CFL Homework 2 **John Pham**

1. **What is the difference between basic regular expressions and extended regular expressions?**

   A regular language can be expressed with just a basic regular expression. That is, any extended regular expression can be converted into a basic one, but not the other way round.

2. **What is the language recognised by the regular expression (0*)*?**

   L((**0***)*) = {[]}
   The language recognised by this expression is a language consisting of just the empty string.

3. **Assuming the alphabet is the set {a, b}, decide which of the following equations are true in general for arbitrary languages A, B, and C:**

(A ∪ B) @ C	**=?**	**A @ C ∪ B @ C**
A* ∪ B*	**=?**	**(A ∪ B)***
A* @ A*	**=?**	**A***
(A ∩ B) @ C	**=?**	**(A @ C) ∩ (B @ C)**

   (A ∪ B) @ C = A @ C ∪ B @ C **is true** because (A ∪ B) gives a choice to select a string from the language A, or language B, but we can only select from one language. Then @C means we must append a string from the language C to the choice we made earlier. As we only have one choice between A or B, the form of the resulting language must be A @ C OR B @ C. Regular expression equivalences (A + B) . C == (A . C) + (B . C)

   A* ∪ B* = (A ∪ B)* **is true**

   A* @ A* = A* **is true** because the star indicates that there can be 0 or more occurrences of a string from the language A. Therefore, A* @ A* means that we can have 0 or more occurrences of A followed by 0 or more occurrences of A, which is the same as having 0 or more occurrences of A.
   This works because both sides of the concatenation have the same language A.

   (A ∩ B) @ C = (A @ C) ∩ (B @ C) **is true**

4. **Given regular expressions r1 = 1 and r2 = 0 and r3 = a. How many strings can the regular expression r1*, r2* and r3* match?**

   r1*: one -> {[]}
   r2*: one -> {
   r3*: infinite number of strings -> {[], a, aa, aaa, aaaa, aaaaa…}

5. **Give regular expressions for (a) decimal numbers and for (b) binary numbers.**

   Decimal numbers: ([1-9] • [0-9]* • . • [0-9]*) + ([0-9] • . • [0-9]*)

   Binary numbers: (0 • 1) • (0 • 1)*

6. Decide whether the following two regular expressions are equivalent

(1 + a)* =? a*

(a • b)* • a =? a • (b • a)*

(1 + a)* = a* is true.

(a • b)* • a = a • (b • a)* is true.

7. Given r = (a • b + b)*. Compute derivative of r with respect to a, b, and c. Is r nullable?

der a r = der a (a·b+b) · r

(der a (a·b) + der a b) · r

(((der a a)· b) + der a b) · r

((1 · b) + 0) · r

Not nullable

der b r = der b (a·b+b) · r

(der b (a.b) + der b b) · r

(((der b a) · b) + der b b) · r

(((0 · b) + 1) · r

((0 · b) + 1) · r

nullable

der c r = der c (a·b+b) · r

(der c (a.b) + der c b) · r

(((der c a) · b + der c b) · r

(0 · b + 0) · r

((0·b) + 0) · r

Not nullable

8. Prove for all regular expressions r we have: nullable(r) if and only if [] ∈ L(r).

P(r): property that nullable(r) if and only if [] ∈ L(r)

P(0) → nullable(0) is false

the language L(0) = {} so [] is not in L(0)

so nullable(0) iff [] ∈ L(0), both sides are false

P(1) → nullable(1) is true

L(1) = {[]} so [] is in L(1)

P(r1+r2) → assume nullable(r1) iff [] ∈ L(r1)

and nullable(r2) iff [] ∈ L(r2)

nullable(r1+r2) = nullable(r1) ∨ nullable(r2)

[] ∈ L(r1) ∨ [] ∈ L(r2) so [] ∈ L(r1) ∪ L(r2)

∴ [] ∈ L(r1+r2)

P(r1·r2) → assume P(r1) and P(r2)

nullable(r1 · r2) = nullable(r1) ∧ nullable(r2)

[] ∈ L(r1) ∧ [] ∈ L(r2) = [] ∈ L(r1) @ L(r2)

P(r*) → assume P(r)

nullable(r*) = true

L(r*) = {[], r, rr...} so [] ∈ L(r*)

9. **Define what is meant by the derivative of a regular expression with respect to a character**

Der c (0) = 0
Der c (1) = 1
Der c (d) = if (c == d) 1 else 0
Der c (r1 + r2) = der c r1 + der c r2
Der c (r1 • r2) = if (nullable(r1)) der c r1 • r2 + der c r2
Else der c r1 • r2

Der c (r*) = der c r • r*

10. **Assume the set Der is defined as Der c A = {s | c :: s ∈ A}. What is the relation between Der and the notion of derivative of regular expressions?**

Der c A = L(Der c A)

11. **Give a regular expression over the alphabet {a, b} recognising all strings that do not contain any substring bb and end in a.**

(Question is quite confusing, is it
all strings that don't contain substring bb **and** don't end in a? **or**
all strings that don't contain substring bb **and** end in a?)
Assumption: I've picked the first one.

(a* • (b + 1))* • a

12. **Do (a + b)* • b^+ and (a* • b^+) + (b* • b^+) define the same language?**

No. First regular expression can match **abab,** second regular expression can't.

13. **Define a function zeroable by recursion over regular expressions.**

Zeroable(0) = true
Zeroable(1) = false
Zeroable(c) = false
Zeroable(r1 + r2) = zeroable(r1) OR zeroable(r2)
Zeroable(r1 • r2) = zeroable(r1) OR zeroable(r2)
Zeroable(r*) = false

Function nullable for the not-regular expression can be defined by:
Nullable(~r) = not (nullable(r))

Unfortunately, a similar definition for zeroable does not satisfy the property P(zeroable):
Zeroable(~r) = not (zeroable(r))

Find a counter example

Zeroable(1) = false
Zeroable(not 1) = Zeroable(c) = false, so doesn't satisfy the property P(zeroable).

14. Give a regular expression that can recognise all strings from the language $\{a^n \mid \exists k.n = 3k + 1\}$

? Does this mean a^n where n is 3k + 1 where k is a number?

15. Give a regular expression that can recognise an odd number of as or an even number of bs.

$(a \bullet (a \bullet a)^*) + (1 + (b \bullet b)^*)$