fare è di premere il tasto del tè, e di prelevarlo. Cosa è successo?
  Dopo l'immissione dei 2p, essendo la macchina non deterministica, questa ha deciso (di sua iniziativa) di evolvere nello stato in cui l'unica cosa che si possa fare è premere il tasto del tè e prendersele. Ma io volevo un caffè!!!
– A quel punto ci rassegnamo e prendiamo il tè. L'indomani, pensando che il caffè non funzioni, immettiamo direttamente 2p con l'intenzione di prendere un tè. Questa volta però la macchina decide (sempre di sua iniziativa) di andare in uno stato in cui richiede altri 2p e non consente di prendere né tè né caffè fino a quando non vengano sborsati altri 2p. Nella speranza di ottenere qualcosa da bere immettiamo altri 2p e a questo punto la macchina mi consente di prendere un caffè, ma solo quello, niente tè, neppure se siamo disposti a pagarlo 4p, il doppio. Ma io volevo un tè!!!

- Diventa così chiaro che le due macchine (o automi), pur essendo equivalenti da un punto di vista dei linguaggi riconosciuti, non lo sono quando li vediamo da un punto di vista reattivo.

- **Lezione appresa:** La nozione di equivalenza classica per automi a stati finiti non ha senso quando si è interessati al comportamento interattivo di automi. In tal caso, un automa non determistico non può essere considerato equivalente ad uno deterministico.

- **Riepilogando:** dal nostro punto di vista i sistemi reattivi sono costituiti da una scatola nera (di cui non conosciamo il contenuto) e da una *interfaccia*.

- L'interfaccia è costituita da un insieme di *porte di comunicazione* che chiameremo *canali*.

- È attraverso i canali che il sistema reattivo interagisce con l'ambiente utilizzandoli per operazioni di input o di output.

- Nell'esempio della macchina del caffè l'interfaccia è composta da tre canali: $2p$, tea, and $coffee$.

- La macchina usa il canale $2p$ in input e gli altri due in output.

Diamo adesso alcune definizioni utili.

- Sia $\mathcal{N}$ un insieme di *nomi*. Esempi di nomi sono $a, b, c, \ldots$ ma anche tea, $coffee$, $2p$.
- Sia $\overline{\mathcal{N}} = \{\overline{a} \mid a \in \mathcal{N}\}$ l'insieme dei *co-nomi*.
- Sia $\mathcal{L} = \mathcal{N} \cup \overline{\mathcal{N}}$ l'insieme delle *etichette*. Le etichette identificheranno i pulsanti delle nostre scatole nere.
- Il nostro insieme di azioni Act coinciderà con $\mathcal{L}$. Useremo $\alpha, \beta, \ldots$ per denotare, indistintamente, nomi o co-nomi. $\alpha, \beta, \ldots \in Act$.
- Possiamo pensare ad a e $\overline{a}$ come azioni complementari eseguite da due processi che vogliono interagire.
- Nell'esempio della macchina del tè, in realtà, ci sono sempre due automi che interagiscono tra loro: la macchina e l'uomo. Quando l'uomo preme il pulsante 'tea', da un punto di vista dell'automa

accade la seguente cosa. L'uomo esegue un'azione tea (richiedi un tè in input) mentre la macchina esegue un'azione complementare $\overline{tea}$ (fornisci un tè in output)