

Chapter 1

Exercises

Course “Compiler Construction”

Martin Steffen

Spring 2022



Section

Exercises 02

2.1 First and follow set
2.2 Nullability
2.3 Associativity and precedence
2.4 Tiny language

Chapter 1 “Exercises”
Course “Compiler Construction”
Martin Steffen
Spring 2022



Exercises 02

2.1 First and follow set

2.2 Nullability

2.3 Associativity and precedence

2.4 Tiny language

Exercises 02

2.1 First and follow set

2.2 Nullability

2.3 Associativity and precedence

2.4 Tiny language

2.1: First and follow for this

$$
\begin{array}{rcl}
exp & \rightarrow & term\ exp' \\
exp' & \rightarrow & addop\ term\ exp' \mid \epsilon \\
addop & \rightarrow & + \mid - \\
term & \rightarrow & factor\ term' \\
term' & \rightarrow & mulop\ factor\ term' \mid \epsilon \\
mulop & \rightarrow & * \\
factor & \rightarrow & (\ exp\ ) \mid \mathbf{number}
\end{array}
$$



2.1: Follow set: fill out this



production	init	pass 1	pass 2
$exp \rightarrow term\ exp'$			
$exp' \rightarrow addop\ term\ exp'$			
$exp' \rightarrow \epsilon$			
$addop \rightarrow +$			
$addop \rightarrow -$			
$term \rightarrow factor\ term'$			
$term' \rightarrow mulop\ factor\ term'$			
$term' \rightarrow \epsilon$			
$mulop \rightarrow *$			
$factor \rightarrow (\ exp\ )$			
$factor \rightarrow \mathbf{number}$			

2.2: Nullable algo?

Describe an algo for checking nullability.



Exercises 02

2.1 First and follow set

2.2 Nullability

2.3 Associativity and precedence

2.4 Tiny language

2.3: Associtativity and precedence

op	precedence	associativity
$+, -$	low	left assoc.
$*, /$	higher	left. assoc.
$\uparrow$	highest	right. assoc



Flat (simple, but highly ambiguous)

$$
\begin{aligned}
exp &\rightarrow \mathbf{number} \mid (\, exp\,) \mid exp\ op\ \ exp \\
op &\rightarrow + \mid - \mid * \mid \uparrow
\end{aligned}
$$



Grammar from the lecture as inspiration

$$
\begin{array}{rcl}
exp & \rightarrow & exp\ addop\ term \mid term \\
addop & \rightarrow & + \mid - \\
term & \rightarrow & term\ mulop\ factor \mid factor \\
mulop & \rightarrow & * \\
factor & \rightarrow & (\ exp\ ) \mid \mathbf{number}
\end{array}
$$



2.4: TINY grammar



$$
\begin{array}{rcl}
\textit{program} & \rightarrow & \textit{stmts} \\
\textit{stmts} & \rightarrow & \textit{stmts} \,;\, \textit{stmt} \mid \textit{stmt} \\
\textit{stmt} & \rightarrow & \textit{if-stmt} \mid \textit{repeat-stmt} \mid \textit{assign-stmt} \\
 & \mid & \textit{read-stmt} \mid \textit{write-stmt} \\
\textit{if-stmt} & \rightarrow & \textbf{if}\ \textit{expr}\ \textbf{then}\ \textit{stmt}\ \textbf{end} \\
 & \mid & \textbf{if}\ \textit{expr}\ \textbf{then}\ \textit{stmt}\ \textbf{else}\ \textit{stmt}\ \textbf{end} \\
\textit{repeat-stmt} & \rightarrow & \textbf{repeat}\ \textit{stmts}\ \textbf{until}\ \textit{expr} \\
\textit{assign-stmt} & \rightarrow & \textbf{identifier} := \textit{expr} \\
\textit{read-stmt} & \rightarrow & \textbf{read identifier} \\
\textit{write-stmt} & \rightarrow & \textbf{write}\ \ \textit{expr} \\
\textit{expr} & \rightarrow & \textit{simple-expr}\ \textit{comparison-op}\ \textit{simple-expr} \mid \textit{simple-expr} \\
\textit{comparison-op} & \rightarrow & < \mid = \\
\textit{simple-expr} & \rightarrow & \textit{simple-expr}\ \textit{addop}\ \textit{term} \mid \textit{term} \\
\textit{addop} & \rightarrow & + \mid - \\
\textit{term} & \rightarrow & \textit{term}\ \textit{mulop}\ \textit{factor} \mid \textit{factor} \\
\textit{mulop} & \rightarrow & * \mid / \\
\textit{factor} & \rightarrow & (\ \textit{expr}\ ) \mid \textbf{number} \mid \textbf{identifier}
\end{array}
$$

Exercises 02

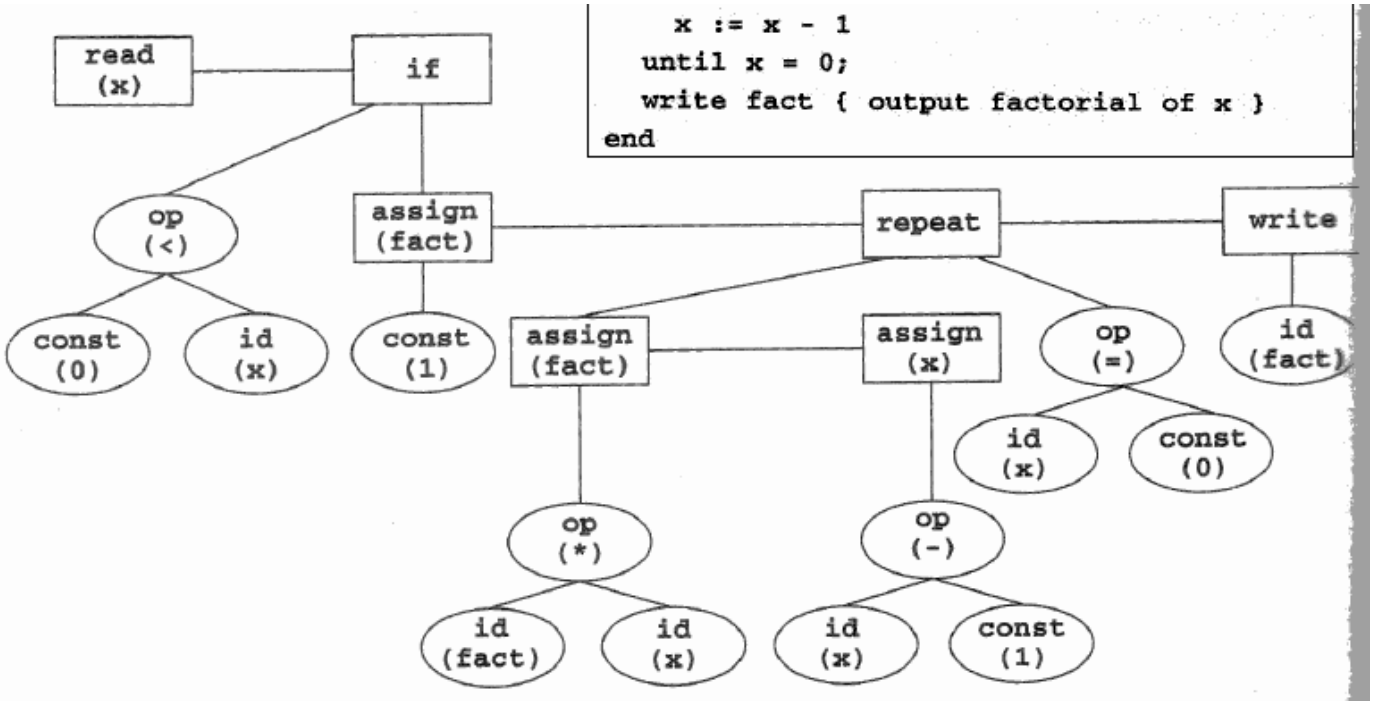
2.1 First and follow set

2.2 Nullability

2.3 Associativity and precedence

2.4 Tiny language

2.5 TINY AST


x := x - 1
until x = 0;
write fact { output factorial of x }
end
read (x)
if
op (<)
assign (fact)
repeat
write
const (0)
id (x)
const (1)
assign (fact)
assign (x)
op (=)
id (fact)
id (x)
const (0)
op (*)
op (-)
id (fact)
id (x)
id (x)
const (1)

Sample program

```
1  read x; { input as integer }
2  if 0 < x then { don't compute if x <= 0 }
3    fact := 1;
4    repeat
5      fact := fact * x;
6      x := x -1
7    until x = 0;
8    write fact    { output factorial of x }
9  end
```

