[20231221] INFOB3TC - Talen en compilers - 2 - USP

Cursus: BETA-INFOB3TC Talen en compilers (INFOB3TC)

Tijdsduur: 2 uur

Aantal vragen: 18

[20231221] INFOB3TC - Talen en compilers - 2 - USP

Cursus: Talen en compilers (INFOB3TC)

Aantal vragen: 18

1
6 pt.

Which abstract syntax is the best fit for the following grammar:

V -> Va | bV | V[V] = V

a. data V a b = SingleA a | SingleB b | Assign (V a b) (V a b) (V ab)

b. data V = Av V | Bv V | Assign V V V

c. data T = SingleA Char | SingleB Char | Assign String String String

d. data V = V ((Maybe V), (Maybe V), [V,V,V])

2 Given is the following Haskell datatype:

data NonEmpty a = NonEmpty a (List a)
data List a = Empty | Cons a (List a)

9 pt. **a.** We define a combined algebra for *NonEmpty and List* as follows

type NonEmptyListAlgebra a n l = (..., ..., ...)

What is the type of the component corresponding to the constructor *NonEmpty*?

9 pt. **b.** The fold function for *NonEmpty* has the following template:

```
foldNonEmpty (nonempty, empty, cons) = f where
  f (NonEmpty x l) = ...
  g Empty = ...
  g (Cons x l) = ...
```

What code should be written for the *NonEmpty* case?

8 pt. **c.** We want to use the function *foldNonEmpty* to count the number of *Cons*'s in a *NonEmpty a*. The following template is given

countCons ne = foldNonEmpty (\x y -> y, const 0, ...)

What should be written on the dots?
Give your answer as a single line of simple code.

3
6 pt.

A grammar has the following productions:

S -> aSb | Sa | ε

We need one more production to be able to derive the sentence *aabbaabb*. Which of the following productions should we add?

a. S -> aSbb

b. S -> bbSaa

c. S -> SS

d. All the other answers are correct.

4
4 pt.

Is the following statement **false**, **true** or plain **nonsense** (nonsense means that you can't even decide that it is true or false, it is as it were type incorrect)?

If $A \rightarrow bC$ is a production of a context-free grammar G , then $AXY \Rightarrow^* bCXY$

a. Nonsense

b. False

c. True

5
4 pt.

Is the following statement **false**, **true** or plain **nonsense** (nonsense means that you can't even decide that it is true or false, it is as it were type incorrect)?

If $AX \rightarrow bC$ is a production of a context-free grammar G , then $XA \Rightarrow^* XbC$

a. True

b. False

c. Nonsense

6 Is the following statement **false**, **true** or plain **nonsense** (nonsense means that you can't even decide
4 pt. that it is true or false, it is as it were type incorrect)?

If $A \to bCX$ is the only production with A as a left hand side in a context-free grammar G, then it can never be the case that $A \Rightarrow^* bC$

a. True

b. False

c. Nonsense

7 Is the following statement **false**, **true** or plain **nonsense** (nonsense means that you can't even decide
4 pt. that it is true or false, it is as it were type incorrect)?

A context-free grammar that employs a finite alphabet can describe a language with infinitely long sentences.

a. True

b. False

c. Nonsense

8 Is the following statement **false**, **true** or plain **nonsense** (nonsense means that you can't even decide
4 pt. that it is true or false, it is as it were type incorrect)?

A context-free grammar that employs a finite alphabet can describe a language with infinitely many sentences.

a. True

b. False

c. Nonsense

9 Assume that the function *abs* computes the absolute value of an integer.
6 pt.

Which of the following grammars generates the language

$$\{a^k b^n c^m \mid k = abs(n-m), k, n, m \geq 0\}$$

a. S -> U | T
U -> aUb | bUc | ε
T -> bTc | aTc | ε

b. S -> TU
T -> A | B
A -> aAb | ε
B -> bBc | ε
U -> V | W
W -> aWc | ε
V -> bVc | ε

c. S -> aSc | T
T -> bTc | ε

d. S -> T | U
T -> AB
A -> aAb | ε
B -> bBc | ε
U -> aUc | V
V -> bVc | ε

10
6 pt.

What is the language generated by the following grammar:

S -> AB

A -> aAb | ab

B -> bbBa | bba

a. $\{vwx \mid v, w \in a^n, x \in b^{3n}, n \geq 1\}$

b. $\{a^n b^{3n} a^n \mid n \geq 1\}$

c. $\{a^n b^n b^{2m} a^m \mid n, m \geq 1\}$

d. All other answers are wrong.

11
1 pt.

Given is the following grammar:

X → X T T X | S
S → ab | cd
T → Nat

X is the start symbol. Which of the following grammars is not left recursive, but equivalent?

a.

X → X (T T S)*
S → ab | cd
T → Nat

b.

X → S | S Z
Z → T T X | T T X Z
S → ab | cd
T → Nat

c.

X → Z T T X | S
Z → X | ε
S → ab | cd
T → Nat

d.

X → S T T Z
Z → S | X
S → ab | cd
T → Nat

12
1 pt.

Consider the following parser:

```
pX :: Parser Token Int
pX  =  pY <*>  pInt
```

The parser pInt parses a single token to an integer. It has the type Parser Token Int.

Consider the parser pY used by pX above. What would be the type of the result obtained by applying pY to a list of tokens? In other words, what is the type of:

```
runParser pY input
```

where input is a list of tokens?

a. Int → Int

b. (Int → Int , [[Token]])

c. [(Int , [Token])]

d. [(Int → Int , [Token])]

13
1 pt.

Consider the following parser:

```
pZ :: Parser Token Int
pZ  =  f <$>  pInt <*> many pInt
```

The parser pInt parses a single token to an integer. It has the type Parser Token Int.

What is the type of f ?

a. Parser Token (Int → [Int] → Int)

b. Parser Token [Int]

c. Int → Int

d. Int → [Int] → Int

14
1 pt.

What follows are several properties involving parser combinators. Which of these properties is **NOT** *true*?

a. *many (some p) ≅ some (many p)*

b. (:) <$> p <*> many p ≅ some p

c. satisfy (== c) ≅ symbol c

d. *greedy1 (greedy1 p) ≅ (\x -> [x]) <$> greedy1 p*

15
6 pt.

Given that L, L_1, L_2 are languages over the same alphabet Σ , Which of the following statements is false?

a. $L^* \cup L = L^+$

b. $(L^R)^* = (L^*)^R$

c. $(LL^*)^+ = L^+$

d. $(L_1 L_2)^R = L_2^R L_1^R$

16
5 pt.

Write a short RegExp that matches all the 'good' email addresses below, and matches none of the 'bad' addresses below.
Note that you only need to consider these examples, not anything else you may know about email addresses!

0 points: tests fail
2 points: tests pass
4 points: tests pass & regexp fits in an SMS (i.e. ≤ 140 characters)
6 points: tests pass & regexp fits in a terminal line (i.e. ≤ 80 characters)
8 points: tests pass & regexp fits in an email subject line (i.e. ≤ 48 characters)
10 points: tests pass & regexp fits in a linux username (i.e. ≤ 32 characters)

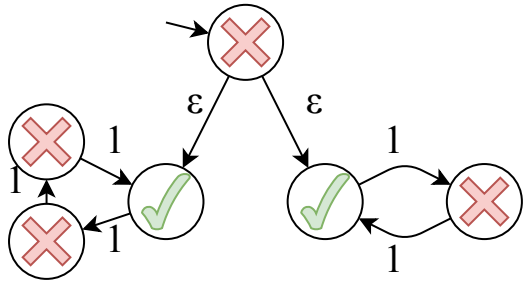
Good examples:

email@example.com
firstname+lastname@example.com
firstname.lastname@example.com
firstname_lastname@example.com
first.middle.+even.more_last@example.com
1234567890@example.com
_____@example.com
email@example.com.
email@subdomain.example.com
email@192.168.1.1
email@example.name
email@example.museum
email@example.co.jp

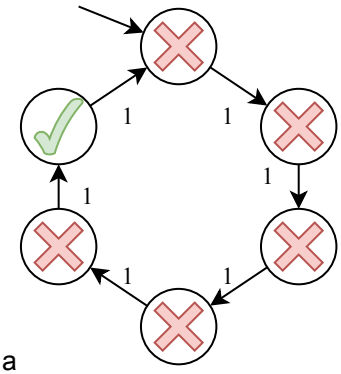
Bad examples:

@example.com
firstname..lastname@example.com
email@example
email@example.
email@_example.com
email@example..com
.email@example.com
email.@example.com
email@example+one.com
plainaddress
email.example.com
email@example@example.com

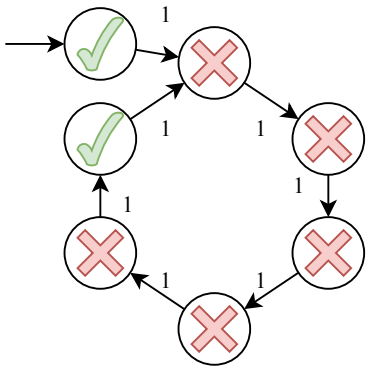
17 This question is about one NFA**ε:**



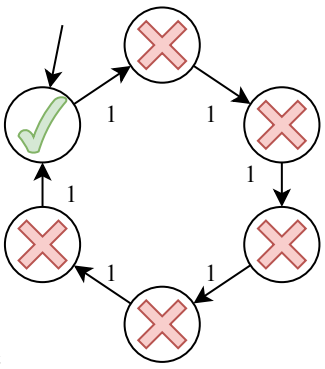
and several DFAs:



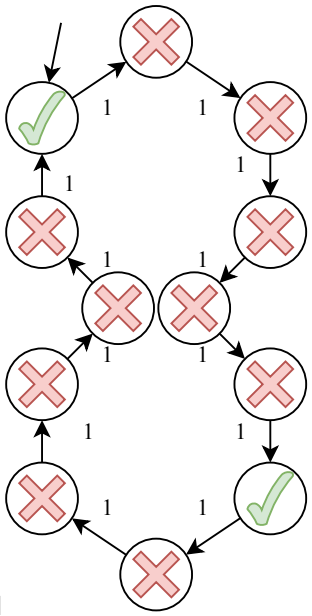
a



b



c



d

All of these automata work with the alphabet {1}.

3 pt. **a.** Which of these DFAs represent the same language as the NFA**ε**? Check all that apply.

- **a.** a
- **b.** b
- **c.** c
- **d.** d

1 pt. **b.** Which of these DFAs is the result of the subsect construction on the NFA?

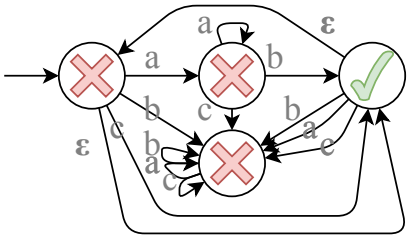
- **a.** a
- **b.** b
- **c.** c
- **d.** d

18 This question is about the following Regular Expression:

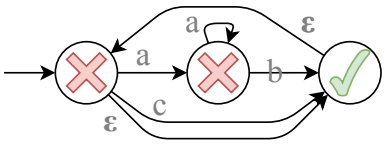
2 pt.

$(a^{+}b|c?)^{*}$

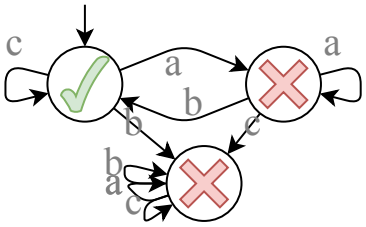
And the following four diagrams:



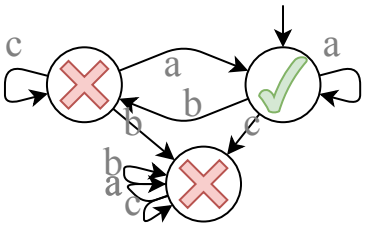
a



b



c



d

Which of these are valid NFAs that represent the regular expression? Check all that apply.

a. a

b. b

c. c

d. d