basicexcersises

```
Function Application and Types

constants of type e : a-o s-w
variables of type t: p-r
constants of type <e,t> : P Q; display as: one-place predicate
constants of type <e*e,t> : R; display as: two-place predicate
variables of type e : x-z
variables of type <et> : X Y; display as: one-place predicate

define Sam: s
define Frodo: f
define Gimli: g
define Sauron: n

# Predicates (APs and NPs)
define hobbit,a-hobbit: Lx.hobbit(x)
define elf,an-elf:      Lx.elf(x)
define dwarf,a-dwarf:   Lx.dwarf(x)
define brave:           Lx.brave(x)
define short:           Lx.short(x)
define runs,run:        Lx.runs(x)
define trusts,trust:    LxLy.trusts(y,x)
define chases,chase:    LxLy.chases(y,x)

# Copula
define is,are: LX[X]

# Booleans and quantifiers
define it-is-not-the-case-that: Lp[~p]
define and: Lp.[Lq.[p & q]]
define or:  Lp.[Lq.[p V q]]

define every,Every: LX[LY[Ax[X(x) -> Y(x)]]]
define some,Some:   LX[LY[Ex[X(x) & Y(x)]]]
define no,No:       LX[LY[~Ex[X(x) & Y(x)]]]


use rule function application
```

```
############################
# EXERCISES
############################

exercise tree
title Function Application
directions Combine the meanings of the following expressions compositionally.

# Simple predications
[.S [.DP Frodo ] [.VP is [.NP a-hobbit ] ] ]
[.NegP it-is-not-the-case-that [.S [.DP Gimli ] [.VP is [.NP a-hobbit ] ] ] ]
[.S [.DP Sam ] [.VP is [.AP brave ] ] ]
[.S [.DP Frodo ] [.VP runs ] ]

# Transitives
[.S [.DP Frodo ] [.VP [.V trusts ] [.DP Sam ] ] ]
[.S [.DP Gimli ] [.VP [.V chases ] [.DP Sauron ] ] ]

# Truth-Functional Connectives
[.S [.S [.DP Sam ] [.VP runs ] ] [.ConjP and [.S [.DP Frodo ] [.VP runs ] ] ] ]
[.S [.S [.DP Frodo ] [.VP is [.AP brave ] ] ] [.ConjP and [.S [.DP Gimli ] [.VP is [.AP short ] ] ] ] ]
[.S [.S [.DP Frodo ] [.VP [.V trusts ] [.DP Sam ] ] ] [.ConjP or [.S [.DP Gimli ] [.VP [.V trusts ] [.DP Frodo ] ] ] ] ]
[.NegP it-is-not-the-case-that [.S [.DP Sam ] [.VP is [.AP short ] ] ] ]
[.NegP it-is-not-the-case-that [.S [.DP Frodo ] [.VP [.V chases ] [.DP Sauron ] ] ] ]

# Quantification (subject position only)
[.S [.DP Every [.NP hobbit ] ] [.VP runs ] ]
[.S [.DP Some [.NP dwarf ] ] [.VP is [.AP brave ] ] ]
[.S [.DP No [.NP elf ] ] [.VP [.V trusts ] [.DP Sauron ] ] ]
[.S [.DP Some [.NP hobbit ] ] [.VP [.V trusts ] [.DP Frodo ] ] ]

# Complex combinations (no quantifiers in object position)
[.S [.S [.DP Every [.NP hobbit ] ] [.VP is [.AP brave ] ] ] [.ConjP and [.S [.DP No [.NP elf ] ] [.VP is [.AP short ] ] ] ] ]


exercise semantic types
title Semantic Types
directions Determine the semantic type of each λ-expression.

Lx[runs(x)]
LxLy[trusts(y,x)]
Lp.[Lq.[p & q]]
```

```
LX[LY[Ax[X(x) -> Y(x)]]]
Lx.~runs(x)
LxLy[brave(x) -> trusts(y,x)]
LX[LY[Ex[X(x) & Y(x)] -> Ay[Y(y)]]]
Lp.[Lq.[~(p -> q) & (p & q)]]
LxLy[Lz[chases(z,x)] -> brave(y)]
LX[LY[~Ex[X(x) & Y(x)]]]
```