Functional-Structural Plant Modelling with GroIMP and XL

Tutorial and Workshop at Agrocampus Ouest, Angers,
5-7 May, 2015

Winfried Kurth

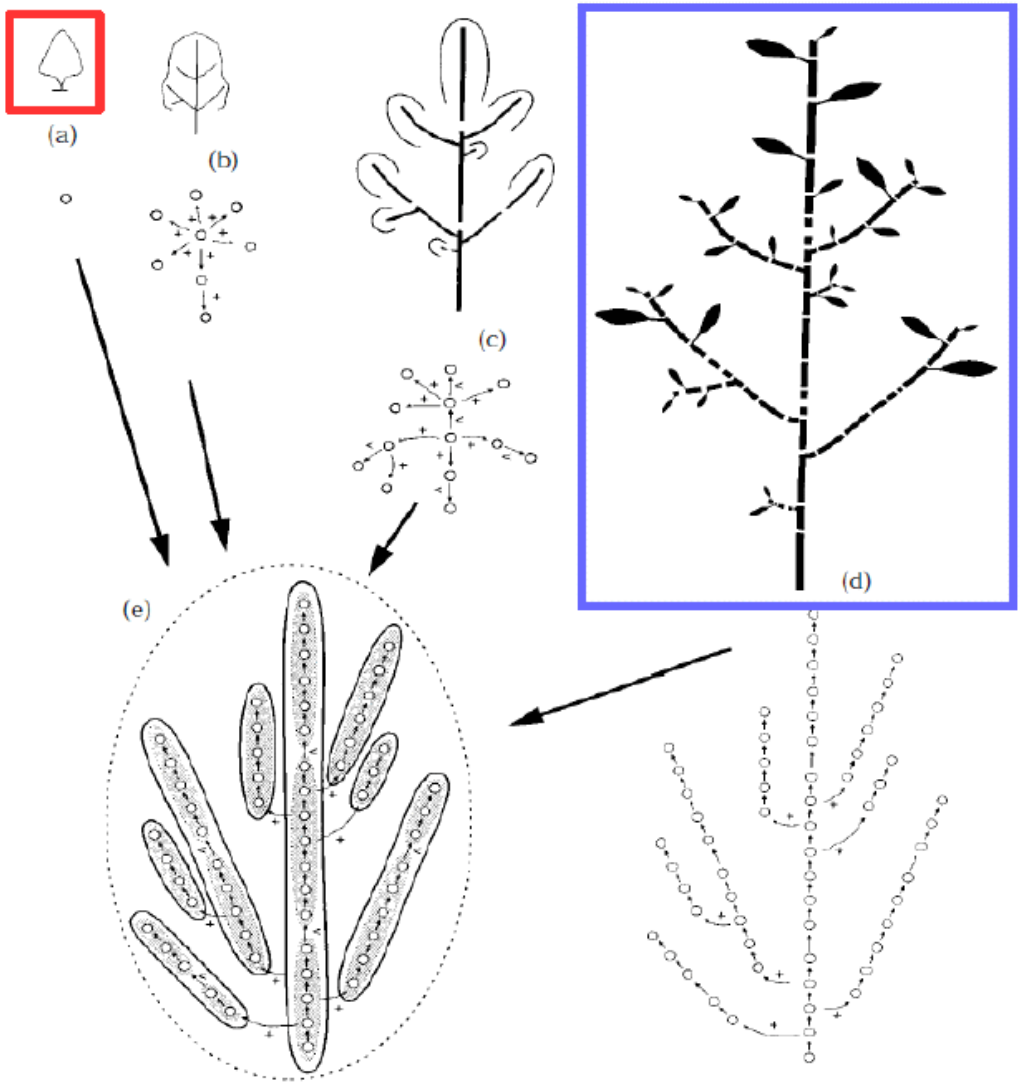
University of Göttingen,
Department Ecoinformatics, Biometrics and Forest Growth

Modelling multiscale structures in XL

(parts from the doctoral defense of Yongzhi Ong, 27.04.2015)

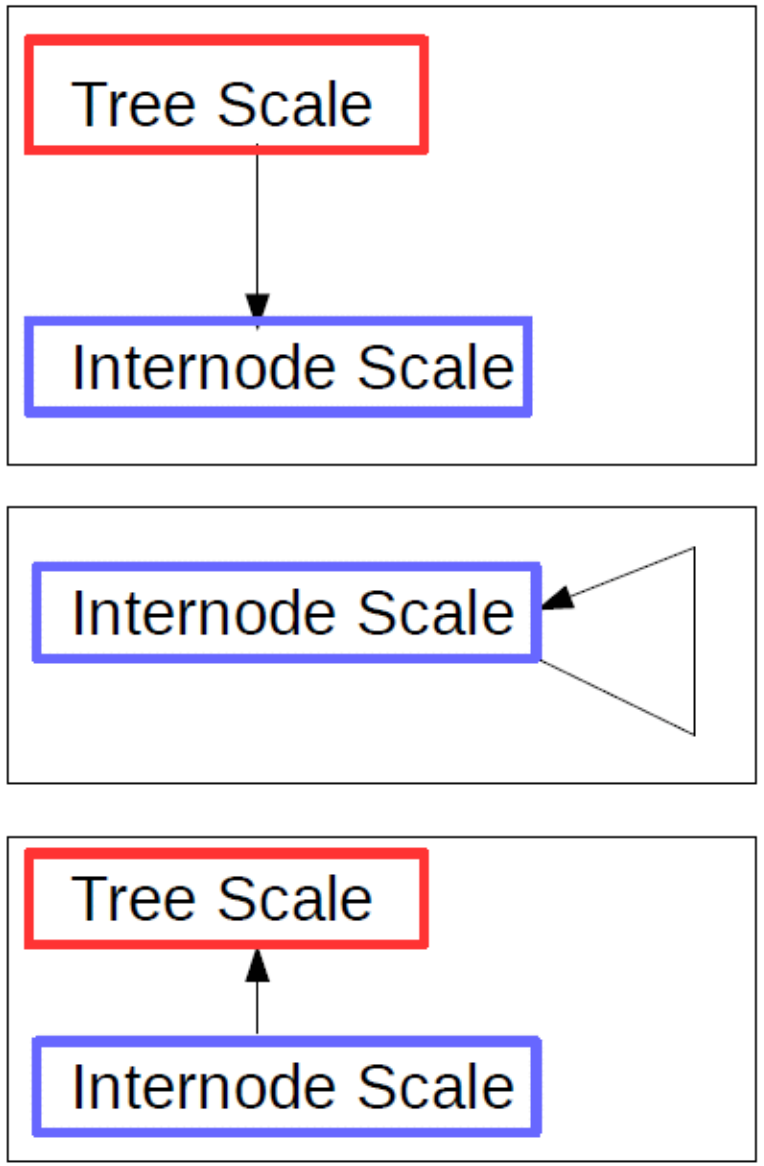
1. Context – Scales / Levels-of-Detail

Individual Trees Axes Growth Units Internodes / Buds / Leaves / etc.



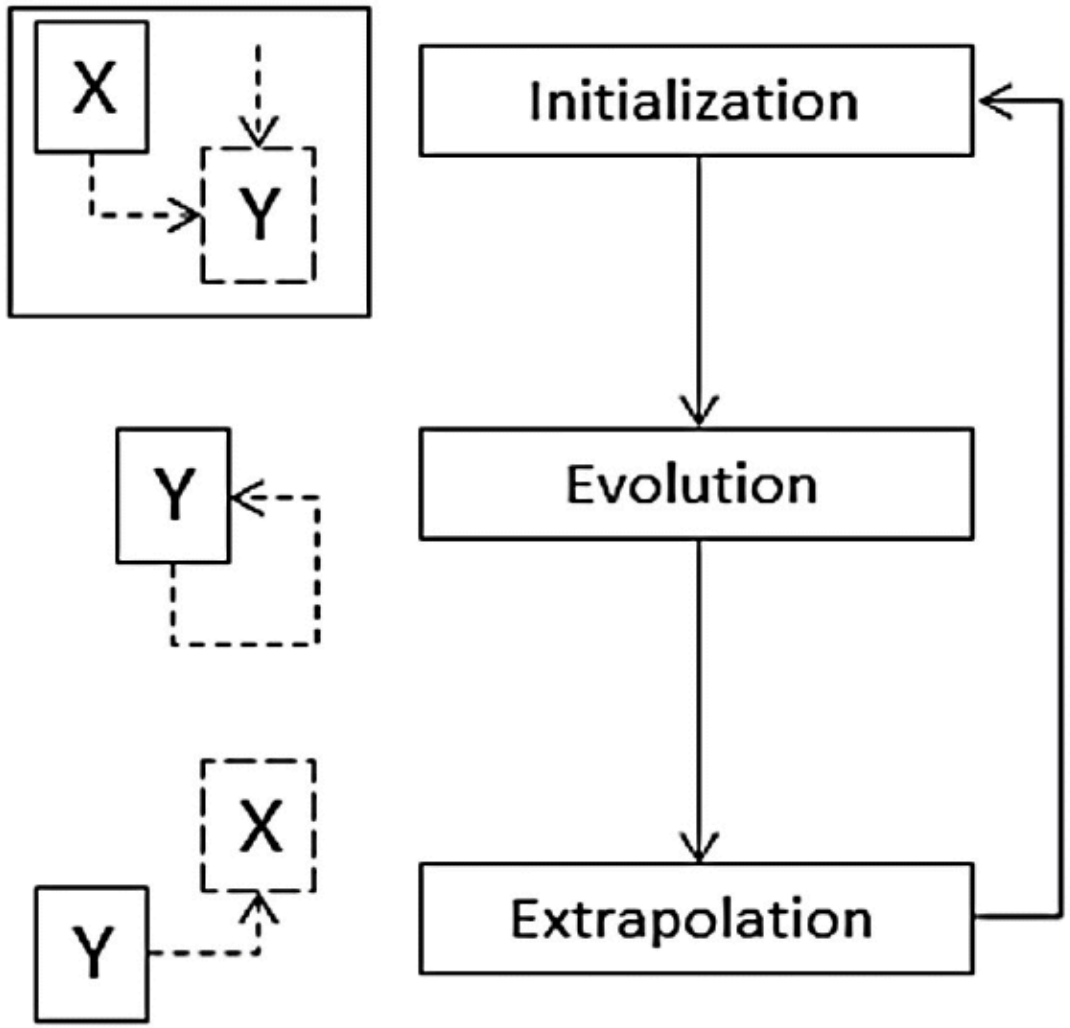
* Godin & Caraglio (1998)

Multiscale Tree Graph (MTG) *



Multiscale Modelling Framework #

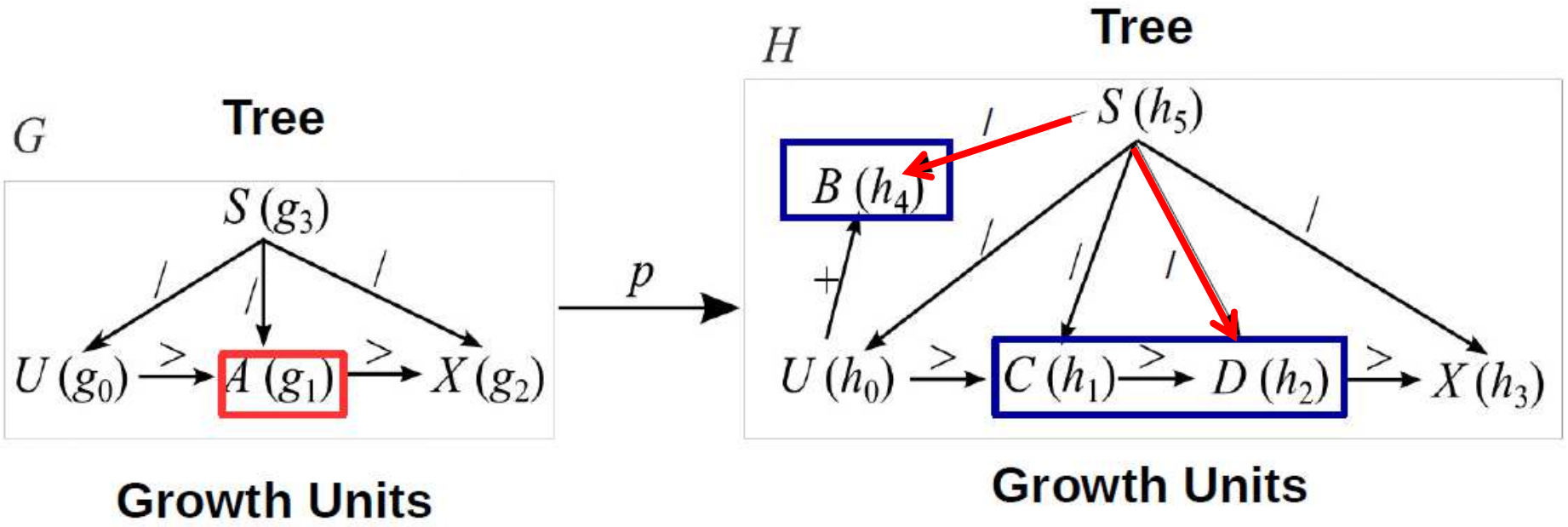
Ong et al. (2014), E (2011)



Multiscale Modelling Framework #

Ong et al. (2014), E (2011)

2. Multiscale (MS) Grammar. - Problem

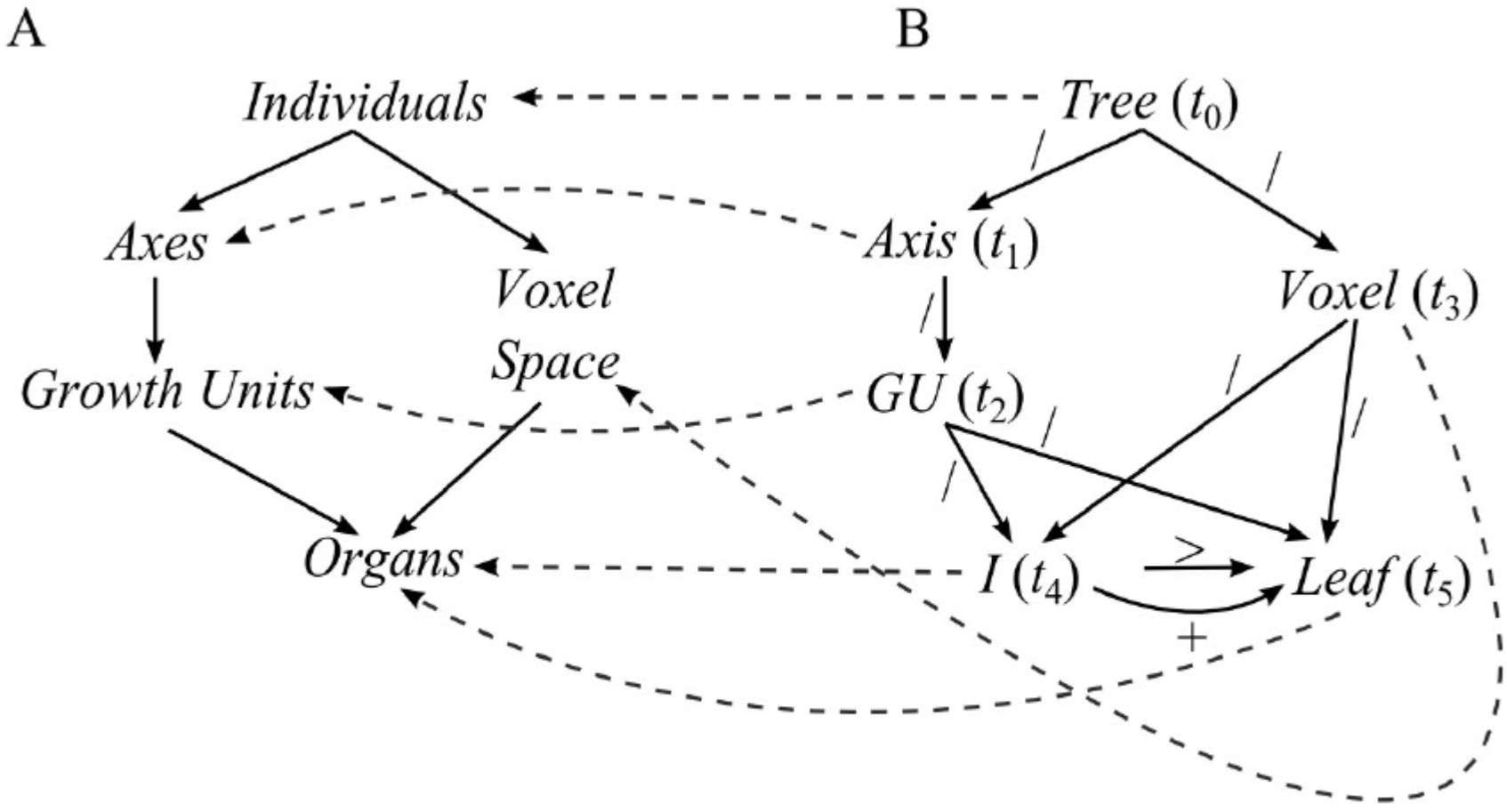


XL Syntax / Rule: A ==> [B] C D;

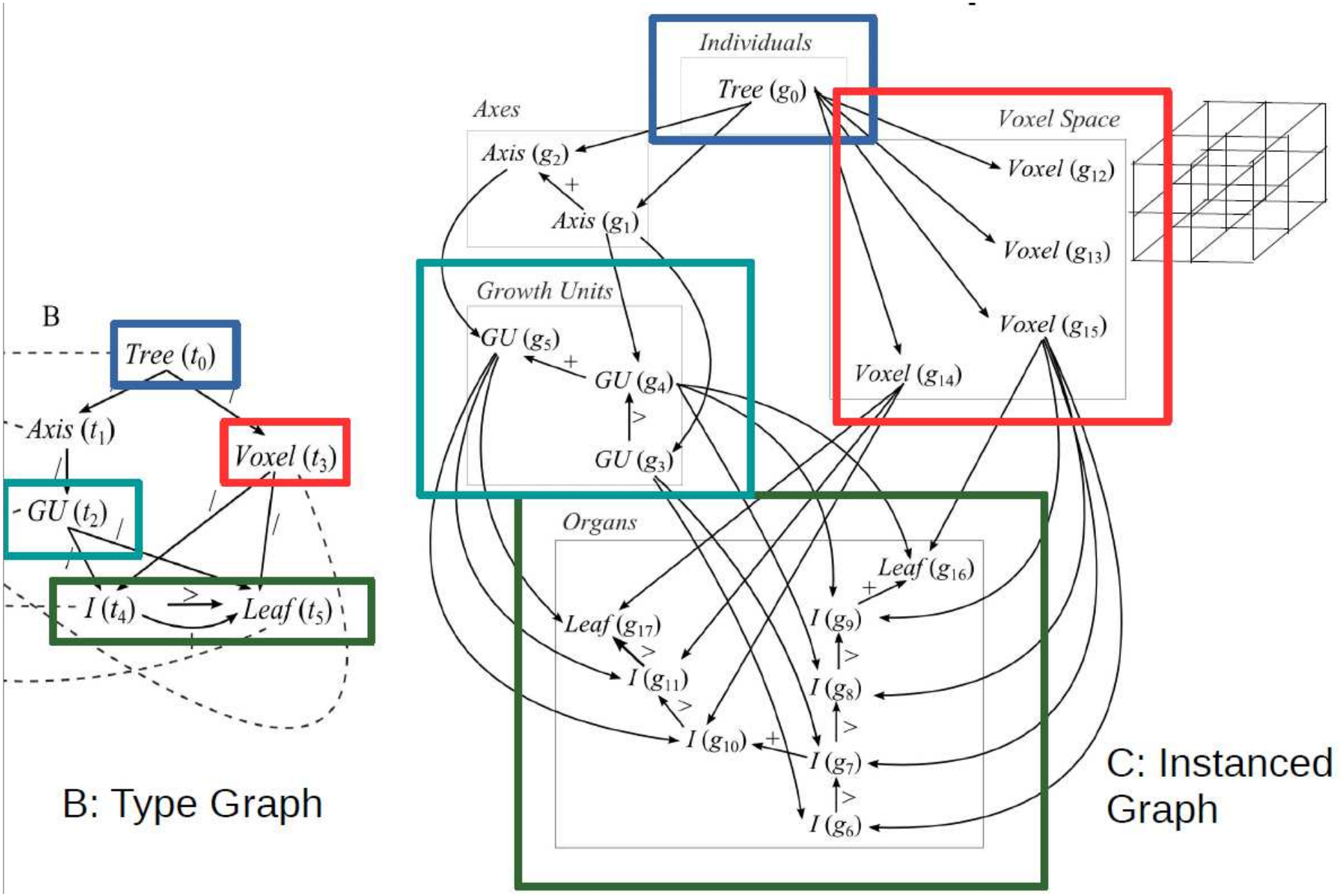
Edge Labels:

> Successor
+ Branching
/ Refinement

A data structure consisting of 3 graphs: Structure of scales, type graph, instanced graph

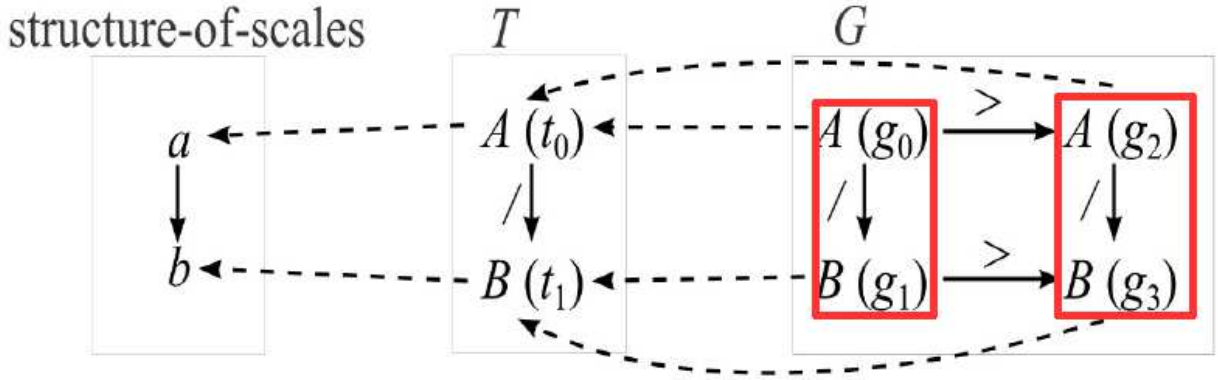


A: Structure-of-Scales B: Type Graph

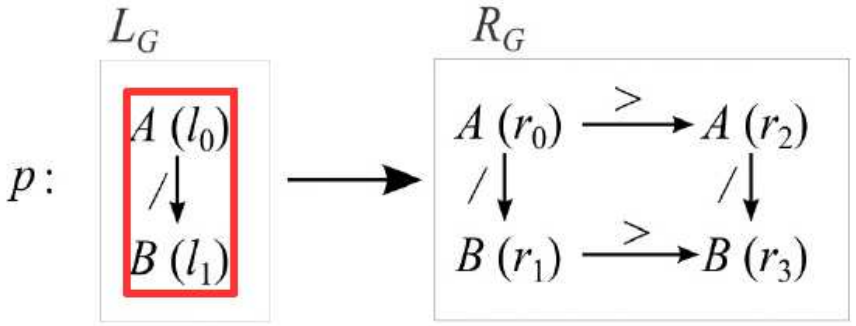

B
Tree (t_0)
Axis (t_1)
Voxel (t_3)
GU (t_2)
I (t_4)
Leaf (t_5)
B: Type Graph
Individuals
Tree (g_0)
Axes
Axis (g_2)
Axis (g_1)
Voxel Space
Voxel (g_{12})
Voxel (g_{13})
Voxel (g_{15})
Voxel (g_{14})
Growth Units
GU (g_5)
GU (g_4)
GU (g_3)
Organs
Leaf (g_{16})
Leaf (g_{17})
I (g_9)
I (g_{11})
I (g_8)
I (g_{10})
I (g_7)
I (g_6)
C: Instanced Graph

Rewriting: The simplest case (l.h.s. of rule specified for each scale)

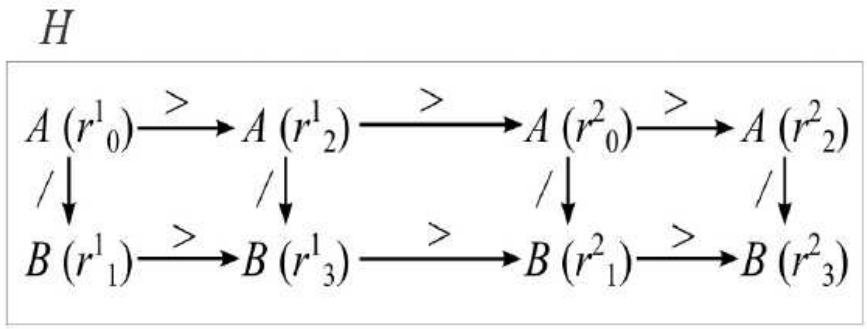
3-Part Graph Model



Production (Rule)

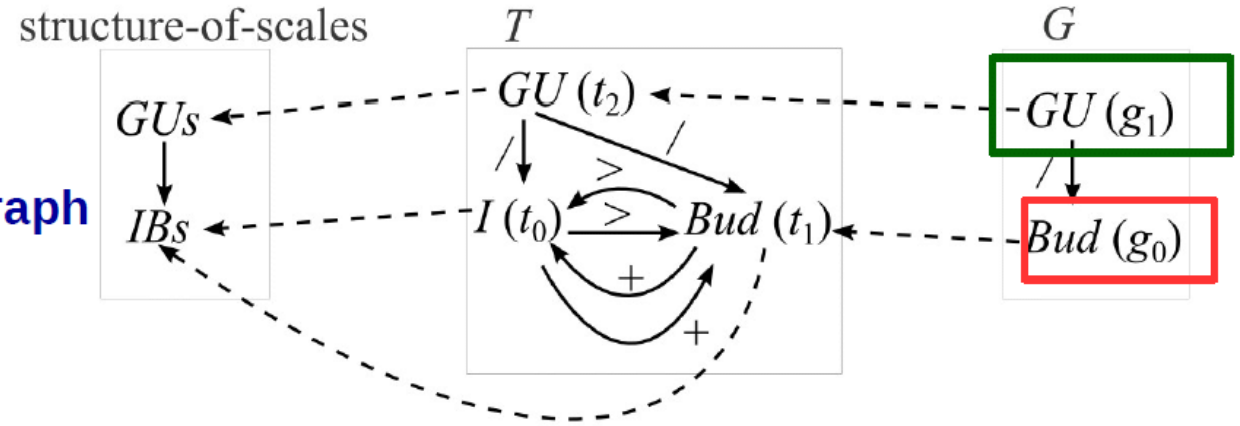


Derived Instance Graph

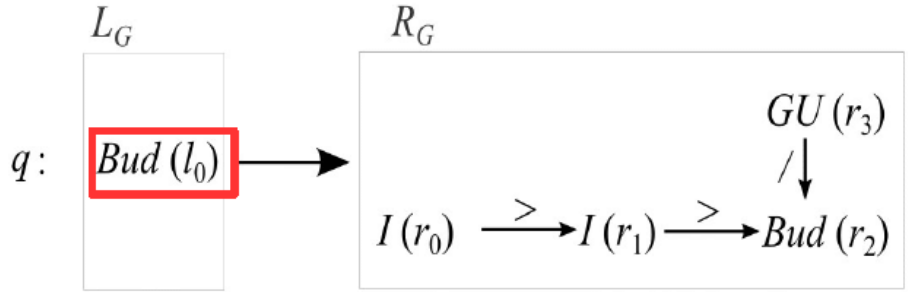


Rewriting: L.h.s. specifies only an organ at finest scale, generation of new entities at higher scales in r.h.s.

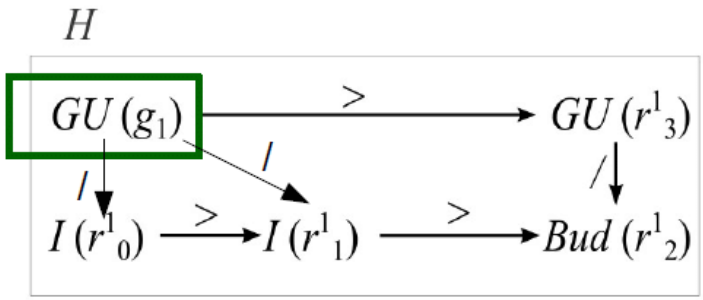
3-Part Graph Model



Production (Rule)



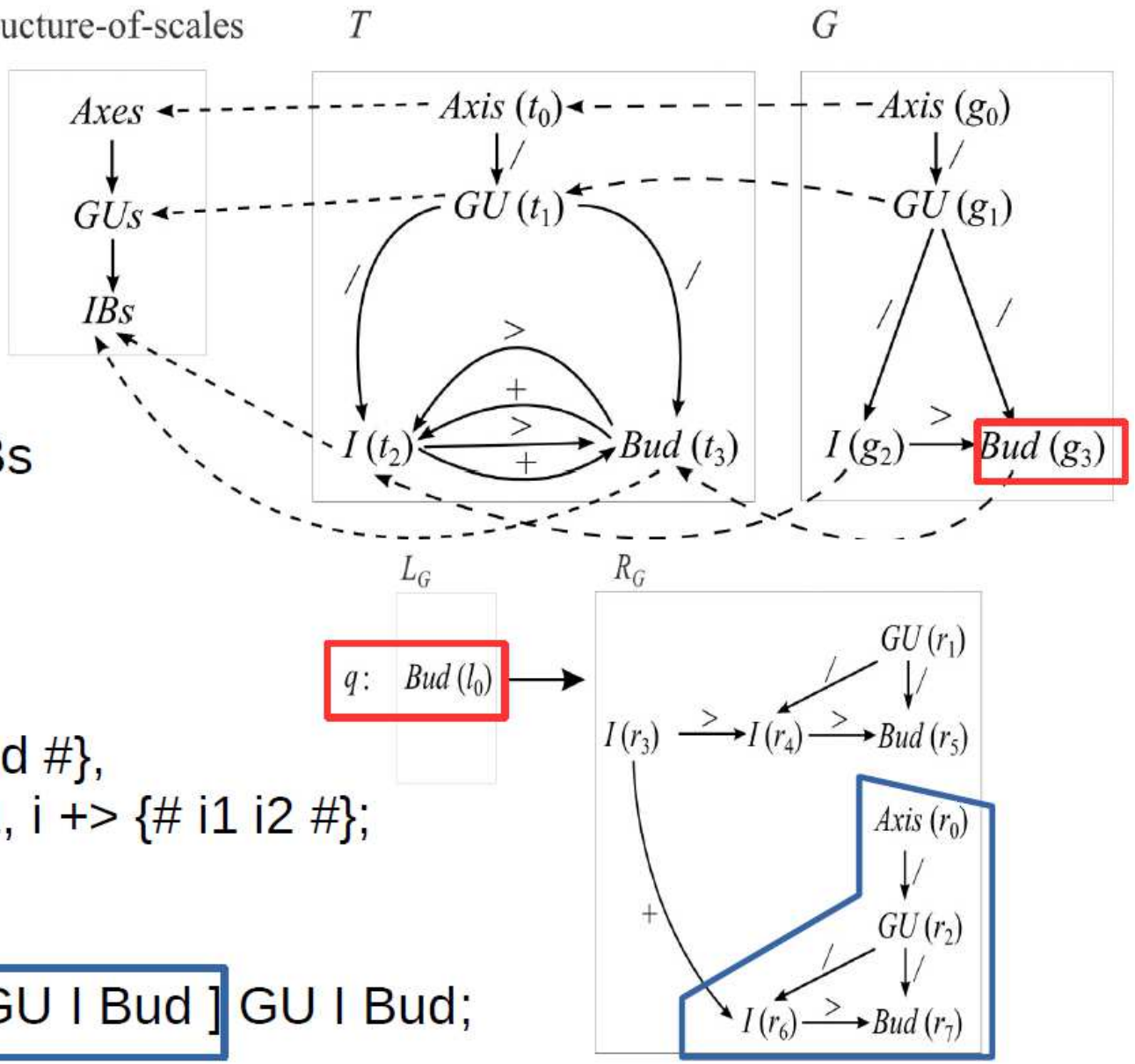
Derived Instance Graph



XL syntax for 3-part graph structure and for rewriting

3-Part Graph Model

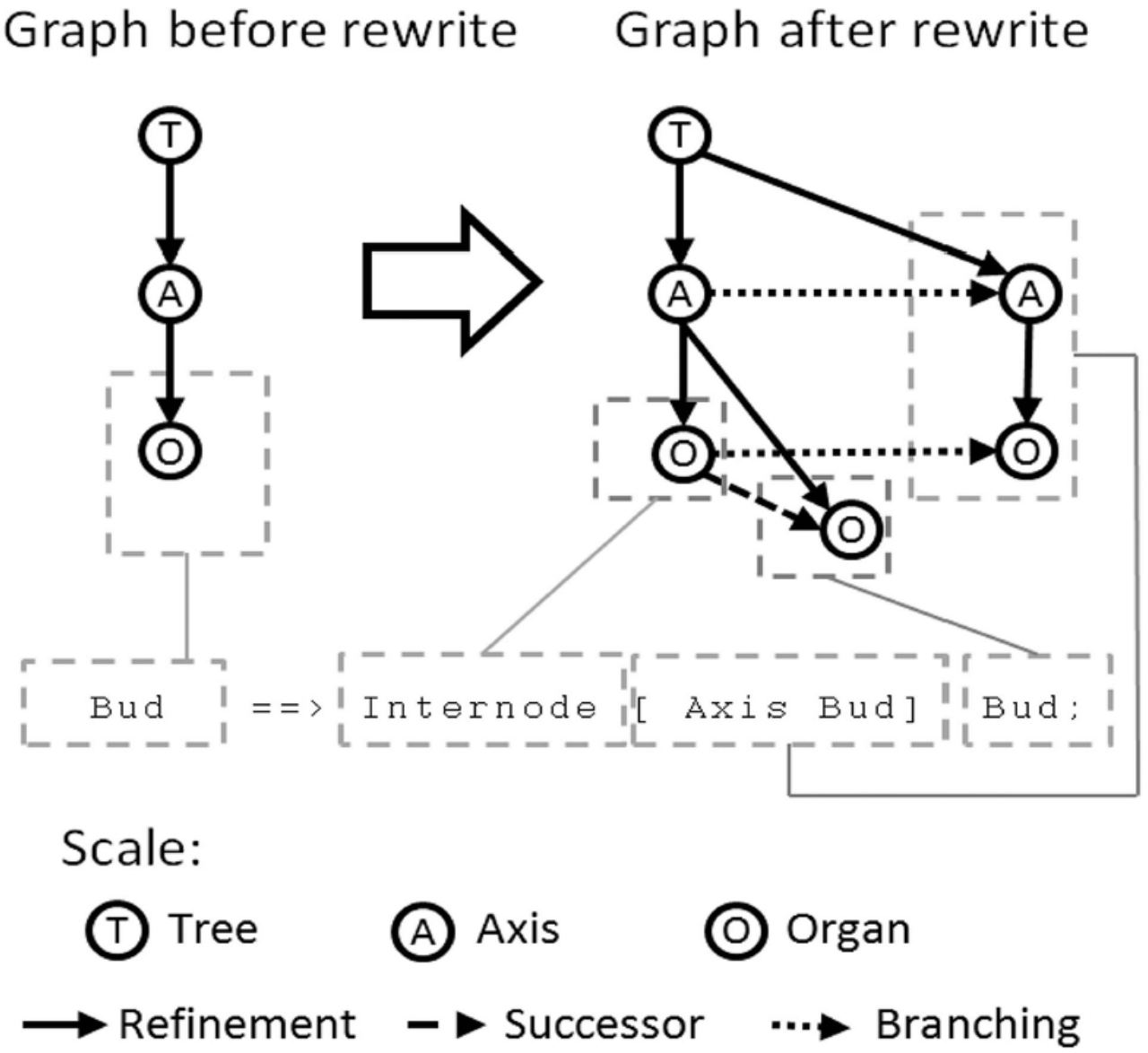
```
==>> ^
[ /> SRoot
  /> a:Axes
  /> g:GUs /> i:IBs
]
  /> TypeRoot
    /> a1:Axis
    /> g1:GU
    /> {# i1:I i2:Bud #},
a +> a1, g +> g1, i +> {# i1 i2 #};
```



Production (Rule)

```
Bud ==> I [Axis GU I Bud ] GU I Bud;
```

Example


Graph before rewrite
Graph after rewrite
T
A
O
Bud ==> Internode [Axis Bud] Bud;
Scale:
T Tree
A Axis
O Organ
Refinement
Successor
Branching

Solution using multiscale grammar:

```
Bud ==> Internode [ Axis Bud ] Bud;
```

Solution using indexes to associate objects:

```
module Internode(int axis) implements Organ;
module Bud(int axis) implements Organ;

b:Bud ==> {
    removeOrganFromAxis(b.axis, b);
    int latAxis = createAxis(b.axis);
 }
 i:Internode(b.axis, 1)
 [bl:Bud(latAxis)]
 ba:Bud(b.axis)
 {addOrganToAxis(b.axis, i); ...}
 ;
```

XL implementation of the observer pattern for information exchange between scales

```
scale Axes;
scale GUs;
scale IBs;

Module Axis;
Module GU;
Module I;
Module Bud;
```

```
protected void init()
[
    ==>> ^
            [ /> SRoot /> a:Axes
                 /> g:GUs /> i:Ibs ]
            /> TypeRoot  /> a1:Axis
                 /> g1:GU /> {# i1:I i2:Bud #},
            a +> a1, g +> g1, i +> {# i1 i2 #}
            {a.observe(i,0,"axesGrow");};

    Axiom ==> ...
]
```

```
public void organGrow()[ Bud ==> I [Axis GU I Bud ] GU I Bud ];
                         i:IBs ::> {i.notify(0);} ]

protected void axesGrow() [
   a:Axis ::> {a.length = sum((* a /> /> I *).length);} ]
```

The most simple example:

```
static int EVOLVE = 1;

scaleclass A(float len);
scaleclass B;

protected void init ()
[
	Axiom ==> a:A(1) b:B, {b.observe(a, EVOLVE, "m");};
]

public void run ()
[
	a:A ::> {a.notify(EVOLVE);}
]

protected void m()
[
	b:B ::> {println("someting has changed");}
]
```

Thank you for your attention!