Corso di Laurea in Sicurezza dei sistemi e delle reti informatiche

Fondamenti di informatica per la sicurezza

anno accademico 2004–2005

docente: Stefano Ferrari

21.04.2005 — Soluzione della seconda parte — vers. A

valutazioni **1** (4) _____ **2** (4) _____ **3** (4) _____ **4** (6) _____ **5** (6) _____ **6** (8) _____

Cognome _____

Nome _____

Matricola _____ **Firma** _____

Esercizio 1

Siano dati i linguaggi L_1 e L_2:

- $L_1 = \{a, ab, ba\}$
- $L_2 = \{c, dz\}$

Descrivere i linguaggi:

a) $L_3 = L_1 \cap L_2$

b) $L_4 = L_1 \cup L_2$

c) $L_5 = L_1 L_2$

d) $L_6 = {L_1}^2$

e) $L_7 = {L_1}^* {L_2}^*$

f) $L_8 = ({L_1}^2 L_2)^*$

Per quegli insiemi di cui sia troppo lungo (o impossibile) dare una descrizione estensionale, elencare almeno tre elementi, indicando le caratteristiche degli elementi che li compongono. In particolare, chiarire se la stringa vuota ϵ appartiene al linguaggio.

Soluzione

a) $L_3 = L_1 \cap L_2 = \emptyset$
Gli insiemi L_1 e L_2 non hanno elementi in comune, quindi la loro intersezione è vuota.

Nota: L'insieme vuoto $\emptyset$ è diverso dall'insieme costituito dalla sola stringa vuota, $\{\epsilon\}$.

b) $L_4 = L_1 \cup L_2 = \{a, ab, ba, c, dz\}$

c) $L_5 = L_1 L_2 = \{ac, abc, bac, adz, abdz, badz\}$

d) $L_6 = {L_1}^2 = \{aa, aba, baa, aab, abab, baab, abba, baba\}$

e) $L_7 = {L_1}^* {L_2}^*$
L'insieme L_7 è dato dalle stringhe formate come concatenazione di un numero arbitrario (eventualmente nullo) di elementi di L_1 seguito da una concatenazione di un numero arbitrario (eventualmente nullo) di elementi di L_2. Poiché sia ${L_1}^*$ che ${L_2}^*$ sono composto da infiniti elementi, anche L_7 avrà infiniti elementi. L'insieme $\{\epsilon, aaab, dzcdz, accdz\}$ è un sottoinsieme di L_7.

f) $L_8 = ({L_1}^2 L_2)^*$
L'insieme L_8 è formato dalla concatenazione di un numero arbitrario (eventualmente nullo) di stringhe composte da due elementi di L_1 e da un elemento di L_2. Pertanto, L_8 è composto da infiniti elementi. L'insieme $\{\epsilon, aadz, baabcbaadz\}$ è un sottoinsieme di L_8.

Esercizio 2

Sia data la seguente grammatica, $G = \langle T, V, P, S \rangle$, definita su $\Sigma = \{a, b, c, d\}$:

- insieme dei simboli terminali, T: $T = \Sigma$
- insieme dei metasimboli, V: $V = \{K, H\}$
- insieme delle regole di produzione, P: $P = \{S ::= K, K ::= a|Hb|Hc, H ::= b|Kd|Hc\}$

Quali fra le seguenti stringhe vengono generate da G?

a) $adcb$

b) *addb*

c) *bbdc*

d) *cccc*

e) *acdb*

Riportare la successione di regole da applicare per la generazione di tali stringhe e le stringhe parziali ottenute, spiegando perché non si possono ottenere le stringhe che eventualmente non risultassero appartenere al linguaggio generato da G.

Soluzione

a)

adcb	
	S
$S ::= K$	K
$K ::= Hb$	Hb
$H ::= Hc$	Hcb
$H ::= Kd$	$Kdcb$
$K ::= a$	$adcb$

La stringa *adcb* è generata da G: $adcb \in \mathcal{L}(G)$.

b)

addb	
	S
$S ::= K$	K
$K ::= Hb$	Hb
$H ::= Kd$	Kdb

Non esiste regola che generi il simbolo d dal metasimbolo K.

La stringa *addb* non può essere generata da G: $addb \notin \mathcal{L}(G)$.

c)

bbdc	
	S
$S ::= K$	K
$K ::= Hc$	Hc
$H ::= Kd$	Kdc
$K ::= Hb$	$Hbdc$
$H ::= b$	$bbdc$

La stringa *bbdc* è generata da G: $bbdc \in \mathcal{L}(G)$.

d)

cccc	
	S
$S ::= K$	K
$K ::= Hc$	Hc
$H ::= Hc$	Hcc
$H ::= Hc$	$Hccc$
$H ::= Hc$	$Hcccc$

La stringa *cccc* è stata ottenuta, ma non è possibile eliminare il metasimbolo H.

La stringa *cccc* non è generata da G: $cccc \notin \mathcal{L}(G)$.

e)

acdb	
	S
$S ::= K$	K
$K ::= Hb$	Hb
$H ::= Kd$	Kdb
$K ::= Hc$	$Hcdb$

Non esiste regola che generi il simbolo a dal metasimbolo H.

La stringa *acdb* non è generata da G: $acdb \notin \mathcal{L}(G)$.

Esercizio 3

Sia dato il seguente automa a stati finiti, A, $A = \langle Q, \Sigma, \delta, q_0, F \rangle$:

- insieme degli stati, Q: $Q = \{q_0, q_1, q_2, q_3\}$
- alfabeto di input, Σ: $\Sigma = \{a, b, c, d, e\}$
- funzione di transizione δ:

	a	b	c	d	e
q_0	q_2	q_1	q_2	q_3	q_1
q_1	q_1	q_3	q_2	q_1	q_1
q_2	q_3	q_0	q_1	q_0	q_3
q_3	q_3	q_2	q_0	q_0	q_2

- stato iniziale, q_0
- insieme di stati finali, F: $F = \{q_1\}$

Indicare:

a) quattro stringhe accettate da A

b) quattro stringhe rifiutate da A

Soluzione

a) quattro stringhe accettate da A:

- *aabc*
- *bad*
- *cebc*
- *dbce*

b) quattro stringhe rifiutate da A:

- *ab*
- *adbc*
- *eba*
- *cabd*

Esercizio 4

Modellare l'utilizzo di un ascensore mediante un automa a stati finiti.

L'ascensore opera in una palazzina a due piani (piano terra e primo piano) ed è dotato di una porta esterna e di una porta interna.

Per potere attivare l'ascensore dall'interno, entrambe le porte devono essere chiuse. L'operatore che si deve modellare può aprire e chiudere una porta (alla volta), entrare o uscire dall'ascensore e attivare l'ascensore.

Ipotizzare che non si possano verificare contemporaneamente più azioni. Modellare l'automa in modo che esso accetti solo le stringhe che descrivono il funzionamento sicuro dell'ascensore. In particolare, individuare possibili situazioni fisicamente irrealizzabili o pericolose e formalizzarle in modo che l'automa rifiuti le successioni di azioni che porterebbero l'ascensore in tali situazioni.

Stati e simboli riportati nel testo sono solo indicativi: possono essere modificati, ridotti ed estesi a secondo delle esigenze del progetto.

Soluzione

L'automa deve modellare il comportamento di un utente. L'insieme dei simboli di input modella quindi le azioni che l'utente opera (o, più in generale, gli scambi di informazione con l'esterno, sotto forma di azioni o segnali) e gli stati descrivono le situazioni in cui l'utente viene a trovarsi.

Questo permette di vedere l'automa come un simulatore dell'utente in esame: l'automa deve accettare le stringhe che rappresentano le sequenze di azioni fisicamente realizzabili oppure quelle che rappresentano una sequenza di eventi di particolare interesse.

L'insieme degli stati dell'utente è descrivibile come l'insieme delle situazioni in cui si trova ad operare. Tali situazioni sono date dalla combinazione di diversi fattori:

- il piano a cui si trova (piano terra e primo piano);
- la posizione rispetto all'ascensore (dentro e fuori);
- lo stato della porta esterna (aperta e chiusa);
- lo stato della porta interna (aperta e chiusa).

Eventualmente, è possibile aggiungere anche lo stato di attività dell'ascensore. Tuttavia, quest'ultimo è un fattore secondario, da tenere conto solo per una modellizzazione molto dettagliata. Non sarà perciò considerato, e l'attività dell'ascensore (lo spostamento tra i piani) sarà ritenuto istantaneo e privo di problemi.

Anche il piano a cui l'utente si trova è secondario e in un primo momento non sarà considerato.

Poichè i fattori rimanenti sono tra loro indipendenti (porta interna, porta esterna e posizione utente), gli stati risulteranno dalla combinazione dei singoli fattori. Per semplificare la trattazione, associamo ad ogni stato un nome che codifichi le proprietà dello stato stesso tramite una sequenza di tre simboli (per esempio, DI_AE_C). Il primo simbolo codifica la posizione dell'utente rispetto all'ascensore: dentro, D, o fuori, F. Il secondo simbolo codifica lo stato della porta interna: aperta, I_A, o chiusa, I_C. Analogamente, il terzo simbolo codifica lo stato della porta esterna: aperta, E_A, o chiusa, E_C.

Pertanto, l'insieme degli stati, Q, può essere:

$$\begin{aligned} Q \;=\; & \{FI_AE_A,\, FI_AE_C,\, FI_CE_A,\, FI_CE_C,\\ & DI_AE_A,\, DI_AE_C,\, DI_CE_A,\, DI_CE_C,\, err\} \end{aligned}$$

Agli stati già citati è stato aggiunto lo stato *err* che verrà utilizzato per modellare le situazioni di errore.

Le azioni che possono essere effettuate dall'utente sono l'apertura e la chiusura delle porte, entrare e uscire dall'ascensore, e azionare l'ascensore. Esse saranno rappresentate dall'insieme dei simboli, Σ:

$$\begin{aligned} \Sigma \;=\; & \{apreI,\, chiudeI,\, apreE,\, chiudeE,\, entra,\\ & esce,\, aziona\} \end{aligned}$$

dove *apreI*, *chiudeI* rappresentano l'apertura e la chiusura della porta interna, e analogo significato l'hanno i simboli *apreE*, *chiudeE*, riferiti alla orta esterna.

Le specifiche descrivono i seguenti vincoli:

- dall'interno, non si può agire sulla porta esterna se la porta interna è chiusa;
- dall'esterno, non si può agire sulla porta interna se la porta esterna è chiusa;
- l'ascensore può essere attivato solo dall'interno;
- l'ascensore può essere attivato solo con entrambe le porte chiuse;

- l'utente non può entrare o uscire dall'ascensore se almeno una delle due porte è chiusa.

Le situazioni che devono essere rifiutate dall'automa sono classificabili in tre categorie:

- situazioni fisicamente irrealizzabili;
- situazioni senza significato (per esempio, entrare quando si è già dentro o chiudere una porta già chiusa);
- situazioni da evitare (cioé che causano pericolo).

L'ultimo tipo di situazione rappresenta un errore grave e quindi l'automa deve catturare ogni sequenza di azioni che porti in una situazione del genere. Questo viene realizzato mediante lo stato *err*, costruendo la tabella delle transizioni che non permetta di lasciare *err* una volta raggiuntolo. Le prime due categorie rappresentano errori lievi, e potrebbero essere gestite facendo rimanere l'automa nello stato in cui si trova al momento dell'azione incriminata. Tuttavia, per rendere l'automa più selettivo, anche queste situazioni verranno catturate dallo stato *err*.

Ogni sequenza di azioni che non comporti il raggiungimento dello stato *err* rappresenta il normale comportamento dell'utente. Pertanto, qualsiasi sequenza di simboli che non porti nello stato *err* deve venire accettata, e, quindi, tutti gli stati tranne *err* compongono l'insieme degli stati finali, F.

Si può ipotizzare che lo stato iniziale sia quello relativo all'utente all'esterno, con entrambe le porte aperte, FI_AE_A.

Con le ipotesi fatte, dovrebbero essere accettate, per esempio, le seguenti sequenze di azioni:

- $entra - chiudeE - chiudeI - aziona - apreI - apreE - esce$,
- $chiudeI - chiudeE - apreE - apreI - entra$.

Al contrario, verrebbero rifiutate, tra le altre, le seguenti sequenze di azioni:

- $entra - chiudiI - esce$,
- $chiudeI - aziona$,
- $chiudeE - entra$,
- $entra - aziona$

Va notato che aggiungendo un qualsiasi suffisso ad una stringa rifiutata, si ottiene sempre una stringa rifiutata: se una certa sequenza di azioni porta in uno stato non accettabile, qualsiasi sequenza di azioni ad essa successiva non può renderla accettabile.

La tabella delle transizioni, $\delta : Q \times \Sigma \to Q$ può essere quella riportata in Tabella 1.

Diverse semplificazioni sono possibili. Per esempio, si possono unificare alcune azioni complementari: *entra* e *esce*, può diventare *transita*, *apreI* e *chiudeI* può diventare *azionaI*, e, analogamente, *apreE* e *chiudeE* può diventare *azionaE*. In questo caso, la tabella delle transizioni, $\delta : Q \times \Sigma \to Q$ può essere quella riportata in Tabella 2.

Un'altra semplificazione, che però parzialmente discorda con lo spirito delle specifiche, si ottiene ignorando la distinzione tra porta interna ed esterna, contando solo il numero di porte aperte. In tal modo, si riesce a catturare gli errori dovuti al tentativo di oltrepassare una porta chiusa e all'azionare l'ascensore con una porta aperta. Non si riesce però a descrivere gli errori causati dalle sequenze sbagliate di azioni sulle porte.

Esercizio 5

Sia data l'espressione regolare E, definita su $\Sigma = \{a, b, c\}$:

- $E = c^2(cb + a)^* + ab^*c^2$

Quali fra le seguenti stringhe vengono descritte da E?

a) *cccbcb*

b) *aaabcc*

c) *abbbcc*

d) *acc*

e) *ccaaacb*

f) *ccaccb*

Soluzione

Le espressioni regolari denotano degli insiemi di stringhe. In tal senso, possiamo applicare l'operatore di relazione insiemistica $\subseteq$ alle espressioni regolari per indicare che l'insieme denotato da un'espressione contiene l'insieme denotato da una seconda espressione regolare. Per esempio, $E_1 \subseteq E_2$ significa che tutte le stringhe descritte da E_1 sono descritte anche da E_2.

$\boldsymbol{\delta}$	$entra$	$esce$	$apreI$	$chiudeI$	$apreE$	$chiudeE$	$aziona$
FI_AE_A	DI_AE_A	err	err	FI_CE_A	err	FI_AE_C	err
FI_AE_C	err	err	err	err	FI_AE_A	err	err
FI_CE_A	err	err	FI_AE_A	err	err	FI_CE_C	err
FI_CE_C	err	err	err	err	FI_CE_A	err	err
DI_AE_A	err	FI_AE_A	err	DI_CE_A	err	DI_AE_C	err
DI_AE_C	err	err	err	DI_CE_C	DI_AE_A	err	err
DI_CE_A	err	err	DI_AE_A	err	err	err	err
DI_CE_C	err	err	DI_AE_C	err	err	err	DI_CE_C
err	err	err	err	err	err	err	err

Tabella 1: Tabella delle transizioni dell'automa dell'esercizio 4.

$\boldsymbol{\delta}$	$transita$	$azionaI$	$azionaE$	$aziona$
FI_AE_A	DI_AE_A	FI_CE_A	FI_AE_C	err
FI_AE_C	err	err	FI_AE_A	err
FI_CE_A	err	FI_AE_A	FI_CE_C	err
FI_CE_C	err	err	FI_CE_A	err
DI_AE_A	FI_AE_A	DI_CE_A	DI_AE_C	err
DI_AE_C	err	DI_CE_C	DI_AE_A	err
DI_CE_A	err	DI_AE_A	err	err
DI_CE_C	err	DI_AE_C	err	DI_CE_C
err	err	err	err	err

Tabella 2: Tabella delle transizioni alternativa dell'automa dell'esercizio 4.

Ricordando che l'espressione regolare s descrive l'insieme di stringhe composto dalla sola s, $\{s\}$, si può dimostrare che tale stringa viene descritta da un'espressione regolare E derivando una catena di inclusioni del tipo $s \subseteq E_1 \subseteq E_2 \subseteq \cdots \subseteq E_k \equiv E$.

Osserviamo innanzitutto che le stringhe che vengono descritte da E sono descritte, in alternativa, o dalla sottoespressione $E_1 = c^2(cb+a)^*$ o dalla sottoespressione $E_2 = ab^*c^2$. Questa premessa semplificherà la spiegazione delle soluzioni di seguito riportate.

a) *cccbcb*
$cccbcb \subseteq (cc)(cb)(cb) \subseteq c^2(cb)^2 \subseteq c^2(cb+a)^2 \subseteq c^2(cb+a)^* \subseteq c^2(cb+a)^* + ab^*c^2$

La stringa *cccbcb* viene descritta da E: $cccbcb \in \mathcal{L}(E)$.

b) *aaabcc*
Le stringhe generate da E iniziano con la doppia c (quando sono generate da E_1) oppure per a (quando sono generate da E_2). In quest'ultimo caso, però, il simbolo a compare solo una volta.

La stringa *aaabcc* ha più di un simbolo a, e quindi non può essere descritta da E: $aaabcc \notin \mathcal{L}(E)$.

c) *abbbcc*
$abbbcc \subseteq (a)(bbb)(cc) \subseteq ab^3c^2 \subseteq ab^*c^2 \subseteq c^2(cb+a)^* + ab^*c^2$

La stringa *abbbcc* viene descritta da E: $abbbcc \in \mathcal{L}(E)$.

d) *acc*
$acc \subseteq (a)(cc) \subseteq ac^2 \subseteq ab^*c^2 \subseteq c^2(cb+a)^* + ab^*c^2$

La stringa *acc* viene descritta da E: $acc \in \mathcal{L}(E)$.

e) *ccaaacb*
$ccaaacb \subseteq (cc)(aaa)(cb) \subseteq c^2(a+cb)^4 \subseteq c^2(a+cb)^* \subseteq c^2(cb+a)^* + ab^*c^2$

La stringa *ccaaacb* viene descritta da E: $ccaaacb \in \mathcal{L}(E)$.

f) *ccaccb*
La stringa *ccaccb* non può essere generata da E_2 perché non ha il simbolo a come prefisso. Le stringhe generate da E_1 hanno cc come suffisso, ma la rimanente sottostringa deve essere composta da simboli a o da stringhe cb. In nessun caso può esserci due simboli c successivi.

La stringa *ccaccb* non viene quindi descritta da E: $ccaccb \notin \mathcal{L}(E)$.

Esercizio 6

Indicare una espressione regolare (non banale) definita su $\Sigma = \{a, b, c\}$ che descriva le seguenti stringhe:

- $aabbc$
- $ccbbbbc$
- ccc
- $aabbbc$

ma non le seguenti:

- $acbbbc$
- $acccbc$
- acc
- $cabbba$

Soluzione

Si può notare che tutte le stringhe da includere hanno a^2 o c^2 come suffisso. Questa caratteristica può essere descritta dall'espressione regolare $(a^2 + c^2)(a + b + c)^*$.

Nessuna delle stringhe del secondo gruppo viene descritta da tale espressione regolare, in quanto tutte le stringhe hanno i primi due simboli diversi.

Altre espressioni regolari che rispettano le specifiche del problema sono:

- $(a^2 + c^2)(b + c)^*$
- $(a^2 + b^*c)^*$
- $a^2b^*c + (b + c)*$