Slide 1)

type SPred a = SF a Bool

notSF sf = sf ≫ arr ( **¬** )

andSF sf1 sf2 = (sf1 &&& sf2) ≫ arr (uncurry (∧))

orSF sf1 sf2 = (sf1 &&& sf2) ≫ arr (uncurry (∨))

implySF sf1 sf2 = orSF sf2 (notSF sf1)

Slide 2)

history :: SPred a → SPred a

history sf = loopPre True $ proc (a, last) →

b ← sf −≺ a

**let** cur = last ∧ b

returnA −≺ (cur,cur)

ever :: SPred a → SPred a

ever sf = loopPre False $ **proc** (a, last) →

b ← sf −≺ a

**let** cur = last ∨ b

returnA −≺ (cur,cur)

Slide 3)

**type** AssertionId = String

**type** DebuggingMonadT = WriterT [(String,DTime)]

assert :: String → SF (DebuggingMonadT m) Bool ()

assert assertionId = **proc** (val) → **do**

t ← localTime −≺ ()

**let** optionallyLog (t1, v1) = when v1 (tell (assertionId, t1))

() ← withSideEffect optionallyLog −≺ (t, val)

returnA −≺ ()

ballAboveFloorM :: SF (DebuggingMonadT m) () Double

ballAboveFloorM = **proc** () → **do**

ballPos ← bouncingBall −≺ ()

() ← assert "Ball must always be above the floor" −≺ (ballPos ⩾ 0)

returnA −≺ ballPos