Lambda kalkul

LET T $= \lambda x\, y.x$
LET F $= \lambda x\, y.y\, x$
LET Not $= \lambda p.p\, \mathsf{F}\, (\lambda r.\mathsf{T})$
LET And $= \lambda a\, b.a\, b\, (\lambda r.\mathsf{F})$
LET Or $= \lambda a\, b.a\, \mathsf{T}\, (\lambda r.b)$

Not (And T F) =
Not $((\lambda a\, b.a\, b\, (\lambda r.\mathsf{F}))\ \mathsf{T}\, \mathsf{F}) =$
Not $(\mathsf{T}\ \mathsf{F}\ (\lambda r.\mathsf{F})) =$
Not $((\lambda x\, y.x)\ \mathsf{F}\ (\lambda r.\mathsf{F})) =$
Not F =
$(\lambda p.p\, \mathsf{F}\, (\lambda r.\mathsf{T}))\ \mathsf{F} =$
F F $(\lambda r.\mathsf{T}) =$
$(\lambda x\, y.y\, x)\ \mathsf{F}\ (\lambda r.\mathsf{T}) =$
$(\lambda r.\mathsf{T})\ \mathsf{F} =$
T

Not (Or T F) =
Not $((\lambda a\, b.a\, \mathsf{T}\, (\lambda r.b))\ \mathsf{T}\, \mathsf{F}) =$
Not $(\mathsf{T}\, \mathsf{T}\, (\lambda r.\mathsf{F})) =$
Not $((\lambda x\, y.x)\, \mathsf{T}\, (\lambda r.\mathsf{F})) =$
Not T =
$(\lambda p.p\, \mathsf{F}\, (\lambda r.\mathsf{T}))\ \mathsf{T} =$
T F $(\lambda r.\mathsf{T}) =$
$(\lambda x\, y.x)\ \mathsf{F}\ (\lambda r.\mathsf{T}) =$
F

Datové struktury

```
data BE a = Var a | Tr | Fa | Neg (BE a) | And (BE a) (BE a) | Or (BE a ) (BE a)
  deriving (Show, Eq)

nneg :: BE a -> BE a
nneg (And e1 e2) = And (nneg e1) (nneg e2)
nneg (Or e1 e2) = Or (nneg e1) (nneg e2)
nneg (Not e) = wasneg e
nneg x = x

wasneg (And e1 e2) = Or (wasneg e1) (wasneg e2)
wasneg (Or e1 e2) = And (wasneg e1) (wasneg e2)
wasneg (Not e) = nneg e
wasneg x = Not x

constF :: BE a -> BE a
constF (And e1 e2) = ev (constF e1) (constF e2)
  where
    ev Tr x = x
    ev x Tr = x
    ev Fa _ = Fa
    ev _ Fa = Fa
    ev a b = And a b
constF (Or e1 e2) = ev (constF e1) (constF e2)
  where
    ev Tr _ = Tr
    ev _ Tr = Tr
    ev Fa x = x
    ev x Fa = x
    ev a b = Or a b
constF (Not e) = ev (constF e)
  where
    ev Fa = Tr
    ev Tr = Fa
    ev x = Not x
constF x = x
```

Důkaz

```
1:   map _ [] = []
2:   map f (x:xs) = f x : map f xs

3:   (++) [] ys = ys
4:   (++) (x:xs) ys = x:(xs ++ ys)

TODO: map f (xs ++ ys) = (map f xs) ++ (map f ys)

1) xs = []
map f ([] ++ ys) =/3  map f ys
(map f []) ++ (map f ys) =/1  [] ++ (map f ys) =/3  map f ys

2) xs = (a:as)
I.H.: map f (as ++ ys) = (map f as) ++ (map f ys)
map f ((a:as) ++ ys) =/4  map f (a:(as++ys)) =/2  f a : map (as ++ ys)
(map f (a:as)) ++ (map f ys) =/2  (f a : map f as) ++ (map f ys) =/4
f a : ((map f as) ++ (map f ys)) =/I.H.  f a : (map f (as ++ ys))
```

Q.E.D.