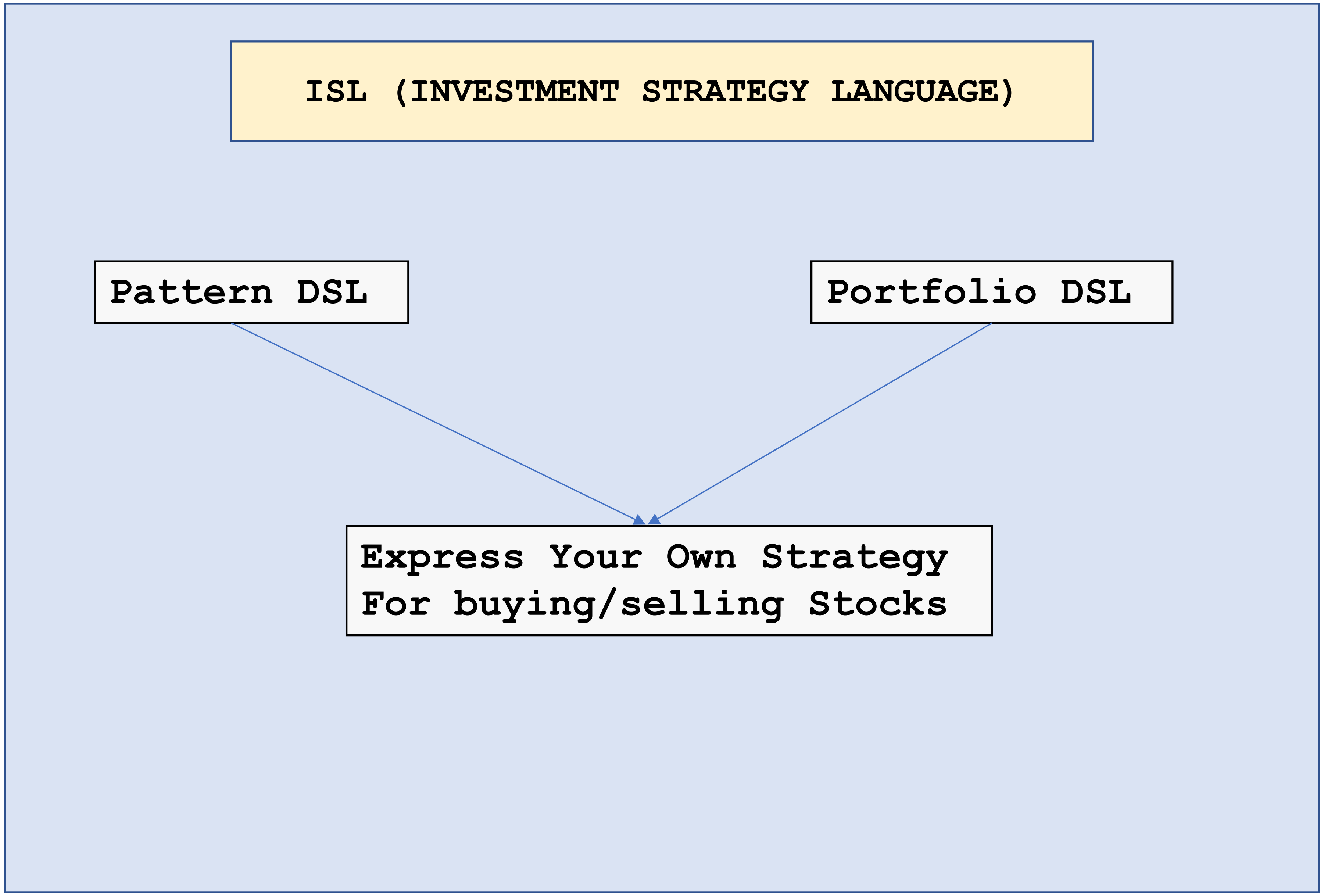
ISL (Investment Strategy Language)

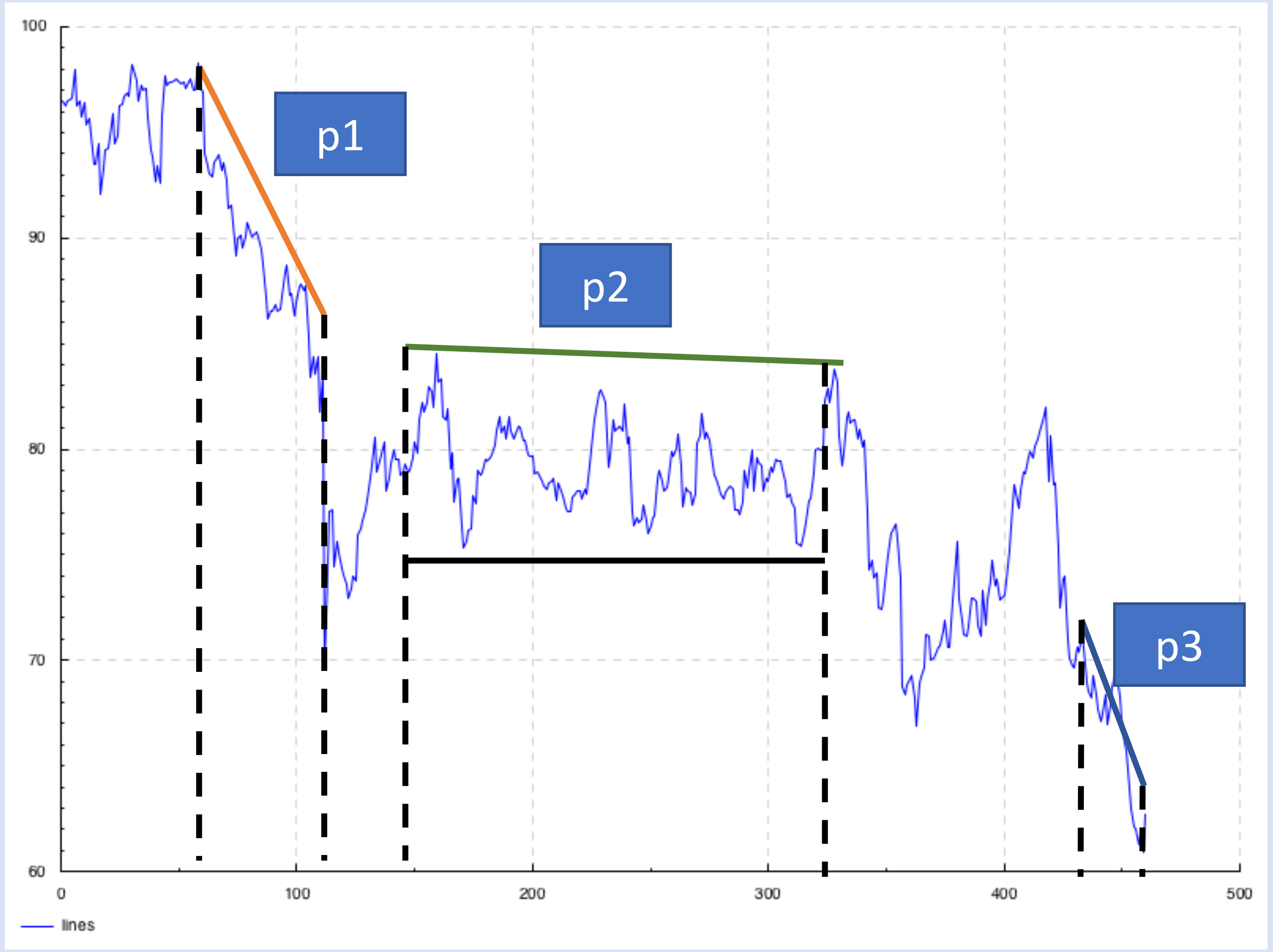
By

Harish

Content


ISL (INVESTMENT STRATEGY LANGUAGE)
Pattern DSL
Portfolio DSL
Express Your Own Strategy For buying/selling Stocks

Pattern DSL



Pattern DSL

```haskell
data Pat = Bas Dir TimeStamp TimeStamp
         | And Pat Pat                -- Both Patterns
         | Or Pat Pat                 -- Or Pattern
         | IfElse Cond Pat Pat -- Cond Pattern

data Dir = Up | Down | Cons | NoPat
```

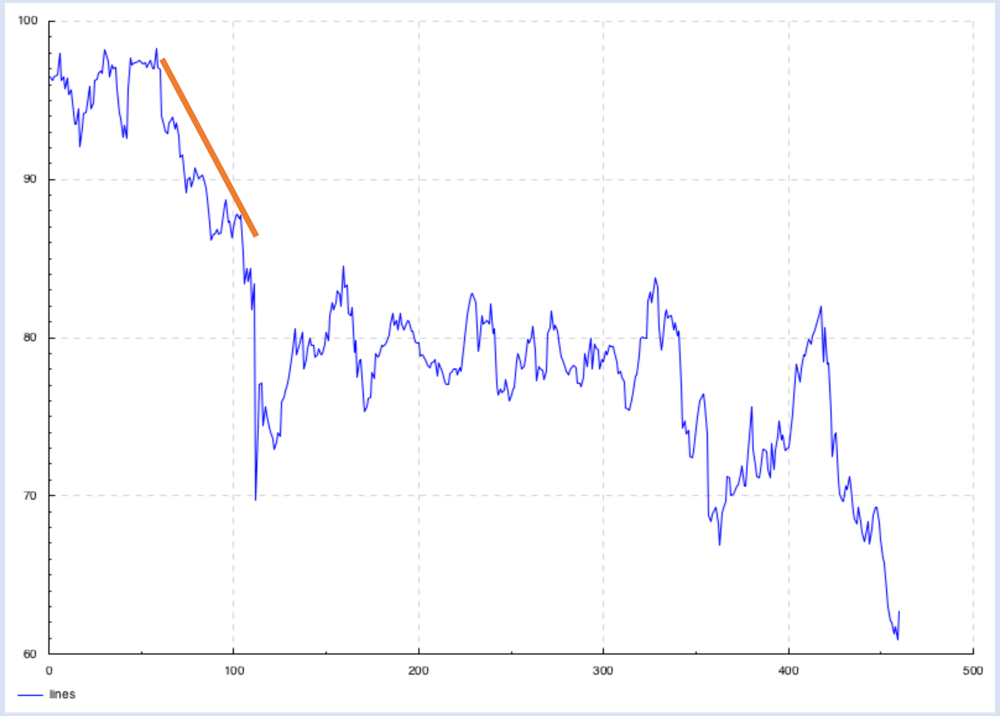
```haskell
type TimeSeries a = TimeStamp -> a
```

Properties of Pattern

- They are Operators to Time Series data
- Patterns are Time Invariant

Properties of Pattern

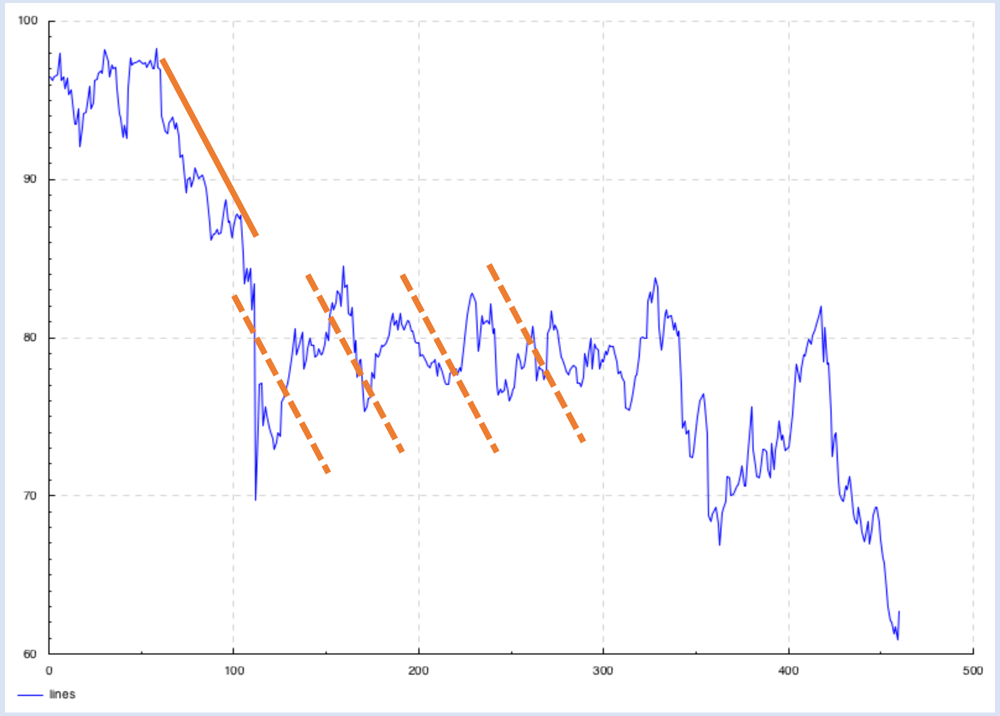
- They are Linear Operator operator to Time Series



Check if Pattern Matches

Properties of Pattern

- **Time Invariant Property**

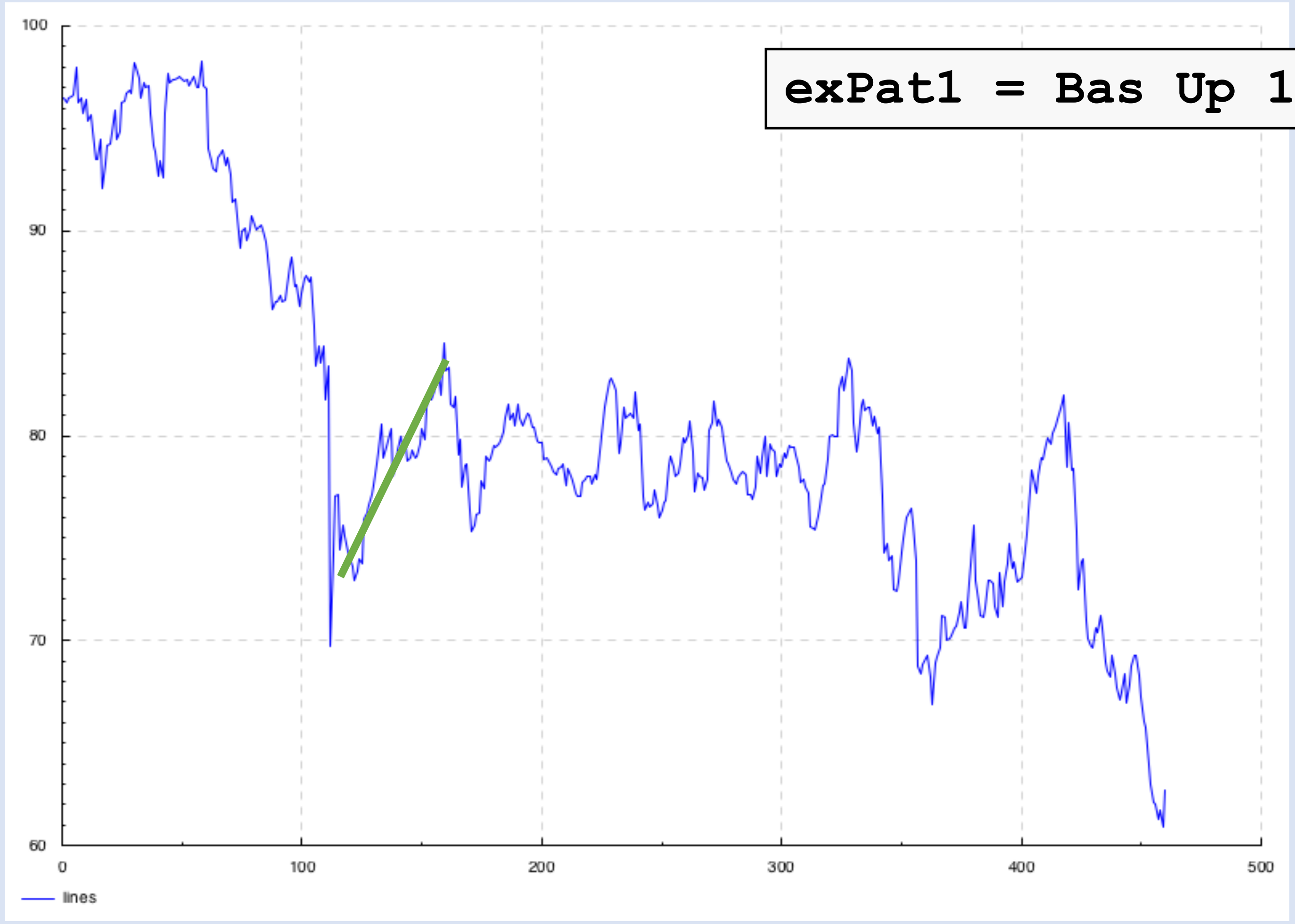


Time Invariant, Contains the Information of Pat

Pattern Algebra

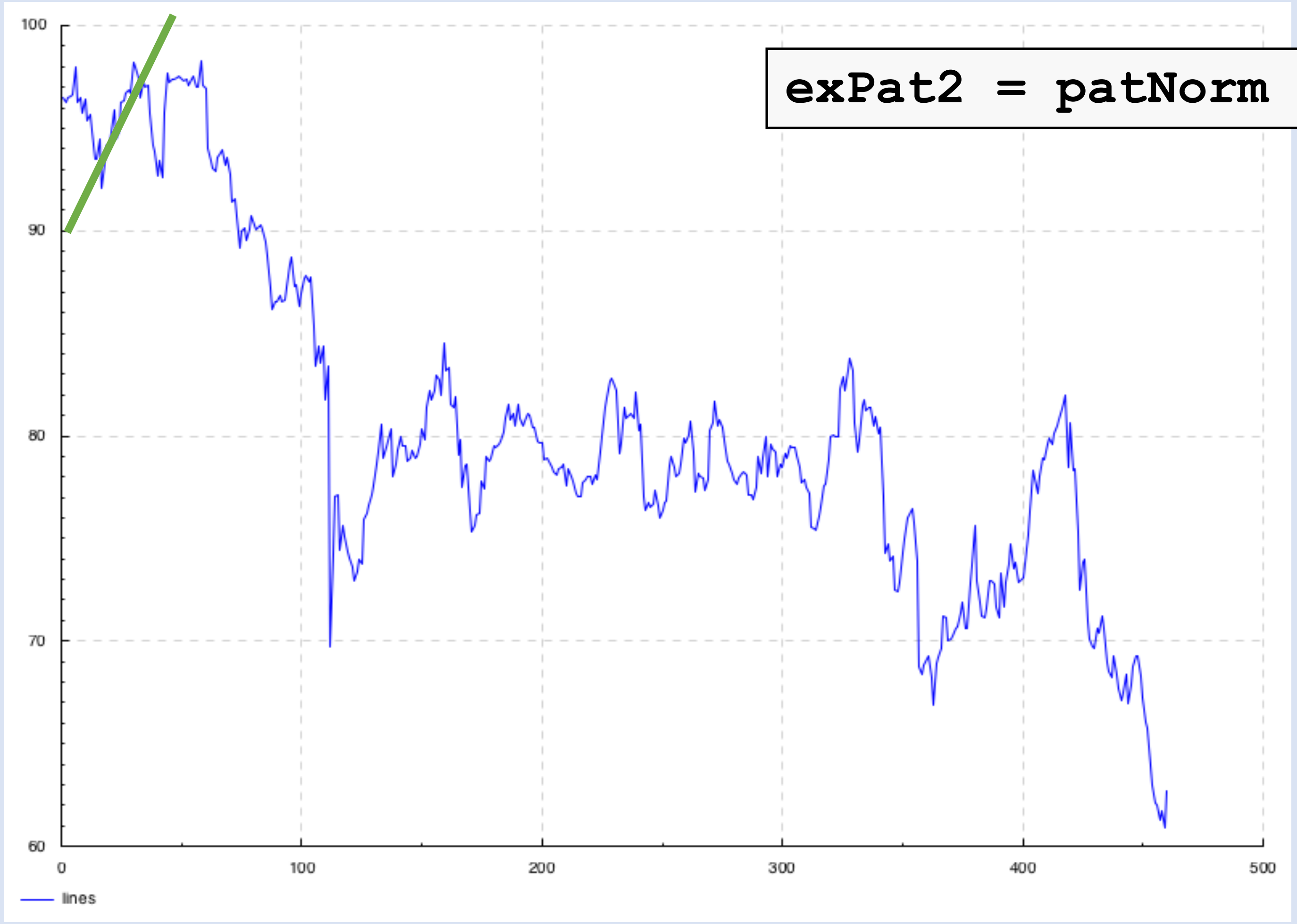
```
Let ‘p1’ be a Pat
Let ‘p2’ be a Pat

Operations on Patterns

• p1 + p2 = p1 and p2            (patAdd      :: Pat -> Pat -> Pat)
• -p1     = Up ’to’ Down         (patMin      :: Pat -> Pat -> Pat)
• Norm p1 = p3                   (patNorm     :: Pat -> Pat)
• p1 + t  = p2                  (shiftPattern:: Pat -> TimeDelta -> Pat)
• p1 ‘equals’p2  = (Norm p1)== (Norm p2)
                          (patEquals :: Pat -> Pat -> Bool)
```

Examples 1 : Basic Pattern



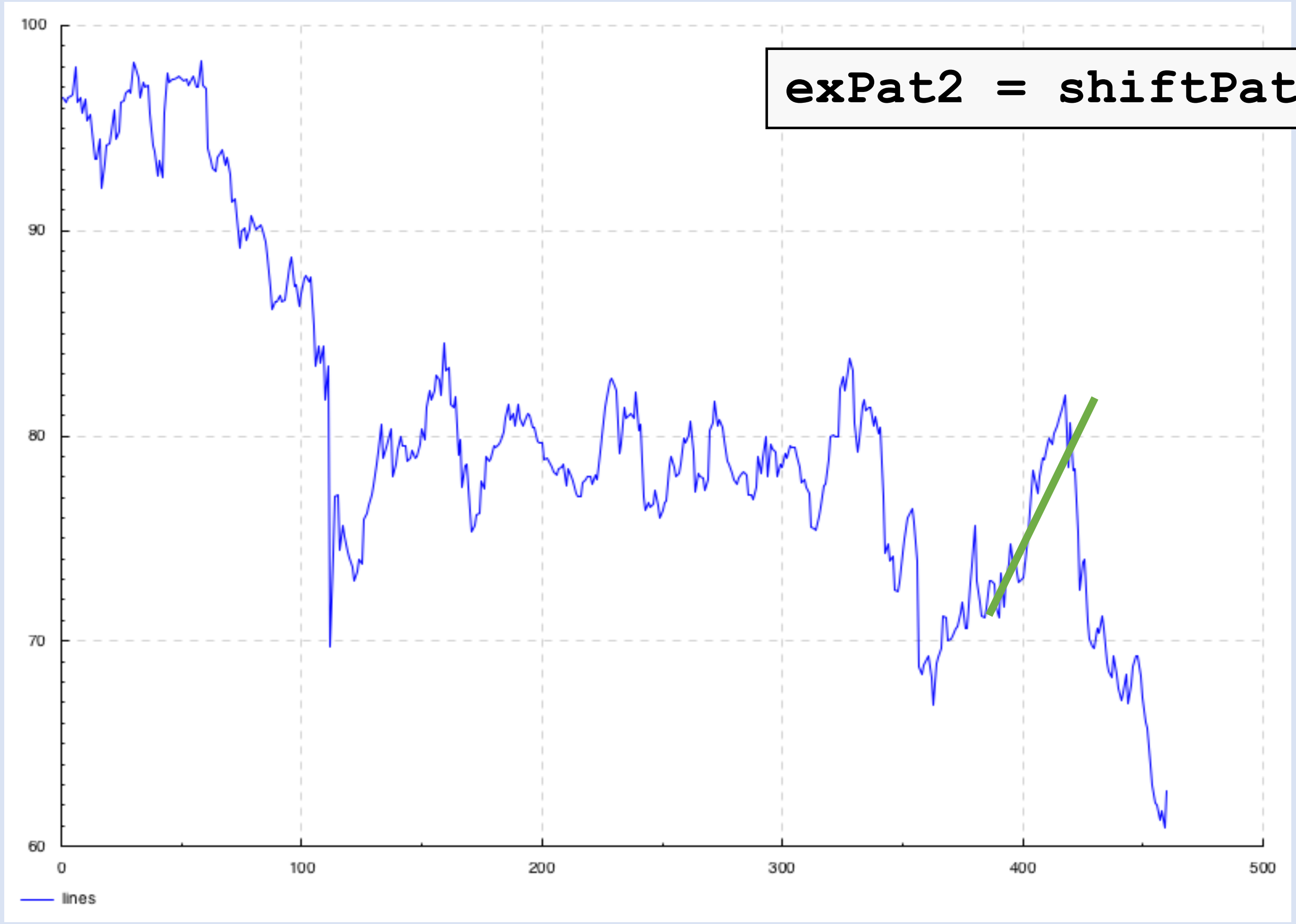
```
exPat1 = Bas Up 110 150
```

Examples 2 : Basic Pattern



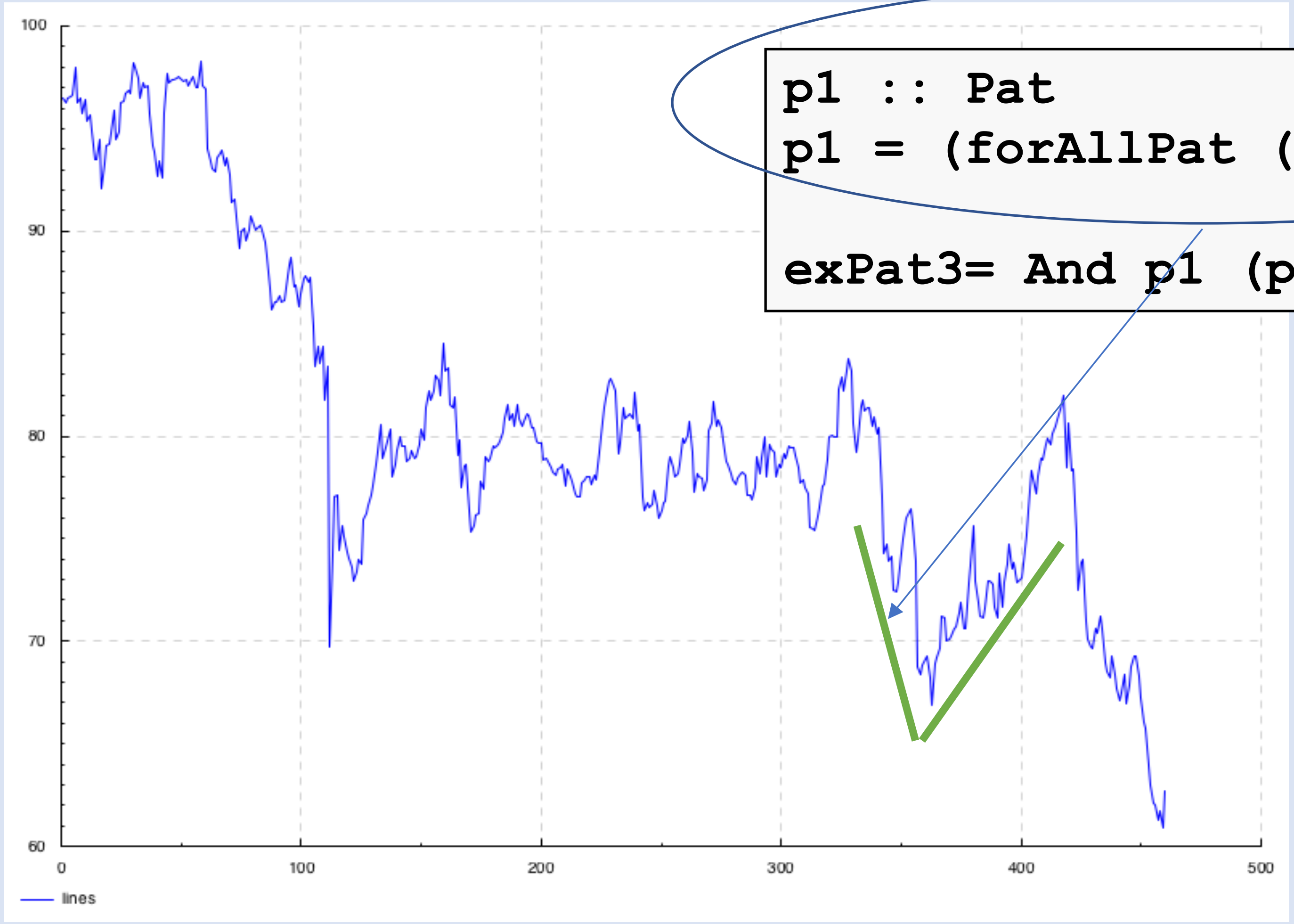
```
exPat2 = patNorm (exPat1)
```

Examples 2 : Shift Pat



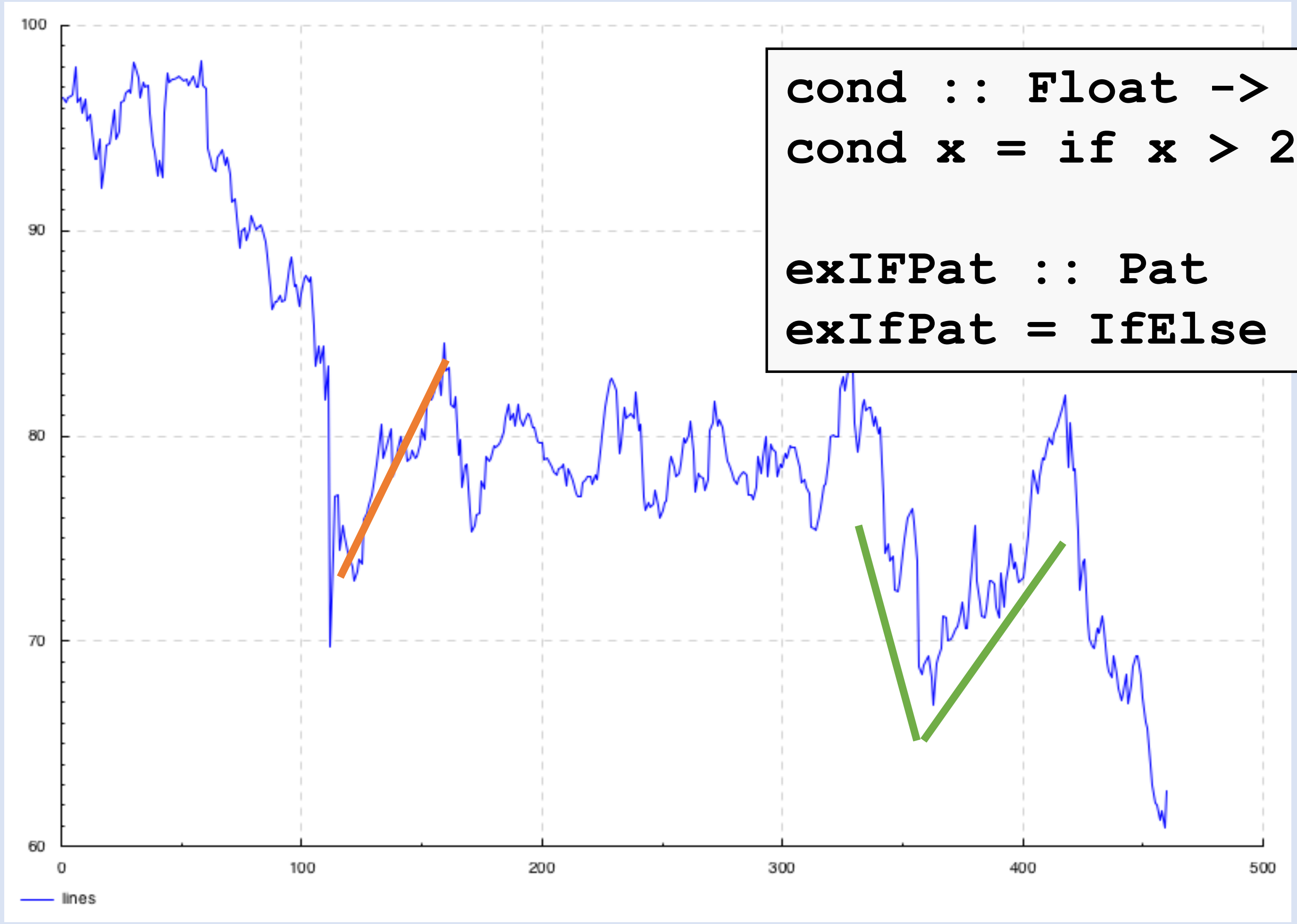
```
exPat2 = shiftPattern (patNorm (exPat1)) 370
```

Examples : foraExample



```
p1 :: Pat
p1 = (forAllPat (320,350) 5 (Down))

exPat3= And p1 (patMin (shiftPattern p1 50) )
```

Examples : IfElse Example



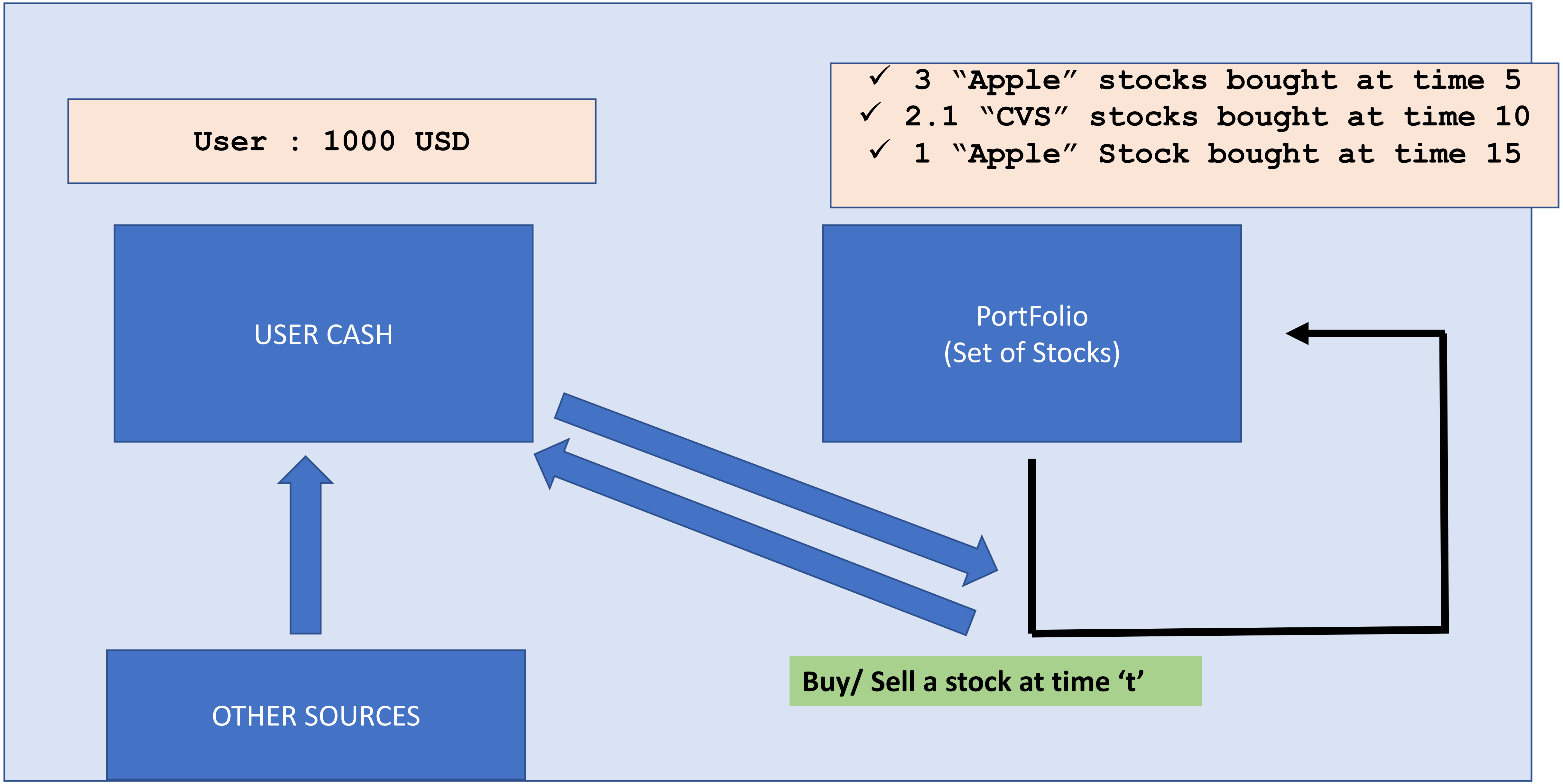
```
cond :: Float -> Bool
cond x = if x > 2000000 then True else False

exIFPat :: Pat
exIfPat = IfElse (cond x ) (exPat1) (exPat3)
```

Semantic Domain

```
match :: Pat -> TimeSeries a -> Bool

Semantic Function

•  S[p1 + p2,ts]              = S[p1,ts] and S[p2,ts]
•  S[-p1,ts]                  = S[-p1,ts] -- Space problem to expand
•  S[IfElse cond p1 p2,ts]    = If cond then S[p1,ts] else S[p2,ts]
•  S[And p1 p2,ts]            = S[p1,ts] and S[p2,ts]
•  S[Or p1 p2,ts]             = S[p1,ts] or S[p2,ts]
•  S[Bas Up t1 t2,ts]         =
•  S[Bas NoPat t1 t2,ts]      = True
```

Semantic Domain

```
matchPat :: Pat -> TimeSeries a -> Bool
--match a single Pattern


lift2Match :: TimeDelta -> Pat -> TimeSeries a -> TimeStamp ->
[(TimeStamp,TimeStamp)]
--matching if a pattern for a moving window pattern till an end
time(t)


lift2PatMatch :: TimeDelta -> [Pat] -> TimeSeries a -> TimeStamp ->
[[(TimeStamp,TimeStamp)]]
--matching a moving list of pattern till an end time (t)
```


User : 1000 USD
✓ 3 "Apple" stocks bought at time 5
✓ 2.1 "CVS" stocks bought at time 10
✓ 1 "Apple" Stock bought at time 15
USER CASH
PortFolio
(Set of Stocks)
OTHER SOURCES
Buy/ Sell a stock at time 't'

Shallow DSL = Investment Language

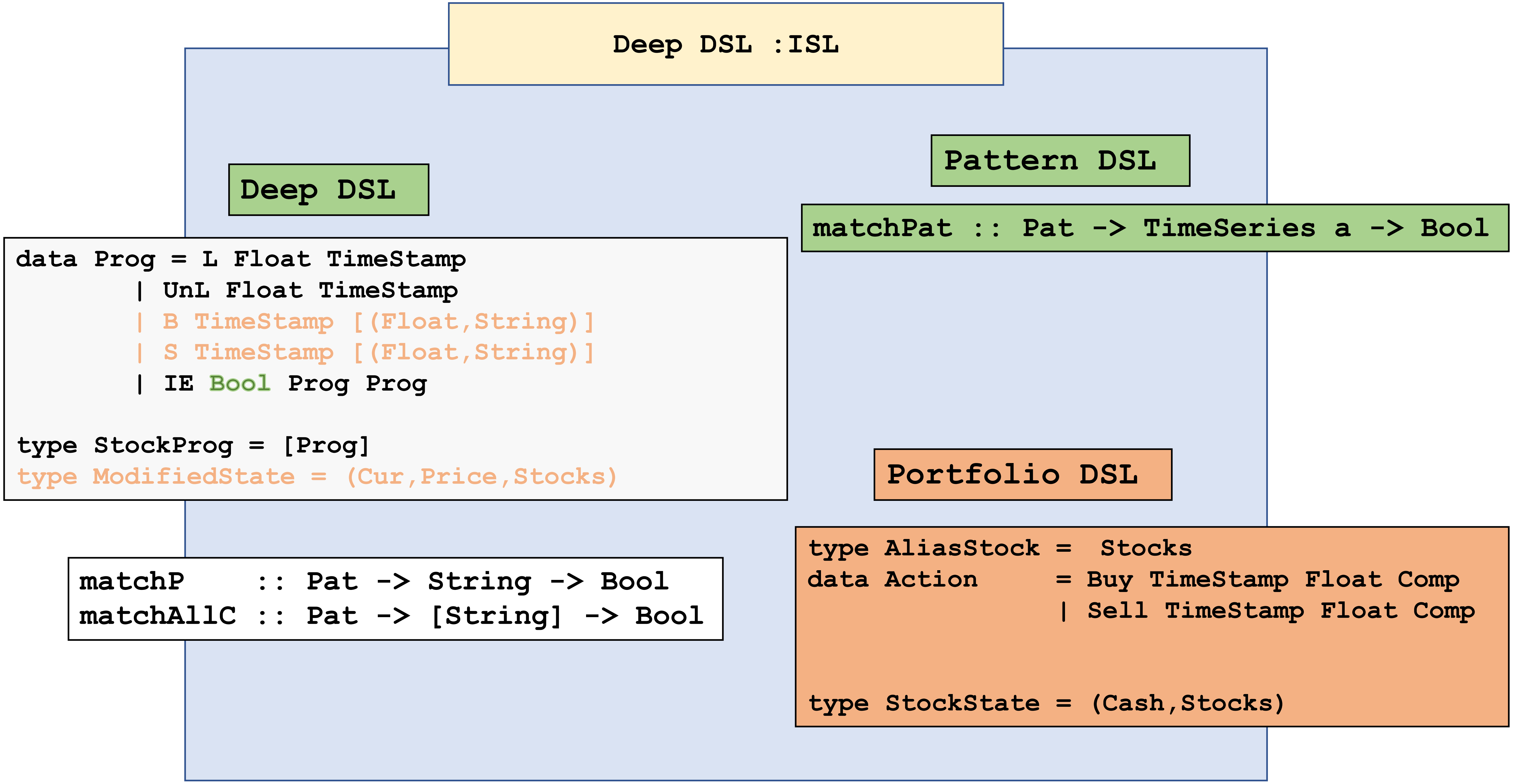
```
data Stocks = Zero
            | Stock TimeStamp Comp
            | Scale Float Stocks
            | A Stocks Stocks


data Cash      = C Cur Price

Comp = C1 | C2 | C3 | C4

Cur = USD | EURO | INR
```

```
data Action = Buy TimeStamp Float Comp
             | Sell TimeStamp Float Comp
type StockState = (Cash,Stocks)
```

Important Function def. for Stocks DSL

```
sellStock :: (Cash,Stocks) -> Action-> (Cash,Stocks)
--sellStock = sells a company(C) stocks at t based on
the max profit at 't'
--Scenario =
--3 "Apple" stocks bought at time 5
--1 "Apple" Stock bought at time 15

buyStock  :: (Cash,Stocks) -> Action -> (Cash,Stocks)
--buyStock = buys stock at time t for company c
```

```
data Action = Buy TimeStamp Float Comp
             | Sell TimeStamp Float Comp
type StockState = (Cash,Stocks)
```



Deep DSL :ISL

Deep DSL

```
data Prog = L Float TimeStamp
          | UnL Float TimeStamp
          | B TimeStamp [(Float,String)]
          | S TimeStamp [(Float,String)]
          | IE Bool Prog Prog

type StockProg = [Prog]
type ModifiedState = (Cur,Price,Stocks)
```

```
matchP    :: Pat -> String -> Bool
matchAllC :: Pat -> [String] -> Bool
```

Pattern DSL

```
matchPat :: Pat -> TimeSeries a -> Bool
```

Portfolio DSL

```
type AliasStock  =  Stocks
data Action      = Buy TimeStamp Float Comp
                 | Sell TimeStamp Float Comp


type StockState = (Cash,Stocks)
```

Example : Investment

1. Load 1000 USD at time 2
2. unLoad 100 USD at time 3
3. Buy 2 Stocks of ’Apple’ and 1 stock ’CVS’ at time 4
4. If (‘Apple’ has shiftPattern(patNorm exPat3) 15) then (Buy 2 Stocks of ‘Apple’at time 25) else (sell 1 stock of ’Apple’ 25)

```
[
   L 1000.0 2,
   UnL 100.0 3,
   B 4 [(2.0,"CVS"),(1.0,"Apple")],
   IE False (B 25 [(2.0,"Apple")]) (B 40[(2.0,"Apple")])
]
```

Conclusion : Future Work

- ✓ Improving the Syntax
- ✓ Adding Observations in general

Thanks

Any Questions ☺