Examination in Compilers, EDAN65

Department of Computer Science, Lund University

2025–10–28, 14.00-19.00

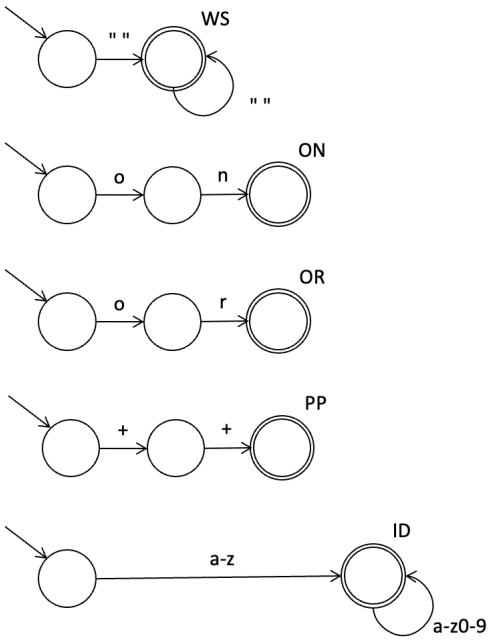
SOLUTIONS

Max points: 60
For grade 3: Min 30
For grade 4: Min 40
For grade 5: Min 50

1 Lexical analysis

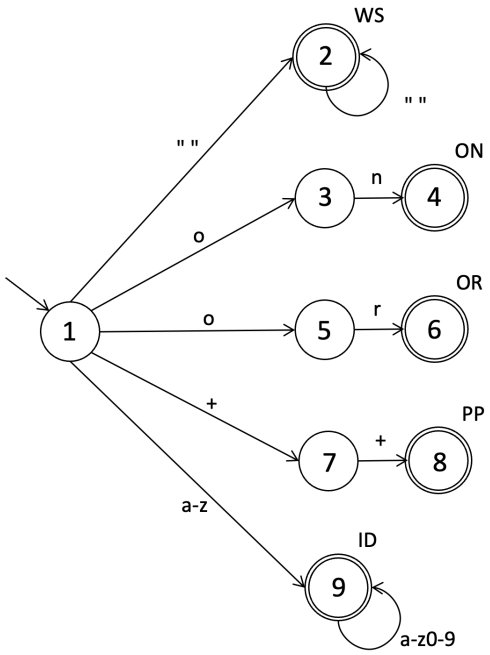
a) (5p)

Finite automata for the five token definitions:



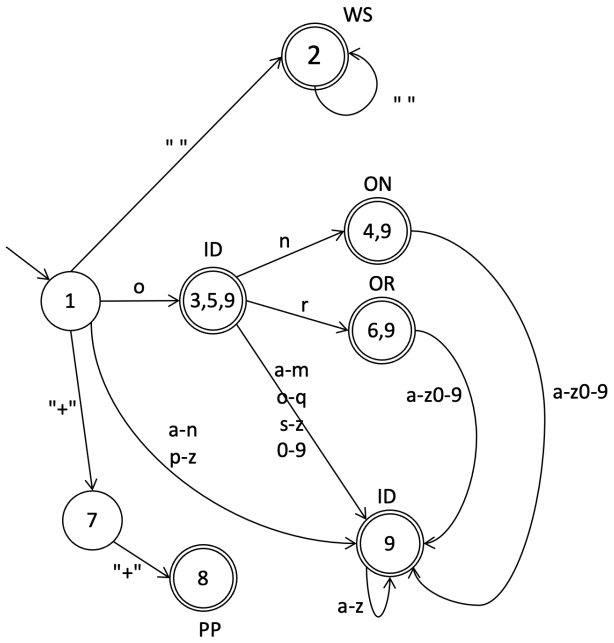
b) (1p)

NFA:



c) (5p)

DFA:



d) (2p)

From the string "`on1 or on+++`", the following sequence of tokens will be generated:

```
ID("on1")
WS
OR
WS
ON
PP
ERR("+")
```

e) (2p)

If earliest match is used, the following sequence would have been generated from "on1 or on++":

```
ID("o")
ID("n")
ERR("1")
WS
ID("o")
ID("r")
WS
ID("o")
ID("n")
ERR("+")
ERR("+")
ERR("+")
```

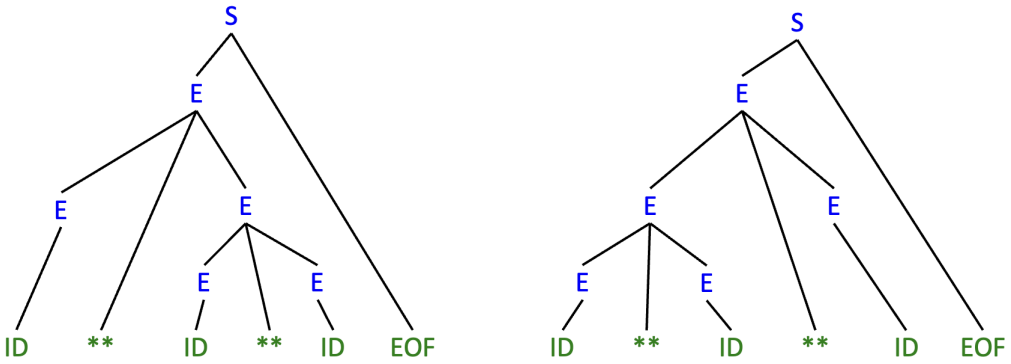
2 Context-Free Grammars

a) (5p)

An example sentence is

```
ID ** ID ** ID EOF
```

For this sentence, the following two parse trees can be constructed:



b) (5p)

An equivalent unambiguous grammar:

p_0: `S -> E EOF`
p_1: `E -> E "+" T`
p_2: `E -> T`
p_3: `T -> F "**" T`
p_4: `T -> F`
p_5: `F -> ID`
p_6: `F -> "(" E ")"`

c) (5p)

The FOLLOW set for `E` is { `EOF`, `","`, `")"` }.

Derivations:

$\texttt{S} \Rightarrow \underline{\texttt{E EOF}}$
$\Rightarrow \texttt{ID "(" L ")" EOF}$
$\Rightarrow \texttt{ID "(" M } \underline{\texttt{E ")"}} \texttt{ EOF}$
$\Rightarrow \texttt{ID "(" M } \underline{\texttt{E ","}} \texttt{ E ")" EOF}$

d) (5p)

The LL(1) parser table

	EOF	ID	"("	")"	","
S		p0			
E		p1,p2			
L		p4		p3	
M		p5,p6			

e) (5p)

```
S -> E EOF
E -> ID [ "(" [ ( E "," )* E ] ")" ]
```

3 Program analysis

a) (6p)

```
syn int LetterList.max();
eq LetterList1.max() = getLetter().facevalue();
eq LetterList2.max() =
    getHead().facevalue() > getTail().max()
    ? getHead().facevalue()
    : getTail().max();

inh boolean Letter.inSubPos();
eq LetterList1.getLetter().inSubPos() = false;
eq LetterList2.getHead().inSubPos() =
    getHead().facevalue() < getTail().max();
```

As an alternative to the `LetterList2.max()` equation, the `java.lang.Math.max` function can be used:

```
// Alternative equation:
eq LetterList2.max() =
    java.lang.Math.max(getHead().facevalue(), getTail().max());
```

b) (4p)

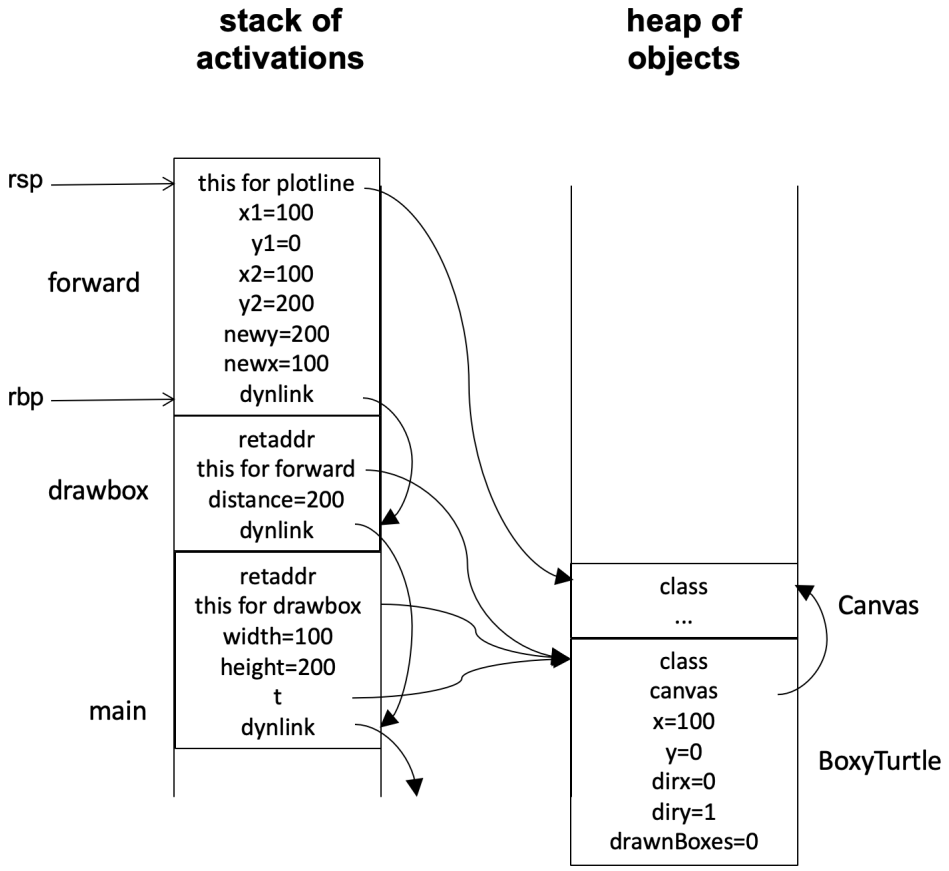
Attribute grammar:

```
syn int Numeral.value() = counter().count();
coll Counter Numeral.counter();
Letter contributes inSubPos()
  ? -facevalue()
  : facevalue()
to Numeral.counter();
```

4 Code generation and run-time systems

a) (7p)

The situation at runtime:



b) (3p)

The `drawbox` method should first follow the `this`-pointer, which is 16 bytes below the basepointer, to get to the `BoxyTurtle` object. In the drawing above, we have assumed that the header of the object consists of one qword (the pointer to the class descriptor). The `drawnBoxes` variable, which is the 6th field in the object, will then be located at 6*8 = 48 bytes below the start of the object.

x86 code for incrementing `drawnboxes`:

```
movq 16(%rbp), %rax   # RAX now points to the BoxyTurtle object
incq 48(%rax)         # Increment the drawnBoxes field
```