
Programming

Expressions

- Arithmetic
- Continuous
- Classified
- Relation and Boolean
- Combinational
- Probability Distributions and Density Functions
- Region and Spatial
- Control
- Output
- Bit-Vector
- Matrix

Arithmetic Expressions

Expression + Expression
Expression ^ Expression (exponent)
Expression % Expression (modulo)

Example:

```
X = (StandAge + 10) / (Year % 100)
Y = (Angle1 + Angle2) % 360
      (this second line give you angles
  between 0 and 360)
```

Continuous Expressions

EXP(Expression) ***- base is e***

LOG(Expression) ***- natural logarithm***

Example:

```
x = EXP(LOG(t))
```

Spatial Expressions

DIRECTION(StartLocation, EndLocation)
DISTANCE(StartLocation, EndLocation)

Example:

```
Alpha = DIRECTION(Location, ClusterStartLoc)
X = COS(Alpha)
```

Control Expressions

IF Expression THEN Expression ELSE Expression

IF Expression
…
ENDFN

IF Expression
…
ELSE
…
ENDFN

Control Expressions

```
WHILE Expression
…
ENDFN
```

```
OVER INDEX SEQUENCE(Start, End)
   x = Index
   …
ENDFN
```

Probability Distributions

Draw a number from a distribution

NORMAL(Mean, StandardDeviation)
LOG NORMAL(Mean, StandardDeviation)
NEGEXP(Mean)
UNIFORM(Min, Max)
POISSON(Lambda)

Example:

```
x = NORMAL(10,5)
makeChoice = p < UNIFORM(0,1)
```

Density Functions

NORMAL PDF(X, Mean, StandardDeviation)
NORMAL CDF(X, Mean, StandardDeviation)

LOG NORMAL PDF(X, Mean, StandardDeviation)
LOG NORMAL CDF(X, Mean, StandardDeviation)

Bounding Expressions

MIN(Expression, Expression)

MAX(Expression, Expression)

CLAMP(Expression, MinExpression, MaxExpression) (keeps the expression within the min and max expressions)

ROUND(Expression) (rounds the expression up or down to the nearest whole number)

FLOOR(Expression) (rounds the expression down to the nearest whole number)

CEIL(Expression) (rounds the value up to the nearest whole number)

| Expression |

Example:

```
StandAge = MIN(StandAge+1, MaxStandAge)
Diff = | x - ROUND(y) |
```

TRIGONOMETRY

SIN(Expression)
COS(Expression)
TAN(Expression)

ARCSIN(Expression)
ARCCOS(Expression)
ARCTAN(Expression)
ARCTAN(Expression, Expression)

Example:

```
Alpha = ARCSIN(x)
```

Output Expressions

```
DISPLAY
   label: Expression
   varName
   …
ENDFN

OUTPUT RECORD(OutputVariable)
 label: Expression
 varName
Label#n1:n2#:var[Index]
   …
ENDFN
```

Output Expressions

OUTPUT RECORD(OutputVariable)
label: $Variable
label: $Variable {LAYER}
label: $Variable {LegendVector}
ENDFN

Example:

```
OUTPUT RECORD(f)
 Soil: $Soil
 newSoil: $x {Soil}
ENDFN
```

Relation Expressions

Expression < Expression
Expression <= Expression
Expression EQ Expression
Expression NEQ Expression
Expression <= Expression <= Expression

Example:

```
isAvailable = (Age > MinAge)
```

Boolean Expressions

Expression AND Expression
Expression OR Expression
NOT Expression

Example:

```
isAvailable = (Age > MinAge) AND (Species NEQ
  Decid)

isAvailable = AND
                Age > MinAge
                Species NEQ Decid
              ENDFN
```

Region Expressions

REGION WHOLE MAP
DECISION Expression

Example:

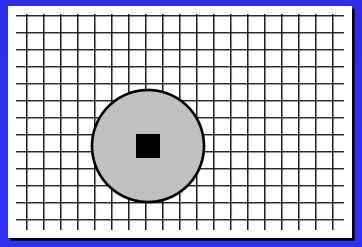
```
EVENTLOCATION
    STATIC REGION WHOLE MAP
        DECISION BEC > 0
ENDEL
```

Region Expressions

REGION CENTRED(Min, Max, options)
DECISION Expression

Example:

```
SPREADLOCATION
    REGION CENTRED(1,1.5, EUCLIDEAN, WRAPPED)
ENDFN
```



Region Expressions

REGION RECT(Bottom, Left, Top, Right)

DECISION Expression

Region Expressions

REGION VECTOR(StartLocation, EndLocation)
DECISION Expression

- follow cells along closest straight line

Region Expressions

REGION LOCATION LIST[X]
DECISION Expression

- use a set of pre-computed locations

Region Expressions

REGION COST SURFACE(endLoc, MaxCost, CostSurface, LeastCostNeighbs, AnchorLoc)

COST costExpr

- ***create a cost surface. Stop spreading when either "endLoc" reached or "MaxCost" achieved***
- ***Use "CostSurface" layer to store cost values***
- ***Use "LeastCostNeighbs" surface to store gradient***
- ***Use "AnchorLoc" to store "turn points"***

Region Expressions

REGION LEAST COST PATH(startLoc, endLocation, LeastCostNeighbs, AnchorLoc)

DECISION Expression

- ***follow a least-cost path from start to end***
- ***leastCostNeighbs and AnchorLoc from Cost Surface (with "endLocation" at the start of the cost surface)***

Region Expressions

OVER REGION …
Expression
…
ENDFN

Example:

```
OVER REGION WHOLE MAP
   DECISION BEC > 0
   StandAge = 0
   ITG = DouglasFir
ENDFN
```

Region Expressions

Change current location temporarily

AT LOCATION(Location)

…

ENDFN

Combinational Expressions

KEYWORD

Expression

…

ENDFN

AND, OR, SUM, PRODUCT

MIN, MAX, MEAN

Combinational Expressions

Example:

```
PRODUCT
    x + 5
    y / 25 - 7
   LOG(z)
ENDFN
```

Classified Expressions

CLASSIFY(Variable)

$value_i$: $Expression_i$

…

ENDFN

INTERPOLATE(Variable)

$value_i$: $Expression_i$

…

ENDFN

Probability Distributions

Draw a number from a distribution

CLASSIFIED_DIST[VectorVariable]

- ***allows drawing from an arbitrary input distribution***

Example:

```
x = CLASSIFIED_DIST[inputDist]
```

Probability Distributions

Setting and getting seed

SEED(Value)

Example:

```
newSeed = SEED(initialSEED)
```

Classified Discrete Distributions

CLASSIFIED_DIST

value$_i$: Expression$_i$

...

ENDFN

Example:

```
x = CLASSIFIED_DIST
        1: 0.1
        2: 0.2
        3: 0.2
      ENDFN
```

Example 2:

```
SurvProb = CLASSIFIED_DIST
                HAB_A = 0.2
                HAB_B = 0.8
                HAB_C = 0.1
           ENDFN
```

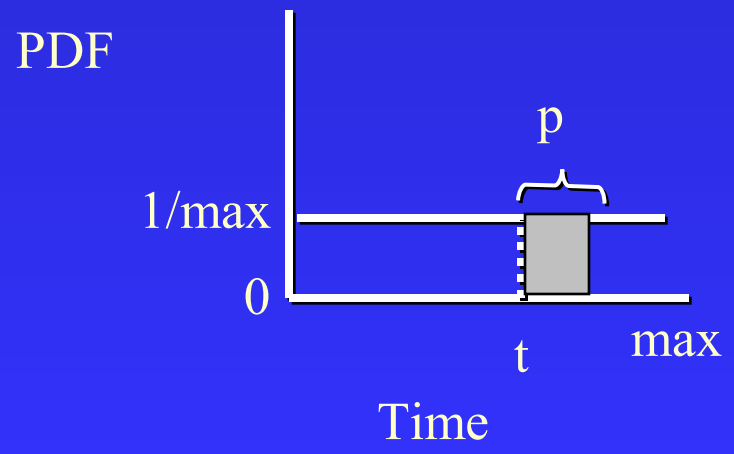
Temporal Density Functions

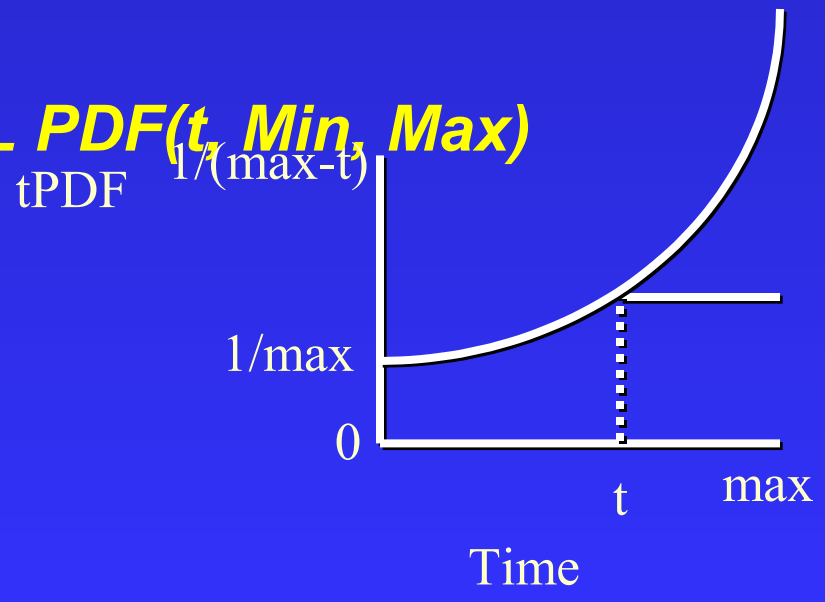
Conditional distribution, where t is time, and

Condition is: "given than a value has not yet been selected"

Designed for incremental testing of values from a distribution

UNIFORM TEMPORAL PDF(t, Min, Max)

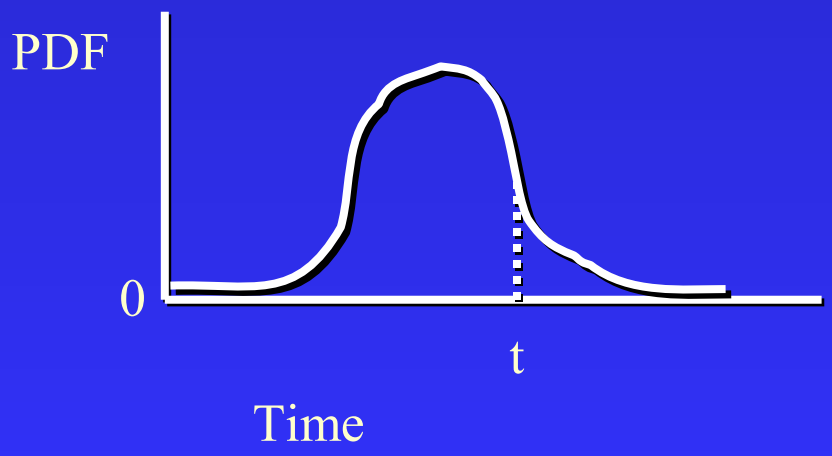




Temporal Density Functions

NORMAL TEMPORAL PDF(Value, Mean, StandardDeviation)

LOG NORMAL TEMPORAL PDF(Value, Mean, StandardDeviation)



Voir les prochains .lse et les Tâches B

Bit-Vector Expressions

For interpreting an integer variable as a string of 32 bits
- positions indexed from 1 from the right

SELECTAT(Variable, PositionExpression)
SETAT(Variable, PositionExpression, Value)
SETAT(Variable, PositionExpression, Value, ProbabilityExpression)
SHIFT LEFT(Variable, NumPositions)

Example:

```
X = 0
X = SETAT(X, 3, 1)
```

X has value 8 (0x0008 or 0000….0100)

Bit-Vector Expressions

BITWISE AND(Variable1, Variable2)
BITWISE OR(Variable1, Variable2)
BITWISE XOR(Variable1, Variable2)
BITWISE NOT(Variable)
MAX POSITION(Variable)
MIN POSITION(Variable)

Example:

```
X = BITWISE AND(Y, Z)
```

Matrix Expressions

X [=] MatrixExpression

X [=] MatrixExpression + MatrixExpression
X [=] MatrixExpression * MatrixExpression
X [=] TRANSPOSE(MatrixExpression)
X [=] INVERT(MatrixExpression)

Example:

```
X [=] M * N
X [=] M * c
```

Debugging Expressions

PAUSE IF Expression
PAUSE
PAUSE(milliseconds)

DEBUG

Example:

```
X = (StandAge + 10) / (Year % 100)
```

Set, List, Tree and Graph Functions

REMOVE ALL(S)

n = SIZE(S)

b = IS EMPTY(S)

Pos = FIRST(S)

Pos = NEXT(S, *Pos*)

Pos = PREV(S, *Pos*)

REMOVE(S, *Pos*)

Set, List, Tree and Graph Functions

X [=] GET(S, *Pos*)

X [=] GET(S, *Pos, Index*)

SET(S, *Pos, X*)

SET(S, *Pos, entryIndex, x*)

Pos = FIND(S, , *Xtmp, Condition*)

Pos = FIND NEXT(S, *Pos*, *Xtmp, Condition*)

SORT(S, *Xtmpr1, Xtmp2*, *Condition*)

Set Variables

GLOBAL SET {10} VARIABLE: S

CONTAINS(S, *X*)
INSERT(S, *X*)

S = UNION(S1, S2)
S = INTERSECTION(S1, S2)
S = SUBTRACT(S1, S2)

List Variables (An ordered set: First in first out lists…FIFO, like a cashier at grocery store)

GLOBAL LIST {10} VARIABLE: L

Pos = HEAD(L)
Pos = TAIL(L)

INSERT HEAD(L, X)
INSERT TAIL(L, X)
INSERT BEFORE(L, Index, X)
INSERT AFTER(L, Index, X)
INSERT AT(L, Index, X)

Expressions

- **Set**
- **List**
- **Graph**
- **Sorting**
- **Tree**

List Variables

GLOBAL LIST {10} VARIABLE: L

REMOVE HEAD(L)
REMOVE TAIL (L)
REMOVE AT INDEX(L, Index)

X [=] GET HEAD(L)
X[=] GET TAIL(L)
X [=] GET AT INDEX(L, Index)
Pos = POS AT INDEX(L, Index)

Tree Variables

GLOBAL SET {10} VARIABLE: T

ADD ROOT(T, X)
INSERT LEFT CHILD(T, *Pos*, X)
INSERT RIGHT CHILD(T, *Pos*, X)
INSERT CHILD(T, *Pos*, X, Index)
n = CHILDREN(T, *Pos*)

Tree Variables

GLOBAL SET {10} VARIABLE: T

Pos = PARENT(T, *Pos*)
Pos = LEFT CHILD(T, *Pos*)
Pos = RIGHT CHILD(T, *Pos*)
Pos = CHILD(T, *Pos, Index*)
Pos = NEXT SIBLING(T, *Pos*)
Pos = PREV SIBLING(T, *Pos*)

Tree Variables

GLOBAL SET {10} VARIABLE: T

X [=] GET LEFT CHILD(T, *Pos*)
X [=] GET RIGHT CHILD(T, *Pos*)
X [=] GET CHILD(T, *Pos*, *Index*)

Tree Variables

GLOBAL SET {10} VARIABLE: T

Pos = NEXT DFS(T, *Pos*)

- position of next node in depth-first pre-order traversal of a tree. Set Pos to NULL to start

Pos = NEXT POSTORDER DFS(T, *Pos*)

- position of next node in depth-first post-order traversal of a tree. Set Pos to NULL to start

Graph Variables

GLOBAL GRAPH {10,5} VARIABLE: G

- nodes and links
- Use list and general functions for nodes

REMOVE ALL LINKS(G)

n = SIZE LINKS(G)

- number of links in graph

b = IS EMPTY LINKS(G)

Pos = FIRST LINK(G)

Pos = NEXT LINK (G, *Pos*)

Pos = PREV LINK (G, *Pos*)

Graph Variables

GLOBAL GRAPH {10,5} VARIABLE: G

X [=] GET LINK(G, *Pos*)
b = CONTAINS LINK(G, *Xlink*)
REMOVE LINK(G, *Pos*)

SET LINK(G, *Pos, Xlink*)
SET LINK(G, *Pos, entryIndex, x*)

Graph Variables

GLOBAL GRAPH {10,5} VARIABLE: G

Pos = FIND LINK(G, *XtmpLink, Condition*)

Pos = FIND NEXT LINK(G, *Pos*, *XtmpLink, Condition*)

SORT LINKS(G, *XtmpLink1*, *XtmpLink1*, *Condition*)

b = LINKED(G, *Xnode1*, *Xnode2*, *LinkType*)

- Set LinkType to 0 for direct links and 1 for indirect