# Linguaggi Formali e Compilatori
# (Formal Languages and Compilers)

## prof. S. Crespi Reghizzi, prof. Angelo Morzenti
## (prof. Luca Breveglieri)

## Prova scritta - 11 settembre 2008 - 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 sostenere 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), rispondendo alle domande obbligatorie.
- 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. Dato un alfabeto $\Sigma = \{a, b, c\}$, si consideri il linguaggio (regolare) $L \subseteq \Sigma^*$ contenente tutte e sole le stringhe che soddisfano le condizioni seguenti (si intende che ogni stringa di $L$ le soddisfa tutte):

   - il digramma $a\,b$ è vietato (nella stringa non ne deve comparire nessuno)
   - il digramma $c\,b$ è obbligatorio (nella stringa ne deve comparire almeno uno)
   - il carattere iniziale è $a$
   - il carattere finale è diverso da $b$

   Si risponda alle domande seguenti:

   (a) Si elenchino tutte le stringhe del linguaggio $L$:
   - di lunghezza minore o uguale a 4
   - di lunghezza uguale a 5 che finiscono con la lettera $a$

   (b) Si progetti, a scelta in modo (semi)intuitivo o algoritmico, un'espressione regolare $R$ che genera il linguaggio $L$.

   (c) (facoltativa) Se l'espressione regolare $R$ è ambigua, si ricavi un'espressione regolare $R'$ non ambigua equivalente a $R$.

---

## Soluzione

(a) È intuitivo come qualunque stringa del linguaggio $L$, che dunque soddisfa a tutte le condizioni date sopra, debba contenere almeno i tre fattori seguenti (nell'ordine):

$$\underbrace{a}_{\text{inizio}} \qquad \underbrace{c\,b}_{\text{digramma obbligatorio}} \qquad \underbrace{a}_{\text{fine (o } c\text{)}}$$

Questi tre fattori non sono sovrapponibili. Così è chiaro che non ci sono stringhe di lunghezza minore di 4. Le due sole stringhe di lunghezza uguale a 4 sono allora le seguenti:

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

Quelle di lunghezza uguale a 5 che finiscono con la lettera $a$, sono ottenibili dalla prima inserendo una lettera, come ammissibile (cioè evitando di formare il digramma $a\,b$), o tra l'inizio $a$ e il digramma $c\,b$, o tra il digramma $c\,b$ e la fine $a$. Eccole:

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

Ce ne sono altrettante di lunghezza 5, identiche a queste ma terminanti con $c$.

(b) Riflettendo sul punto precedente, in modo semi-intuitivo, si vede subito che si può modellare l'espressione regolare $R$ tramite la forma generale seguente:

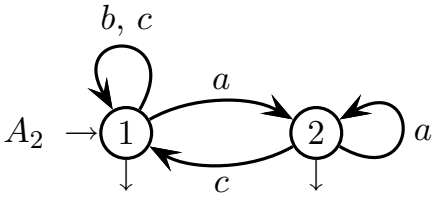
$$R = a\ R_1\ c\,b\ R_2\ (\ a\ \mid\ c\ )$$

dove $R_1$ e $R_2$ sono sottoespressioni tali che:

$R_1$ = fattori non inizianti con $b$ e non contenenti il digramma $a\,b$

$R_2$ = fattori non contenenti il digramma $a\,b$

Un facile automa (minimo) deterministico $A_2$ equivalente a $R_2$ è il seguente:




Chiaramente l'automa $A_2$ non riconosce il digramma $a\,b$ e accetta tutto il resto. Da $A_2$ si ricava intuitivamente l'espressione regolare $R_2$ equivalente:

$$R_2 = (\ a^+\ c\ \mid\ b\ \mid\ c\ )^*\ a^*$$

Senza ricorrere all'intuizione, basterebbe usare l'algoritmo di Berri-Seti, l'eliminazione dei nodi (Brozozowski) o le equazioni linguistiche. Ora è facile derivare intuitivamente l'espressione regolare $R_1$ da $R_2$, impedendo alla stringa di iniziare con la lettera $b$ (o modificando l'automa $A_2$), per esempio come segue:

$$R_1 = a^*\ \mid\ (\ a^+\ c\ \mid\ c\ )\ (\ a^+\ c\ \mid\ b\ \mid\ c\ )^*\ a^*$$

e infine sostituendo $R_1$ e $R_2$ (non è indispensabile farlo esplicitamente) si ottiene l'espressione regolare $R$. Eccola:

$$R = a\ \big(a^*\ \mid\ (a^+\ c\ \mid\ c)\ (a^+\ c\ \mid\ b\ \mid\ c)^*\ a^*\big)\ c\,b\ (\ a^+\ c\ \mid\ b\ \mid\ c)^*\ a^*\ (\ a\ \mid\ c\ )$$

In alternativa, si può provare a tracciare direttamente l'automa $A$ del linguaggio $L$ (deterministico o non) e poi ricavarne l'espressione regolare $R$ equivalente (ambigua o meno che risulti).

(c) L'espressione regolare $R$ data prima è ambigua. Per esempio, la stringa $a\,c\,b\,c\,b\,a$ seguente è generata in due modi differenti:

$$a\ \underbrace{c\,b}_{R_1}\ c\,b\ \underbrace{\varepsilon}_{R_2}\ a \qquad \text{oppure} \qquad a\ \underbrace{\varepsilon}_{R_1}\ c\,b\ \underbrace{c\,b}_{R_2}\ a$$

Si vede subito che il problema consiste (unicamente) nella generazione ambigua del secondo (o ulteriore) digramma $c\,b$, che può essere prodotto da $R_1$ o $R_2$. Per disambiguare l'espressione $R$ basta per esempio impedire alla sottoespressione $R_1$ di produrre il digramma $c\,b$:

$R_1$ = fattori non inizianti con $b$ e non contenenti i digrammi $a\,b$ e $c\,b$

È subito evidente che di fatto la nuova definizione di $R_1$ esclude del tutto la presenza della lettera $b$ (giacché $b$ non può comparire né all'inizio né a seguito di $a$ o $c$), mentre qualunque stringa di sole lettere $a$ e $c$ è ammissibile. Pertanto ora $R_1$ genera il linguaggio universale di alfabeto $\{a, c\}$:

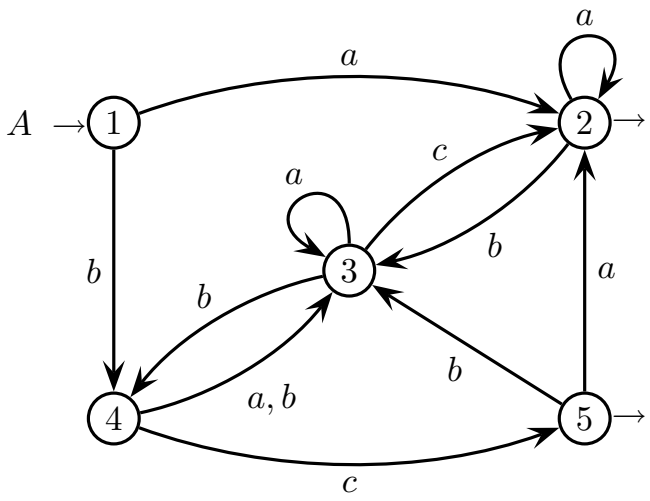
$$R_1 = (\ a\ \mid\ c\ )^*$$

Sostituendo, si ottiene l'espressione regolare $R'$ non ambigua che genera il linguaggio $L$. Eccola:

$$R' = a\ (\ a\ \mid\ c\ )^*\ c\, b\ (\ a^+\, c\ \mid\ b\ \mid\ c\ )^*\ a^*\ (\ a\ \mid\ c\ )$$

Una volta tanto, una forma non ambigua risulta più semplice di una ambigua.

2. È dato l'automa deterministico $A$ mostrato in figura, di alfabeto $\{a, b, c\}$.




Si risponda alle domande seguenti:

(a) Si minimizzi l'automa deterministico $A$ e si disegni il grafo stato-transizione dell'automa minimo $A'$.

(b) Si metta l'automa minimo $A'$ in forma di sistema di equazioni linguistiche lineari a destra, e lo si risolva ricavando un'espressione regolare $R$ equivalente ad $A'$.

---

## Soluzione

(a) Intanto si osservi che l'automa deterministico $A$ è in forma ridotta (o pulita): tutti gli stati sono utili (raggiungibili e definiti). Il grafo di $A$ è connesso ma non fortemente connesso (non si può tornare nello stato 1).

Per minimizzare l'automa $A$, si dà prima la tabella degli stati.

| | ingresso | | | |
|---|---|---|---|---|
| stato | $a$ | $b$ | $c$ | finale? |
| 1 | 2 | 4 | – | no |
| 2 | 2 | 3 | – | sì |
| 3 | 3 | 4 | 2 | no |
| 4 | 3 | 3 | 5 | no |
| 5 | 2 | 3 | – | sì |

Si può procedere in modo intuitivo. Si vede subito che le righe 2 e 5 della tabella sono identiche, pertanto gli stati 2 e 5 sono indistinguibili. La tabella si riduce come segue:

| | ingresso | | | |
|---|---|---|---|---|
| stato | $a$ | $b$ | $c$ | finale? |
| 1 | 2 | 4 | – | no |
| 2 | 2 | 3 | – | sì |
| 3 | 3 | 4 | 2 | no |
| 4 | 3 | 3 | 2 | no |

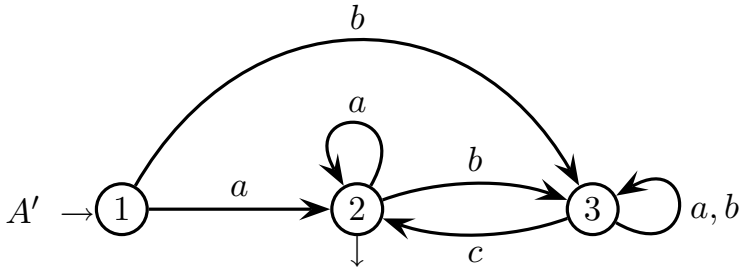
Ora si può osservare intuitivamente che gli stati 3 e 4 sono indistinguibili, in quanto presupponendo l'equivalenza $3 \sim 4$ le righe 3 e 4 della tabella risultano identiche, tranne che nella colonna $b$ dove però sussiste l'equivalenza supposta $3 \sim 4$. Pertanto la tabella si riduce ulteriormente come segue:

| | ingresso | | | |
|---|---|---|---|---|
| stato | $a$ | $b$ | $c$ | finale? |
| 1 | 2 | 3 | – | no |
| 2 | 2 | 3 | – | sì |
| 3 | 3 | 3 | 2 | no |

Questa tabella è in forma minima. Infatti lo stato finale 2 è distinguibile dagli stati non finali 1 e 3, e questi ultimi sono distinguibili nelle colonne $a$ e $c$ (dove 1 va nello stato di errore che nel grafo non è messo esplicitamente in evidenza).

In alternativa si sarebbe potuto procedere algoritmicamente tracciando la tabella triangolare delle implicazioni e risolvendola.

Ecco il grafo stato-transizione dell'automa minimo $A'$:




L'automa $A'$ è ovviamente in forma ridotta (pulita).

(b) Ecco il sistema di equazioni linguistiche lineari a destra equivalente all'automa deterministico minimo $A'$ (con variabili 1, 2 e 3):

$$\begin{cases} 1 &= a\,2 \mid b\,3 \\ 2 &= a\,2 \mid b\,3 \mid \varepsilon \\ 3 &= a\,3 \mid b\,3 \mid c\,2 \end{cases}$$

Il linguaggio $L$ è la (minima) soluzione per la variabile 1 (stato iniziale).

Per risolvere il sistema lineare e trovare l'espressione regolare $R$ che esprime la soluzione della variabile 1 (cioè che genera il linguaggio $L$), si applicano il metodo di sostituzione e l'identità di Arden per trattare la ricorsione, dove serve.
Equazione 3 (raccoglimento di 3 e Arden):

$$\begin{aligned} 3 &= a\,3 \mid b\,3 \mid c\,2 \\ &= (\,a \mid b\,)\;3 \mid c\,2 && \text{- raccoglimento} \\ &= (\,a \mid b\,)^*\;c\,2 && \text{- Arden} \end{aligned}$$

Equazione 2 (sostituzione di 3, raccoglimento di 2 e Arden):

$$\begin{aligned} 2 &= a\,2 \mid b\,3 \mid \varepsilon \\ &= a\,2 \mid b\;(\,a \mid b\,)^*\;c\,2 \mid \varepsilon && \text{- sostituzione} \\ &= (\,a \mid b\;(\,a \mid b\,)^*\;c\,)\;2 \mid \varepsilon && \text{- raccoglimento} \\ &= (\,a \mid b\;(\,a \mid b\,)^*\;c\,)^* && \text{- Arden} \end{aligned}$$

Equazione 3 (sostituzione di 2):

$$\begin{aligned} 3 &= (\,a \mid b\,)^*\;c\,2 \\ &= (\,a \mid b\,)^*\;c\;(\,a \mid b\;(\,a \mid b\,)^*\;c\,)^* && \text{- sostituzione} \end{aligned}$$

Equazione 1 (sostituzione di 2 e 3):

$$\begin{aligned} 1 &= a\,2 \mid b\,3 = \\ &= a\;(\,a \mid b\;(\,a \mid b\,)^*\;c\,)^* \\ &\qquad \mid b\;(\,a \mid b\,)^*\;c\;(\,a \mid b\;(\,a \mid b\,)^*\;c\,)^* && \text{- sostituzione} \end{aligned}$$

Si ha dunque l'espressione regolare $R$ seguente:

$$R = a\;(\,a \mid b\;(\,a \mid b\,)^*\;c\,)^* \mid b\;(\,a \mid b\,)^*\;c\;(\,a \mid b\;(\,a \mid b\,)^*\;c\,)^*$$

Si noti che l'espressione regolare $R$ è non ambigua, poiché è ricavata strutturalmente da un automa deterministico.
Beninteso non è detto che questa formulazione di $R$ sia la più breve o elegante possibile, nonostante sia ricavata partendo da un automa minimo. Per esempio, questa è migliore:

$$R = (\,a \mid b\;(\,a \mid b\,)^*\;c\,)^+$$

e si ricava facilmente dalla precedente per semplificazione algebrica (si lascia l'esercizio al lettore).

## 2 Grammatiche libere e automi a pila 20%

1. Dato l'alfabeto $\Sigma = \{a, c, f\}$, si consideri il linguaggio $L \subseteq \Sigma^*$ contenente certe espressioni in formato polacco postfisso, spiegate nel seguito. Una frase di $L$ è un'espressione avente l'operatore postfisso $f$, preceduto da $n \geq 1$ argomenti. Subito prima degli argomenti si trova la stringa $c^n$, che funge da contatore del numero di argomenti. Un argomento può essere atomico (scritto $a$) oppure una (sotto)espressione posfissa.

   Esempi di frase corretta (per aiutare il lettore, a destra con le graffe, che non fanno parte del linguaggio, sono messe in evidenza le eventuali sottoespressioni):

   | frase | sottoespressioni |
   |---|---|
   | $c\,a\,f$ | |
   | $c\,c\,a\,a\,f$ | |
   | $c\,c\,a\,f\,f$ | $c\ \underbrace{c\,a\,f}\ f$ |
   | $c\,c\,a\,c\,a\,f\,f$ | $c\,c\,a\ \underbrace{c\,a\,f}\ f$ |
   | $c\,c\,c\,a\,f\,c\,c\,a\,a\,f\,f$ | $c\,c\ \underbrace{c\,a\,f}\ \underbrace{c\,c\,a\,a\,f}\ f$ |

   Esempi di frase scorretta (in tutte manca una marca $c$):

   $a\,f$

   $c\,a\,a\,f$

   $c\,a\,f\,f$

   Si risponda alle domande seguenti:

   (a) Si progetti una grammatica $G$ non estesa (BNF) che genera il linguaggio $L$, e si verifichi se essa è ambigua.

   (b) Nel caso la grammatica $G$ precedente sia ambigua, si progetti una grammatica $G'$ non ambigua equivalente a $G$.

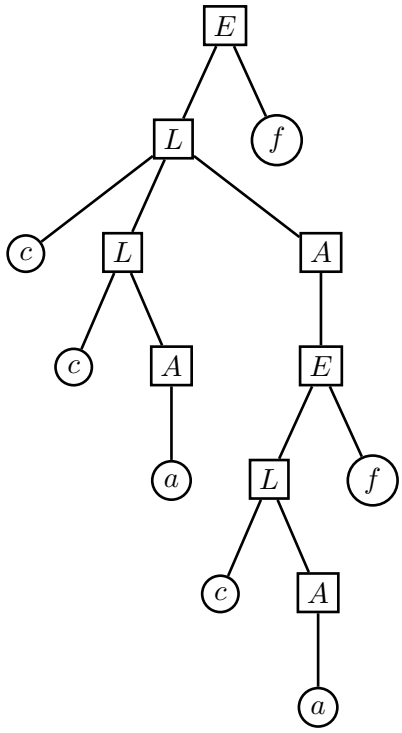
---

## Soluzione

(a) Una possibile soluzione intuitiva $G$ è la seguente (assioma $E$):

| regola | commento |
|---|---|
| $E \rightarrow L\ f$ | l'operatore postfisso $f$ è preceduto dal nonterminale $L$ che genera la lista (non vuota) di argomenti di $f$ |
| $L \rightarrow c\ L\ A$ | l'argomento $A$ (atomico $a$ o sottoespressione $E$) è preceduto dalla marca di conteggio $c$ e la regola ricorsiva autoinclusiva genera tanti argomenti $A$ quante marche $c$ |
| $L \rightarrow c\ A$ | l'argomento $A$ (idem come sopra) è preceduto dalla marca di conteggio $c$ e la regola non ricorsiva termina la lista (non vuota) di argomenti |
| $A \rightarrow a$ | l'argomento $A$ è atomico e si riduce al terminale $a$ |
| $A \rightarrow E$ | l'argomento $A$ è una sottoespressione (postfissa) generata ricorsivamente dall'assioma $E$ |

Come verifica di correttezza (peraltro non richiesta) ecco l'albero sintattico della stringa $c\,c\,a\,c\,a\,f\,f$, che contiene una sottoespressione:




In alternativa si potrebbe partire dalla grammatica classica delle espressioni con un solo operatore infisso binario, modificarla in modo che l'operatore sia $n$-ario con marche di conteggio $c$, e metterla in forma postfissa.

(b) La grammatica $G$ è non ambigua: infatti ogni operatore $f$ determina univocamente la lista dei propri argomenti grazie alla presenza del contatore $c^n$; l'ultimo argomento generato è il più interno.

Ecco un altro modo di presentare lo stesso ragionamento. Si supponga di volere parsificare le stringhe da destra verso sinistra: chiaramente l'algoritmo può operare deterministicamente perché i contatori $c$ permettono sempre di attribuire univocamente gli argomenti (atomici o sottoespressioni) agli operatori $f$, pertanto l'albero sintattico così prodotto è unico e la grammatica $G$ è non ambigua per definizione.

Un modo alternativo per constatare che la grammatica $G$ è non ambigua, consiste nel verificare che è deterministica di tipo $LR(0)$, tracciandone il grafo pilota (di tipo $LR(0)$; non è difficile e neppure lungo, e si lascia il compito al lettore (si noti che però $G$ non è $LL(k)$ per nessun $k \geq 1$).

2. È dato il linguaggio $L$ delle dichiarazioni di array e record specificato sotto:

   - la frase di $L$ è una dichiarazione di array a una dimensione oppure di record
   - la dimensione di array è un intero positivo, non nullo
   - il tipo di elemento di array può essere:
     - integer denotato dal terminale int
     - record o array denotato dal terminale record o array, rispettivamente
   - il campo di record ha etichetta denotata dal terminale id e può essere di tipo:
     - integer (denotato dal terminale int)
     - record o array (denotato dal terminale record o array)

     e i campi di record sono terminati da ";" (punto e virgola), tranne l'ultimo
   - se la dichiarazione di array ha integer come tipo di elemento, può (ma non necessariamente deve) essere inizializzata con una lista di numeri denotati dal terminale num; negli altri casi l'inizializzazione non è consentita
   - le convenzioni lessicali sono illustrate negli esempi seguenti:

     prima frase

     ```
     id record
        id int ;
        id array [1] int = ( num ) ;
        id array [2] int = ( num, num ) ;
        id array [3] int
     end
     ```

     seconda frase

     ```
     id array [30] record
        id int ;
        id record
           id int ;
           id int
        end ;
        id int
     end
     ```

   Si risponda alle domande seguenti:

   (a) Si progetti una grammatica non ambigua, usando dove conviene regole estese (Extended BNF).

   (b) (facoltativa) Si dica se ci sono vincoli di correttezza delle dichiarazioni che non sono tenuti in conto nella grammatica progettata.

---

## Soluzione

(a) Una soluzione è la grammatica estesa seguente (assioma DECL):

$$
\begin{aligned}
\langle \text{DECL} \rangle &\rightarrow \text{id } \langle \text{S\_TYPE} \rangle \\
\langle \text{S\_TYPE} \rangle &\rightarrow \langle \text{RECORD} \rangle \mid \langle \text{ARRAY} \rangle \\
\langle \text{FIELD} \rangle &\rightarrow \text{int} \mid \langle \text{S\_TYPE} \rangle \\
\langle \text{RECORD} \rangle &\rightarrow \text{record } \langle \text{FIELD} \rangle \ (\ \text{';'}\ \langle \text{FIELD} \rangle\ )^*\ \text{end} \\
\langle \text{ARRAY} \rangle &\rightarrow \text{array '['}\ \langle \text{DIM} \rangle\ \text{']'} \Big(\ \text{int}\ \big[\ \text{'='}\ \langle \text{I\_LIST} \rangle\ \big]\ \mid\ \langle \text{S\_TYPE} \rangle\ \Big) \\
\langle \text{I\_LIST} \rangle &\rightarrow \text{'('}\ \text{num}\ (\ \text{','}\ \text{num}\ )^*\ \text{')'} \\
\langle \text{DIM} \rangle &\rightarrow [\text{'1'} - \text{'9'}]\ [\text{'0'} - \text{'9'}]^*
\end{aligned}
$$

Le parentesi quadre usate come metacaratteri indicano opzionalità oppure intervallo di simboli terminali. I nomi delle classi sintattiche (nonterminali) sono autoesplicativi della semantica rispettiva.

La grammatica proposta è non ambigua, perché formata da semplici strutture non ambigue (come liste, ecc). È anche facile constatare che essa è di tipo $LL(1)$ (in senso esteso), dunque a maggior ragione non ambigua.

(b) La grammatica proposta non impone che la lunghezza della lista degli interi che inizializza l'array sia uguale alla dimensione dell'array stesso; del resto ciò non si può fare con una grammatica libera. Oltre a ciò, essa presenta le usuali debolezze delle grammatiche libere usate per definire la sintassi linguaggi di programmazione con variabili: non evita la duplicazione di identificatori, la dichiarazione ripetuta di tipo, ecc.

## 3 Analisi sintattica e parsificatori 20%

1. È data la grammatica $G$ seguente, non estesa (BNF) (assioma $S$):

$$G \begin{cases} S & \rightarrow & d\ S\ b\ b \\ S & \rightarrow & d\ S\ b \\ S & \rightarrow & d \end{cases}$$

Si risponda alle domande seguenti, motivando brevemente la risposta:

(a) Si costruisca l'automa pilota $LR(0)$ della grammatica $G$, e si dica se $G$ è $LR(0)$.

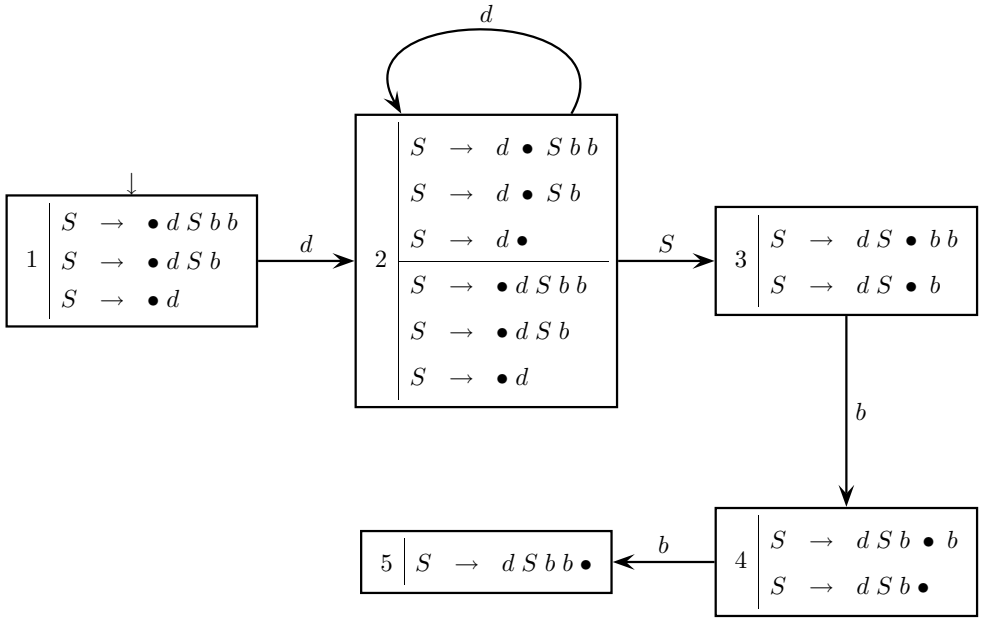
(b) Si stabilisca se la grammatica $G$ è $LL(1)$.

(c) (facoltativa) Se $G$ non è $LL(1)$, si stabilisca se è $LL(k)$ per qualche $k \geq 2$.

---

### Soluzione

(a) Ecco il grafo pilota di tipo $LR(0)$ della grammatica $G$:

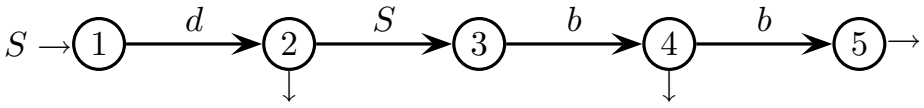



Il grafo pilota $LR(0)$ ha 5 macrostati. Si vede subito che i macrostati 2 e 4 sono inadeguati, in quanto presentano entrambi conflitto di tipo riduzione-spostamento. Pertanto la grammatica $G$ non è di tipo $LR(0)$.

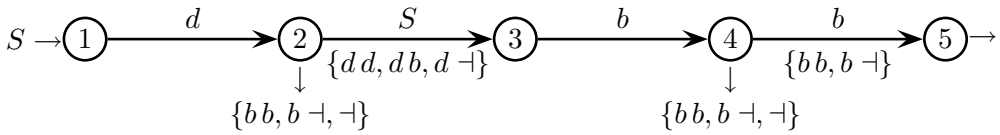
Volendo, si potrebbe prima mettere la grammatica $G$ in forma di rete di automi, e poi tracciarne il grafo pilota di tipo $LR(0)$. Qui per modellare $G$ basta un solo automa e il grafo pilota risulterebbe piuttosto semplice. Ovviamente si arriverebbe alla stessa conclusione di prima, ossia che $G$ non è di tipo $LR(0)$.

(b) È immediato osservare come le tre regole alternative che espandono l'assioma $S$ abbiano insiemi guida identici per $k = 1$, cioè $\{d\}$, senza bisogno di porre $G$ in forma di rete di automi e di studiarla in modo sistematico. Pertanto la grammatica $G$ non è di tipo $LL(1)$.

(c) Per effettuare l'analisi $LL(k)$ con $k \geq 2$, conviene porre la grammatica $G$ in forma di rete di automi. Eccola:



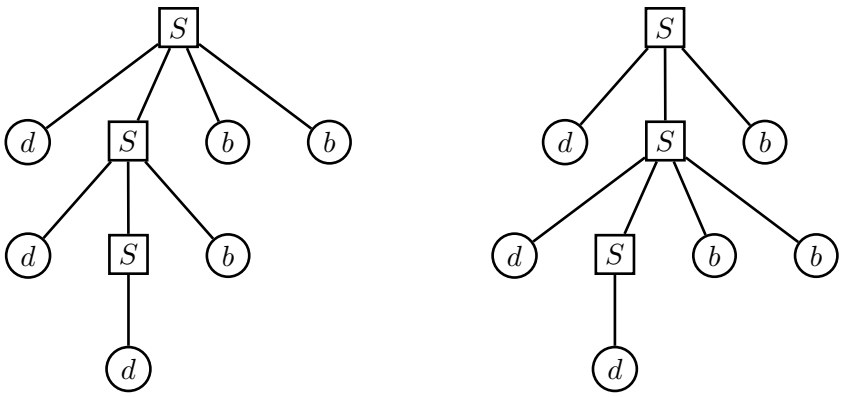

Analisi $LL(2)$ (solo dove serve):




Le biforcazioni sono solo agli stati 2 e 4. Allo stato 4 gli insiemi guida non sono disgiunti (hanno in comune la stringa $b\,b$), pertanto la grammatica $G$ non è di tipo $LL(2)$.

Per l'analisi $LL(k)$ con $k \geq 3$, non è difficile vedere che allo stato 4 gli insiemi guida hanno sempre in comune l'elemento $b^k$, pertanto la grammatica $G$ non è $LL(k)$ per nessun $k \geq 3$ (e dunque da prima per nessun $k \geq 1$).

In alternativa, e molto più rapidamente, si può osservare che la grammatica $G$ è ambigua: le regole alternative $S \to d\ S\ b$ e $S \to d\ S\ b\ b$ sono commutative. Si consideri la stringa $d\,d\,d\,b\,b\,b$, che ha i due alberi sintattici seguenti:




Tale stringa è dunque ambigua di grado 2, e in generale stringhe più lunghe sono via via maggiormente ambigue. Il motivo è sempre lo stesso: le due regole al-

ternative indicate prima sono commutative, essendo indifferente applicare prima l'una o l'altra.

Dato che la grammatica $G$ è ambigua, non è $LL(k)$ per nessun $k \geq 1$. Si noti che $G$ non è neppure $LR(k)$ per nessun $k \geq 1$, e in particolare dunque non è né $LR(0)$ né $LR(1)$.

Volendo precisare (non è affatto richiesto dal testo dell'esercizio), il linguaggio $L$ generato dalla grammatica $G$ è il seguente:

$$L = \left\{ d^h\ d\ b^k \mid \quad h \geq 0\ \wedge\ h \leq k \leq 2h \right\}$$

Ora si consideri una stringa generica $d^h\ d\ b^k$ ($h \geq 0$ e $h \leq k \leq 2h$). È evidente che ogni derivazione di tale stringa contiene esattamente $2h-k$ regole $S \to d\ S\ b$ e $k-h$ regole $S \to d\ S\ b\ b$, liberamente commutabili, perché i numeri $\#d$ e $\#b$ di lettere $d$ (escludendo la $d$ centrale) e $b$ risultano effettivamente:

$$\begin{aligned} \#d &= (2h-k) + (k-h) = 2h - k + k - h = h \\ \#b &= (2h-k) + 2\,(k-h) = 2h - k + 2k - 2h = k \end{aligned}$$

Inoltre la derivazione contiene una sola regola $S \to d$ finale, obbligatoria per generare la $d$ centrale. La derivazione consta pertanto di $(2h-k)+(k-h)+1 = h+1$ passi, dei quali i primi $h$ sono liberamente commutabili e corrispondono alle due regole indicate (se si preferisce, gli alberi sintattici hanno profondità $h+1$, e i primi $h$ livelli sono liberamente interscambiabili come si vede dai due alberi di esempio dati prima). Il numero di commutazioni possibili è $\binom{h}{2h-k} = \binom{h}{k-h}$: si tratta della nota formula di analisi combinatoria[1], qui istanziata in modo da dare il numero di possibili liste di lunghezza $h$ con elementi di due tipi (ossia le due regole dette prima), dove i due tipi di elemento compaiono nella lista esattamente $2h-k$ e $k-h$ volte ciascuno. Pertanto il grado di ambiguità della stringa generica $d^h\ d\ b^k$ vale $\binom{h}{2h-k}$ o indifferentemente $\binom{h}{k-h}$.

La stringa proposta prima, $d\,d\,d\,b\,b\,b$ ossia $d^h\ d\ b^k$ con $h,k = 2,3$, ha grado di ambiguità $\binom{h}{2h-k} = \binom{2}{2\times 2-3} = \binom{2}{1} = 2$, come si vede dagli alberi. Le stringhe più corte $d$ ($h,k = 0,0$), $d\,d\,b$ ($h,k = 1,1$), $d\,d\,b\,b$ ($h,k = 1,2$) e $d\,d\,d\,b\,b$ ($h,k = 2,2$) hanno grado di ambiguità $\binom{0}{0} = 1$, $\binom{1}{1} = 1$, $\binom{1}{0} = 1$ e $\binom{2}{2} = 1$, rispettivamente, e pertanto non sono ambigue. Dunque quella proposta è la più corta stringa ambigua generata dalla grammatica $G$.

Dato che il coefficiente binomiale $\binom{h}{2h-k}$ cresce arbitrariamente (per esempio si ha $\lim\limits_{h\to\infty} \binom{h}{2h-k}\Big|_{k=2h-1} = \lim\limits_{h\to\infty} \binom{h}{1} = \lim\limits_{h\to\infty} h = \infty$), la grammatica $G$ ha grado di ambiguità illimitato (però non infinito perché $G$ non genera stringhe con infiniti alberi sintattici). Ecco un caso particolare: la stringa $d^{10}\,d\,b^{15}$ ($h,k = 10,15$) ha grado di ambiguità $\binom{h}{2h-k}\Big|_{h,k=10,15} = \binom{10}{20-15} = \binom{10}{5} = \frac{10\cdot 9\cdot 8\cdot 7\cdot 6}{5\cdot 4\cdot 3\cdot 2\cdot 1} = \frac{36240}{120} = 252$.

Ciò non significa che il linguaggio $L$ sia inerentemente ambiguo, e in effetti è facile trovare una grammatica non ambigua equivalente a $G$; si lascia l'esercizio al lettore. Suggerimento: rendere non commutative le due regole responsabili dell'ambiguità, imponendo loro un ordine prestabilito di applicazione.

[1] Ecco la formula generale: il numero di liste di lunghezza $a \geq 1$ con elementi di due tipi, che compaiono $b$ e $c$ volte ciascuno ($b,c \geq 0$ e $b+c = a$), vale $\binom{a}{b} = \binom{a}{c}$; per simmetria i due coefficienti binomiali coincidono.

2. È data la grammatica $G$ seguente, non estesa (BNF) (assioma $S$):

$$G \begin{cases} S & \rightarrow & d\ S\ b\ b \\ S & \rightarrow & d\ S\ b \\ S & \rightarrow & \varepsilon \end{cases}$$

Si risponda alla domanda seguente. Si svolga l'analisi di Earley per la stringa:

$$d\,d\,b\,b\,b$$

e si disegnino gli alberi sintattici di tale stringa, mostrandone la corrispondenza con i passi dell'algoritmo di Earley. Per l'analisi si usi la tabella predisposta di seguito.

---

| **Schema di simulazione dell'algoritmo di Earley** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| stato 0 | pos. $d$ | stato 1 | pos. $d$ | stato 2 | pos. $b$ | stato 3 | pos. $b$ | stato 4 | pos. $b$ | stato 5 | |
| | | | | | | | | | | | |

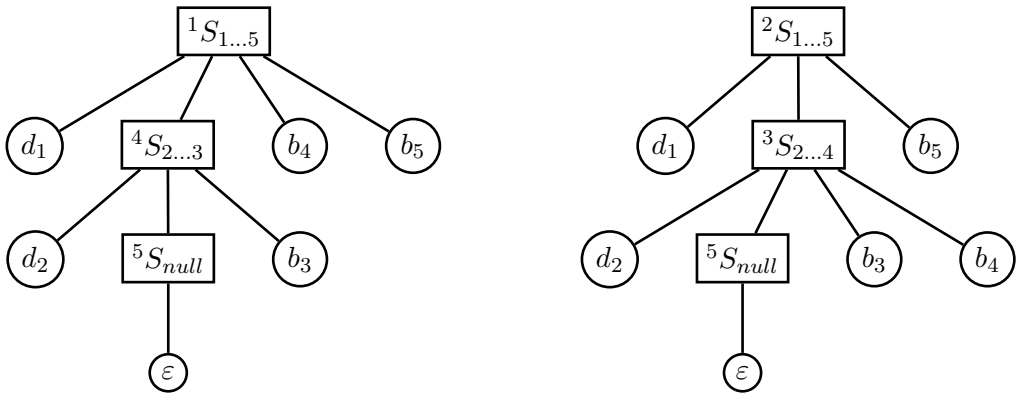
## Soluzione

**Schema di simulazione dell'algoritmo di Earley**

| stato 0 | pos. $d$ | stato 1 | pos. $d$ | stato 2 | pos. $b$ | stato 3 | pos. $b$ | stato 4 | pos. $b$ | stato 5 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $S \rightarrow \bullet\, d\, S\, b\, b$ | 0 | $S \rightarrow d \bullet S\, b\, b$ | 0 | $S \rightarrow d \bullet S\, b\, b$ | 1 | $S \rightarrow d\, S\, b \bullet b$ | 1 | ${}^{3}S \rightarrow d\, S\, b\, b \bullet$ | 1 | ${}^{1}S \rightarrow d\, S\, b\, b \bullet$ | 0 |
| $S \rightarrow \bullet\, d\, S\, b$ | 0 | $S \rightarrow d \bullet S\, b$ | 0 | $S \rightarrow d \bullet S\, b$ | 1 | ${}^{4}S \rightarrow d\, S\, b \bullet$ | 1 | $S \rightarrow d\, S\, b \bullet b$ | 0 | $S \rightarrow d\, S\, b \bullet b$ | 0 |
| $S \rightarrow \bullet\, \varepsilon = \varepsilon \bullet$ | 0 | $S \rightarrow \bullet\, d\, S\, b\, b$ | 1 | $S \rightarrow \bullet\, d\, S\, b\, b$ | 2 | $S \rightarrow d\, S \bullet b\, b$ | 0 | $S \rightarrow d\, S\, b \bullet$ | 0 | ${}^{2}S \rightarrow d\, S\, b \bullet$ | 0 |
| | | $S \rightarrow \bullet\, d\, S\, b$ | 1 | $S \rightarrow \bullet\, d\, S\, b$ | 2 | $S \rightarrow d\, S \bullet b$ | 0 | $S \rightarrow d\, S \bullet b\, b$ | 0 | | |
| | | $S \rightarrow \bullet\, \varepsilon = \varepsilon \bullet$ | 1 | ${}^{5}S \rightarrow \bullet\, \varepsilon = \varepsilon \bullet$ | 2 | | | $S \rightarrow d\, S \bullet b$ | 0 | | |
| | | $S \rightarrow d\, S \bullet b\, b$ | 0 | $S \rightarrow d\, S \bullet b\, b$ | 1 | | | | | | |
| | | $S \rightarrow d\, S \bullet b$ | 0 | $S \rightarrow d\, S \bullet b$ | 1 | | | | | | |

Lo stato 5 della tabella contiene due candidate di riduzione con puntatore allo stato iniziale 0.
Pertanto la stringa è riconosciuta, e ci sono almeno due (o più) alberi sintattici
(per sapere esattamente quanti bisogna effettuare la ricostruzione).

Ecco gli alberi sintattici riscostruiti, sono esattamente due (non ce ne sono altri):




L'apice a sinistra del nonterminale indica la corrispondenza tra nodo e candidata di riduzione nella tabella. La coppia di pedici a destra del nonterminale (o la scritta *null*) dà le posizioni della lettera iniziale e finale del fattore generato dal nodo. Il pedice a destra del terminale dà la posizione della lettera nella stringa (e ovviamente nessuna indicazione nel caso di $\varepsilon$).

Chiaramente la stringa $d\,d\,b\,b\,b$ ha grado di ambiguità 2, e pertanto la grammatica $G$ è ambigua. Si noti che il motivo di fondo dell'ambiguità è dato dalla commutabilità delle regole alternative $S \to d\ S\ b$ e $S \to d\ S\ b\ b$. Si riveda l'esecizio precedente (che ha una grammatica molto simile) per approfondimenti al riguardo.

# 4 Traduzione e analisi semantica 20%

1. Dato l'alfabeto $\{a, b, c\}$, si considerino i linguaggi sorgente $L_S$ e destinazione $L_D$ seguenti (il numero 0 è pari):

$$\begin{aligned} L_S &= \left\{ a^r\, b^s\, c^t \mid \quad r, s, t \geq 0 \ \wedge \ \Big( \, (\, r \,\text{pari} \, \wedge \, t \,\text{disp.}\,) \ \vee \ (\, r \,\text{disp.} \, \wedge \, t \,\text{pari}\,) \, \Big) \right\} \\ L_D &= \{\, a^r \mid \quad r \geq 0 \ \wedge \ r \,\text{pari} \,\} \ \cup \ \{\, b^s \mid \quad s \geq 0 \,\} \end{aligned}$$

   Si definisce la traduzione sintattica $\tau \colon L_S \to L_D$ nel modo seguente:

$$\tau\left(a^r\, b^s\, c^t\right) \mapsto \begin{cases} a^r & \text{se } r \text{ è pari e } t \text{ è dispari} \\ b^s & \text{se } r \text{ è dispari e } t \text{ è pari} \end{cases}$$

   Si risponda alle domande seguenti:

   (a) Si argomenti se la traduzione sintattica $\tau$, calcolata tramite un automa trasduttore a stati finiti, sia deterministica o inerentemente indeterministica.

   (b) Si tracci un automa trasduttore a stati finiti $T$ che realizza la trasduzione $\tau$, deterministico o meno secondo quanto concluso al punto precedente.

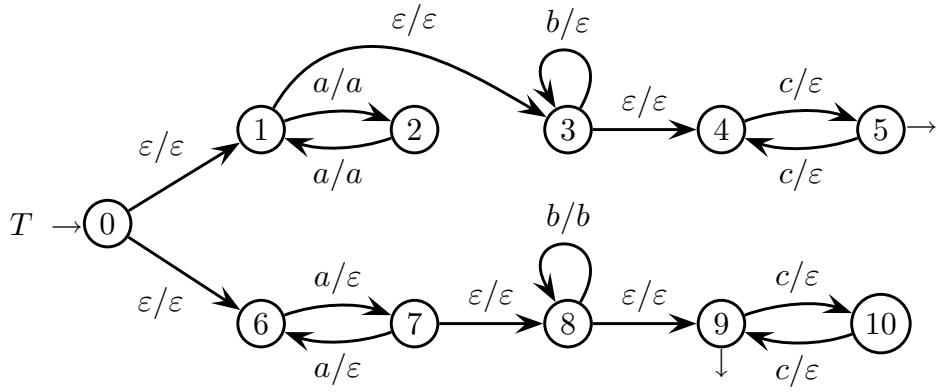
   (c) (facoltativa) Si descriva a parole un automa a pila deterministico $P$, con riconoscimento a stato finale, che realizza la trasduzione $\tau$.

---

## Soluzione

(a) La trasduzione $\tau$, se calcolata a stati finiti, è inerentemente indeterministica: per decidere se emettere il fattore di ingresso $a^r$ piuttosto che $b^s$ (entrambi di lunghezza arbitraria), occorre conoscere la parità del fattore successivo $c^t$, il che non è fattibile deterministicamente solo con gli stati finiti perché la quantità di memoria disponibile è limitata.

(b) L'automa trasduttore a stati finiti $T$ di $\tau$ deve essere indeterministico: occorre fare una scelta iniziale indeterministica su quali siano le parità dei fattori di ingresso $a^r$ e $c^t$. Il trasduttore $T$ si traccia senza difficoltà. Eccolo (con 11 stati):



Alcuni aspetti di indeterminismo di $T$ sarebbero eliminabili (per esempio alcune transizioni spontanee), a prezzo di qualche complicazione strutturale del grafo; tuttavia in ogni caso $T$ non può essere completamente deterministico; tanto vale dunque mettere $T$ nella forma più leggibile possibile sfruttando aspetti indeterministici, come qui per esempio l'uso di numerose transizioni spontanee.

(c) Per rendere deterministica la trasduzione, l'idea di fondo è usare la pila per memorizzare in via precauzionale informazione la cui necessità effettiva non è chiara al momento, per in seguito sfruttarla, se servirà, oppure scartarla, se risulterà inutile. Nel caso specifico l'informazione da impilare precauzionalmente è il fattore di ingresso $a^r$, giacché non si sa se andrà emesso o scartato.

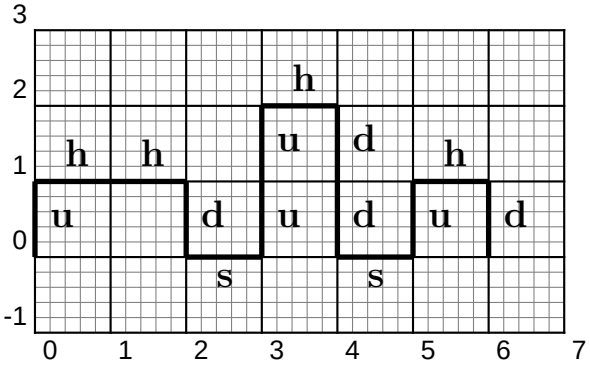
L'automa a pila deterministico $P$ che realizza la trasduzione $\tau$ può dunque operare nel modo seguente (con riconoscimento a stato finale):

- legge il fattore $a^r$, per precauzione lo impila codificando il carattere $a$ con il simbolo di pila $A$, e usa gli stati finiti per valutare la parità di $r$
- se $r$ è pari:
  - legge il fattore $b^s$ senza né toccare la pila né emettere nulla
  - legge il fattore $c^t$ senza né toccare la pila né emettere nulla, usando gli stati finiti per valutare la parità di $t$
  - se $t$ è dispari svuota la pila e per ogni simbolo $A$ emette $a$, e termina riconoscendo (va nello stato finale)
  - altrimenti termina senza riconoscere
- se $r$ è dispari:
  - svuota la pila (eliminando tutti i simboli $A$) senza né leggere né emettere nulla
  - legge il fattore $b^s$ e lo impila, codificando il carattere $b$ con il simbolo di pila $B$
  - legge il fattore $c^t$ senza né toccare la pila né emettere nulla, usando gli stati finiti per valutare la parità di $t$
  - se $t$ è pari svuota la pila e per ogni simbolo $B$ emette $b$, e termina riconoscendo (va nello stato finale)
  - altrimenti termina senza riconoscere

L'automa a pila $P$ così descritto è certamente deterministico, in quanto le sue mosse sono decise univocamente dal contenuto dell'ingresso (nonché dal contenuto della pila), senza bisogno di fare scelte indeterministiche.

Non sarebbe difficile tracciare il grafo stato-transizione di questo automa a pila deterministico, sebbene sia un po' lungo (ma il grafo ha struttura piuttosto modulare); si lascia l'esercizio al lettore.

2. Una forma d'onda costituita da una sequenza d'impulsi che iniziano e tornano all'asse delle ascisse, è mostrata in figura:




La forma d'onda in figura contiene tre impulsi ed è codificata dalla stringa di alfabeto { $u\,(= up)$, $d\,(= down)$, $h\,(= horizontal)$, $s\,(= separator)$ } seguente:

$$u\,h\,h\,d\;\;s\;\;u\,u\,h\,d\,d\;\;s\;\;u\,h\,d$$

Inoltre l'area della forma d'onda ha il valore seguente:

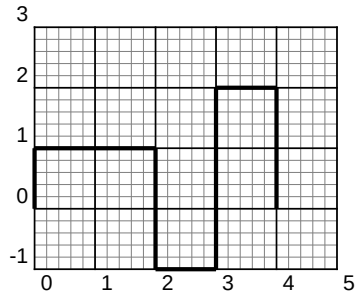
$$\text{area} = 2 \times 1 + 1 \times 2 + 1 \times 1 = 5$$

Il problema richiede la costruzione di una grammatica $G$ con attributi per definire tale famiglia di forme d'onda.

La sintassi di riferimento $G$ è la seguente, e non deve essere modificata (assioma $S$):

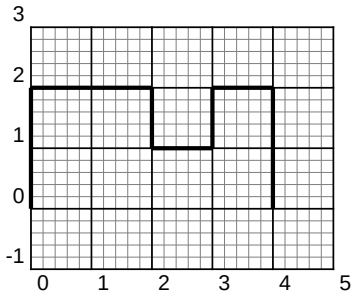
$$\begin{array}{lcl@{\qquad\qquad}lcl}
S & \rightarrow & W & & & \\
W & \rightarrow & U\ H\ D & H & \rightarrow & h\ H \\
W & \rightarrow & W\ s\ U\ H\ D & H & \rightarrow & h \\
U & \rightarrow & u\ U & D & \rightarrow & d\ D \\
U & \rightarrow & u & D & \rightarrow & d
\end{array}$$

Molte forme d'onda generate da $G$ non sono valide, come per esempio le seguenti:




$u\,h\,h\,d\,d\;s\;u\,u\,u\,h\,d\,d$

impulso che scende sotto l'asse delle ascisse




$u\,u\,h\,h\,d\;s\;u\,h\,d\,d$

impulso che non torna all'asse delle ascisse

Si risponda alle domande seguenti:

(a) Si completi la sintassi data con le funzioni semantiche e gli attributi necessari per calcolare l'attributo booleano $well\_formed$, abbreviato $wf$, che risulta vero se, e solo se, la stringa data rappresenta una forma d'onda corretta. Si scrivano le funzioni semantiche nello spazio apposito. Si precisi quali attributi sono sinistri e quali destri.

(b) Si spieghi se la grammatica con attributi progettata è di tipo a una passata (one-sweep) o addirittura di tipo L, ed eventualmente se l'analizzatore sintattico è integrabile con quello semantico.

(c) (facoltativa) Si scrivano altre funzioni semantiche per calcolare l'attributo $\alpha$ che esprime l'area compresa tra la forma d'onda e l'asse delle ascisse (si veda la prima figura), se la curva è corretta (altrimenti $\alpha$ assume un valore di errore). Si scrivano le funzioni semantiche nello spazio apposito. Si precisi quali attributi sono sinistri e quali destri.

Nota bene: oltre a definire eventuali nuovi attributi, quelli già dati prima sono disponibili e usabili, senza bisogno di riscriverne le funzioni semantiche.

---

attributi da usare per la grammatica

| tipo | nome | (non)terminali | dominio | significato |
|---|---|---|---|---|
| già dati nel testo dell'esercizio | | | | |
| | $wf$ | $S$ | booleano | onda ben formata |
| | $\alpha$ | $S$ | intero | area sottesa dall'onda |
| eventualmente da estendere e / o aggiungerne di nuovi | | | | |
| | | | | |

| *sintassi* | *funzioni semantiche* (domanda (a)) |
|---|---|
| $S_0 \rightarrow W_1$ | |
| $W_0 \rightarrow U_1\ H_2\ D_3$ | |
| $W_0 \rightarrow W_1\ s\ U_2\ H_3\ D_4$ | |
| $U_0 \rightarrow u\ U_1$ | |
| $U_0 \rightarrow u$ | |
| $H_0 \rightarrow h\ H_1$ | |
| $H_0 \rightarrow h$ | |
| $D_0 \rightarrow d\ D_1$ | |
| $D_0 \rightarrow d$ | |

| *sintassi* | *funzioni semantiche* (domanda (c)) |
|---|---|
| $S_0 \rightarrow W_1$ | |
| $W_0 \rightarrow U_1\ H_2\ D_3$ | |
| $W_0 \rightarrow W_1\ s\ U_2\ H_3\ D_4$ | |
| $U_0 \rightarrow u\ U_1$ | |
| $U_0 \rightarrow u$ | |
| $H_0 \rightarrow h\ H_1$ | |
| $H_0 \rightarrow h$ | |
| $D_0 \rightarrow d\ D_1$ | |
| $D_0 \rightarrow d$ | |

## Soluzione

(a) Ecco l'elenco degli attributi per questa domanda:

attributi da usare per la grammatica

| tipo | nome | (non)terminali | dominio | significato |
|---|---|---|---|---|
| già dati nel testo dell'esercizio | | | | |
| | $wf$ | $S$ | booleano | onda ben formata |
| eventualmente da estendere e / o aggiungerne di nuovi | | | | |
| sx | $wf$ | $S, W$ | booleano | onda ben formata |
| sx | $\upsilon$ | $U$ | intero | altezza del fronte di salita dell'impulso |
| sx | $\delta$ | $D$ | intero | altezza del fronte di discesa dell'impulso |

La soluzione proposta usa gli attributi interi $\upsilon$ e $\delta$, di tipo sinistro, associati ai nonterminali $U$ e $D$, rispettivamente, per calcolare le altezze dei fronti di salita e discesa dell'impulso, rispettivamente, ovvero come contatori dei segmenti *up* e *down*. L'attributo booleano $wf$ è il predicato di correttezza, pure sinistro, associato ai nonterminali $W$ e $S$.

L'idea è di calcolare le altezze dei fronti di salita e di discesa dell'impulso, e poi di confrontarle propagando l'esito del confronto fino alla radice dell'albero sintattico.

| regola | | | funzione semantica | | |
|---|---|---|---|---|---|
| $S_0$ | $\rightarrow$ | $W_1$ | $wf_0$ | $\leftarrow$ | $wf_1$ |
| $W_0$ | $\rightarrow$ | $U_1\ H_2\ D_3$ | $wf_0$ | $\leftarrow$ | $(\ \upsilon_1 = \delta_3\ )$ |
| $W_0$ | $\rightarrow$ | $W_1\ s\ U_2\ H_3\ D_4$ | $wf_0$ | $\leftarrow$ | $wf_1\ \wedge\ (\ \upsilon_2 = \delta_4\ )$ |
| $U_0$ | $\rightarrow$ | $u\ U_1$ | $\upsilon_0$ | $\leftarrow$ | $\upsilon_1 + 1$ |
| $U_0$ | $\rightarrow$ | $u$ | $\upsilon_0$ | $\leftarrow$ | $1$ |
| $H_0$ | $\rightarrow$ | $h\ H_1$ | | | |
| $H_0$ | $\rightarrow$ | $h$ | | | |
| $D_0$ | $\rightarrow$ | $d\ D_1$ | $\delta_0$ | $\leftarrow$ | $\delta_1 + 1$ |
| $D_0$ | $\rightarrow$ | $d$ | $\delta_0$ | $\leftarrow$ | $1$ |

Volendo si potrebbero riunire gli attributi $\upsilon$ e $\delta$ in uno solo, usato in regole diverse per calcolare le due altezze di fronte. Comunque averne due con nomi differenziati aumenta la leggibilità delle funzioni semantiche.

(b) La grammatica con attributi completa è di tipo puramente sintetizzato, pertanto è certamente di tipo a una passata (one-sweep). Il supporto sintattico è ricorsivo a sinistra (terza regola) e dunque non è di tipo $LL(k)$, per nessun $k \geq 1$; pertanto la grammatica con attributi completa non è di tipo L e certamente non è integrabile con l'analizzatore sintatico $LL$ (che del resto non esiste).

(c) Ecco l'elenco degli attributi per questa domanda (ovvero l'elenco complessivo in quanto direttamente o indirettamente tutti gli attributi vengono usati):

attributi da usare per la grammatica

| tipo | nome | (non)terminali | dominio | significato |
|---|---|---|---|---|
| già dati nel testo dell'esercizio | | | | |
| | $wf$ | $S$ | booleano | onda ben formata |
| | $\alpha$ | $S$ | intero | area sottesa dall'onda |
| eventualmente da estendere e / o aggiungerne di nuovi | | | | |
| sx | $wf$ | $S, W$ | booleano | onda ben formata |
| sx | $\alpha$ | $S, W$ | intero | area sottesa dall'onda |
| sx | $\upsilon$ | $U$ | intero | altezza del fronte di salita dell'impulso |
| sx | $\delta$ | $D$ | intero | altezza del fronte di discesa dell'impulso |
| sx | $\eta$ | $H$ | intero | larghezza dell'impulso |

La soluzione proposta usa gli attributi interi $\upsilon$ e $\eta$, sinistri, associati ai nonterminali $U$ e $H$, rispettivamente, e l'attributo intero $\alpha$ (area), pure sinistro, associato ai nonterminali $W$ e $S$. Le funzioni semantiche calcolano le aree degli impulsi successivi (altezza $\times$ larghezza) e le addizionano; il valore negativo $-1$ indica errore (impulso malformato); non appena compare un impulso malformato (ciò si rileva tramite l'attributo $wf$), si genera errore e lo si propaga alla radice dell'albero sintattico.

| regola | | | funzione semantica | | |
|---|---|---|---|---|---|
| $S_0$ | $\rightarrow$ | $W_1$ | $\alpha_0$ | $\leftarrow$ | $\alpha_1$ |
| $W_0$ | $\rightarrow$ | $U_1\ H_2\ D_3$ | $\alpha_0$ | $\leftarrow$ | **if** $wf_0$ **then** $\upsilon_1 \times \eta_2$ **else** $-1$ |
| $W_0$ | $\rightarrow$ | $W_1\ s\ U_2\ H_3\ D_4$ | $\alpha_0$ | $\leftarrow$ | **if** $wf_0$ **then** $\alpha_1 + \upsilon_2 \times \eta_3$ **else** $\alpha_1$ |
| $U_0$ | $\rightarrow$ | $u\ U_1$ | | | |
| $U_0$ | $\rightarrow$ | $u$ | | | |
| $H_0$ | $\rightarrow$ | $h\ H_1$ | $\eta_0$ | $\leftarrow$ | $\eta_1 + 1$ |
| $H_0$ | $\rightarrow$ | $h$ | $\eta_0$ | $\leftarrow$ | $1$ |
| $D_0$ | $\rightarrow$ | $d\ D_1$ | | | |
| $D_0$ | $\rightarrow$ | $d$ | | | |

Si noti che l'attributo sinistro $\alpha$ dipende dall'attributo sinistro $wf$ dello stesso nodo padre (come è ammesso per attributi sinistri).

Si sottintende che ci siano anche gli attributi $\upsilon$ e $\delta$, e le funzioni semantiche direttamente o indirettamente necessarie per calcolare il predicato $well\_formed$ (qui non sono riportate).