Werkcollege Compilerconstructie
Vrijdag 27 september 2019

1. Beschouw de context-vrije grammatica G' met startvariabele S en de volgende producties:

$$\begin{aligned} S &\rightarrow \textbf{if } B\ S \mid \textbf{id} = \textbf{id} \\ B &\rightarrow EB' \\ B' &\rightarrow \textbf{boolop } EB' \mid \epsilon \\ E &\rightarrow \textbf{id } E' \\ E' &\rightarrow \textbf{relop id} \mid \epsilon \end{aligned}$$

   In G' zijn S, B, B', E, E' dus de variabelen en $\textbf{if}, \textbf{id}, =, \textbf{boolop}, \textbf{relop}$ de terminalen. G' is het resultaat van opgave 5a) uit werkcollege 3,

   (a) Bepaal voor elke variabele in de grammatica G' zowel de FIRST- als de FOLLOW-verzameling.

   (b) Construeer de LR(0)-automaat bij grammatica G'.

   (c) Construeer de SLR *parsing table* bij grammatica G'.

   (d) Is G' een SLR grammatica? Motiveer je antwoord.

   (e) Geef (ad hoc) een afleidingsboom (*parse tree*) in G voor de string **if id id = id** (bijvoorbeeld overeenkomend met de instructie `if OK x=y`).

   (f) Parse de string **if id id = id** met de SLR parsing table van onderdeel (c). Laat bij iedere stap duidelijk zien wat je doet, bijvoorbeeld met behulp van een tabel van de volgende vorm:

States on stack	Corresponding Symbols on stack	Input	Action
...	...	...	...

2. Beschouw de context-vrije grammatica G met startvariabele S en de volgende producties:

$$\begin{aligned} S &\rightarrow aTa \\ T &\rightarrow bSb \mid ba \end{aligned}$$

   (a) Construeer de LR(0)-automaat bij grammatica G.

   (b) Stel dat je geen toegevoegde productie $S' \rightarrow S$ zou hebben, maar dat je gewoon zou accepteren als je een \$ in de invoer ziet, terwijl je in een toestand met item $S \rightarrow aTa \cdot$ bent (zie de eerste slides over 'LR(0) Automaton (Introduction)' uit het hoorcollege).

      i. Hoe zou de automaat er dan uitzien?

      ii. Ga na wat er met deze automaat zou gebeuren bij de invoer *ababaa*.

3. Verzin een voorbeeld van een *ondubbelzinnige* context-vrije grammatica G waarbij in de SLR *parsing table* reduce/reduce conflicten ontstaan.

4. Bij het opbouwen van een SLR *parsing table* is een van de regels:

   > Als $[A \to \alpha \cdot]$ in I_i zit en $a \in \text{FOLLOW}(A)$, voeg dan de actie 'reduce naar $A \to \alpha$' toe aan ACTION$[i, a]$.

   Verzin een voorbeeld van een context-vrije grammatica G, zódat

   - de LR(0)-automaat een itemset I_i bevat met daarin een item $[A \to \alpha \cdot]$,
   - en er een terminal $a \in \text{FOLLOW}(A)$ is,
   - terwijl je door de reductie naar $A \to \alpha$ (en eventueel daarna nog andere reducties) uit te voeren, toch niet in een itemset I_j kunt komen vanwaaruit er een transitie met een a is.

5. Beschouw de context-vrije grammatica G met startvariabele S en de volgende producties:

   (1) $S \to S - A$
   (2) $S \to A$
   (3) $A \to AB$
   (4) $A \to B$
   (5) $B \to B+$
   (6) $B \to p$
   (7) $B \to q$

   (a) Geef (ad hoc) een afleidingsboom in G voor de string $q + p$.

   (b) Gegeven is dat de SLR parsing table bij grammatica G er als volgt uit ziet:

State	Action					Goto		
	$-$	$+$	p	q	\$	S	A	B
0			s6	s7		1	9	8
1	s2				acc			
2			s6	s7			3	8
3	r1		s6	s7	r1			4
4	r3	s5	r3	r3	r3			
5	r5	r5	r5	r5	r5			
6	r6	r6	r6	r6	r6			
7	r7	r7	r7	r7	r7			
8	r4	s5	r4	r4	r4			
9	r2		s6	s7	r2			4

   Parse de string $q+p$ met deze tabel. Laat bij iedere stap duidelijk zien wat je doet, bijvoorbeeld met behulp van een tabel van de volgende vorm:

States on stack	Corresponding Symbols on stack	Input	Action
...	...	...	...