eventsemantics-1

```
Davidsonian and Neo-Davidsonian Events

constants of type e : a b g l m o u h w
variables of type t : p q
constants of type <e,t> : P Q; display as: one-place predicate
variables of type e : x y z
variables of type v : e
variables of type <e,t> : X Y; display as: one-place predicate
variables of type <e,<v,t>> : V
variables of type <v,t> : U

multiple letter identifiers

use rule function application
use rule non-branching nodes
use rule predicate modification
use rule lambda abstraction

define Aragorn: a
define Boromir: b
define Gimli: g
define Legolas: l
define Merry: m
define the-orc: o
define Ugluk: u
define HelmsDeep: h
define Anduril: w

define dwarf,dwarves: Lx.dwarf(x)
define elf,elves: Lx.elf(x)
define hobbit,hobbits: Lx.hobbit(x)
define orc,orcs: Lx.orc(x)
define ranger: Lx.ranger(x)
define king: Lx.king(x)
define captain: Lx.captain(x)
define warrior: Lx.warrior(x)

define brave:     Lx.brave(x)
define tall:      Lx.tall(x)
define short:     Lx.short(x)
define elven:     Lx.elven(x)
define dwarven:   Lx.dwarven(x)
define gondorian: Lx.gondorian(x)

define experienced: LX.Lx.experiencedas(X,x)
define alleged:     LX.Lx.allegedas(X,x)
define former:      LX.Lx.former(X,x)
define fake:        LX.Lx.fake(X,x)

define runs,run:       Lx.Le.runs(x,e)
define sprints,sprint: Lx.Le.sprints(x,e)
define chases,chase:   Lx.Ly.Le.chases(y,x,e)
define fears,fear:     Lx.Ly.Le.fears(y,x,e)
define kills,kill:     Lx.Ly.Le.kills(y,x,e)

define runE:           Le.runv(e)
```

```
define sprintE:        Le.sprintv(e)
define chaseE:         Le.chasev(e)
define fearE:          Le.fearv(e)
define killE,killedE:  Le.killv(e)

define slowly:  LV.[Lx.[Le.[V(x)(e) & slowly(e)]]]
define quickly: LV.[Lx.[Le.[V(x)(e) & quickly(e)]]]

define in:   Lz.LV.[Lx.[Le.[V(x)(e) & in(e,z)]]]
define with: Lz.LV.[Lx.[Le.[V(x)(e) & with(e,z)]]]

define slowlyE: Le.slowly(e)
define quicklyE: Le.quickly(e)
define inE:     Lz.Le.in(e,z)
define withE:   Lz.Le.with(e,z)

define Agent: Lx.Le.agent(e,x)
define Theme: Lx.Le.theme(e,x)

define E: LU.Ee.U(e)

define every,Every,all: 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)]]]

define is,are: LX.Lx.X(x)

define and: Lp.[Lq.[p & q]]
define or:  Lp.[Lq.[p V q]]

###################################
# SECTION 1: Semantic Types
###################################

exercise semantic types
title Semantic Types with Events and Adverbs
directions Give the semantic type of each of the following lambda-expressions.

Lx.Le.runs(x,e)
Lx.Ly.Le.kills(y,x,e)
LV.[Lx.[Le.[V(x)(e) & slowly(e)]]]
LU.Ee.U(e)
LX.Lx.experiencedas(X,x)
LX[LY[Ax[X(x) -> Y(x)]]]

###################################
# SECTION 2: Davidsonian Events
###################################

exercise tree
title Davidsonian Events
directions Each tree includes an existential closure head E at the top. Use function application and non-branching nodes to compute the truth conditions.

instructions Gimli runs.
[.S' E [.S [.DP Gimli ] [.VP [.V runs ] ] ] ]

instructions Gimli runs slowly.
[.S' E [.S [.DP Gimli ] [.VP [.VP [.V runs ] ] [.AdvP slowly ] ] ] ]
```

```
instructions Legolas runs quickly in Helm's Deep.
[.S' E [.S [.DP Legolas ] [.VP [.VP [.VP [.V runs ] ] [.AdvP quickly ] ] [.AdvP [.P in ] [.DP HelmsDeep ] ] ] ] ]

instructions Aragorn slowly kills the-orc.
[.S' E [.S [.DP Aragorn ] [.VP [.AdvP slowly ] [.VP [.V kills ] [.DP the-orc ] ] ] ] ]

instructions Merry slowly sprints.
[.S' E [.S [.DP Merry ] [.VP [.VP [.V sprints ] ] [.AdvP slowly ] ] ] ]

##################################
# SECTION 3: Neo-Davidsonian Events
##################################

exercise tree
title Neo-Davidsonian Events
directions Here verbs are predicates of events <v,t>, and participant roles and adverbs are also predicates of events. Use predicate modification to conjoin event predicates, and E at the top for existentia

instructions Merry runs.
[.S' E [.S [.AgentP [.Agent Agent ] [.DP Merry ] ] [.VP [.V runE ] ] ] ]

instructions Aragorn killed the-orc.
[.S' E [.S [.AgentP [.Agent Agent ] [.DP Aragorn ] ] [.VP [.VP [.V killE ] [.ThemeP [.Theme Theme ] [.DP the-orc ] ] ] ] ] ]

instructions Merry runs slowly.
[.S' E [.S [.AgentP [.Agent Agent ] [.DP Merry ] ] [.VP [.VP [.V runE ] ] [.AdvP slowlyE ] ] ] ]

instructions Gimli runs in Helm's Deep.
[.S' E [.S [.AgentP [.Agent Agent ] [.DP Gimli ] ] [.VP [.VP [.V runE ] ] [.AdvP [.P inE ] [.DP HelmsDeep ] ] ] ] ]

instructions Legolas killed the-orc with Anduril.
[.S' E [.S [.AgentP [.Agent Agent ] [.DP Legolas ] ] [.VP [.VP [.V killE ] [.ThemeP [.Theme Theme ] [.DP the-orc ] ] ] [.AdvP [.P withE ] [.DP Anduril ] ] ] ] ]

##################################
# SECTION 4: Davidsonian Events + Quantifier Raising
##################################

exercise tree
title Davidsonian Events with Quantifier Raising
directions These trees use Davidsonian event semantics, existential closure (E above S), and quantifier raising with LP, index, and trace. Compute the truth conditions using FA, lambda abstraction, predicat

instructions No orc runs.
[.S [.DP no [.NP orc ] ] [.LP 1 [.S' E [.S [.DP t_1 ] [.VP [.V runs ] ] ] ] ] ]

instructions Every brave dwarf runs slowly.
[.S [.DP Every [.NP [.AP brave ] [.NP dwarf ] ] ] [.LP 1 [.S' E [.S [.DP t_1 ] [.VP [.VP [.V runs ] ] [.AdvP slowly ] ] ] ] ] ]

instructions Some dwarf quickly chases Legolas in Helm's Deep.
[.S [.DP Some [.NP dwarf ] ] [.LP 1 [.S' E [.S [.DP t_1 ] [.VP [.VP [.VP [.V chases ] [.DP Legolas ] ] [.AdvP quickly ] ] [.AdvP [.P in ] [.DP HelmsDeep ] ] ] ] ] ] ]

instructions Every tall elf slowly kills the-orc.
[.S [.DP Every [.NP [.AP tall ] [.NP elf ] ] ] [.LP 1 [.S' E [.S [.DP t_1 ] [.VP [.AdvP slowly ] [.VP [.V kills ] [.DP the-orc ] ] ] ] ] ] ]

##################################
# SECTION 5: Neo-Davidsonian Events + Quantifier Raising
##################################

exercise tree
title Neo-Davidsonian Events with Quantifier Raising
directions Now combine Neo-Davidsonian event predicates with quantifier raising and existential closure. Use predicate modification for event predicates, lambda abstraction for QR, and E at the top.
```

```
instructions Every dwarf runs.
[.S [.DP Every [.NP dwarf ] ] [.LP 1 [.S' E [.S [.AgentP [.Agent Agent ] [.DP t_1 ] ] [.VP [.V runE ] ] ] ] ] ]

instructions Some brave elf quickly sprints.
[.S [.DP Some [.NP [.AP brave ] [.NP elf ] ] ] [.LP 1 [.S' E [.S [.AgentP [.Agent Agent ] [.DP t_1 ] ] [.VP [.VP [.V sprintE ] ] [.AdvP quicklyE ] ] ] ] ] ]

instructions No alleged orc slowly kills the-orc.
[.S [.DP no [.NP [.AP alleged ] [.NP orc ] ] ] [.LP 1 [.S' E [.S [.AgentP [.Agent Agent ] [.DP t_1 ] ] [.VP [.AdvP slowlyE ] [.VP [.V killE ] [.ThemeP [.Theme Theme ] [.DP the-orc ] ] ] ] ] ] ] ]
```