

Bare Bones Programming using Python interpreter

Een programmeertaal gemaakt door Khemin van Gestelen, Niels Van
den Broeck, Elias Dams, Robin Dillen

Taal

- Gebaseerd op de bare bones taal

testcode.txt		double.txt		testCode_1.txt	
1	x = 5;	1	import clear;	1	import double;
2	while x != 0 {	2		2	
3	print(x);	3	inplace clear Aux;	3	Counter = 15;
4	decr x;	4	while Aux < x1 {	4	inplace double Counter;
5	}	5	incr Aux;	5	print(Counter);
6		6	}	6	while 7 < Counter {
7		7	while Aux != 0 {	7	decr Counter;
		8	incr x1;	8	}
		9	decr Aux;	9	print(Counter);
		10	}		

Taal

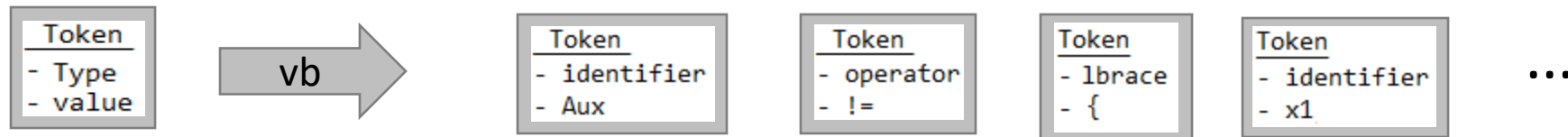
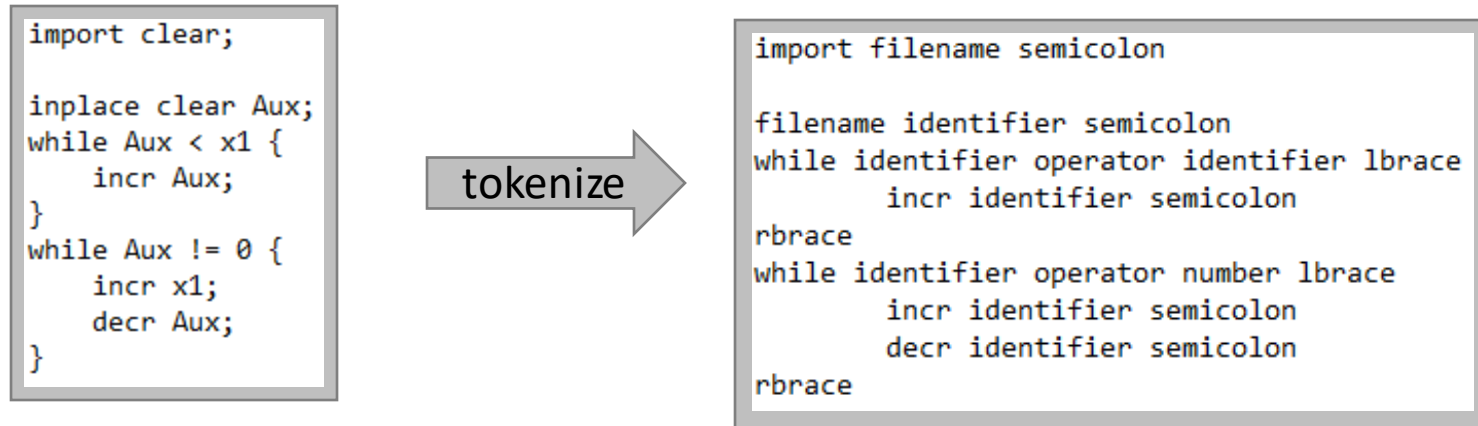
- Gebaseerd op de bare bones taal

ROOT	→	IMPORT BODY
IMPORT	→	TMP_IMPORT IMPORT / ϵ
TMP_IMPORT	→	import FUNCTIONNAME semicolon
LOOP	→	while CONDITION lbrace BODY rbrace
BODY	→	TMP_BODY BODY / ϵ
TMP_BODY	→	ASSIGN / LOOP
ASSIGN	→	ASSIGN ASSIGN / incr IDENTIFIER semicolon / decr IDENTIFIER semicolon / / inplace FUNCTIONNAME ARGUMENT semicolon / FUNCTIONNAME ARGUMENT semicolon / IDENTIFIER OPERATION DIGIT semicolon / const IDENTIFIER OPERATION DIGIT semicolon / print lparen IDENTIFIER rparen semicolon
ARGUMENT	→	IDENTIFIER
CONDITION	→	IDENTIFIER OPERATION IDENTIFIER / IDENTIFIER OPERATION DIGIT / DIGIT OPERATION IDENTIFIER / DIGIT OPERATION DIGIT
IDENTIFIER	→	identifier
FUNCTIONNAME	→	filename
OPERATION	→	operator
DIGIT	→	number

```
import filename semicolon inplace filename identifier semicolon while identifier operator identifier lbrace incr identifier semicolon rbracewhile identifier operator number lbraceincr identifier semicolondecr identifier semicolon rbrace
```

Lexer

- Tokenize de gekende syntaxes



Parser

CFG omzetten naar DFA.

Initiele staat aanmaken door closure van root + symbool toevoegen die locatie bijhoudt (punt)

1. Loop over producties van huidige staat
2. Als punt niet op het einde staat -> goto berekenen, anders naar volgende productie
3. Punt verschuiven van alle goto producties en closure berekenen van item na punt
4. Deze verzameling van producties stelt een DFA staat voor. Met transitie van huidige staat naar de nieuwe
5. Herhaal met huidige staat = nieuwe staat tot er geen nieuwe staten meer worden aangemaakt

$S' \rightarrow S$
 $S \rightarrow AA$
 $A \rightarrow .aA \mid b$
 (originele CFG)

$S \rightarrow A.A$
 $A \rightarrow .aA \mid b$ (stap 0) \Rightarrow $A \rightarrow .aA \mid b$ (stap 1) \Rightarrow $A \rightarrow .aA$ (stap 2) \Rightarrow $A \rightarrow a.A$ (stap 3.1) \Rightarrow $A \rightarrow a.A$ (stap 3.2)
 $A \rightarrow .aA \mid b$

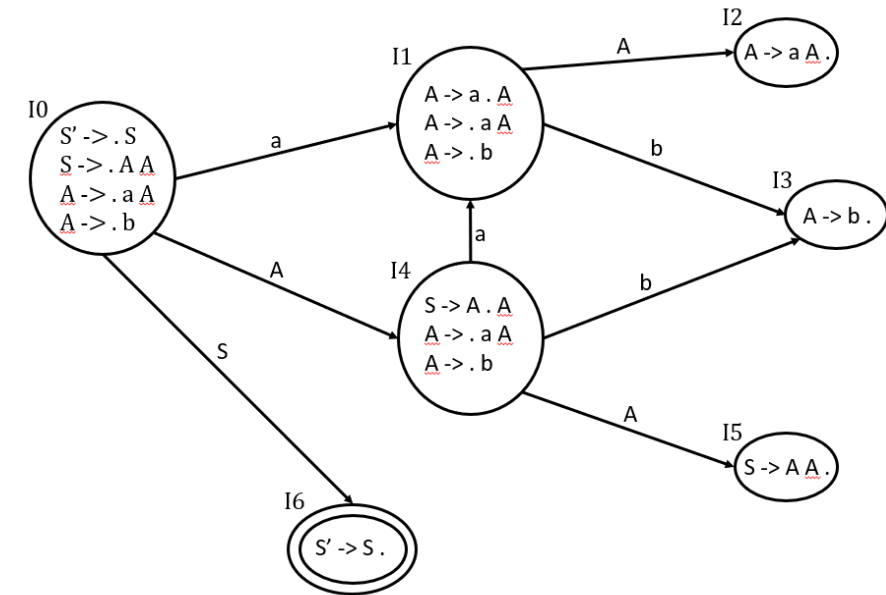
Parse table

- Staten van DFA bepaald tijdens opstellen van de canonieke collectie
- Samengevoegd tot DFA in `getParseTable()`
- DFA meegegeven aan `ParseTable` constructor
- `getParseTable()` geeft de overeenkomstige tabel terug


```

156 ParseTable Parser::getParseTable() {
157     DFA dfa = DFA(dfaStates, terms: cfg->getTerminals(), vars: cfg->getVariables());
158
159     return ParseTable(dfa);
160 }

```



	a	b	EOS	A	S
I0	shift I1	shift I3		I4	I6
I1	shift I1	shift I3		I2	
I2	A -> a A	A -> a A	A -> a A		
I3	A -> b	A -> b	A -> b		
I4	shift I1	shift I3		I5	
I5	S -> A A	S -> A A	S -> A A		
I6			accept		

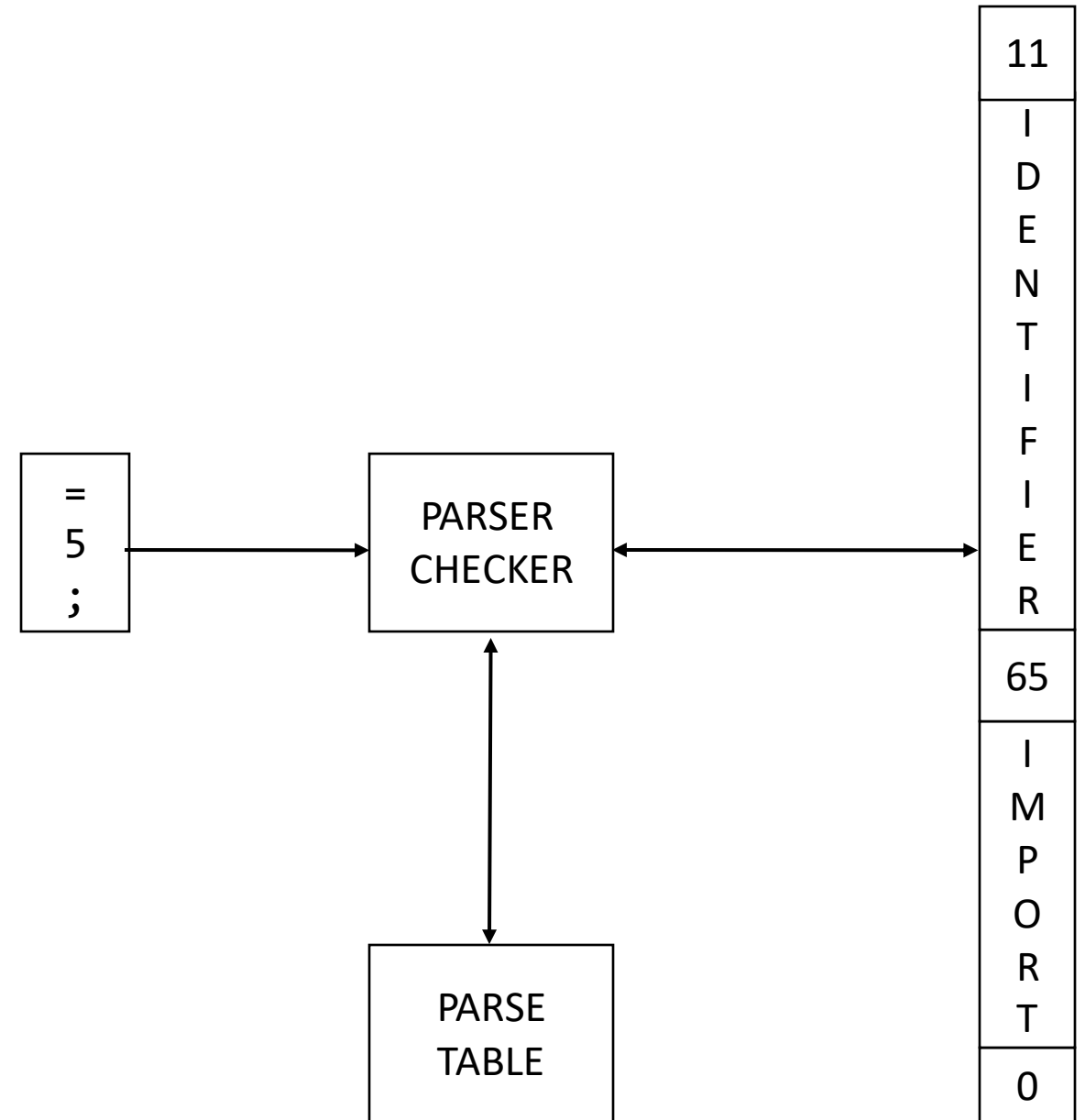
	comma	const	decr	epsilon1_  183
0		shift 13	shift 24	shift 62
1				
10	ARGUMENT -> IDENTIFIER	ARGUMENT -> IDENTIFIER	ARGUMENT -> IDENTIFIER	ARGUMENT -> IDENTIFIER
11				
12	ASSIGN -> <u>FUNCTIONNAME</u> ARGUMENT semicolon	ASSIGN -> <u>FUNCTIONNAME</u> ARGUMENT semicolon	ASSIGN -> <u>FUNCTIONNAME</u> ARGUMENT semicolon	ASSIGN -> <u>FUNCTIONNAME</u> ARGUMENT semicolon
13			shift 18	
14				
15				
16				
17	ASSIGN -> const IDENTIFIER OPERATION DIGIT semicolon	ASSIGN -> const IDENTIFIER OPERATION DIGIT semicolon	ASSIGN -> const IDENTIFIER OPERATION DIGIT semicolon	ASSIGN -> const IDENTIFIER OPERATION DIGIT semicolon
18				
19				
2				
20	ASSIGN -> const decr IDENTIFIER semicolon	ASSIGN -> const decr IDENTIFIER semicolon	ASSIGN -> const decr IDENTIFIER semicolon	ASSIGN -> const decr IDENTIFIER semicolon
21				
22				
23	ASSIGN -> const incr IDENTIFIER semicolon	ASSIGN -> const incr IDENTIFIER semicolon	ASSIGN -> const incr IDENTIFIER semicolon	ASSIGN -> const incr IDENTIFIER semicolon
24				
25				
26	ASSIGN -> decr IDENTIFIER semicolon	ASSIGN -> decr IDENTIFIER semicolon	ASSIGN -> decr IDENTIFIER semicolon	ASSIGN -> decr IDENTIFIER semicolon
27				

ARGUMENT	ASSIGN	BODY	CONDITION	DIGIT	FUNCTIONNAME	IDENTIFIER	IMPORT	LOOP	OPERATION	ROOT	TEMPBODY	TEMPIMPORT
	58			47	9	1	66	59		68	40	60
									2			
						14						
									15			
				16								
						19						
				3		5						
						22						
						25						
						26						

Parser Checker

Input nakijken door:

1. Stack aanmaken
2. DFAState-naam op stack
3. Tokens overlopen en nakijken met tabel en stack.



Parser Checker

4 mogelijke gevallen:

- Shift X: Token pushen op stack samen met getal achter shift.
- Production: Tokens op stack vervangen met originele variabele van productie. Kijken op tabel naar getal voor variabele en variabele zelf, en op stack pushen.
- Accept: "accept" wordt op de stack gepushed en de tokens worden geaccepteerd.
- Empty: Geen geldige tokenstring.

shift 45	
-----	-----
ROOT -> IMPORT BODY	ROOT -> IMPORT BODY
-----	-----
	accept
-----	-----
-----	-----
ARGUMENT -> IDENTIFIER	ARGUMENT -> IDENTIFIER
-----	-----

Parser Checker

Geen geldige tokenstring als:

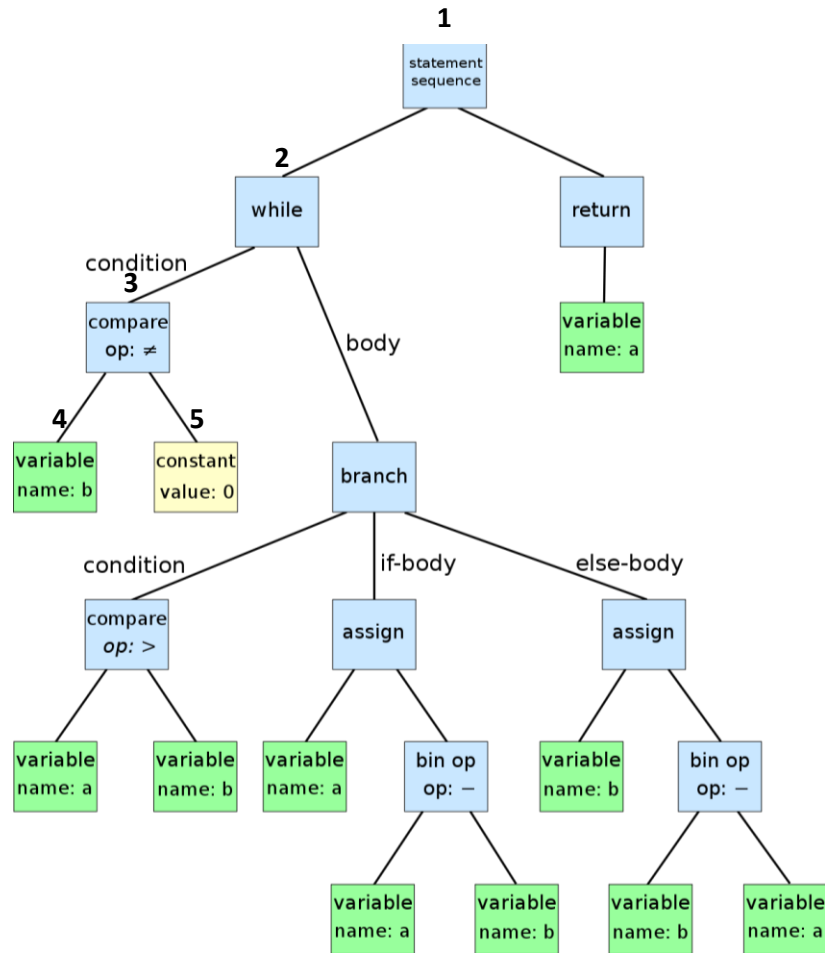
- Leeg vakje in tabel.
- Productie in stack niet kan worden omgewisseld.

```
x = 5;  
while x != 0 {  
    print(x);  
    decr x;  
}
```



```
x = 5;  
while x != 0 {  
    print(x;  
    decr  
}
```





Abstract Syntax tree

- Boom representatie van het programma
- Tokens
 - Keyword
 - Data
- Error detection
 - Syntax error
 - Logic errors

Implementatie AST

- STL implementatie
- Iterator

```
for (auto node : Const_iterator = ast.begin(); node != ast.end(); ++node) {  
    compileNode(node, program);  
}
```

De compiler

- Functional geïmplementeert
- Een incr in python:

```
program << (unsigned char) LOAD_NAME << (unsigned char) name_index << EOL;  
program << (unsigned char) LOAD_CONST << (unsigned char) const_index << EOL;  
program << (unsigned char) INPLACE_ADD << (unsigned char) 0 << EOL;  
program << (unsigned char) STORE_NAME << (unsigned char) name_index << EOL;
```

Vragen?