Linguaggi Formali e Compilatori
(Formal Languages and Compilers)

prof. S. Crespi Reghizzi, prof.ssa L. Sbattella
(prof. Luca Breveglieri)

Prova scritta - 5 luglio 2007 - Parte I: Teoria

CON SOLUZIONI - a scopo didattico le soluzioni sono molto estese e commentate variamente - non si richiede che il candidato svolga il compito in modo altrettanto ampio, bensì che risponda in modo appropriato e a suo giudizio ragionevole

NOME:

COGNOME:

MATRICOLA: FIRMA:

ISTRUZIONI - LEGGERE CON ATTENZIONE:

- L'esame si compone di due parti:
  - I (80%) Teoria:
    1. espressioni regolari e automi finiti
    2. grammatiche libere e automi a pila
    3. analisi sintattica e parsificatori
    4. traduzione sintattica e analisi semantica
  - II (20%) Esercitazioni Flex e Bison
- Per superare l'esame l'allievo deve avere sostenuto con successo entrambe le parti (I e II), in un solo appello oppure in appelli diversi, ma entro un anno.
- Per superare la parte I (teoria) occorre dimostrare di possedere conoscenza sufficiente di tutte le quattro sezioni (1-4).
- È permesso consultare libri e appunti personali.
- Per scrivere si utilizzi lo spazio libero e se occorre anche il tergo del foglio; è vietato allegare nuovi fogli o sostituirne di esistenti.
- Tempo: Parte I (teoria): 2h.30m - Parte II (esercitazioni): 45m

1 Espressioni regolari e automi finiti 20%

1. Il linguaggio regolare L di alfabeto $\{a, b, c\}$ è definito per mezzo delle tre *condizioni* seguenti (ogni frase di L le deve soddisfare tutte e tre):

   (a) una frase di L inizia con la lettera a

   and

   (b) una frase di L termina con la lettera c

   and

   (c) una frase di L non può contenere nessuna delle sottostringhe seguenti:

   $$b\,a \qquad b\,b \qquad c\,a \qquad c\,c$$

   Per esempio la stringa seguente:

   $$a\,b\,c\,b\,c$$

   è valida e appartiene a L, mentre la stringa:

   $$a\ \underbrace{b\,a}_{\text{vietata}}\ b\,c$$

   non è valida (non è una frase di L) perché la sottostringa evidenziata è proibita.

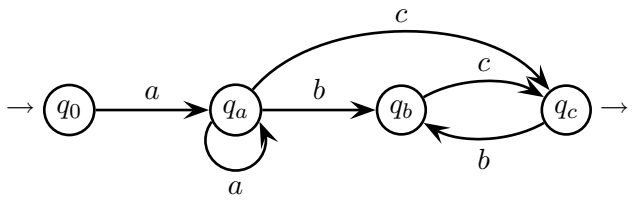
   Si risponda ai punti seguenti:

   (a) Si scriva un'espressione regolare del linguaggio L usando soltanto gli operatori di unione, concatenamento e stella (o croce).

   (b) Si definisca il linguaggio $\overline{L}$, complemento di L, modificando opportunamente le tre condizioni (a, b, c).

Soluzione

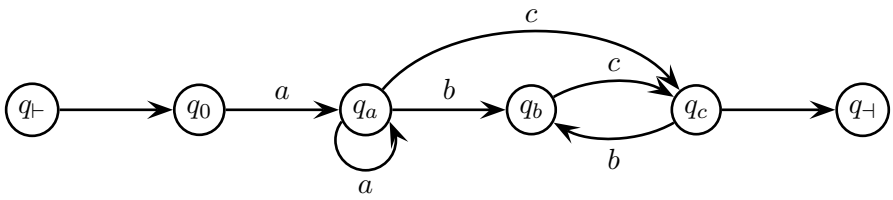
(a) Per ottenere un'espressione regolare del linguaggio L se ne può costruire l'automa. Poiché L è un linguaggio della sottofamiglia detta *locale*, l'automa ha come stati le lettere dell'alfabeto più lo stato iniziale. Esso deve soltanto ricordare l'ultima lettera letta e controllare che la stringa inizi e termini con le lettere prescritte dalle condizioni (a, b, c). Ecco l'automa in questione:



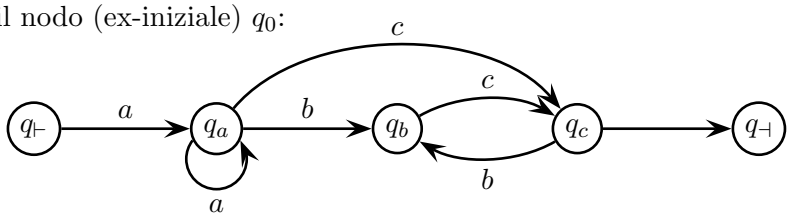
I nomi degli stati (q_a, q_b e q_c) sono in corrispondenza biunivoca con le lettere dell'alfabeto (tranne lo stato iniziale q_0). Gli archi uscenti rappresentano i caratteri iniziali, finali e le coppie ammissibili di caratteri adiacenti.

Per trovare un'espressione regolare R equivalente (con solo gli operatori di concatenamento unione e stella o croce) si può usare per esempio il metodo di eliminazione dei nodi. Eccone i passaggi ($q_\vdash$ e $q_\dashv$ sono i nodi iniziale e finale):

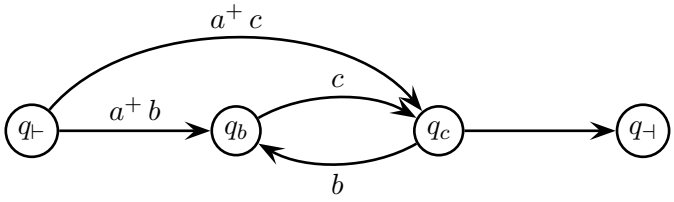
Prepara l'automa:



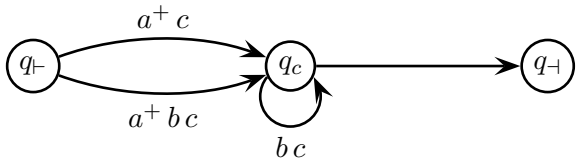
Elimina il nodo (ex-iniziale) q_0:



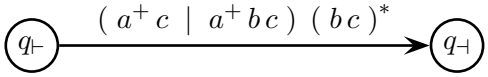
Elimina il nodo q_a:



Elimina il nodo q_b:



Elimina il nodo q_c:



E infine si ha l'espressione regolare R seguente:

$$\begin{aligned} R &= \left(a^+ c \mid a^+ b c \right) (b c)^* \\ &= a^+ c (b c)^* \mid a^+ b c (b c)^* \\ &= a^+ (c \mid b c) (b c)^* \\ &= a^+ (c b)^* c \mid a^+ (b c)^+ \end{aligned}$$

qui trasformata in quattro forme diverse ma equivalenti. La quarta forma si ottiene dalla seconda osservando che vale l'identità $c (b c)^* = (c b)^* c$.

(b) Si ricordi che L è un linguaggio locale, definito come congiunzione logica (and) delle tre condizioni (a, b, c). Pertanto il complemento $\overline{L}$ è definito come disgiunzione logica (or) delle stesse tre condizioni in forma negata (teorema di De Morgan). Si ha:

- una frase di $\overline{L}$ *non inizia* con la lettera a (= inizia con b o c)

  or

- una frase di $\overline{L}$ *non termina* con la lettera c (= termina con a o b)

  or

- una frase di $\overline{L}$ *non può non contenere nessuna* (= deve contenere almeno una) delle sottostringhe seguenti:

  $b\,a$ $\qquad$ $b\,b$ $\qquad$ $c\,a$ $\qquad$ $c\,c$

Inoltre va da sé che $\overline{L}$ contiene ε, giacché L è ε-free. Tanto basta per rispondere degnamente al punto (b). Si badi bene che le tre condizioni ora sono disgiunte, non congiunte, cioè basta che la stringa ne soddisfi una per appartenere a $\overline{L}$ (va bene anche se ne soddisfa due o tutte e tre).

Volendo approfondire un po' la questione, dato che L è un linguaggio locale un'espressione regolare generalizzata R (contenente anche l'operatore di differenza insiemistica o quelli di intersezione e complemento) che lo esprime è la seguente:

$$R = a\Sigma^*c - (\Sigma^*\{b\,a, b\,b, c\,a, c\,c\}\Sigma^*) = a\Sigma^*c \cap \overline{\Sigma^*\{b\,a, b\,b, c\,a, c\,c\}\Sigma^*}$$

Beninteso essa è equivalente all'espressione R trovata al punto (a), benché sia formulata diversamente. Un'espressione regolare ordinaria R' (con solo concatenamento unione e stella) che esprime il complemento $\overline{L}$ è allora la seguente:

$$\begin{aligned}
R' &= \overline{R} \\
&= \overline{a\Sigma^*c \cap \overline{\Sigma^*\{b\,a, b\,b, c\,a, c\,c\}\Sigma^*}} \\
&= \overline{a\Sigma^*c} \cup \overline{\overline{\Sigma^*\{b\,a, b\,b, c\,a, c\,c\}\Sigma^*}} \\
&= \overline{(a\Sigma^* \cap \Sigma^*c)} \cup \Sigma^*\{b\,a, b\,b, c\,a, c\,c\}\Sigma^* \\
&= \overline{a\Sigma^*} \cup \overline{\Sigma^*c} \cup \Sigma^*\{b\,a, b\,b, c\,a, c\,c\}\Sigma^* \\
&= \varepsilon \cup \{b, c\}\Sigma^* \cup \varepsilon \cup \Sigma^*\{a, b\} \cup \Sigma^*\{b\,a, b\,b, c\,a, c\,c\}\Sigma^* \\
&= \varepsilon \cup \{b, c\}\Sigma^* \cup \Sigma^*\{a, b\} \cup \Sigma^*\{b\,a, b\,b, c\,a, c\,c\}\Sigma^*
\end{aligned}$$

Si ha senz'altro $\overline{L} = L(R')$. Come è ragionevole, l'espressione R' ha struttura conforme alle tre condizioni date prima (più ε), dove l'unione rappresenta la disgiunzione (or). Con la simbologia usuale si scrive come segue:

$$\begin{aligned}
R' &= \varepsilon \mid (\ b \mid c\ )\ (\ a \mid b \mid c\ )^* \mid (\ a \mid b \mid c\ )^*\ (\ a \mid b\ ) \\
&\mid (\ a \mid b \mid c\ )^*\ (\ b\,a \mid b\,b \mid c\,a \mid c\,c\ )\ (\ a \mid b \mid c\ )^*
\end{aligned}$$

Si potrebbe ottenere R' anche nel modo seguente: si prenda l'automa deterministico di L trovato prima e se ne costruisca l'automa complemento, che riconosce $\overline{L}$, poi si ricavi da quest'ultimo l'espressione regolare equivalente, per esempio tramite il metodo di eliminazione dei nodi; si lascia la costruzione al lettore.

2. È data l'espressione regolare R seguente (in forma generalizzata giacché contiene l'operatore di intersezione), di alfabeto $\{a, b, c\}$:

$$R = a\ (\ b\ c^*\ )^* \quad \cap \quad (\ a\ \mid\ b\ )\ \left(\ a^*\ b^*\ c^+\ \right)^*$$

Si risponda ai punti seguenti:

(a) Si costruisca in modo sistematico un automa riconoscitore equivalente a R (cioè lo si costruisca come intersezione di automi).

(b) Se occcorre si renda deterministico il riconoscitore così costruito.

Soluzione

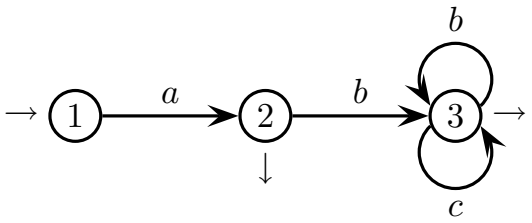
(a) Intuitivamente l'intersezione è data dall'espressione regolare R' seguente:

$$R' = \left(\ a\ \left(\ b^+\ c^+\ \right)^*\ \right)$$

che è del tutto equivalente all'espressione R data nel testo dell'esercizio.

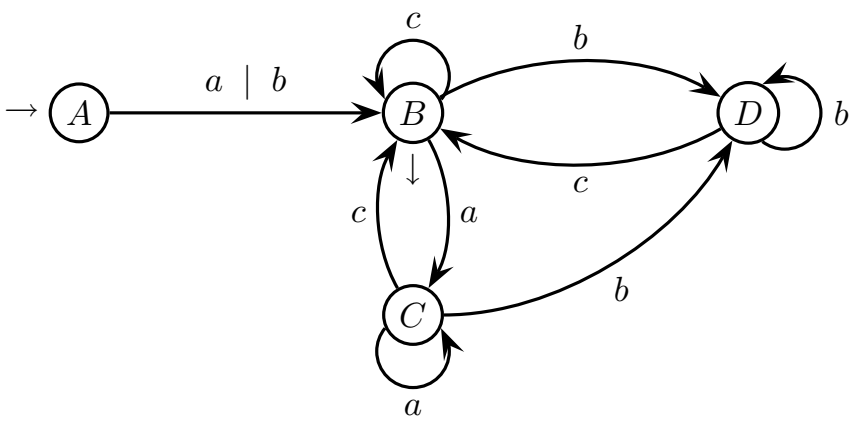
Per trovare l'intersezione in modo sistematico, gli automi delle sottoespressioni $a\ (\ b\ c^*\ )^*$ e $(\ a\ \mid\ b\ )\ (\ a^*\ b^*\ c^+\ )^*$ si possono ricavare con i metodi di Berri-Sethi o McNaughton-Yamada, ma sono facili da trovare anche intuitivamente.

Per la sottoespressione $a\ (\ b\ c^*\ )^*$ si ha l'automa seguente:



Si ottiene subito l'automa osservando che $a\ (\ b\ c^*\ )^* = a\ \mid\ a\,b\ (\ b\ \mid\ c\ )^*$, perché $(\ b\ c^*\ )^*$ è ε oppure una stringa qualunque di b e c che inizia con b.

Per la sottoespressione $(\ a\ \mid\ b\ )\ (\ a^*\ b^*\ c^+\ )^*$ si ha l'automa seguente:

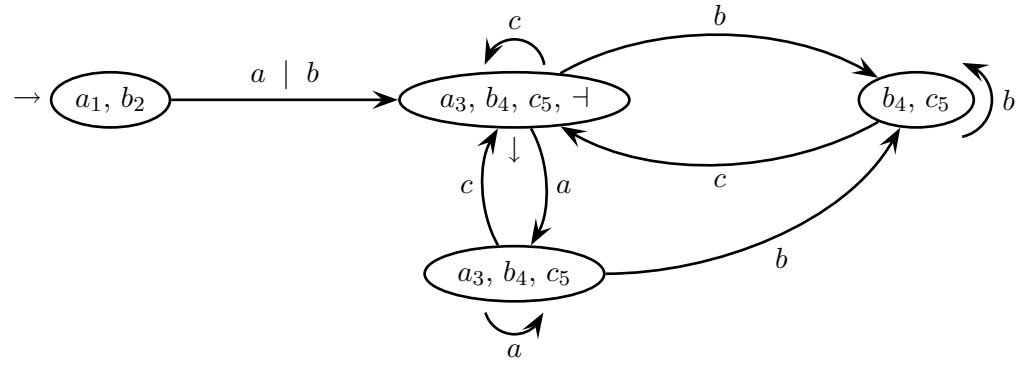


Entrambi gli automi sono già in forma pulita e deterministica.

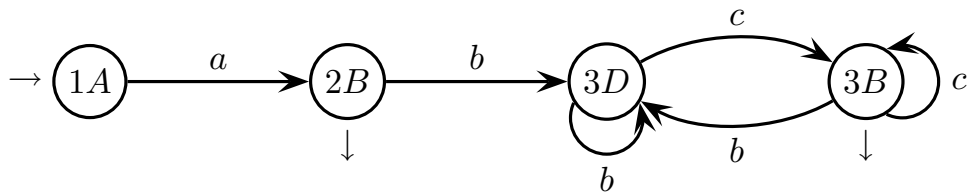
Se il secondo automa paresse un po' troppo complesso da tracciare intuitivamente, ecco come ottenerlo in modo sistematico mediante l'algoritmo di McNaughton-Yamada, esaminando i séguiti dei generatori dell'espressione:

$$\left(a_1 \mid b_2 \right) \left(a_3^* \, b_4^* \, c_5^+ \right)^* \dashv$$

inizî	a_1, b_2
generatori	séguiti
a_1	a_3, b_4, c_5, $\dashv$
b_2	a_3, b_4, c_5, $\dashv$
a_3	a_3, b_4, c_5
b_4	b_4, c_5
c_5	a_3, b_4, c_5, $\dashv$



Per inciso, l'automa così ottenuto coincide con la versione intuitiva. Esso è deterministico (per costruzione) e risulta minimo (come si verifica facilmente).
Ed ecco l'automa intersezione (costruzione del prodotto cartesiano), evitando da subito gli stati irraggiungibili e indefiniti:



L'automa prodotto è evidentemente deterministico e minimo, e genera l'espressione regolare R' di intersezione intuitiva data sopra; si lascia tale verifica al lettore (per esempio mediante il metodo di eliminazione dei nodi).

(b) L'automa ottenuto prima è già deterministico e pertanto non occorre fare altro.

2 Grammatiche libere e automi a pila 20%

1. Una lista x contiene un numero qualsiasi di elementi e separati dal carattere g, ossia è definita da $(e^+\, g)^*\, e^+$. Ecco un esempio:

$$x = \underbrace{e\ e\ e\ e}_{\text{gruppo } G_1}\ g\ \underbrace{e}_{\text{gruppo } G_2}\ g\ \underbrace{e\ e}_{\text{gruppo } G_3}\ g\ \underbrace{e\ e\ e}_{\text{gruppo } G_4}\ g\ \underbrace{e\ e\ e}_{\text{gruppo } G_5}$$

Si chiami "gruppo G_i" ($i \geq 1$) la i^{esima} sottolista di elementi e compresi tra due caratteri g successivi. Un gruppo non può essere vuoto. Come mostra l'esempio x, si considerano gruppi anche il prefisso G_1 e il suffisso G_5.

Una lista siffatta appartiene al linguaggio L se, e solo se, contiene due gruppi G_i e G_j ($1 \leq i < j$) tali che la lunghezza di G_i sia minore di o uguale a quella di G_j, cioè se e solo se $\exists\, i, j\ 1 \leq i < j\ \wedge\ |G_i| \leq |G_j|$. La stringa x data sopra è valida, perché ponendo per esempio $i = 2$ e $j = 4$ si ha che il gruppo G_2 è più corto di G_4, e tanto basta per validarla (ignorando gli altri gruppi). Invece la stringa y seguente:

$$y = \underbrace{e\ e\ e}_{\text{gruppo } G_1}\ g\ \underbrace{e\ e}_{\text{gruppo } G_2}\ g\ \underbrace{e}_{\text{gruppo } G_3}$$

non è valida e non appartiene al linguaggio L.

Si risponda ai punti seguenti:

(a) Si scriva una grammatica G in forma non estesa che generi il linguaggio L e si disegni l'albero sintattico della frase x di esempio data sopra.

(b) Si esamini se la grammatica G progettata sia ambigua oppure no.

Soluzione

(a) Al fine di soddisfare la condizione enunciata sopra, è evidente che le frasi di L devono constare di almeno due gruppi (non vuoti). Si dà prima la soluzione completa, poi la si giustifica in modo progressivo. Ecco la grammatica soluzione G completa (assioma S):

$$\begin{aligned}
S &\rightarrow U\ e\ M\ e\ V \\
M &\rightarrow e\ M\ e \mid M\ e \mid g\ U \quad (\text{oppure } V\ g) \\
U &\rightarrow U\ E\ g \mid \varepsilon \\
V &\rightarrow g\ E\ V \mid \varepsilon \\
E &\rightarrow e\ E \mid e
\end{aligned}$$

Si nota subito la presenza di una regola ricorsiva e autoinclusiva ($M \rightarrow e\ M\ e$), chiaramente deputata a produrre i due gruppi capaci di validare la stringa.

Ed ecco la giustificazione di come funziona la grammatica G proposta. Le sole regole seguenti, con assioma S:

$$\begin{aligned} S &\rightarrow e\ M\ e \\ M &\rightarrow e\ M\ e \mid M\ e \mid g \end{aligned}$$

generano due gruppi consecutivi (non vuoti) che soddisfano alla condizione $|G_i| \leq |G_j|$. La stringa valida più corta generata da S è $e\,g\,e$. Essa soddisfa alla condizione in modo minimale ponendo $i = 1$ e $j = 2$, avendosi allora $|G_1| = |G_2| = 1$. In generale così S produce le stringhe valide di tipo $e^h\,g\,e^k$, con $1 \leq h \leq k$.
Si considerino poi i due nonterminali U e V, espansi dalle regole seguenti (che sono lineari e dunque regolari):

$$\begin{aligned} U &\rightarrow U\ E\ g \mid \varepsilon \\ V &\rightarrow g\ E\ V \mid \varepsilon \\ E &\rightarrow e\ E \mid e \end{aligned}$$

I nonterminali U e V generano le stringhe di tipo $(\,e^+\,g\,)^*$ e $(\,g\,e^+\,)^*$, rispettivamente. In generale da sole esse non sono valide (lo possono essere in casi particolari).
Ora, mettendo U e V a sinistra e destra della regola assiomatica, rispettivamente, si concatenano ai due gruppi cruciali quanti altri gruppi (non vuoti) si vogliano:

$$S \rightarrow U\ e\ M\ e\ V$$

Pertanto così S genera stringhe del tipo $(\,e^+\,g\,)^*\ e^h\ g\ e^k\ (\,g\,e^+\,)^*$, con $1 \leq h \leq k$, che sono tutte valide.
Infine, aggiungendo la regola seguente (che genera $g\ (\,e^+\,g\,)^*$):

$$M \rightarrow g\ U$$

si inseriscono in mezzo ai due gruppi cruciali altri gruppi (non vuoti). Pertanto così S genera stringhe valide del tipo seguente:

$$\left(\,e^+\,g\,\right)^*\ e^h\ g\ \left(\,e^+\,g\,\right)^*\ e^k\ \left(\,g\,e^+\,\right)^*$$

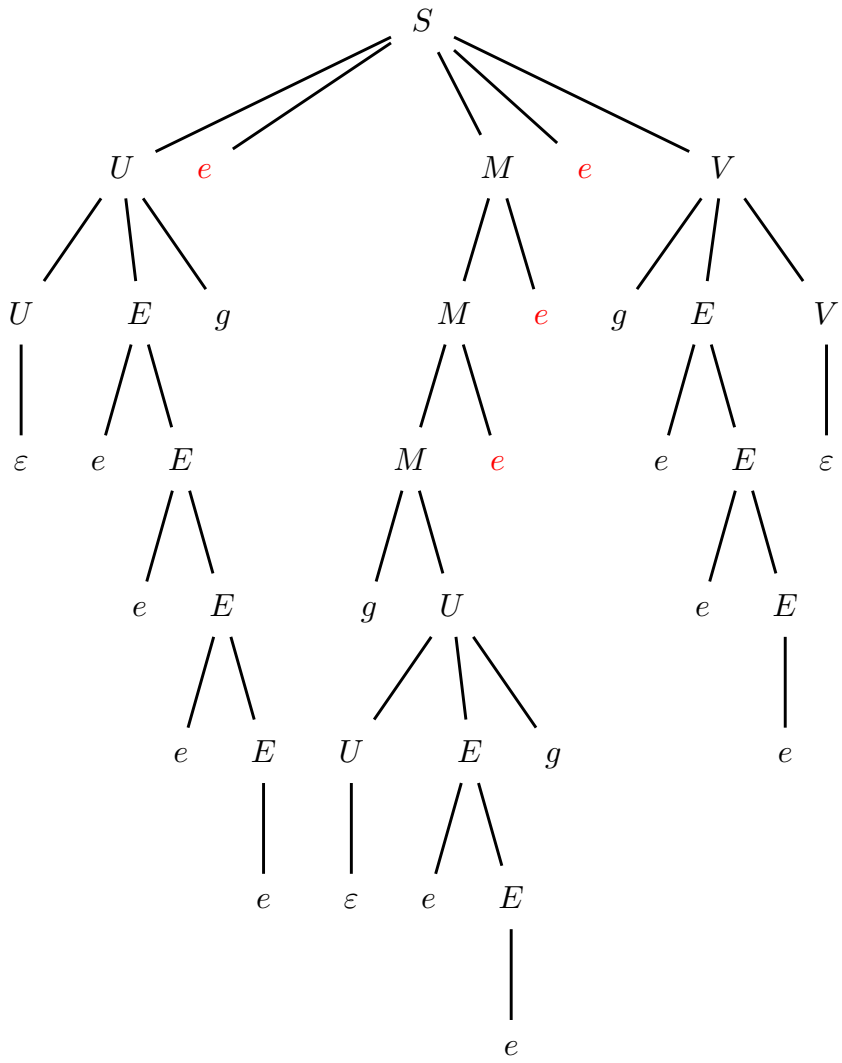
con $1 \leq h \leq k$.
La regola $M \rightarrow V\ g$ funzionerebbe bene quanto la regola $M \rightarrow g\,U$, ma sarebbe del tutto ridondante. È immediato rendersi conto che questa è proprio la forma generale del linguaggio L. Eccola presentata formalmente:

$$L = \left\{\ \left(\,e^+\,g\,\right)^*\ e^h\ g\ \left(\,e^+\,g\,\right)^*\ e^k\ \left(\,g\,e^+\,\right)^*\ \mid\quad 1 \leq h \leq k\ \right\}$$

Così il linguaggio L è finalmente completo. La coppia di esponenti h, k e la sottostringa $e^h\ g\ e^k$ ne rappresentano l'aspetto propriemente libero (non contestuale), il resto è una sorta di "ripieno" puramente regolare.

Il lettore riconsideri la giustificazione leggendola al contrario e avrà un metodo di costruzione progressiva della soluzione.

Ecco l'albero sintattico della stringa valida x di esempio:



I gruppi critici per la validità di x sono messi in evidenza in rosso. Si vede bene come i gruppi $e^h\, g \ldots e^k$ con esponenti $h \leq k$ vengano prodotti da regole con parti destre $e\ M\ e$ ed $M\ e$, la prima delle quali è necessariamente autoinclusiva.

(b) La grammatica G data prima è palesemente ambigua, giacché due gruppi G_i e G_j soddisfacenti alla condizione vengono certamente generati da M, ma altri gruppi, anch'essi soddisfacenti alla condizione e del tutto indistinguibili da quelli generati da M, potrebbero essere generati da U o V in modo puramente regolare. Per esempio la stringa $e\,g\,e\,g\,e$ è ambigua, dato che si può modellare in L come $e^h\, g\, e^k\, g\, e$ oppure $e\, g\, e^h\, g\, e^k$, con $h = k = 1$.

Ragionevolmente il linguaggio L stesso è inerentemente ambiguo, perché non si può impedire a una grammatica che generi obbligatoriamente i due gruppi G_i e G_j necessarî, di generarne in qualche momento anche altri, pure soddisfacenti la condizione e indistinguibili dai primi.

2. Si consideri un formato di documento di testo in stile simile a LaTeX, un po' semplificato. Tale formato ha la struttura così esemplificata:

   - Il documento è una sequenza (anche vuota) di ambienti, ciascuno dei quali è un elenco non numerato (in inglese si dice una "dotted list") o numerato ("numbered list") di elementi di testo (text_item).
   - I due tipi di ambiente si denotano nel modo seguente:

Elenco non numerato:	Elenco numerato:
`%begin_itemize`	`%begin_enumerate`
`%item <text_item>`	`%item <text_item>`
`...`	`...`
`%end_itemize`	`%end_enumerate`

   - Gli ambienti possono essere annidati (anche se di tipo diverso ossia dotted o numbered), a profondità di annidamento arbitraria.
   - Ogni ambiente contiene almeno un elemento di testo (l'elemento può essere un testo semplice senza sottoambienti o un sottoambiente annidato) o più di uno.
   - Il simbolo ⟨text_item⟩ sta a indicare o un testo semplice (senza sottoambienti), schematizzato dal puro terminale c (anche se è un testo di più caratteri), o un sottoambiente, cioè un elenco di tipo non numerato o numerato.

   Esempio di documento:

```
%begin_itemize

%item c

%item %begin_enumerate

%item c

%item c

%end_enumerate

%item c

%end_itemize

%begin_enumerate

...
```

   Si scriva una grammatica EBNF non ambigua che generi il formato descritto.

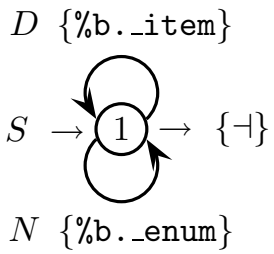
Soluzione

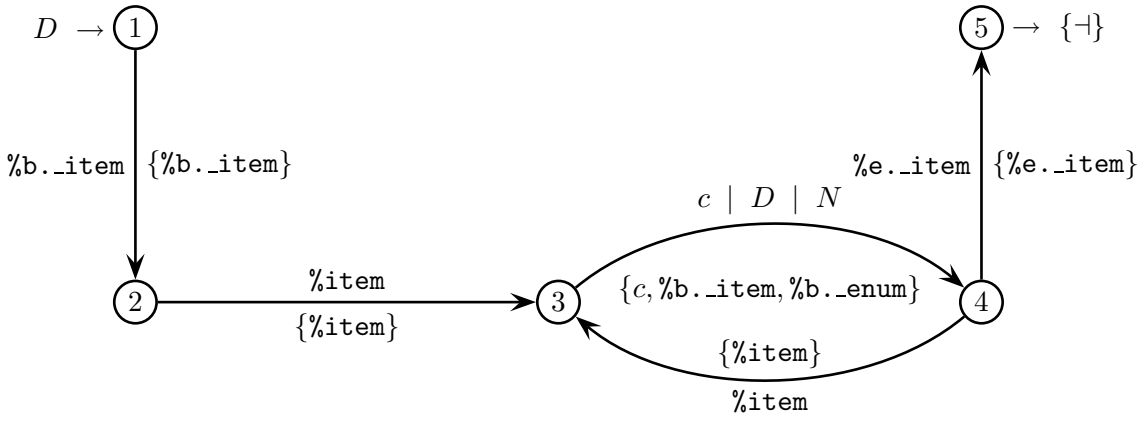
Si denotano con i nonterminali D e N gli ambienti "dotted list" e "numbered list", rispettivamente (abbreviando un po' le parole chiave per ragioni di spazio).

$$
\begin{aligned}
S &\rightarrow \; (\; D \; \mid \; N \;)^{*} \\
D &\rightarrow \; \texttt{``\%b.\_item.''} \; \big( \; \texttt{``\%item''} \; (\; c \; \mid \; D \; \mid \; N \;)^{+} \; \big)^{+} \; \texttt{``\%e.\_item.''} \\
N &\rightarrow \; \texttt{``\%b.\_enum.''} \; \big( \; \texttt{``\%item''} \; (\; c \; \mid \; D \; \mid \; N \;)^{+} \; \big)^{+} \; \texttt{``\%e.\_enum.''}
\end{aligned}
$$

La grammatica è forma EBNF e si verifica facilmente che è $LL(1)$, pertanto certamente non è ambigua.

Per la verifica $LL(1)$ basta per esempio mettere le regole in forma di automa e calcolare gli insiemi guida di profondità uno alle biforcazioni. Eccoli:

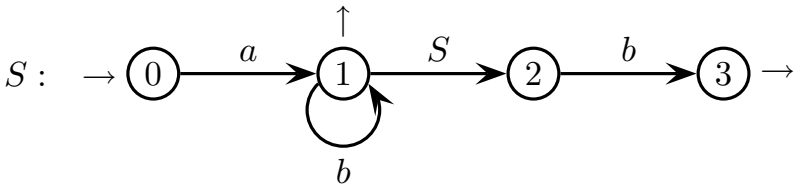




Si procede similmente per l'automa della regola che espande N (qui è omesso), strutturalmente analogo a quello di D. È evidente che vale la condizione $LL(1)$.

3 Analisi sintattica e parsificatori 20%

1. È data la grammatica G seguente, presentata come automa (assioma S):



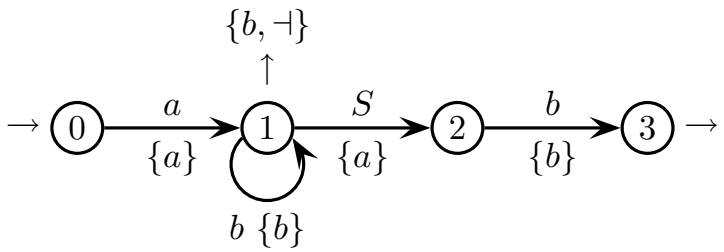
Si risponda ai punti seguenti:

(a) Si calcolino gli insiemi guida su tutti gli archi del grafo e si dica se la grammatica G sia $LL(k)$ per qualche $k \geq 1$.

(b) Si esamini la possibilità di trovare una grammatica equivalente a G con la proprietà $LL(1)$.

Soluzione

(a) Ecco gli insiemi guida:



Chiaramente la grammatica G non è $LL(1)$ allo stato 1. Non è difficile convincersi che G non è neppure $LL(k)$, per nessun $k > 1$, giacchè l'insieme guida con prospezione k sull'autoanello contiene la stringa b^k, come anche l'insieme guida sulla freccia di uscita allo stato 1.

(b) Il linguaggio generato da G contiene farsi del tipo seguente:

$$\underbrace{a\,b^*}_{1}\ \underbrace{a\,b^*}_{2}\ \cdots\ \underbrace{a\,b^*}_{n}\ \underbrace{a\,b^*}_{n+1}\ \underbrace{b}_{n}\ \cdots\ \underbrace{b}_{2}\ \underbrace{b}_{1} \qquad n \geq 0$$

Si tratta di una variante del linguaggio $a^n b^n$, ben noto per essere $LL(1)$. Tuttavia qui un automa a pila a discesa ricorsiva con prospezione di profondità fissata $k \geq 1$ non riuscirebbe mai, esaminando una sequenza di k caratteri b consecutivi nel nastro di ingresso, a distinguere se si tratti di una delle sottrostringhe b^*, le quali sono puramente regolari e non richiedono di usare la pila, oppure di parte del suffisso di n caratteri b finali, i quali bilanciano altrettanti caratteri a e chiaramente richiedono l'uso della pila.

2. Sono date la grammatica G seguente (assioma S):

$$
\begin{aligned}
S &\rightarrow a\ B\ S\ b \\
S &\rightarrow a\ B \\
B &\rightarrow b\ B \\
B &\rightarrow \varepsilon
\end{aligned}
$$

e la stringa di esempio seguente:

$$a\ b\ a\ b$$

Si risponda ai punti seguenti:

(a) Si esegua il riconoscimento della stringa di esempio con il metodo di Earley, compilando la tabella all'uopo predisposta a pagina seguente.

(b) Si disegni l'albero sintattico della stringa di esempio riconosciuta.

Soluzione

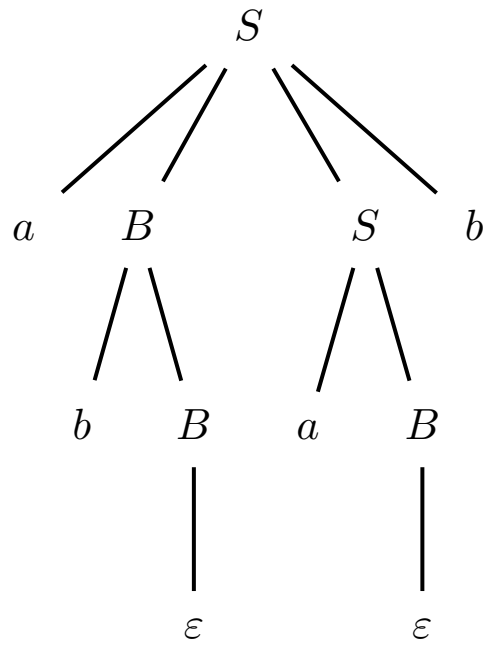
(a) Ecco la simulazione del riconoscimento di $a\,b\,a\,b$ con l'algoritmo di Earley:

Schema di simulazione dell'algoritmo di Earley

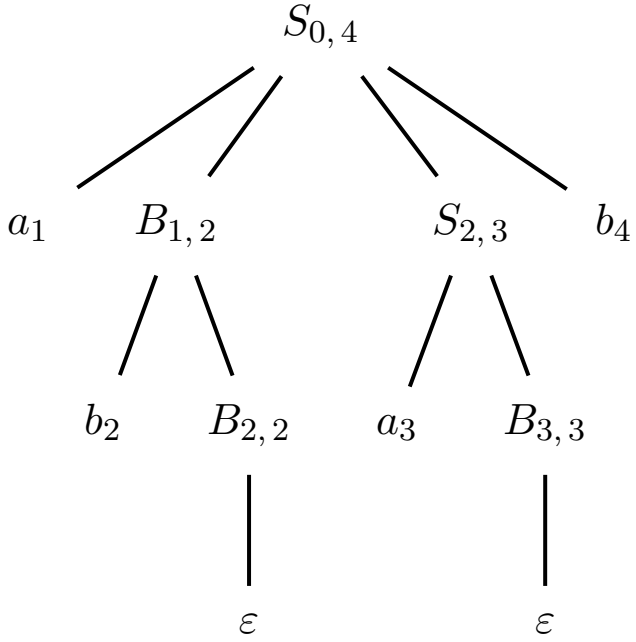
stato 0	pos. a	stato 1	pos. b	stato 2	pos. a	stato 3	pos. b	stato 4
$S \to \bullet\, a\, B\, S\, b$, 0 $S \to \bullet\, a\, B$, 0		$S \to a \bullet B S b, 0$ $S \to a \bullet B$, 0 $B \to \bullet\, b\, B$, 1 $B \to \bullet\, \varepsilon = \varepsilon \bullet$, 1 $S \to a\, B \bullet S\, b$, 0 $S \to a\, B \bullet$, 0 $S \to \bullet\, a\, B\, S\, b$, 1 $S \to \bullet\, a\, B$, 1		$B \to b \bullet B$, 1 $B \to \bullet\, b\, B$, 2 $B \to \bullet\, \varepsilon = \varepsilon \bullet$, 2 $B \to b\, B \bullet$, 1 $S \to a\, B \bullet S\, b$, 0 $S \to a\, B \bullet$, 0 $S \to \bullet\, a\, B\, S\, b$, 2 $S \to \bullet\, a\, B$, 2		$S \to a \bullet B\, S\, b$, 2 $S \to a \bullet B$, 2 $B \to \bullet\, b\, B$, 3 $B \to \bullet\, \varepsilon = \varepsilon \bullet$, 3 $S \to a\, B \bullet S\, b$, 2 $S \to a\, B \bullet$, 2 $S \to \bullet\, a\, B\, S\, b$, 3 $S \to \bullet\, a\, B$, 3 $S \to a\, B\, S \bullet b$, 0		$B \to b \bullet B$, 3 $S \to a\, B\, S\, b \bullet$, 0

Le candidate di riduzione che partecipano alla costruzione dell'albero sintattico (punto (b)) sono messe in evidenza in rosso.

(b) L'albero sintattico della stringa $a\,b\,a\,b$ di esempio è il seguente:



Mettendo in evidenza la numerazione riferita alla tabella di simulazione dell'algoritmo di Earley, si vede la relazione tra nodi dell'albero sintattico e candidate di riduzione (in rosso) presenti nella tabella.



Non sarebbe difficile verificare che la grammatica G è $LR(1)$ (naturalmente non è $LR(0)$ dato che contiene una regola nulla). Pertanto essa non è ambigua e dunque l'albero sintattico della stringa riconosciuta è certamente unico.

4 Traduzione e analisi semantica 20%

1. È dato un testo "italiano", costituito da parole separate da virgola "," o da spazio ⟨blank⟩, o da entrambi. La parola è schematizzata come stringa di caratteri c identici, e non è espansa ulteriormente. Il testo può contenere incisi racchiusi tra parentesi tonde "(" e ")". Gli incisi non sono annidati. La grammatica (sorgente) G_s qui sotto definisce il formato di testo così descritto (assioma S):

$$
\begin{aligned}
S &\rightarrow (\ P \mid I\ )\ (\ \text{“,”} \mid \text{“⟨blank⟩”} \mid \text{“, ⟨blank⟩”}\ )\ S \\
S &\rightarrow (\ P \mid I\ ) \\
P &\rightarrow c^{+} \\
I &\rightarrow \text{“(”}\ P\ (\ (\ \text{“,”} \mid \text{“⟨blank⟩”} \mid \text{“, ⟨blank⟩”}\ )\ P\ )^{*}\ \text{“)”}
\end{aligned}
$$

Si risponda ai punti seguenti:

(a) Si progetti uno schema di traduzione sintattica (ovvero una grammatica di traduzione), modificando nel modo opportuno la grammatica sorgente G_s, al fine di generare un testo identico a quello sorgente, tranne separare le une dalle altre con esattamente uno spazio le parole contenute negli incisi, in qualunque modo esse siano separate nel testo sorgente. Ecco un esempio di traduzione:

stringa sorgente:
ccc ⟨blank⟩ (*cc*, *cc*, ⟨blank⟩ *cccc* ⟨blank⟩ *cc*) *cc*, ⟨blank⟩ *ccc*

traduzione:
ccc ⟨blank⟩ (*cc* ⟨blank⟩ *cc* ⟨blank⟩ *cccc* ⟨blank⟩ *cc*) *cc*, ⟨blank⟩ *ccc*

(b) Si progetti un automa trasduttore per calcolare la traduzione così descritta (si rifletta su quale modello di trasduttore usare).

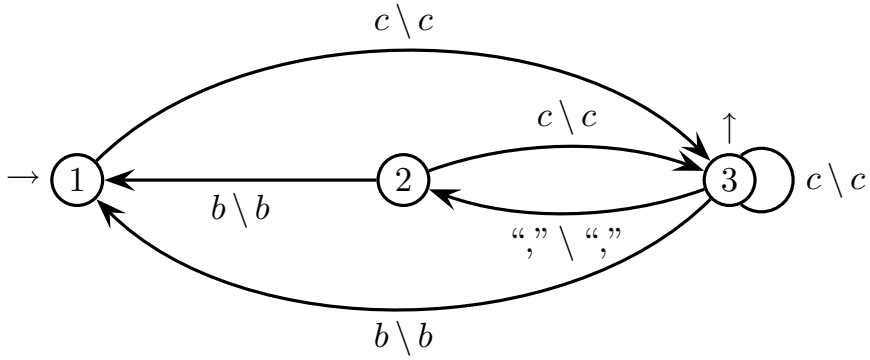
Soluzione

(a) La grammatica di traduzione è la seguente:

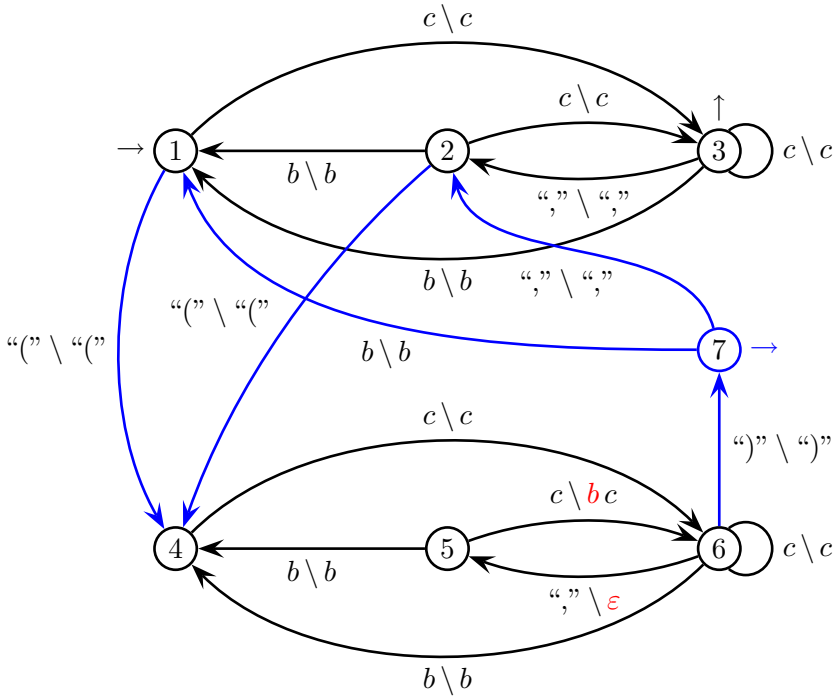
$$
\begin{aligned}
S &\rightarrow (\ P \mid I\ )\ \left(\ \frac{\text{“,”}}{\text{“,”}} \mid \frac{\text{“⟨blank⟩”}}{\text{“⟨blank⟩”}} \mid \frac{\text{“, ⟨blank⟩”}}{\text{“, ⟨blank⟩”}}\ \right)\ S \\
S &\rightarrow (\ P \mid I\ ) \\
P &\rightarrow \left(\frac{c}{c}\right)^{+} \\
I &\rightarrow \text{“(”}\ P\ \left(\ \left(\ \frac{\text{“,”}}{\text{“⟨blank⟩”}} \mid \frac{\text{“⟨blank⟩”}}{\text{“⟨blank⟩”}} \mid \frac{\text{“, ⟨blank⟩”}}{\text{“⟨blank⟩”}}\ \right)\ P\ \right)^{*}\ \text{“)”}
\end{aligned}
$$

Le modifiche strutturali apportate alla grammatica sorgente sono minime. Incidentalmente si noti che la grammatica non ha produzioni ricorsive, tranne una di tipo lineare a destra. Pertanto la traduzione è razionale e se ne potrebbe dare l'espressione regolare di traduzione, invece della grammatica.

(b) Palesemente la traduzione in questione è qualcosa di molto simile a una semplice traslitterazione, da effettuare però solo nel contesto dell'inciso: essa traslittera la virgola "," in spazio ⟨blank⟩, con la sola e lieve generalizzazione di tradurre la coppia ", ⟨blank⟩" in un solo spazio ⟨blank⟩, non in due.
Per calcolare tale traslitterazione basta un trasduttore a *stati finiti*. Esso deve tenere memoria se ci si trovi dentro un inciso o no. Fuori dall'inciso il testo tradotto è identico a quello sorgente, dentro invece si applica la (pseudo)traslitterazione descritta sopra. Si può organizzare la costruzione del trasduttore in due passi. Ecco prima l'automa traduttore delle sole frasi *senza incisi* (*b* sta per ⟨blank⟩):



Per aggiungere gli *incisi*, basta in un certo senso raddoppiare l'automa, collegare le due metà (in blu) e modificare la trasduzione dentro l'inciso (in rosso):



Così il trasduttore è completo e non necessita di spiegazioni particolari, salvo osservare che è deterministico e che sei stati bastano per tracciarlo; non si esclude peraltro ne possano bastare anche di meno, magari mettendolo in forma non deterministica. Naturalmente pure un automa a pila funzionerebbe (deterministico o no), ma sarebbe inutilmente potente per il semplice problema proposto.

2. Un testo (semplificato) in linguaggio naturale, composto da parole separate da uno spazio ⟨blank⟩, va impaginato in una finestra di terminale (rettangolare) larga $W \geq 1$ colonne, in modo da centrare ogni riga nelle W colonne. La parola è schematizzata semplicemente come stringa di caratteri c identici, senza espansione ulteriore.

   Si prenda come riferimento la grammatica con attributi data nel testo del corso a pp. 300, la quale calcola l'allineamento a sinistra del testo (non la centratura) e della quale qui sotto si riporta il solo supporto sintattico (assioma S):

$$
\begin{aligned}
S &\rightarrow T \\
T &\rightarrow T \ \langle \text{blank} \rangle \ T \\
T &\rightarrow V \\
V &\rightarrow c \, V \\
V &\rightarrow c
\end{aligned}
$$

   Si vuole modificare tale grammatica con attributi, aggiungendole se necessario nuovi attributi semantici e / o modificando quelli già noti, per calcolare (almeno) la colonna della lettera finale dell'ultima parola della riga corrente di testo, imponendo appunto che la riga sia centrata. Ecco un esempio (lo stesso del libro a pp. 300 ma centrato invece che allineato a sinistra): la frase "la torta ha gusto ma la grappa ha calore" è disposta come segue in una finestra di larghezza $W = 13$.

1	2	3	4	5	6	7	8	9	10	11	12	13
	l	a		t	o	r	t	a		h	a	
	g	u	s	t	o		m	a		l	a	
		g	r	a	p	p	a		h	a		
			c	a	l	o	r	e				

   La variabile (attributo) *ultimo* vale 3 per la, 9 per torta, 12 per ha, ..., e 9 per calore. L'eventuale spazio ⟨blank⟩ spurio va a destra (qui succede nell'ultima riga).

   Si risponda ai punti seguenti:

   (a) Si scriva la grammatica con attributi così descritta, dandone attributi e regole semantiche (si usino le tabelle all'uopo predisposte alle pagine successive).

   (b) Si verifichi se la grammatica così progettata soddisfi la condizione a una scansione (one-sweep) ed eventualmente la condizione L.

Soluzione

(a) Qui si opta per dare una soluzione ottenuta come variazione di quella presentata nel libro del corso come modello cui ispirarsi (benché non sia obbligatorio). Gli attributi da usare sono dunque adattamenti di quelli presentati nel libro, illustrati a pp. 300. Invece di considerare solo l'ultima colonna occupata a dx

dal testo, si considerano la colonna esterma di sx e quella di dx, giacché si deve centrare la riga di testo e non semplicemente allinearla a sinistra.
Pertanto gli attributi della nuova grammatica si specializzano come segue:

- l'attributo sx *ult* si divide in due attributi sx *sin* e *des*
- l'attributo dx *prec* si divide in due attributi dx *p_sin* e *p_des*
- l'attributo sx *lun* resta com'è

Ecco l'elenco dettagliato dei nuovi attributi, con spiegazione:

attributi da usare per la grammatica

tipo	nome	(non)terminali	dominio	significato
già dati nel testo dell'esercizio ed eventualmente da modificare e / o aggiungerne di nuovi				
sx	*lun*	*V*	num. int.	lunghezza della parola in numero di caratteri
sx	*sin*	*T*	num. int.	colonna di sx della riga centrata
sx	*des*	*T*	num. int.	colonna di dx della riga centrata
dx	*p_sin*	*T*	num. int.	colonna di sx della parte precedente di riga centrata
dx	*p_des*	*T*	num. int.	colonna di dx della parte precedente di riga centrata

A ogni nuova parola concatenata alla parte di testo già generata, la grammatica con attributi centra nuovamente la riga ricalcolando le colonne occupate dagli estremi sinistro e destro, e andando a capo sulla riga successiva quando serve. Tale comportamento ricorda da vicino quello dei comuni sistemi di videoscrittura (come Word e simili) quando si scrive testo con paragrafi in allineamento di tipo centrato: a ogni nuova parola aggiunta (anzi a ogni nuovo carattere introdotto da tastiera) la riga viene subito ricentrata a video, ricalcolando le colonne degli estremi sinistro e destro. Si può dunque immaginare un algoritmo di impaginazione strutturato come grammatica con attributi, il quale si attivi a ogni aggiunta di un carattere o una parola alla riga (esso viene presumibilmente agganciato al sistema e attivato automaticamente come una routine di interrupt associata alla pressione del tasto), ricostruendo (o aggiornando) l'albero sintattico del testo introdotto e ricalcolando gli attributi (e quando ciò avviene il sistema di grafica del calcolatore ridisegna immediatamente la riga a video).

Le funzioni semantiche vengono di conseguenza e si giustificano con lo stesso ragionamento esposto nel libro del corso a pp. 300, cui dunque si rimanda. Si presti attenzione al trattamento dei due attributi ereditati *p_sin* e *p_des*. La nuova soluzione è messa in parallelo con la vecchia, per vedere bene l'analogia.

sintassi	*funzioni semantiche nuove*	*vecchie*
$S_0 \rightarrow T_1$	$p\_sin_1 = 1$	
	$p\_des_1 = 0$	$prec_1 = -1$
$T_0 \rightarrow T_1$ ⟨blank⟩ T_2	$p\_sin_1 = p\_sin_0$	
	$p\_des_1 = p\_des_0$	$prec_1 = prec_0$
	$p\_sin_2 = sin_1$	
	$p\_des_2 = des_1$	$prec_2 = ult_1$
	$sin_0 = sin_2$	
	$des_0 = des_2$	$ult_0 = ult_2$
$T_0 \rightarrow V_1$	**if** $(p\_des_0 - p\_sin_0 + 1 + lun_1 \leq W)$ **then**	$prec_0 + 1 + lun_1 \leq W$
	$sin_0 = \left\lfloor \frac{(W-(p\_des_0-p\_sin_0+1+lun_1))}{2} \right\rfloor$	$ult_0 =$
	$des_0 = \left\lceil \frac{(W+(p\_des_0-p\_sin_0+1+lun_1))}{2} \right\rceil$	$prec_0 + 1 + lun_1$
	else	
	$sin_0 = \left\lfloor \frac{(W-lun_1)}{2} \right\rfloor$	
	$des_0 = \left\lceil \frac{(W+lun_1)}{2} \right\rceil$	$ult_0 = lun_1$
	endif	
$V_0 \rightarrow c\ V_1$	$lun_0 = 1 + lun_1$	
$V_0 \rightarrow c$	$lun_0 = 1$	

I simboli "$\lceil\ \rceil$" e "$\lfloor\ \rfloor$" (si chiamano parentesi di Gauss), applicati a un argomento non intero, danno l'approssimazione all'intero immediatamente superiore e inferiore, rispettivamente (per esempio si ha $\lceil 1,5 \rceil = 2$ e $\lfloor 1,5 \rfloor = 1$). Qui servono per lasciare a destra della riga l'eventuale spazio spurio.

(b) La grammatica con attributi progettata è a una sola scansione (one-sweep), come lo è la grammatica a pp. 300 del libro del corso la quale fa da modello per questa, e come del resto si verifica facilmente in modo diretto controllando la condizione one-sweep. Essa è anche di tipo L perché l'ordine di valutazione degli attributi procede da sinistra verso destra. Si noti comunque che il supporto sintattico è ambiguo (a causa della regola ricorsiva bilaterale $T \to T \perp T$) e che pertanto il valutatore semantico deve avere l'albero sintattico già costruito in qualche modo.

Per convincersi del buon funzionamento della nuova grammatica con attributi, si può riprendere l'albero sintattico decorato con attributi riportato nel libro del corso a pp. 300 e adattarlo ai nuovi attributi, seguendo il frammento di testo di esempio dato prima nell'esercizio. Si lascia il compito al lettore.

Beninteso ci possono essere soluzioni alternative a questa, strutturalmente diverse da quella presentata nel libro ma altrettanto valide.