Esercizi tipo, 2022

Nel seguito, dati

- lo stato P di un automa deterministico A
- la stringa $\beta = X_1X_2 \ldots X_n$

si indica con $P[\![X_1X_2 \ldots X_n]\!]$ lo stato di A che si raggiunge da P tramite il cammino $X_1X_2 \ldots X_n$.

Esercizio 1-1

Sia $\mathcal{L} = \{ww \mid w \in \mathcal{L}((a \mid b)^*)\}$. Se $\mathcal{L}$ è un linguaggio regolare rispondere "SI" e dire quanti stati ha il minimo DFA per il riconoscimento di $\mathcal{L}$ e quanti di questi stati sono finali. Se invece $\mathcal{L}$ non è regolare, allora rispondere "NO" e fornire una stringa z da utilizzare con successo nella dimostrazione per contraddizione rispetto al Pumping Lemma dei linguaggi regolari.

NO $z = a^p b^p a^p b^p$

Esercizio 1-2

Se la seguente affermazione è vera rispondere "VERO", altrimenti rispondere "FALSO": *"Se i linguaggi $\mathcal{L}_1$ e $\mathcal{L}_2$ sono entrambi regolari allora $\mathcal{L}_1 \cup \mathcal{L}_2$ è regolare."*

VERO

Esercizio 1-3

Sia $r = b^* \mid b^*a(\epsilon \mid a \mid b)^*$ e sia $\mathcal{D}$ il DFA minimo per il riconoscimento di $\mathcal{L}(r)$. Dire quanti stati ha $\mathcal{D}$ e quanti di questi stati sono finali.

1 stato, 1 finale

Esercizio 1-4

Sia $\mathcal{N}_1$ lo NFA con stato iniziale A, stato finale E e con la seguente funzione di transizione

	ϵ	a	b
A	$\{B, E\}$	$\emptyset$	$\emptyset$
B	$\{C\}$	$\emptyset$	$\{E\}$
C	$\emptyset$	$\{D\}$	$\emptyset$
D	$\{E\}$	$\emptyset$	$\{B\}$
E	$\emptyset$	$\{E\}$	$\{A\}$

Chiamiamo $\mathcal{D}$ il DFA ottenuto da $\mathcal{N}_1$ per subset construction e Q lo stato iniziale di $\mathcal{D}$. Dire a quale sottoinsieme degli stati di $\mathcal{N}_1$ corrisponde $Q[\![ab]\!]$.

$\{A, B, C, E\}$

Esercizio 1-5

Sia $\mathcal{D}_1$ il DFA con stato iniziale A, stato finale D e con la seguente funzione di transizione

	a	b
A	B	
B	D	C
C	D	
D		B

Chiamiamo $\mathcal{D}_m$ il DFA ottenuto per minimizzazione di $\mathcal{D}_1$ e P lo stato iniziale di $\mathcal{D}_m$. Dire a quale sottoinsieme degli stati di $\mathcal{D}_1$ corrisponde $P[\![abab]\!]$.

$\{B\}$

Esercizio 2-1

Scrivere l'intera riga della tabella di parsing LL(1) per $\mathcal{G}_1$ relativa al non-terminale B.

	a	b	c	\$
B	$B \to \varepsilon$		$B \to \varepsilon$	$B \to \varepsilon$

Esercizio 2-2

Chiamiamo $\mathcal{A}$ l'automa caratteristico per il parsing LR(1) di $\mathcal{G}_1$, I lo stato iniziale di $\mathcal{A}$, T la tabella di parsing LR(1) per $\mathcal{G}_1$. Se T non contiene alcun conflitto nello stato $I[\![BcBa]\!]$, rispondere "NO CONFLICT". Altrimenti, per ciascuna X tale che $T[I[\![BcBa]\!], X]$ contiene un conflitto, dire, specificando a quale X si fa riferimento: (i) di che tipo di conflitto si tratta; (ii) quale/i riduzione/i sono coinvolte.

NO CONFLICT

Esercizio 2-3

Chiamiamo $\mathcal{A}$ l'automa caratteristico per il parsing LR(1) di $\mathcal{G}_1$ e J lo stato iniziale di $\mathcal{A}$. Elencare gli item che appartengono a $J[\![Aa]\!]$.

$S \to Aa \cdot B, \{\$\}$

$B \to \cdot, \{\$\}$

Esercizio 2-4

Chiamiamo $\mathcal{A}$ l'automa caratteristico per il parsing LALR(1) di $\mathcal{G}_1$, H lo stato iniziale di $\mathcal{A}$, T la tabella di parsing LALR(1) per $\mathcal{G}_1$. Se non ci sono conflitti nello stato $H[\![BcBaBc]\!]$ di T, rispondere "NO CONFLICT". Altrimenti, per ciascuna X tale che $T[H[\![BcBaBc]\!], X]$ contiene un conflitto, dire, specificando a quale X fa riferimento: (i) di che tipo di conflitto si tratta; (ii) quale/i riduzione/i sono coinvolte.

NO CONFLICT

Esercizio 3-1

Sia P lo stato iniziale del parser LALR(1) per la grammatica dello SDD $\mathcal{S}_1$. Il parser ha 4 conflitti shift/reduce: uno in $[P[\![EaE]\!], a]$, uno in $[P[\![EaE]\!], b]$, uno in $[P[\![EbE]\!], a]$ e uno in $[P[\![EbE]\!], b]$. Supponiamo che tutti e 4 i conflitti siano risolti a favore dello shift. Supponiamo inoltre che l'attributo $n.lexval$ del terminale n sia il numero intero rappresentato da n. Se l'input $2a3b4$ non è riconosciuto, rispondere "ERROR". Altrimenti dire quale valore viene valutato per $S.v$ su input $2a3b4$.

14

Esercizio 3-2

Sia P lo stato iniziale del parser LALR(1) per la grammatica dello SDD $\mathcal{S}_1$. Il parser ha 4 conflitti shift/reduce: uno in $[P[\![EaE]\!], a]$, uno in $[P[\![EaE]\!], b]$, uno in $[P[\![EbE]\!], a]$ e uno in $[P[\![EbE]\!], b]$. Alcuni di questi conflitti dipendono dal fatto che la grammatica non modella la precedenza dell'operatore di moltiplicazione (operatore a) sull'operatore di somma (operatore b). Si dica quali conflitti sono dovuti alla suddetta carenza della grammatica e si dica come risolvere ciascuno di essi per fare in modo che a abbia precedenza su b.

[P[[EaE]], b] risolvere con REDUCE

[P[[EbE]], a] risolvere con SHIFT

Esercizio 3-3

Sia $\mathcal{S}_{1b}$ il seguente SDD:

$$
\begin{array}{lcll}
S & \rightarrow & A & \{while(T\,not\,empty)\{num = pop(T); print(num,);\}\} \\
A & \rightarrow & CB & \{push(T,1);\} \\
B & \rightarrow & aCB & \{push(T,2);\} \\
B & \rightarrow & \epsilon & \{push(T,3);\} \\
C & \rightarrow & ED & \{push(T,4);\} \\
D & \rightarrow & bED & \{push(T,5);\} \\
D & \rightarrow & \epsilon & \{push(T,6);\} \\
E & \rightarrow & g & \{push(T,7);\}
\end{array}
$$

dove T è una pila inizialmente vuota. Sulla pila sono definite le comuni funzioni di $push(pila, elemento)$ e $pop(pila)$. Si immagini di analizzare $\mathcal{S}_{1b}$ con un parser LALR(1). Se l'input $gagbg$ non è riconosciuto dal parser scrivere "ERROR". Altrimenti scrivere il risultato della valutazione dell'input $gagbg$.

1, 2, 3, 4, 5, 6, 7, 7, 4, 6, 7

Esercizio 3-4

Sia $\mathcal{G}$ la seguente grammatica ambigua per un linguaggio con identificatori *id* e operatori binari *a* e *b*

$$S \rightarrow S\,a\,S \mid S\,b\,S \mid (S) \mid id$$

Fornire una grammatica LL(1) per la generazione di $\mathcal{L}(\mathcal{G})$ in cui l'ambiguità è risolta rispettando le seguenti convenzioni: l'operatore *a* ha precedenza sull'operatore *b*; entrambi gli operatori associano a sinistra.

$E \rightarrow TE'$

$E' \rightarrow bTE' \mid \varepsilon$

$T \rightarrow FT'$

$T' \rightarrow aFT' \mid \varepsilon$

$F \rightarrow (E) \mid id$

Esercizio 3-5

Sia $\mathcal{D}$ la seguente porzione di *syntax directed translation*:

$$\begin{array}{lcll} P & \rightarrow & S & \{\ S.next = newlabel() \\ & & & \phantom{\{}\ P.code = S.code \triangleright label(S.next)\ \ \} \\ S & \rightarrow & \mathsf{for}\ (S_1\ ;\ B\ ;\ S_2)\ S_3 & \end{array}$$

Assumendo che:

- B è gestita con gli usuali attributi $B.code$, $B.true$ e $B.false$
- la semantica del comando "$\mathsf{for}\ (S_1\ ;\ B\ ;\ S_2)\ S_3$" è la stessa di "$S_1\ ;\ \mathsf{while}\ (B)\ \{S_3\ ;\ S_2;\};$"

dire quali regole semantiche vanno associate all'ultima produzione per ottenere la traduzione del for-statement.

$S_1.next = newlabel()$; $S_2.next = S_1.next$
$S_3.next = newlabel()$; $B.true = newlabel()$;
$B.false = S.next$;

$S.code = S_1.code \triangleright label(S_1.next) \triangleright B.code \triangleright$
$label(B.true) \triangleright S_3.code \triangleright$
$label(S_3.next) \triangleright S_2.code \triangleright$
$gen(GOTO\ S_1.next)$

Esercizio 3-6

Sia data la seguente *syntax-directed translation* per array

$S \rightarrow id = E$	$gen(table.get(id)\ '='\ E.addr)$
$S \rightarrow L = E$	$gen(L.array\_base\ '['\ L.addr\ ']'\ '='\ E.addr)$
$E \rightarrow E_1 + E_2$	$E.addr = newtemp()$ $gen(E.addr\ '='\ E_1.addr\ '+'\ E_2.addr)$
$E \rightarrow id$	$E.addr = table.get(id)$ $E.code = ''$
$E \rightarrow L$	$E.addr = newtemp()$ $gen(E.addr\ '='\ L.array.base\ '['L.addr']')$
$L \rightarrow id[E]$	$L.array = table.get(id)$ $L.type = arg2(table.getType(id))$ $L.width = width(L.type)$ $L.addr = newtemp()$ $gen(L.addr\ '='\ E.addr\ '*'\ L.width)$
$L \rightarrow L_1[E]$	$L.array = L_1.array$ $L.type = arg2(L_1.type)$ $L.width = width(L.type)$ $t = newtemp()$ $gen(t\ '='\ E.addr\ '*'\ L.width)$ $L.addr = newtemp()$ $gen(L.addr\ '='\ L_1.addr\ '+'\ t)$

Si assumano le seguenti condizioni: il tipo di a è $array(2, array(3, integer))$; la *base* di a è 0; c, i, j sono interi; la dimensione di un intero è 4. Si assuma inoltre di risolvere l'ambiguità della grammatica assegnando all'operatore di somma la usuale associtività a sinistra. Dire quale codice viene generato nell'analisi bottom-up della stringa

$$b = c + a[i][j]$$

$t_1 = i * 12$

$t_2 = j * 4$

$t_3 = t_1 + t_2$

$t_4 = a[t_3]$

$t_5 = c + t_4$

$b = t_5$