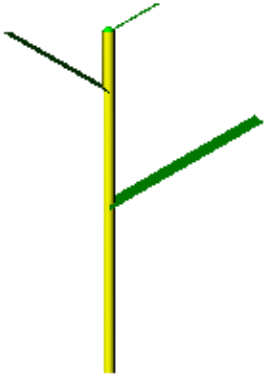
XL language: Queries & Co

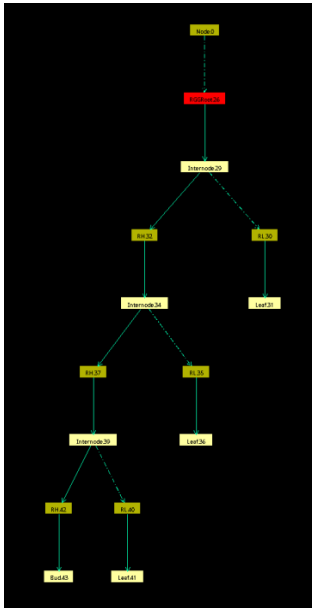
Katarína Smoleňová

FSPM & GroIMP Workshop

PMBDA 2025, Nanchang, China

November 1, 2025

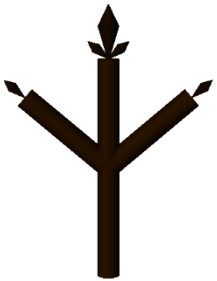


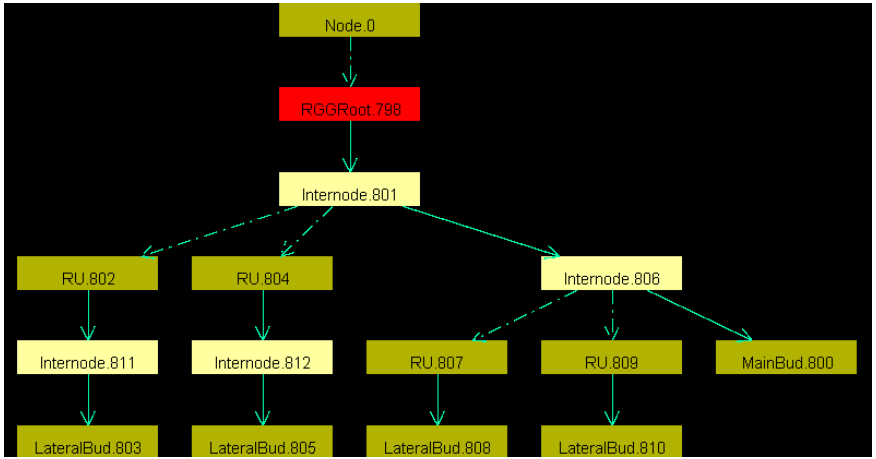


WAGENINGEN
UNIVERSITY & RESEARCH

Everything is represented as a graph...

- Plant structure described by a graph composed of nodes (plant organs), connected via edges (relations between organs)
- We can search for a specific pattern in the graph and collect information





Graph operators

- *Queries*
  to search for a specific pattern
- *Aggregators*
  to collect values
- Analyse *relations* between nodes

Find:

all leaves

total leaf area

stem length

first internode below bud

etc.

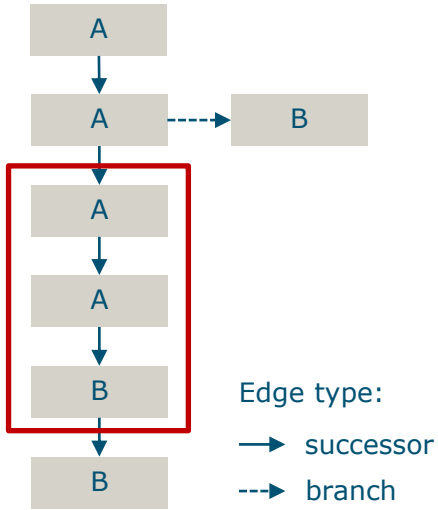
Graph query

- *XL syntax:*
  enclosed in asterisked parentheses **(* *)**

- The elements are given in their expected order, e.g.:

  (* A A B *)
  searches for a subgraph that consists of a sequence of nodes of the types A A B, connected by *successor* edges

  ([blank] – successor edge notation)



Query: Examples

- Find all nodes B, connected to A with a branch edge (+>)
- Find all internodes
- ... and print them out

A query can contain a condition

- Find all newly created leaves (with age 0)
- Search for all leaves with length > 0.05
- Find all pairs of F-s with distance < 1

```
(* A +> B *)


(* Internode *)

println((* Internode *));



(* l:Leaf, (l[age] == 0) *)

(* l:Leaf, (l[length] > 0.05) *)

(* f:F, g:F, ((f != g) && (distance(f, g) < 1)) *)
```

WAGENINGEN
UNIVERSITY & RESEARCH

Aggregators

- Collect multiple values when traversing the graph and return one single value as a result
- Standard aggregate operations:

  count, **sum**, **empty**, **exist**, **forall**, **first**, **last**, **max**, **min**, **mean**, **selectRandomly**, **selectWhereMin**, **selectWhereMax**, ...

- Applied to the results of a query:

  count((* Leaf *) **)**

Aggregators: Examples

- *Get stem length:*

  Search for all internodes and sum up their lengths

- *Get total leaf area:*

  Search for all leaves and sum up their surface areas

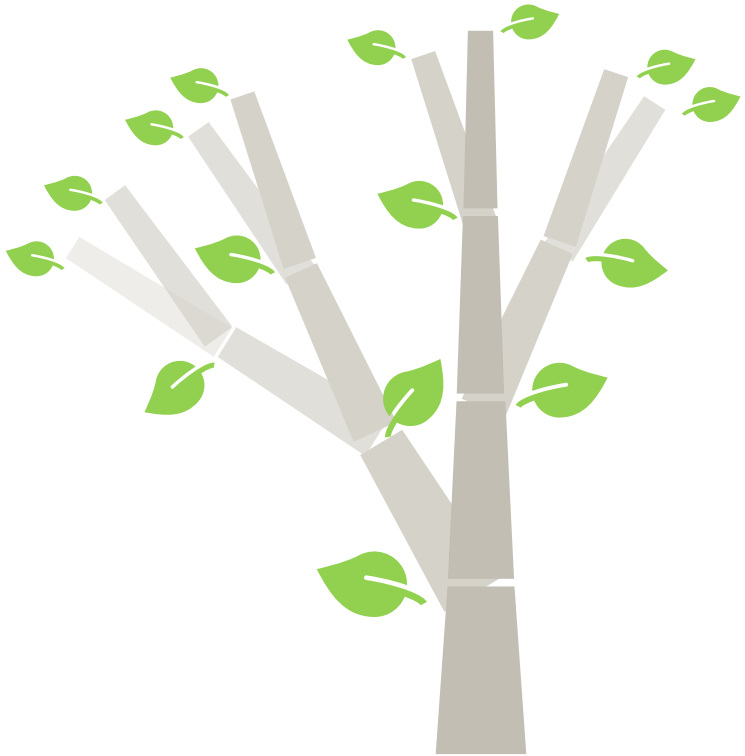
- *Get total absorbed light by a plant:*

  Search for all leaves and sum up the amount of light they absorbed

```
sum((* Internode *)[length])


sum((* Leaf *)[area]) I


sum((* Leaf *)[absLight]);
```

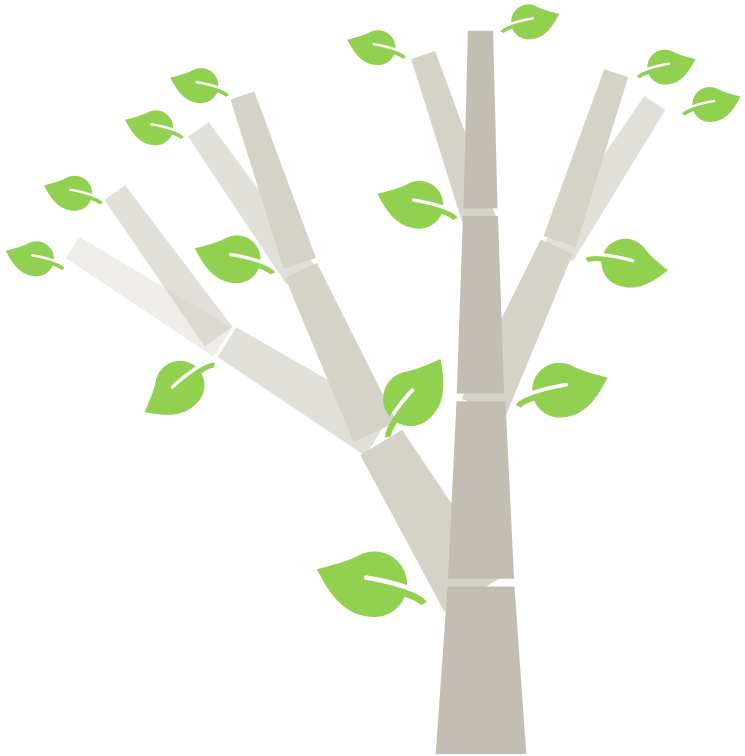
Aggregators: More examples

```
selectRandomly((* F *))

selectWhereMax((* f:F *), (f[diameter]))

first((* l:Leaf, (l[order] == 1 && l[rank] == 1) *))
```



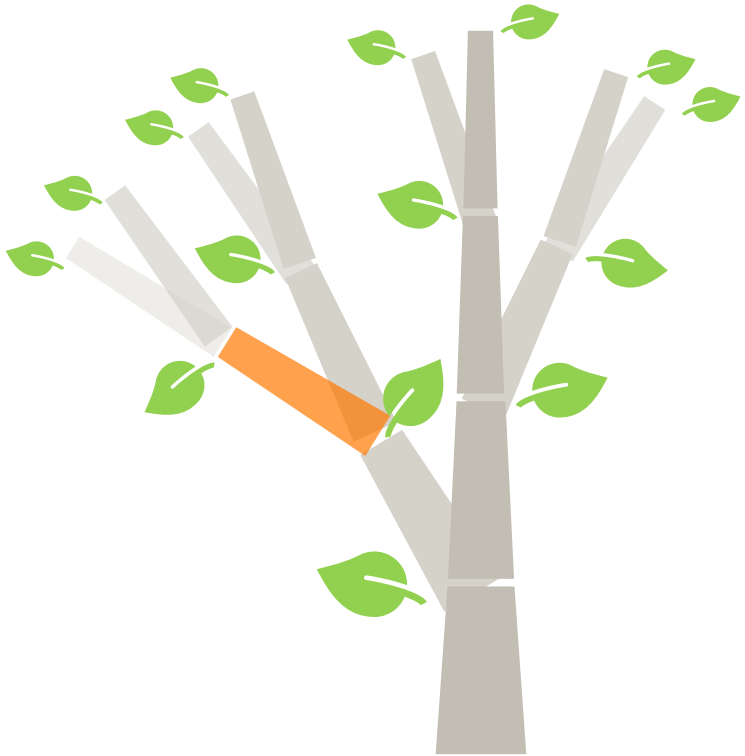
Aggregators: More examples

```
selectRandomly((* F *))

selectWhereMax((* f:F *), (f[diameter]))

first((* l:Leaf, (l[order] == 1 && l[rank] == 1) *))
```



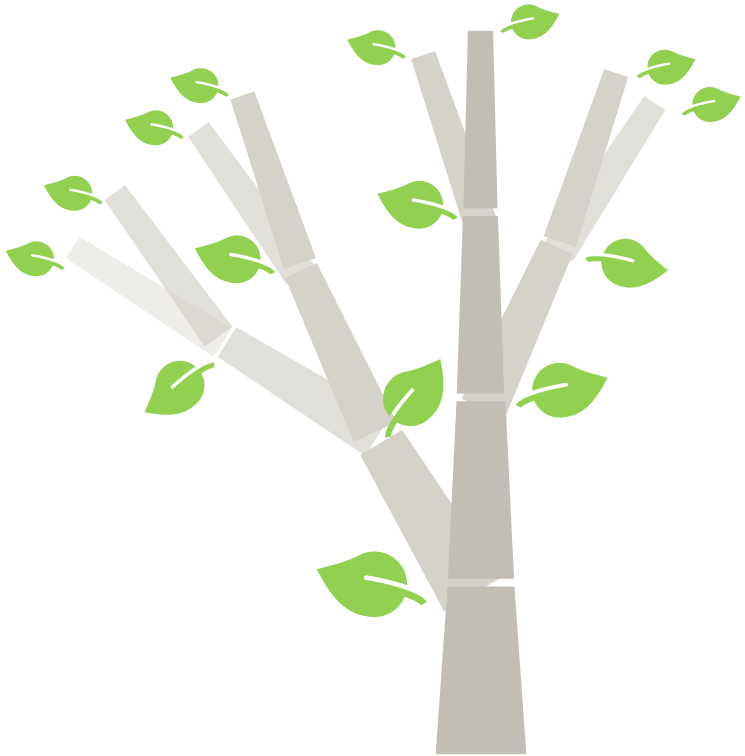
Aggregators: More examples

```
selectRandomly((* F *))

selectWhereMax((* f:F *), (f[diameter]))

first((* l:Leaf, (l[order] == 1 && l[rank] == 1) *))
```



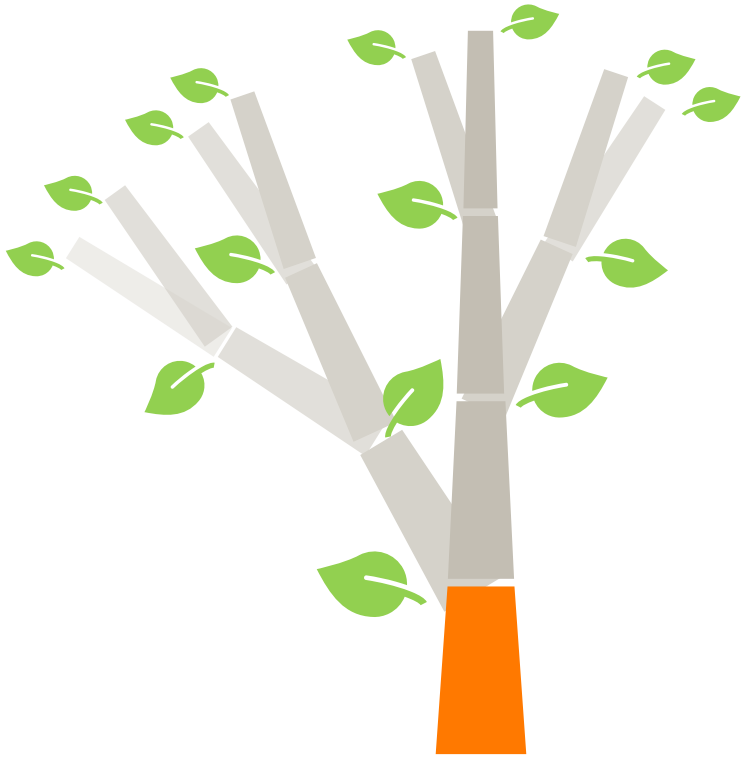
Aggregators: More examples

```
selectRandomly((* F *))

selectWhereMax((* f:F *), (f[diameter]))

first((* l:Leaf, (l[order] == 1 && l[rank] == 1) *))
```



Aggregators: More examples

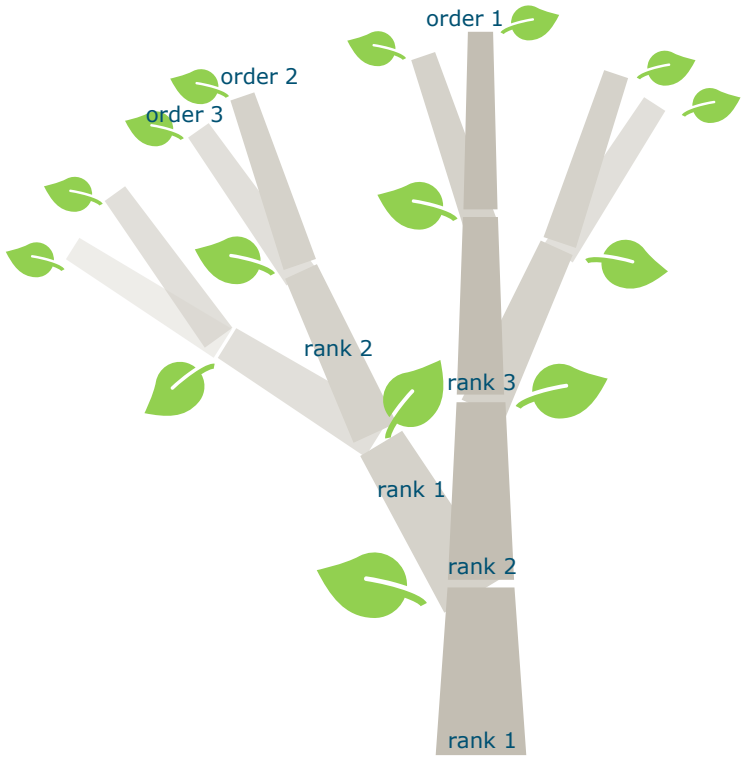
```
selectRandomly((* F *))

selectWhereMax((* f:F *), (f[diameter]))

first((* l:Leaf, (l[order] == 1 && l[rank] == 1) *))
```



Aggregators: More examples

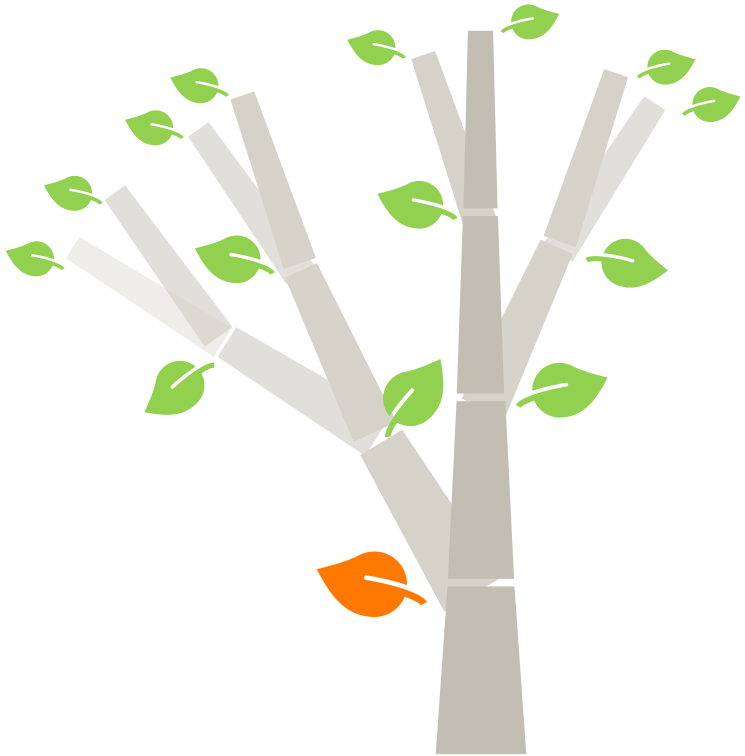
```
selectRandomly((* F *))

selectWhereMax((* f:F *), (f[diameter]))

first((* l:Leaf, (l[order] == 1 && l[rank] == 1) *))
```

(order refers to branching order; main stem has order 1;
each branch has one order higher;
rank refers to organ number within a branch)



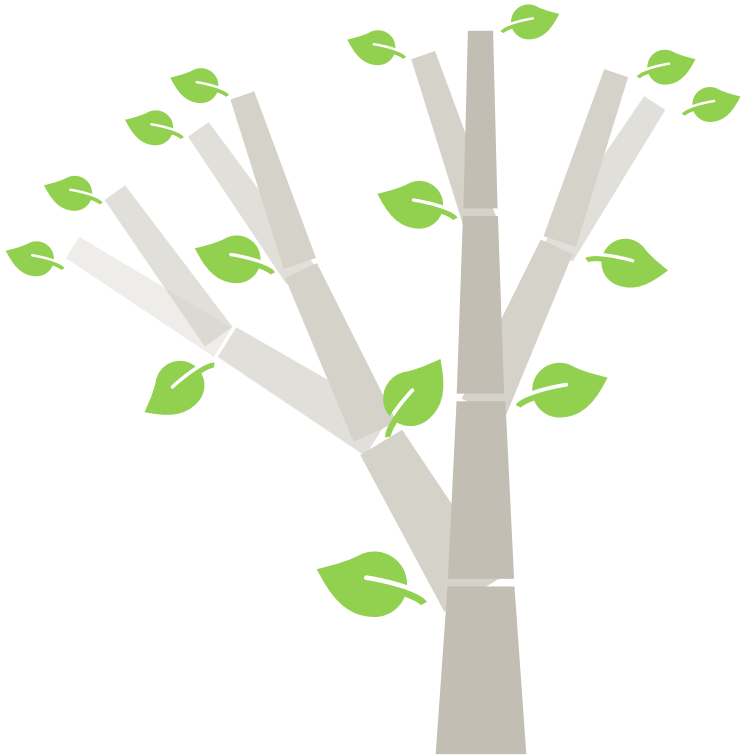
Aggregators: More examples

```
selectRandomly((* F *))

selectWhereMax((* f:F *), (f[diameter]))

first((* l:Leaf, (l[order] == 1 && l[rank] == 1) *))
```



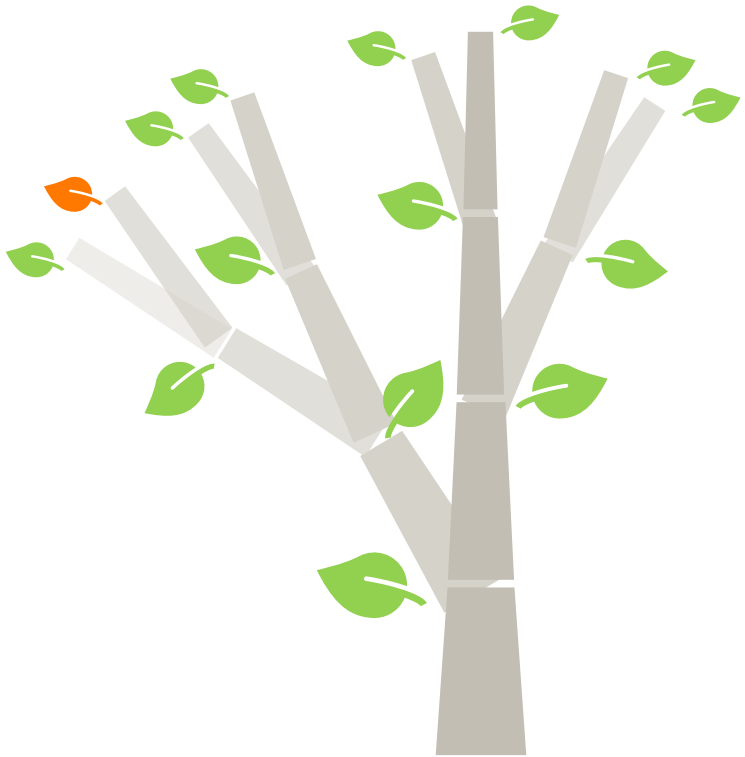
Aggregators: More examples

```
selectRandomly((* F *))

selectWhereMax((* f:F *), (f[diameter]))

first((* l:Leaf, (l[order] == 3 && l[rank] == 2) *))
```



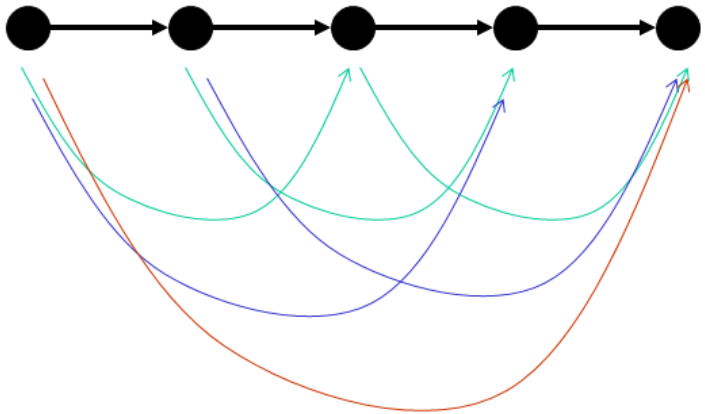
Aggregators: More examples

```
selectRandomly((* F *))

selectWhereMax((* f:F *), (f[diameter]))

first((* l:Leaf, (l[order] == 3 && l[rank] == 2) *))
```



Derived relations

- Relation between nodes connected by several edges (one after the other) of the same type
- *Example*:
  find *all descendants* of some given node that are of type Internode



- *"Transitive closure"* of the original relation (iteration of edges)

XL syntax for the transitive closure

- 1-to-n repetitions: **+**

```
A (-edgetype->)+ B
```

- 0-to-n repetitions: ***

```
A (-edgetype->)* B
```

- Minimal elements
  (stop searching once a match has been found):

```
A (-edgetype->)+ :(B)
```

```
A (-edgetype->)* :(B)
```

- *Examples:*

Find all the internodes connected to the bud

```
(* Bud (<--)+ Internode *)
```

Find the first internode connected to the bud

```
(* Bud (<--)+ :(Internode) *)
```

Edge types

r = relation

- Edge notation:

Arbitrary	**-->**	
Successor	(blank)	or **>** or **–successor->**
Branch	**+>**	or **[** or **–branch->**
Decomposition	**/>**	
User-defined	**-r->**	

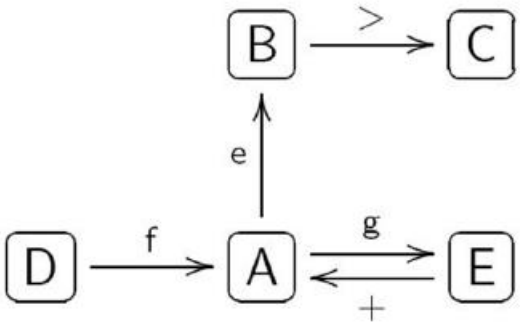
- Edge direction:

Forward	**-r->**
Backward	**<-r-**
Forward or backward	**-r-**
Forward and backward	**<-r->**

Graphs in XL

successor: > or (blank)

branch: + or [ ]




```
a:A [-e-> B C] [<-f- D] -g-> E [a]
```

multiple representations possible!

Some predefined relations

- *ancestor*

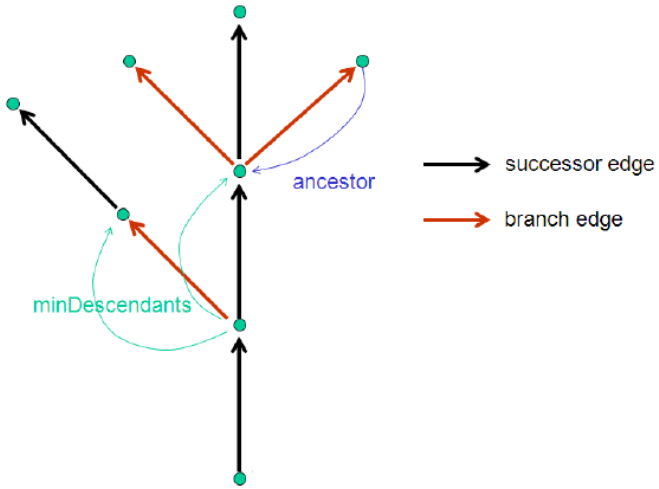
  nearest preceding node of a certain node type

- *minDescendants*

  nearest successors of a certain node type
  (nodes of other types are skipped)

- *descendants*

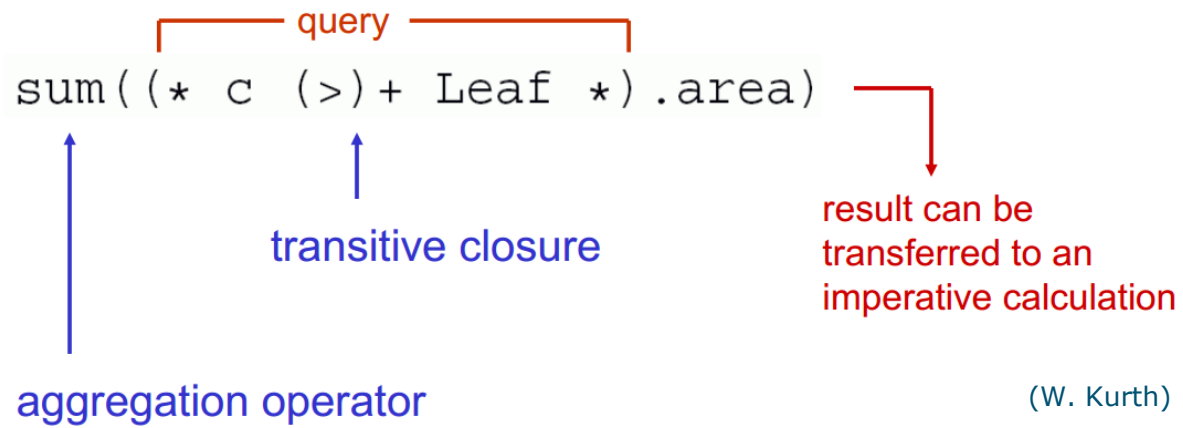
  all successors of a certain node type



Syntax:
-ancestor->
-descendants->
-minDescendants->

Summary – Graph operations

- Provide possibilities to connect structure and function
- *Example*:
  search for all leaves that are successors of node c and sum up their surface areas

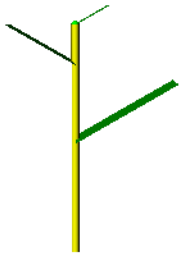


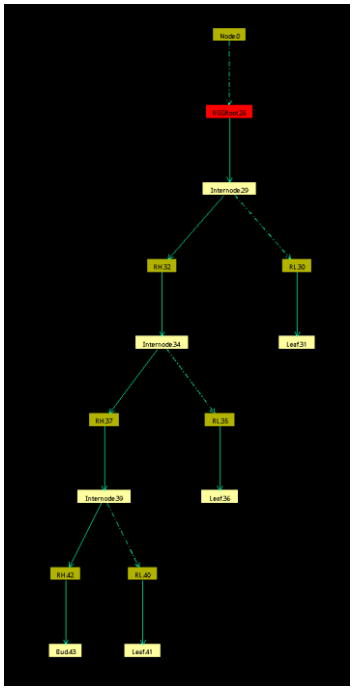
(W. Kurth)

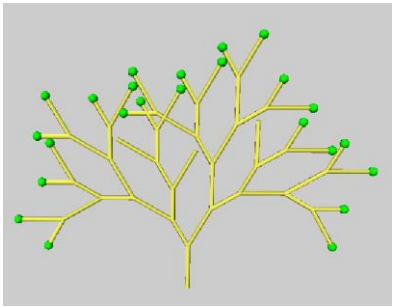
Examples

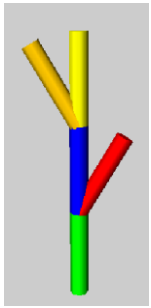
- Tutorial:
  “tutorials:xl-queries-and-operators [GroIMP wiki]

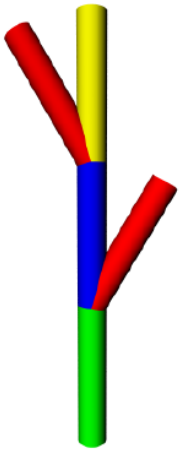
- Further examples:
  binaryTree_distance.gsz
  nodeRelations.gsz











```
// all modules extend the module Internode
Axiom ==>
    InternodeFirst [RU(30) BranchFirst]
    InternodeSecond [RU(-30) BranchSecond]
    InternodeThird
;
// queries and their outputs:
(* InternodeFirst -minDescendants-> Internode *)
    BranchFirst
    InternodeSecond

(* InternodeFirst -descendants-> Internode *)
    InternodeSecond
    InternodeThird
    BranchSecond
    BranchFirst

(* InternodeThird -ancestor-> Internode *)
    InternodeSecond

(* InternodeThird (-ancestor->)+ Internode *)
    InternodeSecond
    InternodeFirst
```

wiki.grogra.de

fspm.discourse.group