FACULTEIT INGENIEURSWETENSCHAPPEN
EN ARCHITECTUUR

Compilers - Voorbeeldexamen

Bert De Saffel

Master in de Industriële Wetenschappen: Informatica Academiejaar 2018–2019

Gecompileerd op 6 juni 2019

UNIVERSITEIT
GENT

Inhoudsopgave

Hoofdstuk 1

Vragen

1.1 Compilerfasen - I

Gegeven de opeenvolgende fasen van een compiler, geef aan de programmavoorstellingen, grafen of datastructuren die aan het *eind* van elke fase geproduceerd worden. Geef uw antwoord in de vorm: a-7, b-3, ...

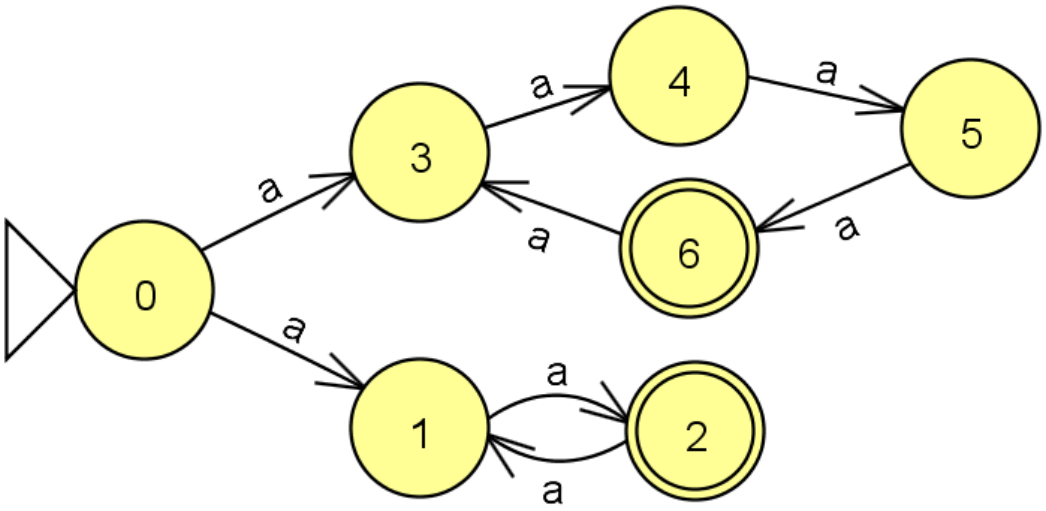
Compilerfase	Programmavoorstelling
(a) scanning (lexicale analyse)	1. intermediaire boomtaal
(b) parsing (syntactische analyse)	2. control flow graaf (CFG)
(c) type checking (semantische analyse)	3. frame layout, activation records
(d) vertaling (translate)	4. symbooltabellen
(e) canonicalisering	5. interferentiegraaf
(f) instructieselectie	6. gekleurde interference graph
(g) controlestroom analyse	7. tokens
(h) dataflow analyse (liveness)	8. assembler code
(i) register allocatie	9. assembler instructies
(j) code emission	10. abstracte syntax tree (AST)

Antwoord

a-7, b-10, c-?, d-?, e-?, f-9, g-?, h-6, i-?, j-8

1.2 Lexicale analyse

Volgende nondeterministische eindige automaat herkent geldige tokens `(aa)+` en `(aaa)+`.



1. Beschrijf de manier om in het algemeen verschillende NFA's samen te voegen tot 1 DFA. Bespreek daarbij de definitie van *sluiting*, *DFAedge* en het *algoritme* dat de NFA-toestanden samenvoegt tot toestanden in een DFA.
2. Pas dit toe op dit voorbeeld: de herkenning van twee- en drievouden van `a` en teken de bekomen DFA.
3. Zijn er in de gevonden DFA nog equivalente toestanden die men kan vereenvoudigen?

Antwoord

1. Een deterministische eindige automaat (DFA) is een eindige automaat waarbij elke transitie uniek is. Het algoritme om een NFA om te vormen naar een DFA maakt enerzijds gebruik van sluitingen. De sluiting T van een verzameling van toestanden S, of `closure(S)`, bevat alle toestanden die kunnen bereikt worden voor de lege transitie ϵ voor elke staat in S. Dit kan wiskundig gedefinieerd worden als:

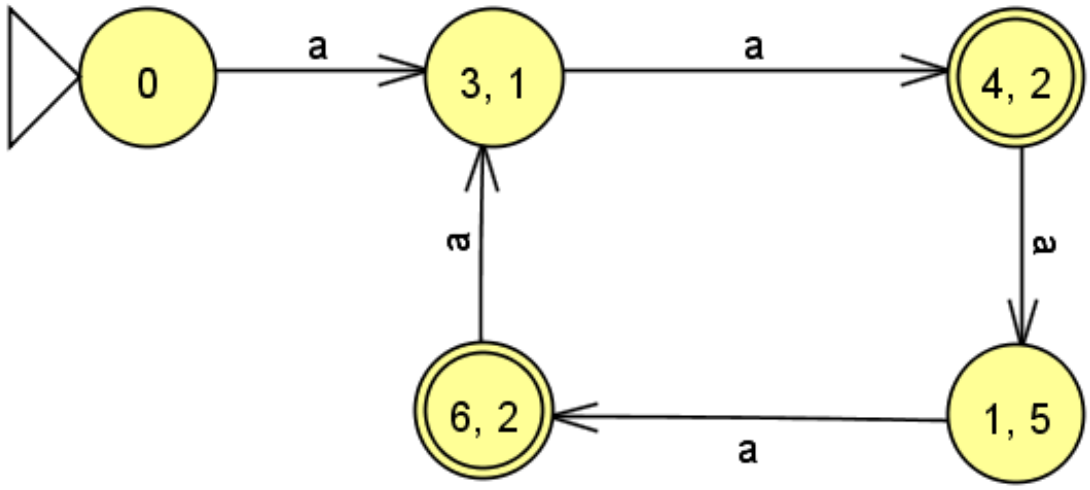
   $$T = S \cup \left(\bigcup_{s \in T} edge(s, \epsilon) \right)$$

   waarbij `edge(s, c)` de staten geeft die vanuit s via symbool c kan bereikt worden.

   Anderzijds maakt het algoritme ook gebruik van de functie `DFAedge(d, c)`, die de staten teruggeeft die vanuit d kunnen bereikt worden bij symbool c.

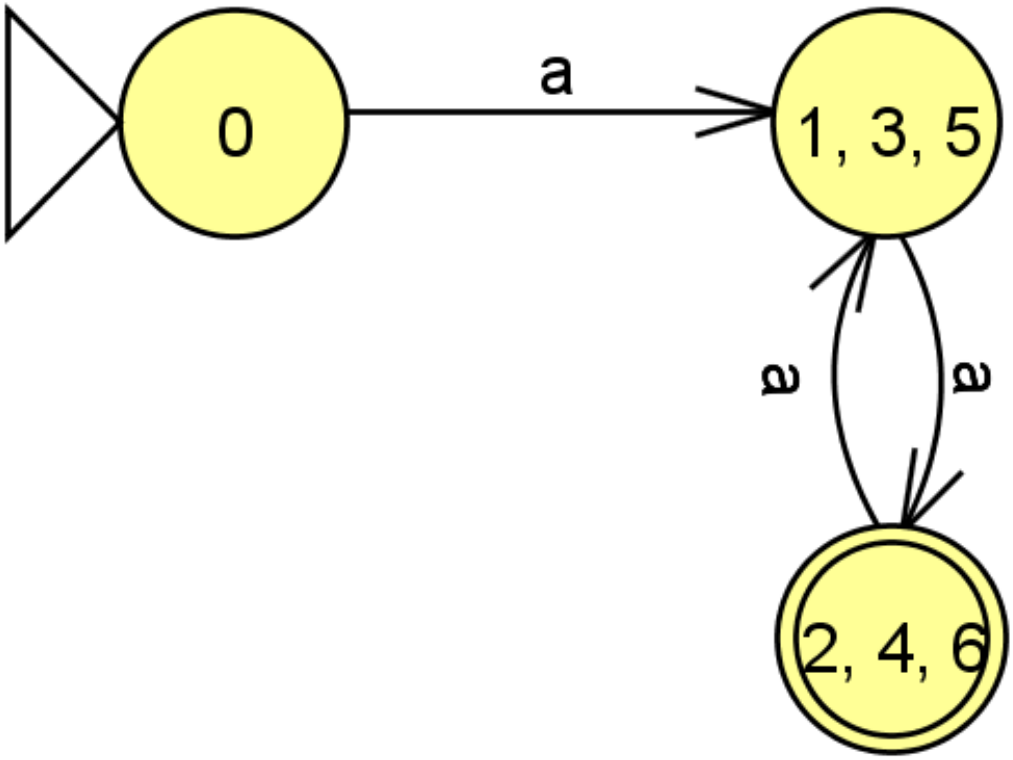
   _ToDo: verder uitleggen

2. De automaat is enkel in staat om enkelvoudige a symbolen te verwerken en aangezien er geen lege strings zijn, draagt de `closure(S)` functie hier niets bij. De eerste verwerking zorgt ervoor dat staten 1 en 3 bereikt worden. Vanuit 1 kan er naar 2 gegaan worden en vanuit 3 kan er naar 4 gegaan worden. In dit geval moet dit herhaald worden tot dat er een combinatie van staten gevonden is die al eerder gecombineerd zijn. Dit is het geval als uiteindelijk de staten



6 en 2 bereikt worden. Vanuit 6 kan naar 3 gegaan worden en vanuit 2 kan naar 1 gegaan worden. Die combinatie bestaat al, en da DFA is voltooid.

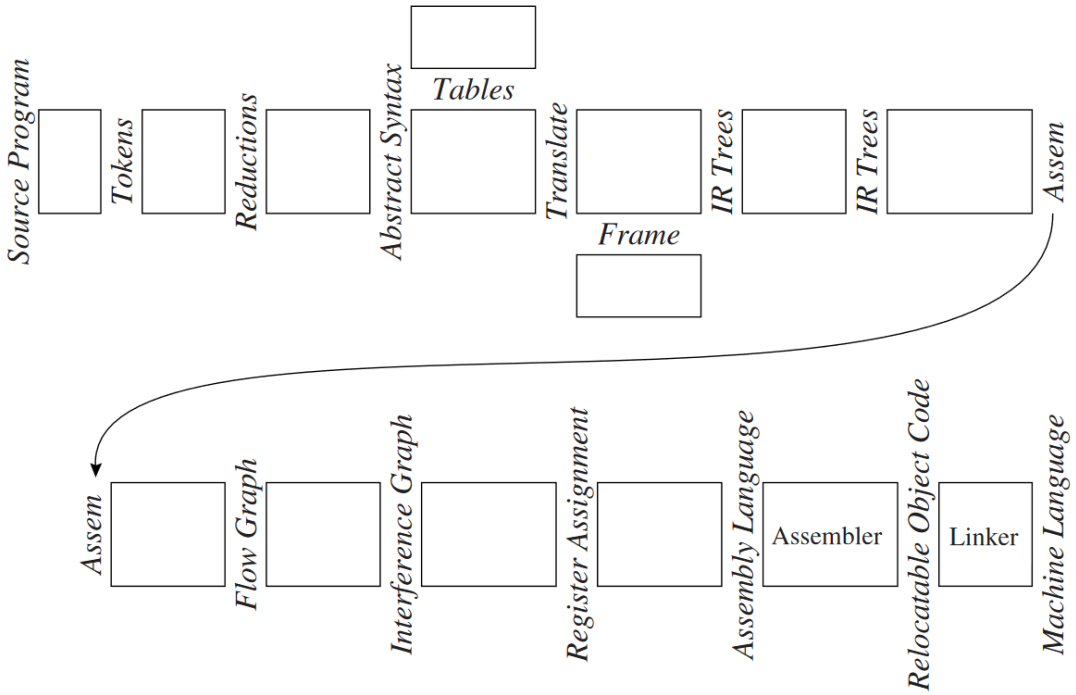
3. Het algoritme garandeerd niet dat de opgeleverde DFA optimaal is, maar er kunnen nadien wel nog optimalisaties doorgevoerd worden. Staten die equivalent zijn kunnen samengenomen worden. Een staat s_1 is equivalent met staat s_2 als ze beiden finaal of niet finaal zijn voor dezelfde symbolen en als voor elk symbool c, $\text{trans}[s_1, c] = \text{trans}[s_1, c]$. In dit geval is dit waar voor staat $\{3, 1\}$ met $\{1, 5\}$ en voor staat $\{4, 2\}$ met $\{6, 2\}$. Deze kunnen gecombineerd worden.



1.3 Compilerfasen - II

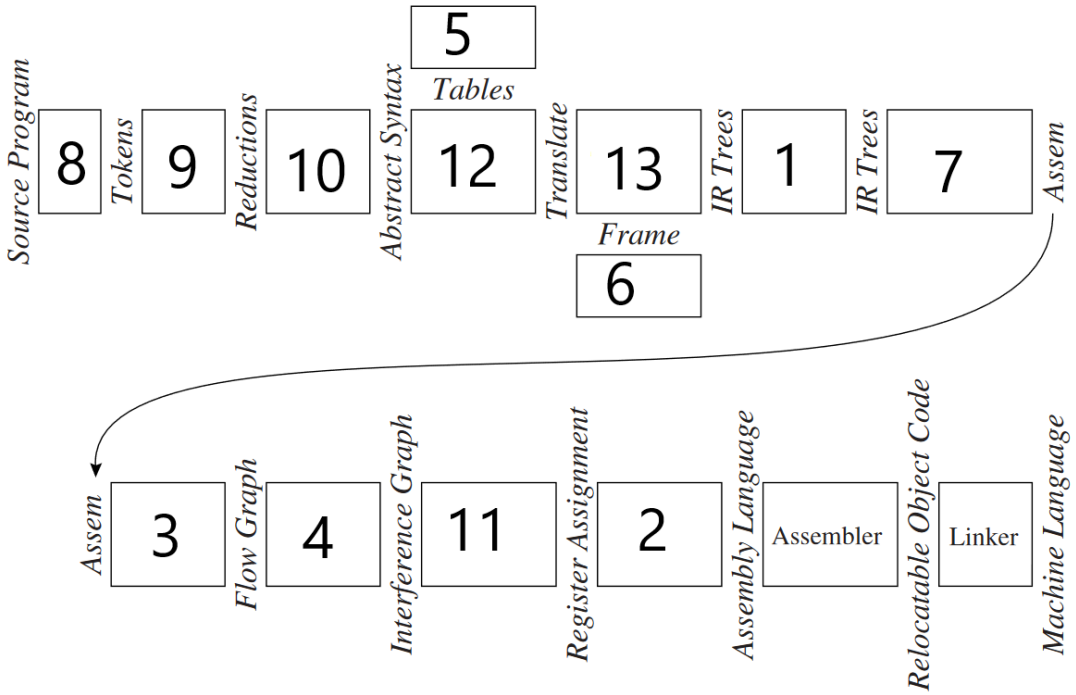
De volgende figuur toont de intermediaire voorstellingen die gebruikt worden als interface tussen de verschillende fasen van een compiler. Vul de nummers in van de corresponderende fasen in een compiler.

1. Canonicalize
2. Code Emission
3. Control Flow Analysis
4. Data Flow Analysis
5. Environments
6. Frame Layout
7. Instruction Selection
8. Lex
9. Parse
10. Parsing Actions

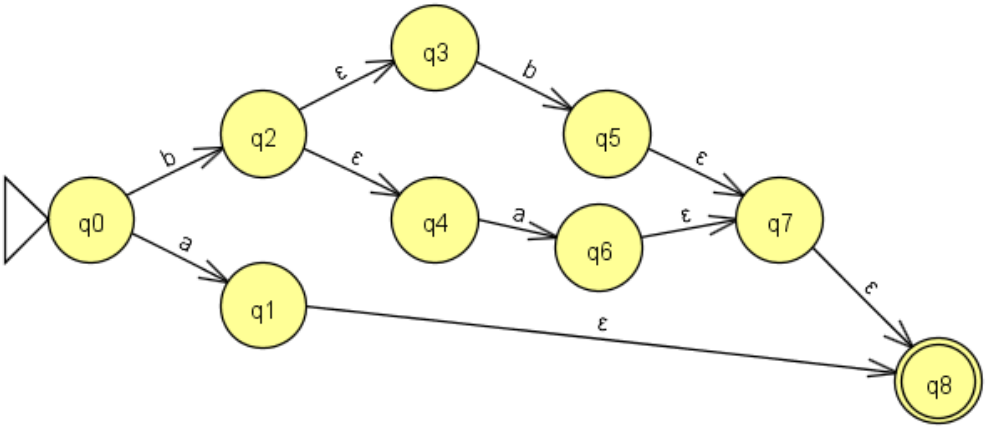


11. Register Allocation
12. Semantic Analysis
13. Translate

Antwoord



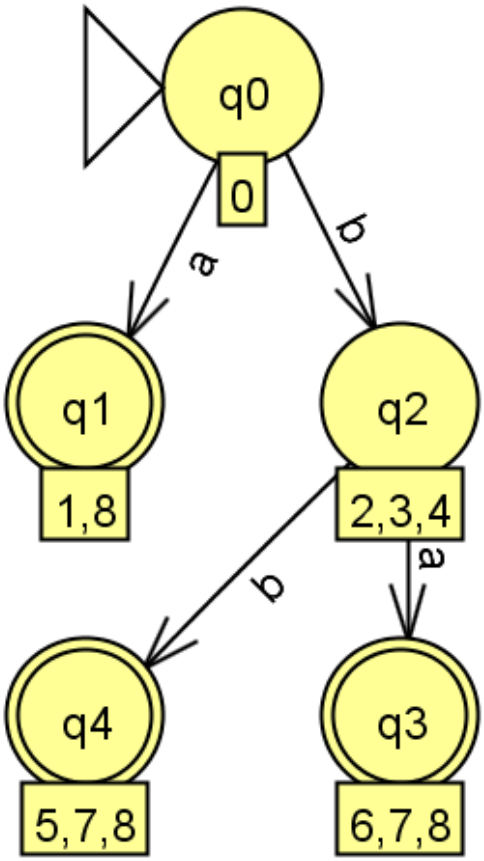
1.4 NFA



1. Converteer deze NFA naar een DFA.
2. Welke reguliere expressie wordt hiermee voorgesteld?

Antwoord

1. Gebruik dezelfde methodologie als in vraag 1.2.



2. De reguliere expressie die voorgesteld wordt is

```
a+ba+bb
```

ToDo: hoe bekomen

1.5 Parsing - I

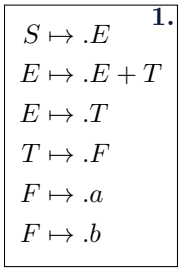
Gegeven de grammatica

$$
\begin{aligned}
S &\mapsto E \\
E &\mapsto E + T \\
E &\mapsto T \\
T &\mapsto F \\
F &\mapsto a \\
F &\mapsto b
\end{aligned}
$$

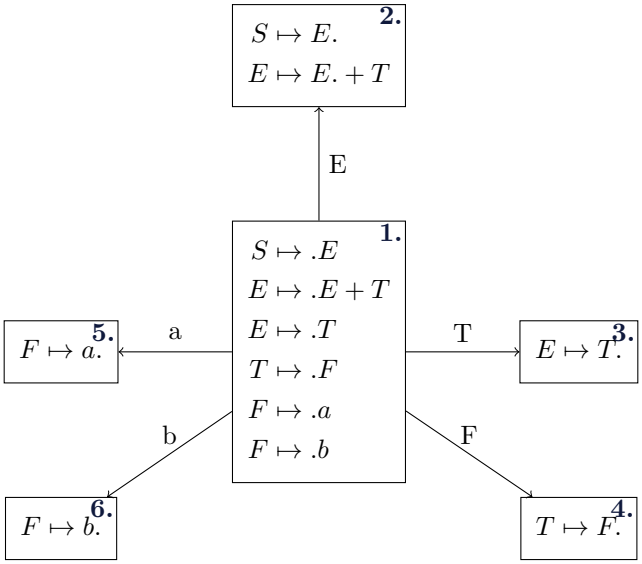
1. Welke zijn de items in de initiële toestand (1) van de LR(0) parser generator?
2. Teken de toestanden die bereikt worden na één shift of reduce actie van de LR(0) parser generator vanuit toestand 1. Teken ook de items in elke toestand.

Antwoord

1. Elke productieregel wordt een item in deze initiële toestand voor deze grammaatica.



2. Vanuit de eerste toestand zijn er enkel shift acties mogelijk.



1.6 Parsing - II

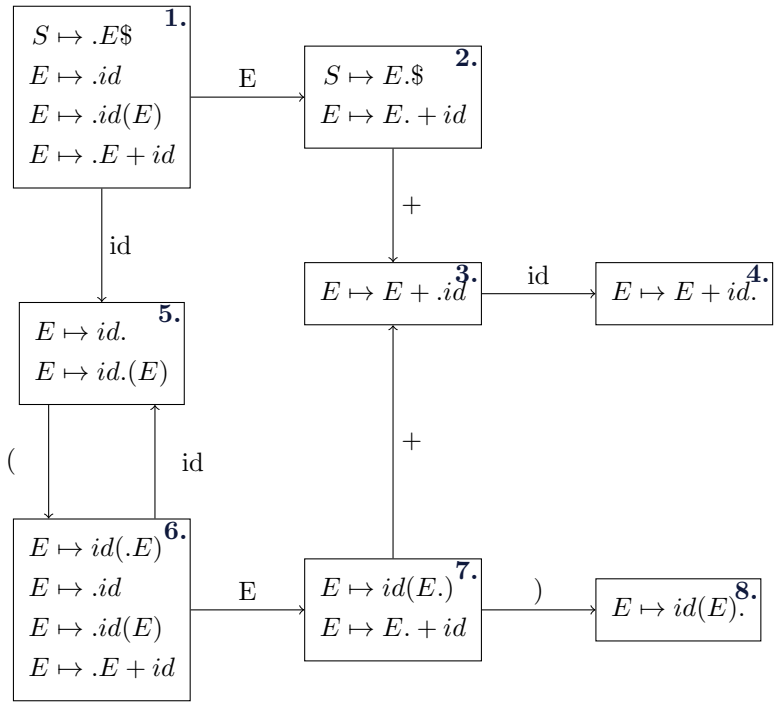
Gegeven de volgende context-vrije grammatica:

0. $S \mapsto E\$$
1. $E \mapsto id$
2. $E \mapsto id(E)$
3. $E \mapsto E + id$

1. Bouw een LR(0) DFA voor deze grammatica.
2. Is dit een LR(0) grammatica? Toon aan Waarom.
3. Is dit een SLR grammatica? Toon aan Waarom.
4. Is dit een LR(1) grammatica? Toon aan Waarom.

Antwoord

1. De LR(0) toestandenautomaat:



2. Dit is geen LR(0) grammatica. De LR(0) parsetabel (tabel 1.1) toont aan dat er een shift-reduce conflict is in toestand 5 voor symbool (.
3. Voor de SLR parsetabel moet eerst de follow set bepaald worden van elke niet-terminaal dat niet S is.
   - FOLLOW(E) = +)

	id	(	)	+	$	S	E
1	s5						g2
2				s3	a		
3	s4						
4	r3	r3	r3	r3	r3		
5	r1	s6, r1	r1	r1	r1		
6	s5						g7
7			s8	s3			
8	r2	r2	r2	r2	r2		

Tabel 1.1: De LR(0) parsetabel.

	id	(	)	+	$	S	E
1	s5						g2
2				s3	a		
3	s4						
4			r3	r3			
5		s6	r1	r1			
6	s5						g7
7			s8	s3			
8			r2	r2			

Tabel 1.2: De SLR parsetabel.

De reductie voor elke niet-terminaal moet nu enkel uitgevoerd worden voor de terminalen in de follow set. De SLR parsetabel (tabel 1.2) toont aan dat er geen shift-reduce conflicten zijn. Dit is een SLR grammatica.

4. Als een grammatica SLR is, is het ook een LR(1) grammatica vanwege de hiërarchische ordening.