A Primer on Inverse Procedural Modeling

CS434

Daniel G. Aliaga
Department of Computer Science
Purdue University

Recall: Procedural Modeling

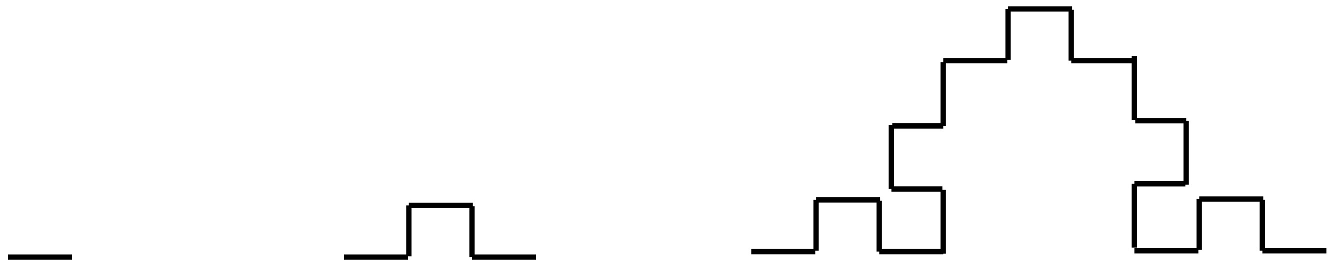
- Apply algorithms for producing objects and scenes
- The rules may either be embedded into the algorithm, configurable by parameters, or externally provided

Procedural Modeling

- Fractals
- Terrains
- Image-synthesis
  - Perlin Noise
  - Clouds
- Plants
- Cities
- And procedures in general…

L-system

- Variables: a
- Constants: +, - (rotations of + or – 90 degrees)
- Initial string (axiom): s=a
- Rules: a → a+a-a-a+a



(Context-Free) L-system for Plants

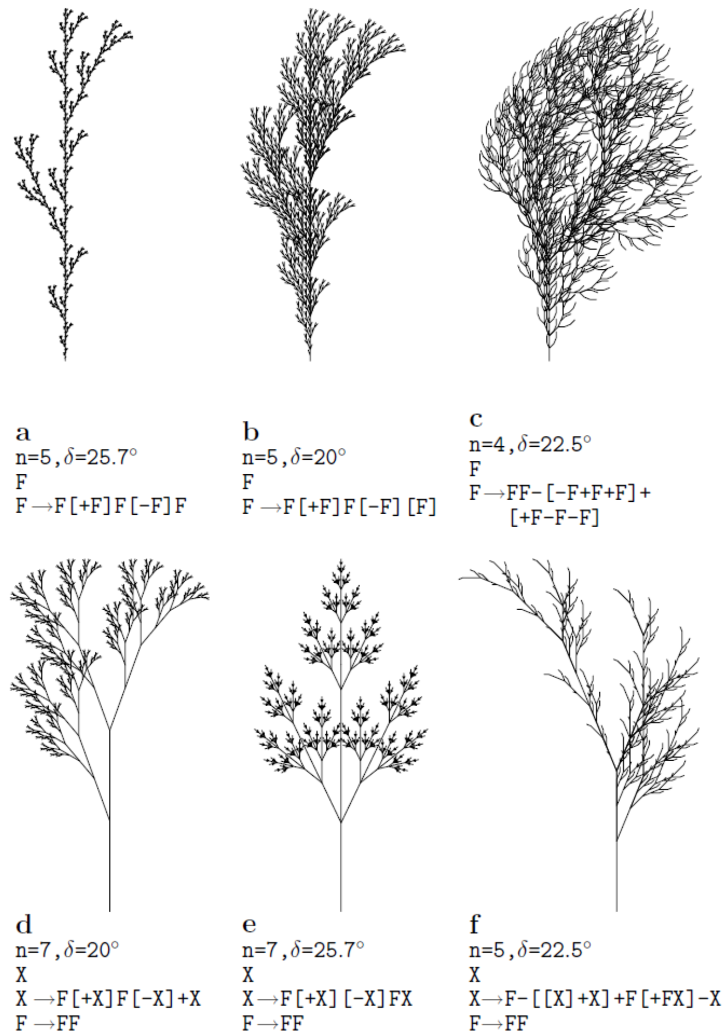


Figure 1.24: Examples of plant-like structures generated by bracketed OL-systems. L-systems (a), (b) and (c) are edge-rewriting, while (d), (e) and (f) are node-rewriting.

L-system for Plants (stochastic)

$$
\begin{aligned}
\omega &: \quad F \\
p_1 &: \quad F \xrightarrow{.33} F[+F]F[-F]F \\
p_2 &: \quad F \xrightarrow{.33} F[+F]F \\
p_3 &: \quad F \xrightarrow{.34} F[-F]F
\end{aligned}
$$

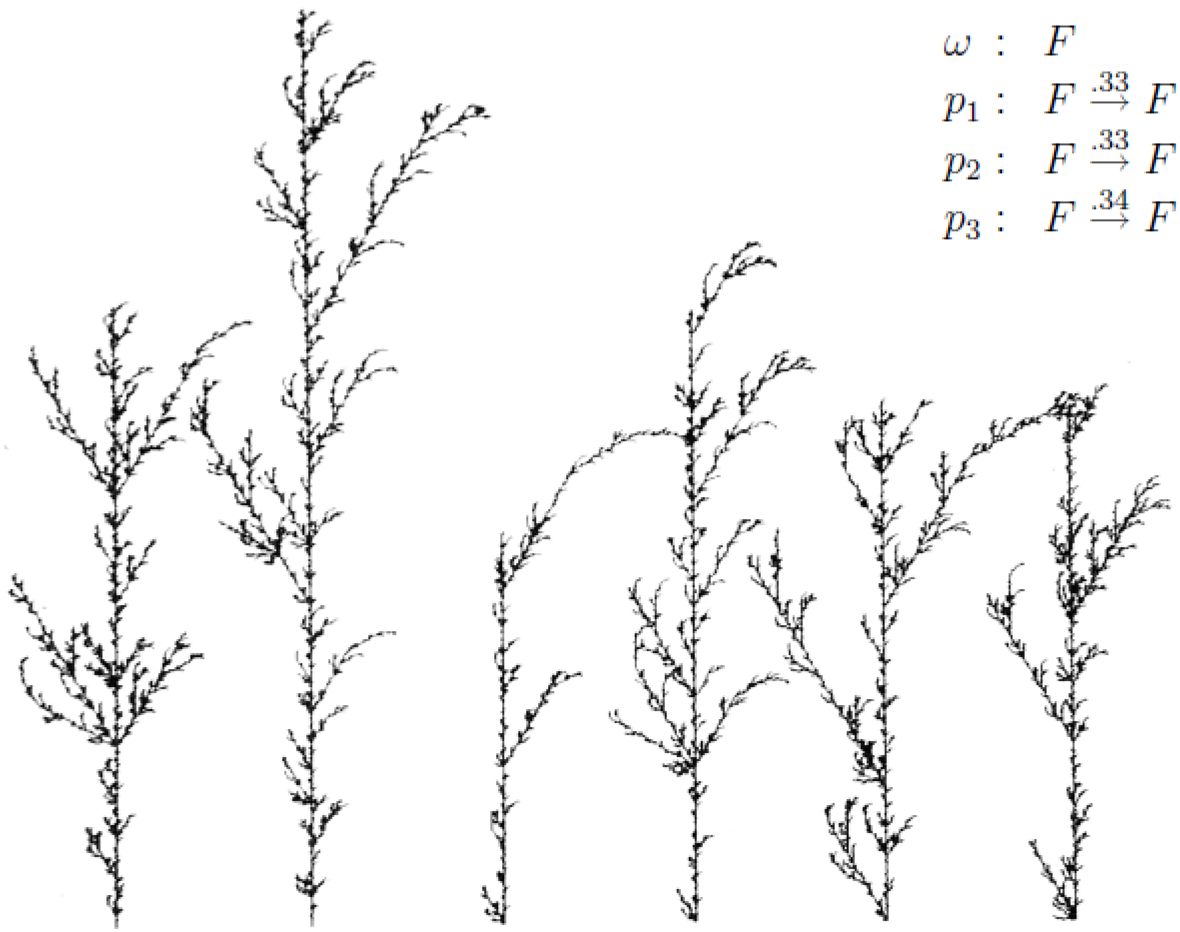


Figure 1.27: Stochastic branching structures

L-system for Plants (3D)

n=5, δ=18°

ω : plant
p_1 : plant → internode + [plant + flower] − − //
[− − leaf] internode [+ + leaf] −
[plant flower] + + plant flower
p_2 : internode → F seg [// & & leaf] [// ∧ ∧ leaf] F seg
p_3 : seg → seg F seg
p_4 : leaf → [' { +f−ff−f+ | +f−ff−f }]
p_5 : flower → [& & & pedicel ‘ / wedge //// wedge ////
wedge //// wedge //// wedge]
p_6 : pedicel → FF
p_7 : wedge → [‘ ∧ F] [{ & & & & −f+f | −f+f }]

Figure 1.26: A plant generated by an L-system

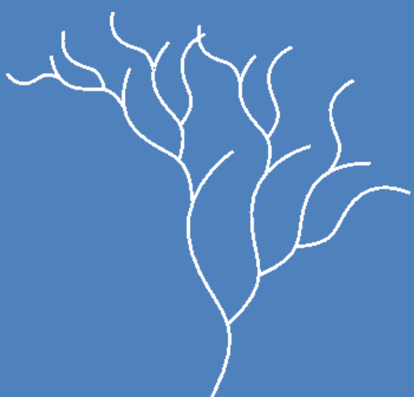
Figure 1.28: Flower field

Inverse Procedural Modeling (by Automatic Generation of L-systems)

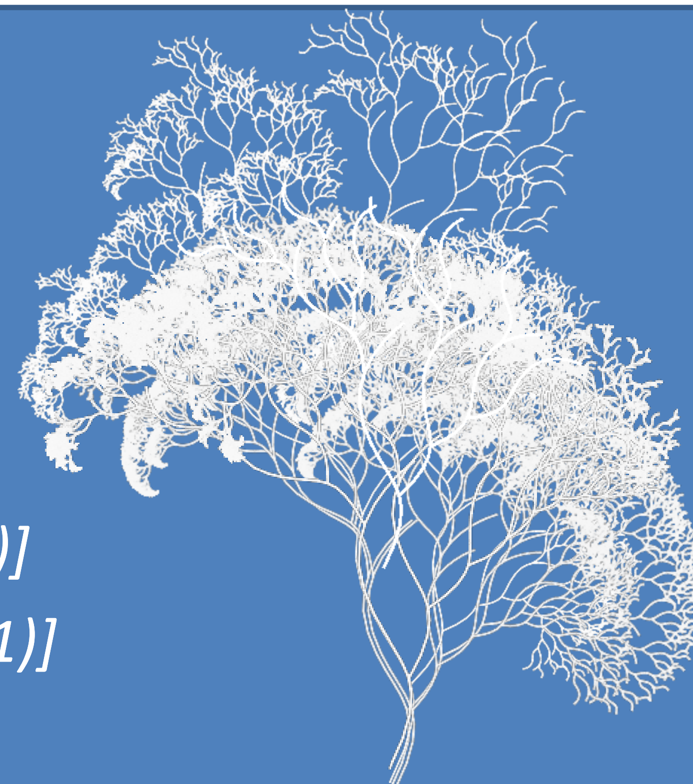
O. Šťava, B. Beneš, R. Měch*,
D. Aliaga, P. Krištof
Purdue University, *Adobe Inc

Overview

2D Vector Image → Analysis → L-system → Modifications



$$R(m) \rightarrow$$
$$A[-(\alpha)\, f(d) * (s) - (\beta) R(m-1)]$$
$$[+(\alpha')\, f\,(d') * (s') + (\beta') R(m-1)]$$



Introduction

- **Procedural Modeling**
  - Rules → Scene
  - Extensively studied
  - The most important: L-systems
  - Applied to plants, buildings, rivers, etc.
- **Inverse Procedural Modeling**
  - Scene → Rules
  - Open problem

Motivation

- What is the L-system for these?

Motivation

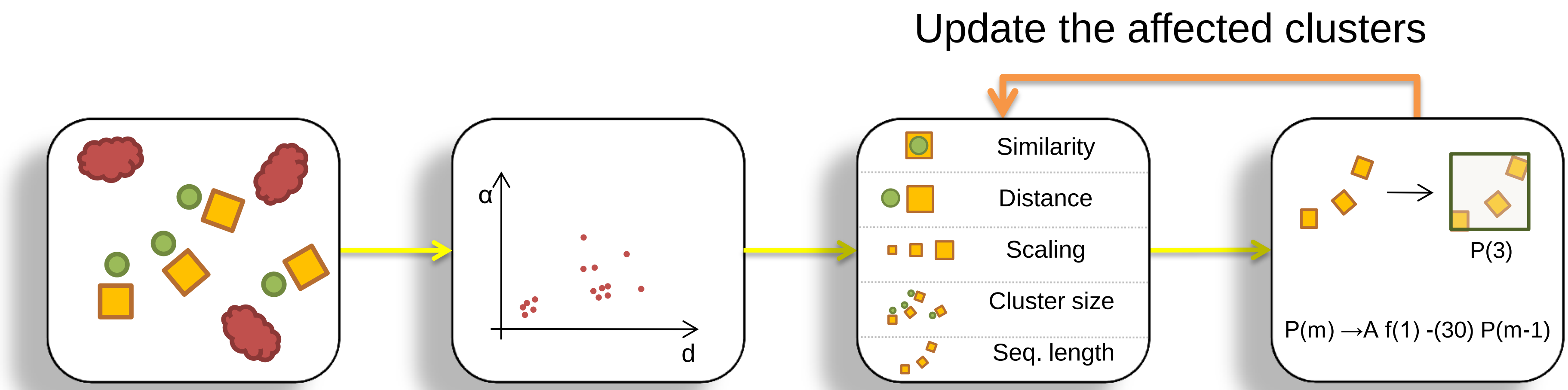
- Writing procedural models,
  be it L-systems or others is hard
- Setting parameters is not intuitive
- Difficult for a non-expert to create a model for a desired shape,
  especially when recursive branching

Approach

- Input: 2D vector image
- Output: L-system

- Inspired by symmetry detection
  - Mitra et al., "Partial and approximate symmetry detection for 3D geometry"
  - Pauly et al., "Discovering structural regularity in 3D geometry"

Approach



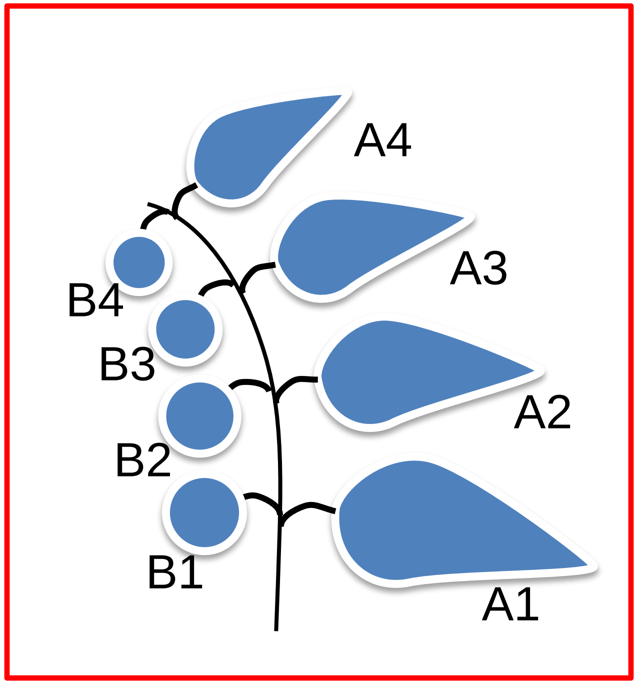
1. Calculate terminals by inverse instancing

2. Calculate transformations, fill transformation spaces, and perform clustering

3. Analyze clusters and calculate cluster significance

4. Create non-terminal symbols and L-system rules

Approach



L-system 1

$$P1(m)\ :m>0 \rightarrow [A]\ T_1\ P1(m\text{-}1)$$

$$m=0 \rightarrow [A]$$

$$P2(m)\ :m>0 \rightarrow [B]\ T_2\ P2(m\text{-}1)$$

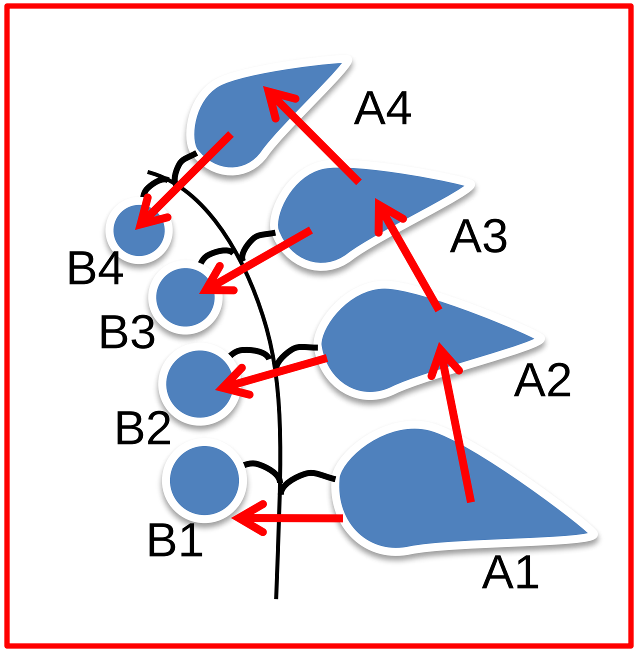
$$m=0 \rightarrow [B]$$

$$P3(m) \rightarrow [P1(3)]\ T_3\ [P2(3)]$$

$$S \rightarrow T_s\ [P3]$$

Approach

- Different rules can generate the same result



L-system 2

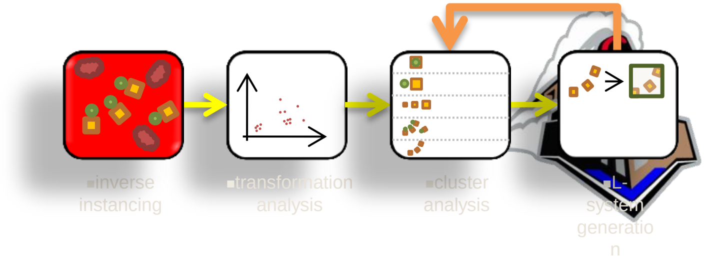
$P1(m) \rightarrow [A]\ T_1\ [B]$

$P2(m)\ :m>0 \rightarrow [P1]\ T_2\ P2(m-1)$

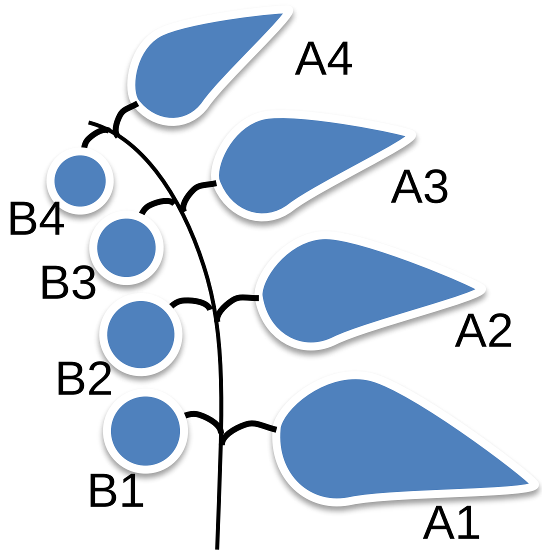
$m=0 \rightarrow [P1]$

$S \rightarrow T_s\ [P2(3)]$

Inverse Instancing



- **Terminal symbols**
  - Similar vector elements



L-system 2

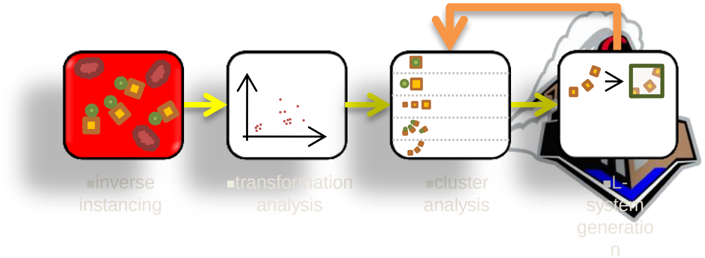
$P1(m) \rightarrow [A]\ T_1\ [B]$

$P2(m)\ :m>0 \rightarrow [P1]\ T_2\ P2(m-1)$

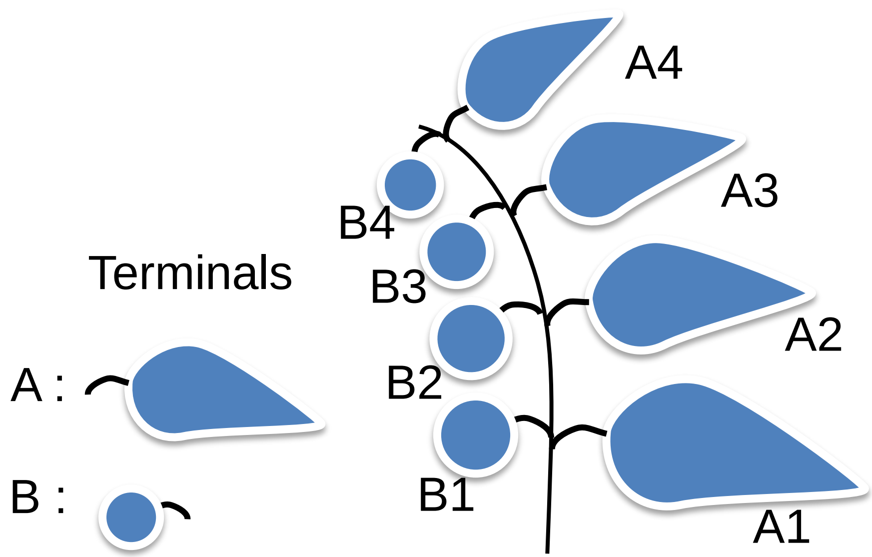
$m=0 \rightarrow [P1]$

$S \rightarrow T_s\ [P2(3)]$

Inverse Instancing



- Compute similarity between all input elements
- Similar elements are represented by a terminal



L-system 2

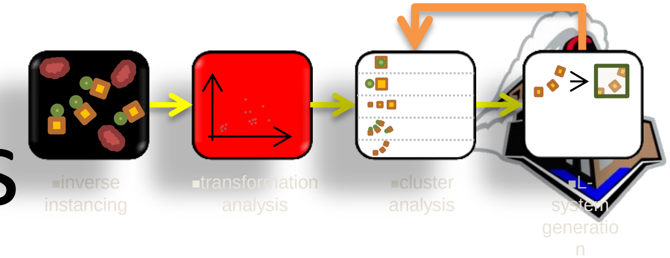
$$P1(m) \rightarrow [A]\, T_1\, [B]$$

$$P2(m)\ : m>0 \rightarrow [P1]\ T_2\ P2(m-1)$$

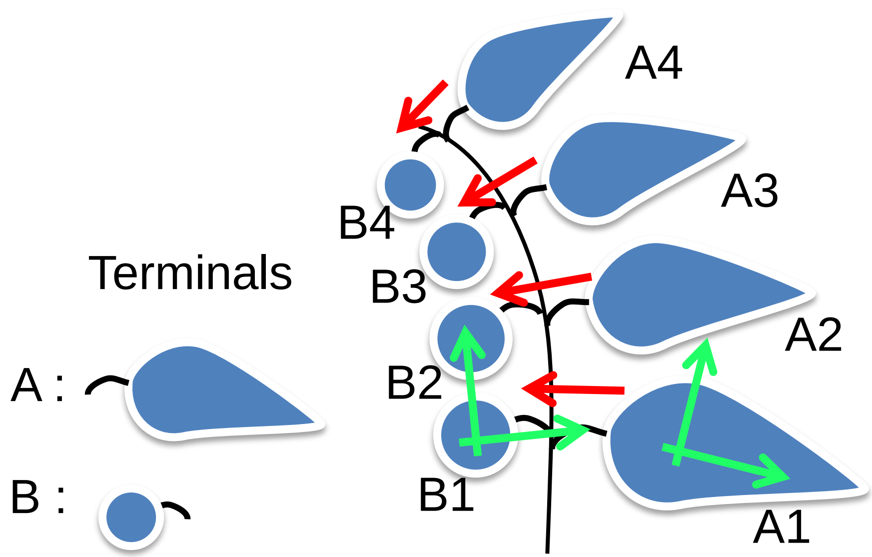
$$m=0 \rightarrow [P1]$$

$$S \rightarrow T_s [P2(3)]$$

Transformation Analysis



- Procedural rule
  - Transformation between two symbols



L-system 2

$$P1(m) \rightarrow [A]\ T_1\ [B]$$

$$P2(m)\ :m>0 \rightarrow [P1]\ T_2\ P2(m-1)$$

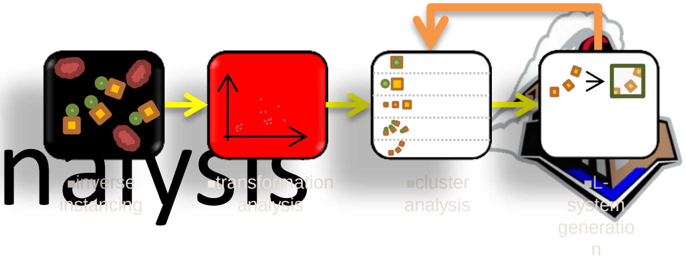
$$m=0 \rightarrow [P1]$$

$$S \rightarrow T_s\ [P2(3)]$$

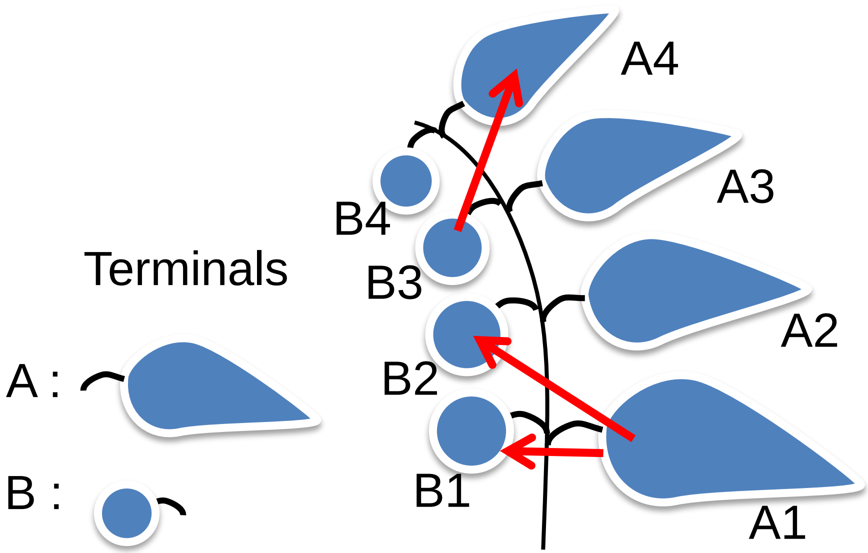
$$-(\alpha)\ f(d) * (s) - (\beta)$$

Transformation between two coordinate systems

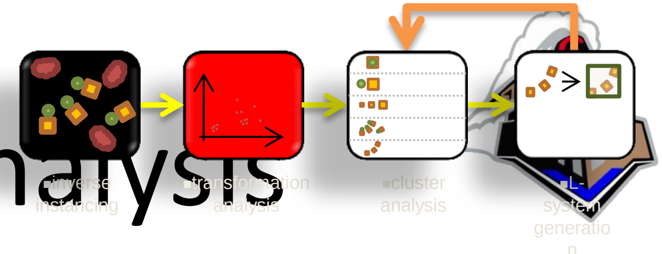
Transformation Analysis



- Many possible transformations
  - Use *significant* transformations for rules

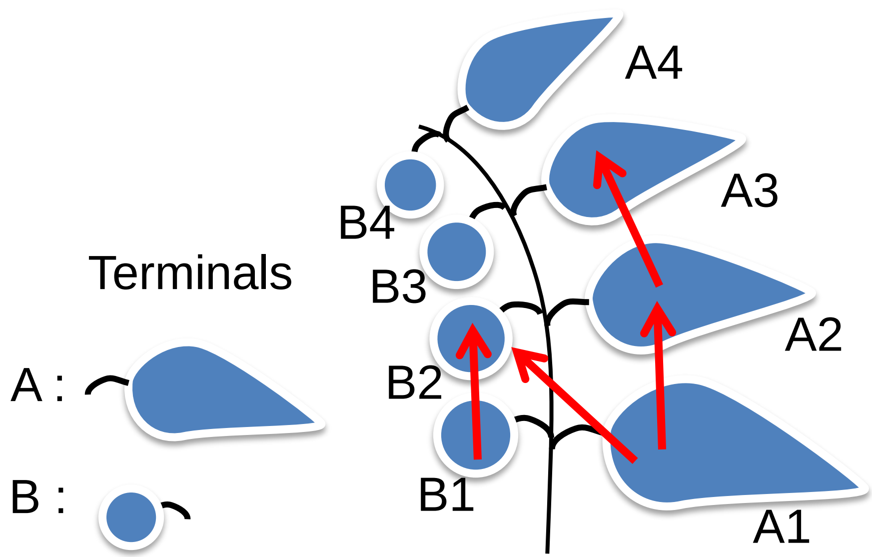


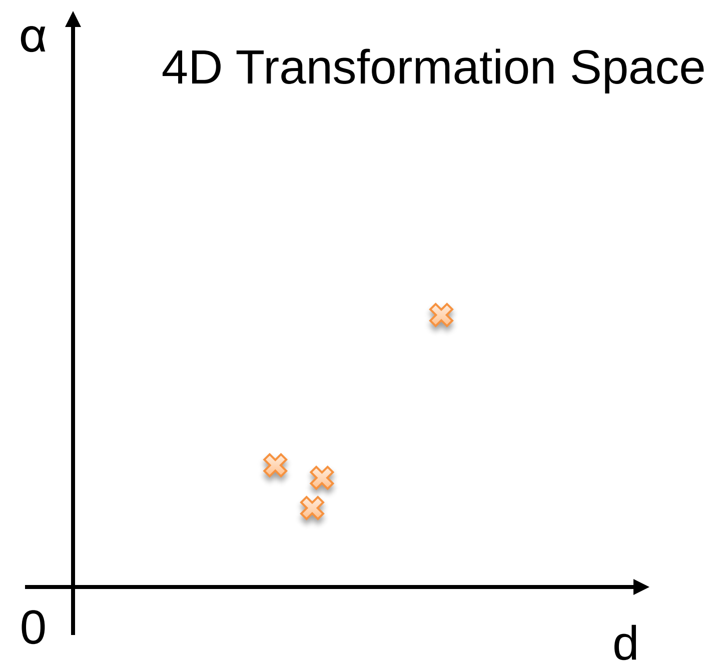
Transformation Analysis



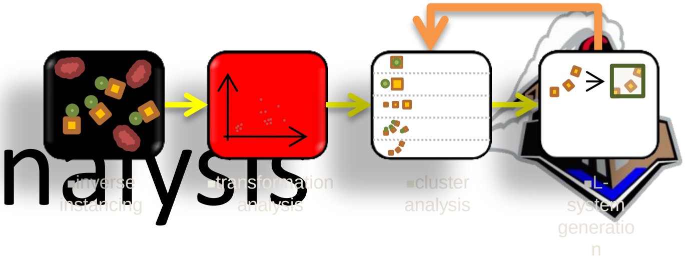
- Detect significant transformations
  - Put all transformations into Transformation space

Transformation = 4D Vector (2D transl., rotation, scale)

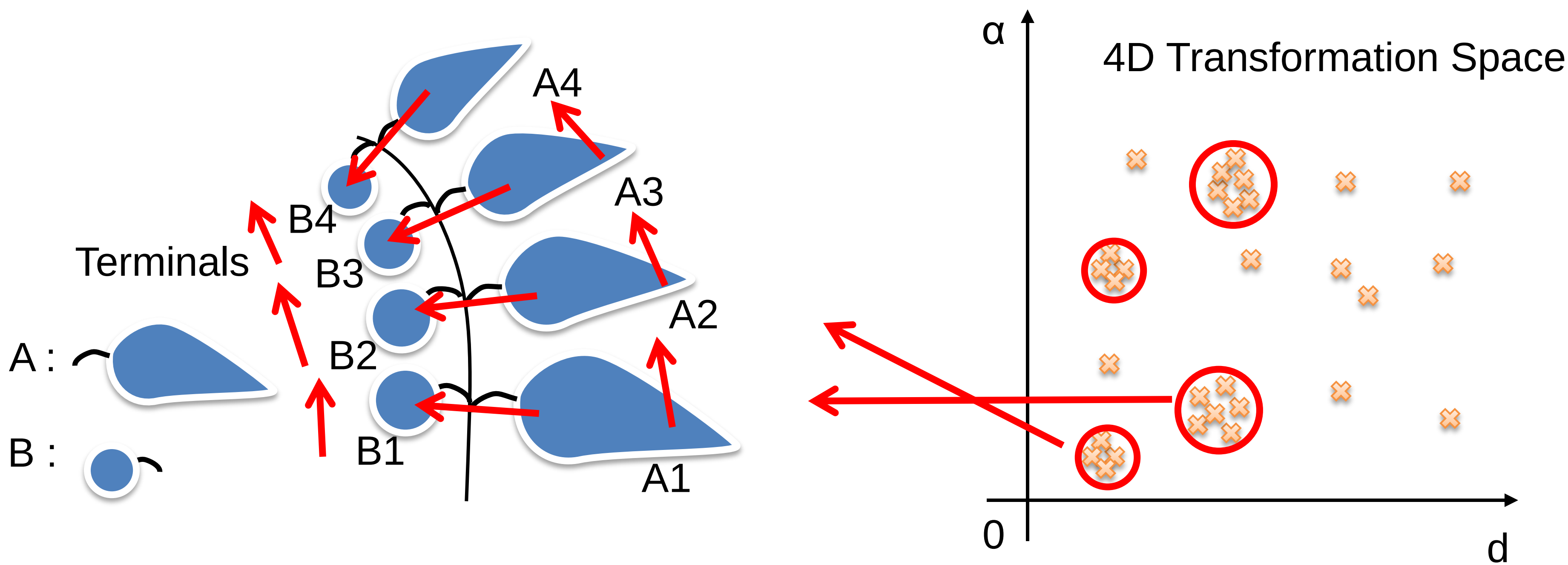




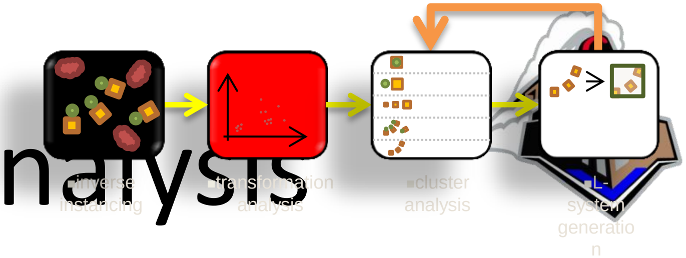
Transformation Analysis



- Clustering in the transformation space
  - Large clusters ~ significant transformations



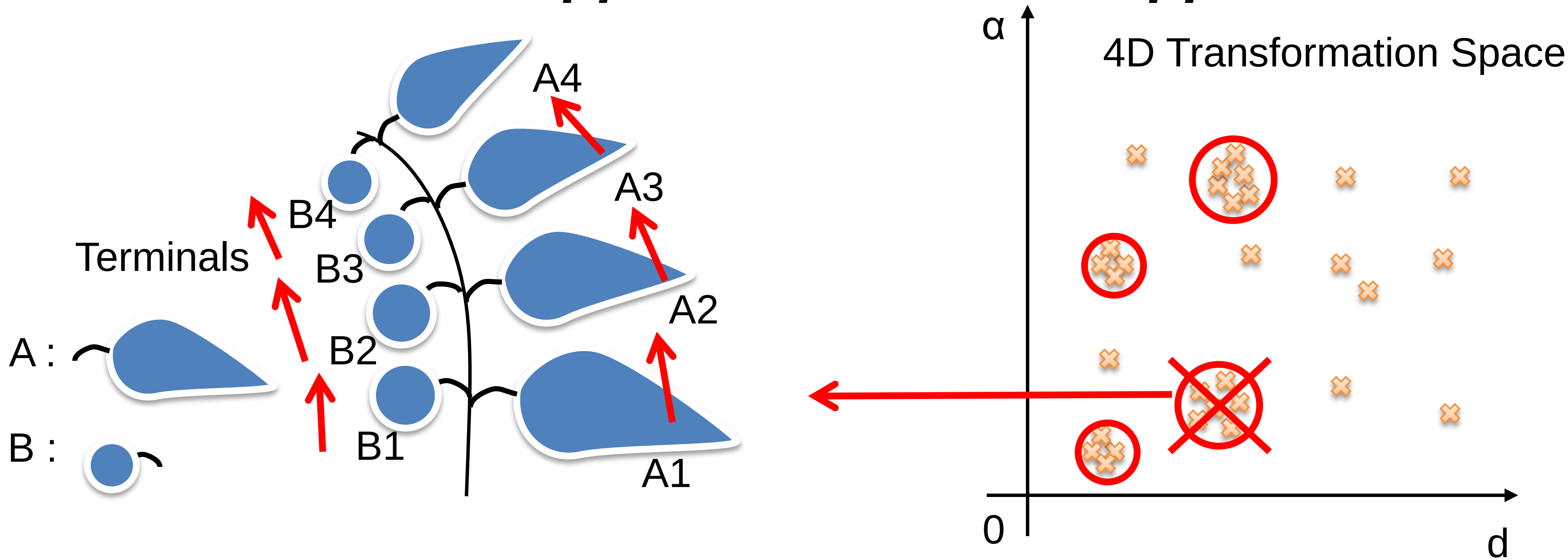
Transformation Analysis



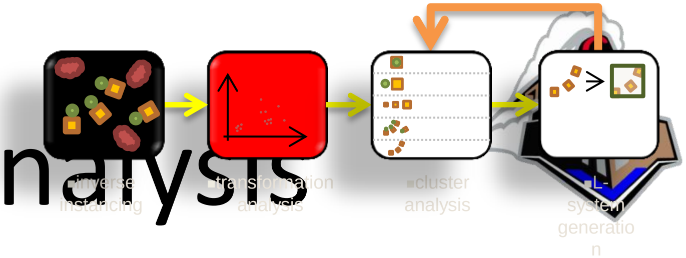
- One rule might not represent one cluster
  - All transformations must be between same symbols

$P2(m) : m>0 \rightarrow [B]\ T_2\ P2(m-1)$ $P1(m) : m>0 \rightarrow [A]\ T_1\ P1(m-1)$

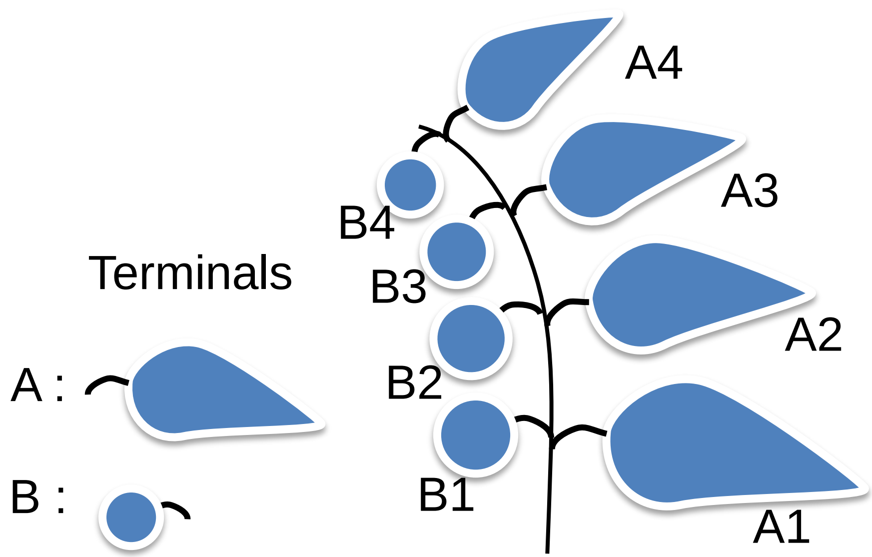
$m=0 \rightarrow [B]$ $m=0 \rightarrow [A]$



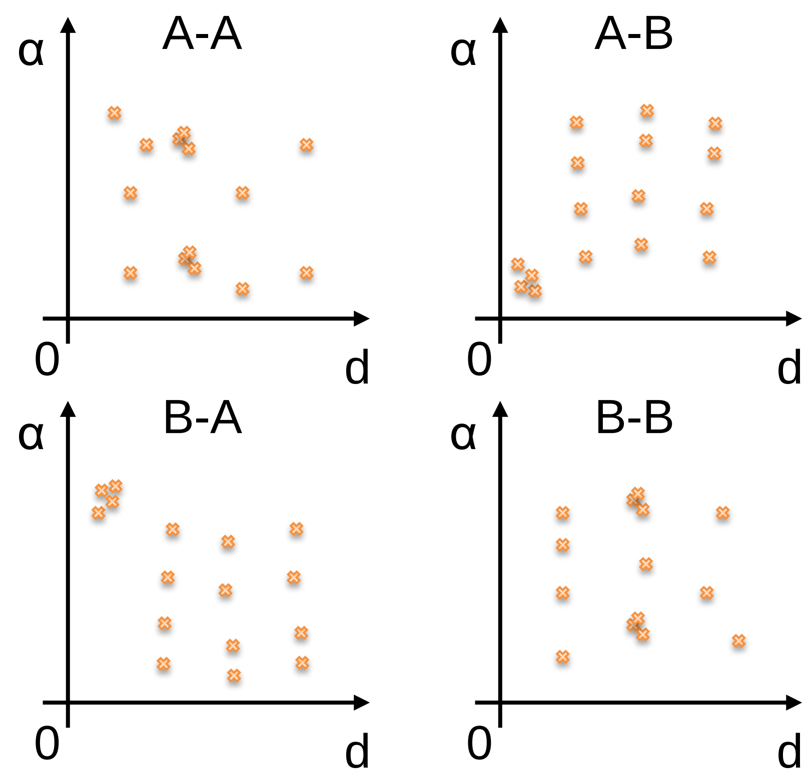
Transformation Analysis



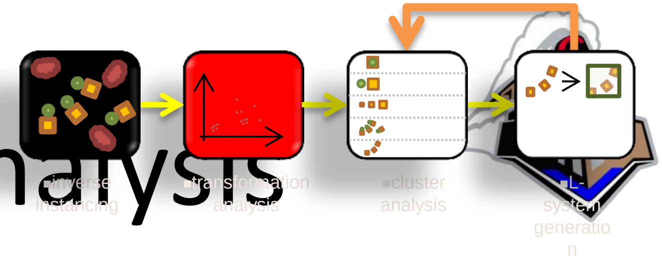
- One transformation space for each pair of terminal symbols



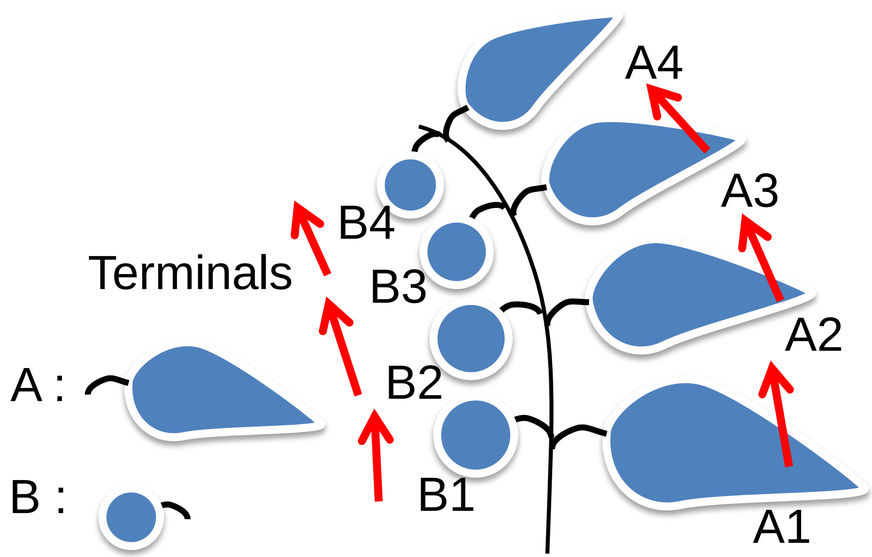
4x 4D Transformation Spaces



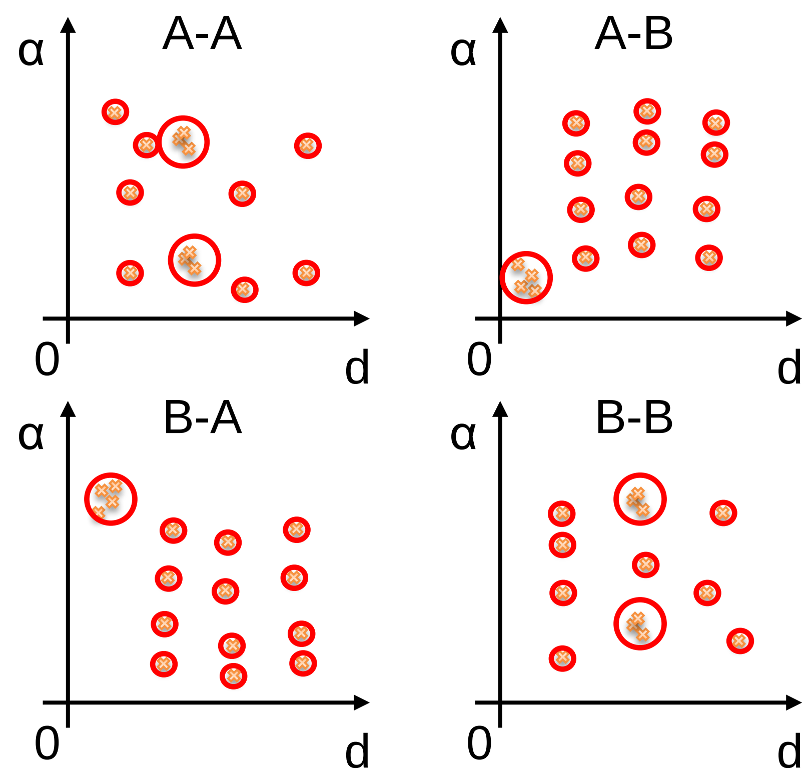
Transformation Analysis



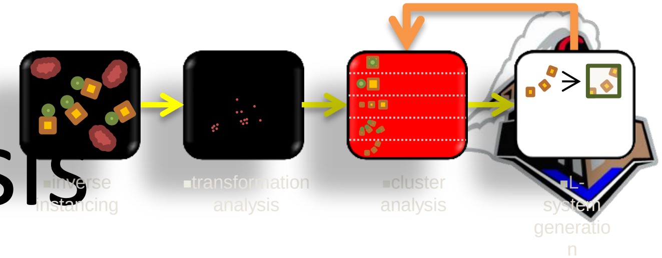
- Cluster = Transformations between the same symbols



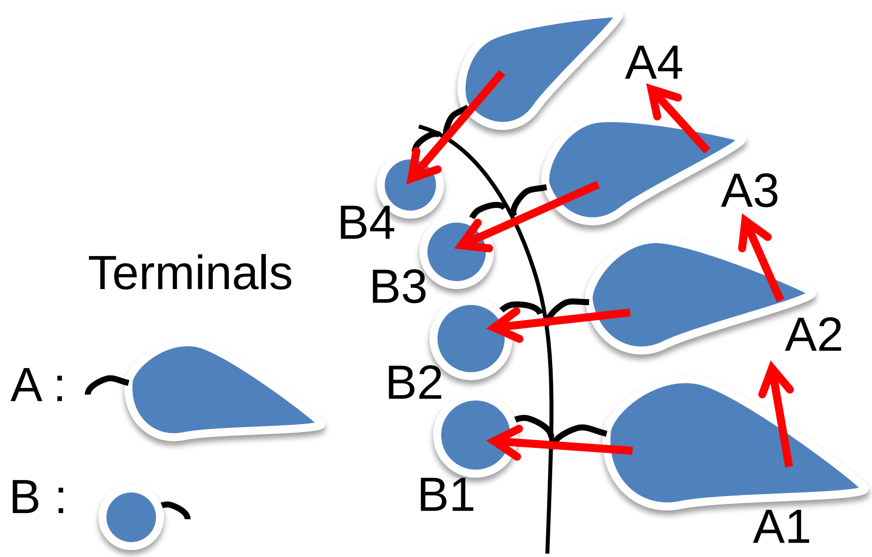
4x 4D Transformation Spaces



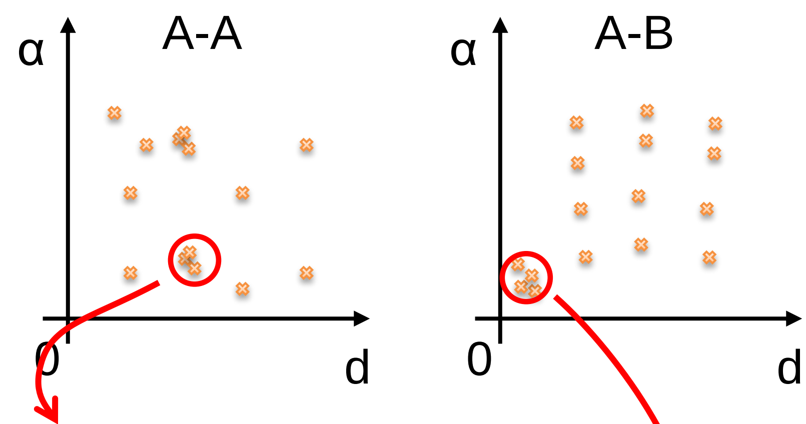
Cluster Analysis



- One cluster → One rule
  - Each rule is unique



4x 4D Transformation Spaces

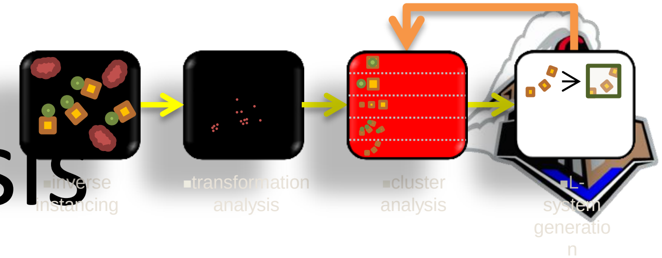


P(m) :m>0 → [A] T P(m-1)

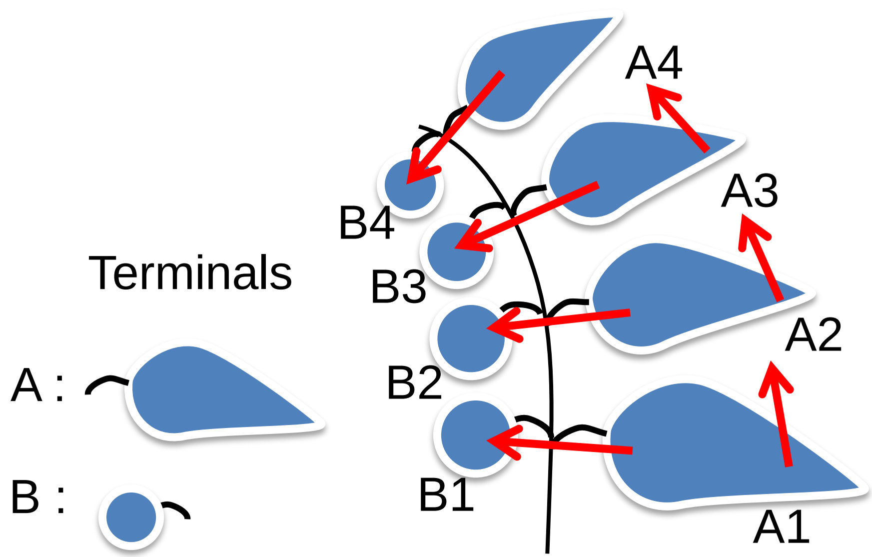
m=0 → [A]

P(m)→ [A] T' [B]

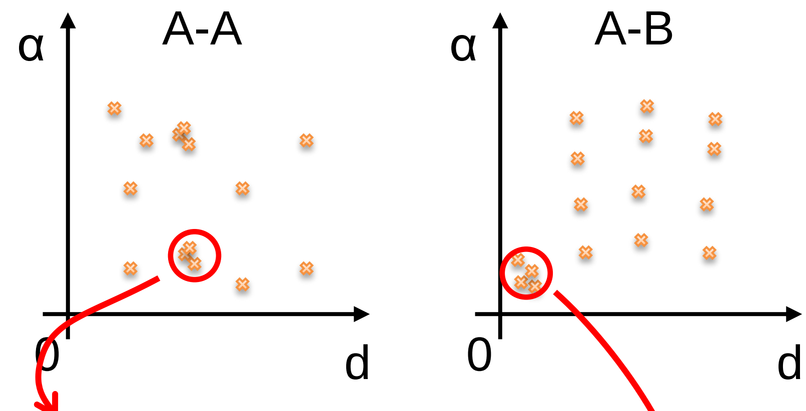
Cluster Analysis



- Rules are generated from clusters sequentially
  - Order of clusters is important



4x 4D Transformation Spaces

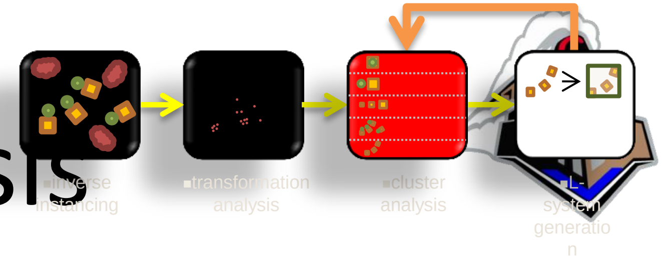


P(m) :m>0 → [A] T P(m-1)

m=0 → [A]

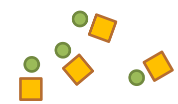
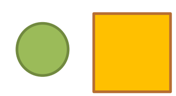
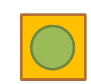
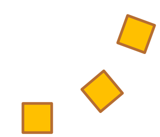
~~*P(m)→ [A] T’ [B]*~~

Cluster Analysis

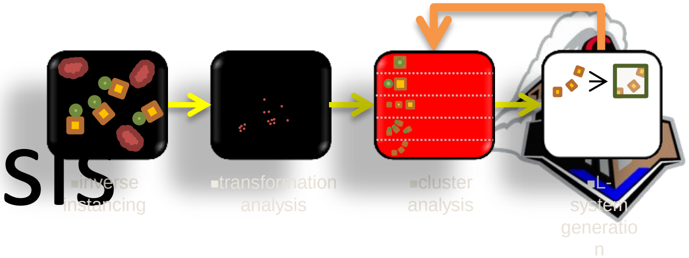


- Compute importance of each cluster
  - Weighted cluster importance function

$$w=w_n n + w_h h + w_\Phi \Phi + w_l l$$

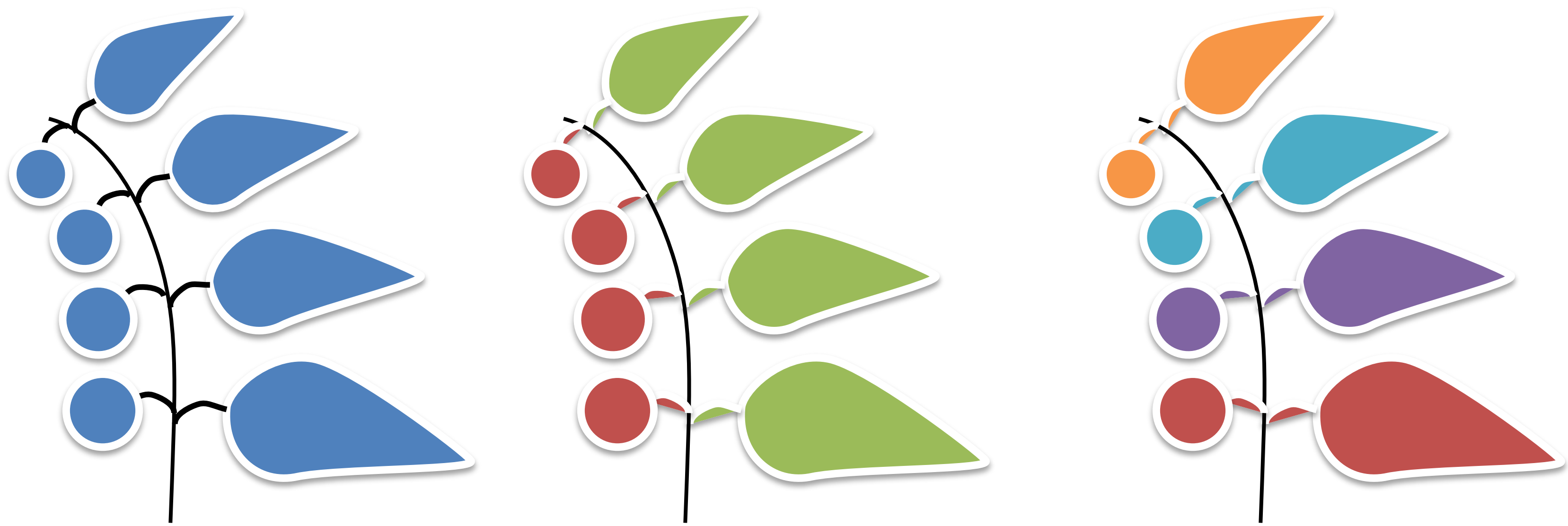
  - n: number of points in the cluster 
  - h: proximity of two elements 
  - *Φ:* similarity between terminals 
  - l: average length of the sequences in a cluster 
- Sort clusters according to their importance

Cluster Analysis

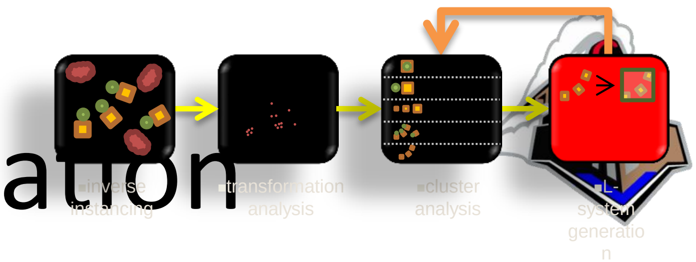


– Weighted cluster importance function
  - Weights determine the final rules

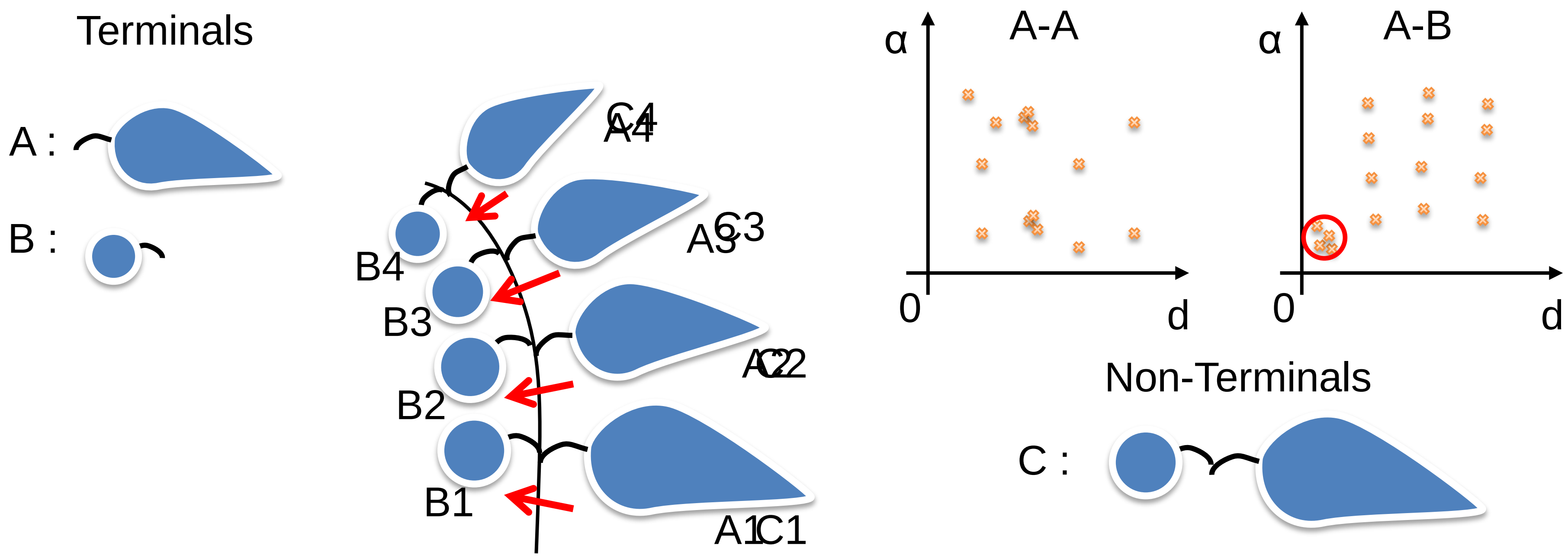
Prefer sequences *Prefer proximity*



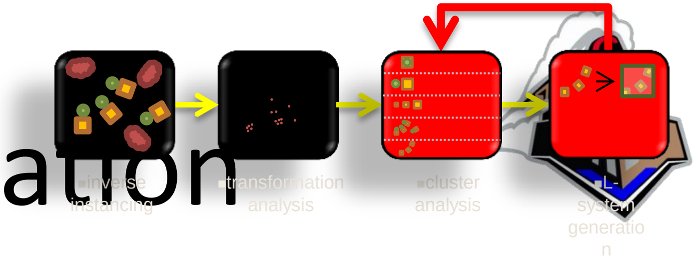
L-system Generation



- New rule = new non-terminal symbol

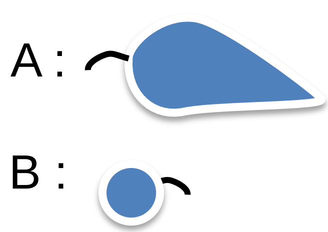


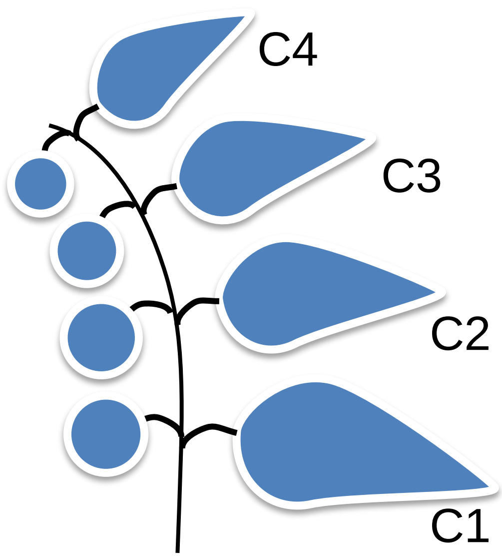
L-system Generation

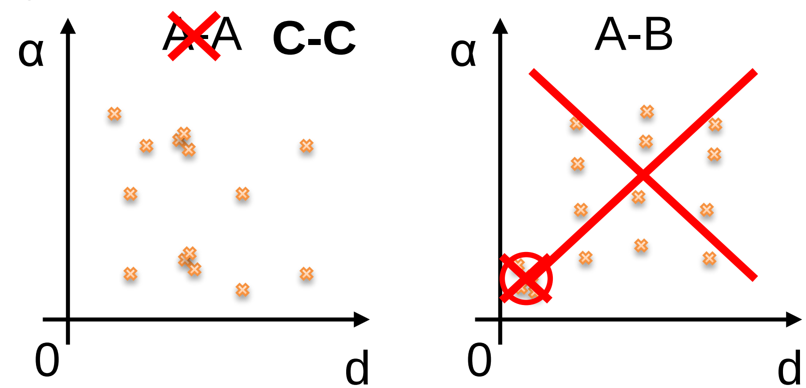


- Clusters no longer valid
  - Update them using the new non-terminal symbol
  - Compute importance of updated clusters

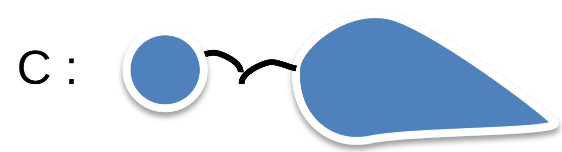
Terminals

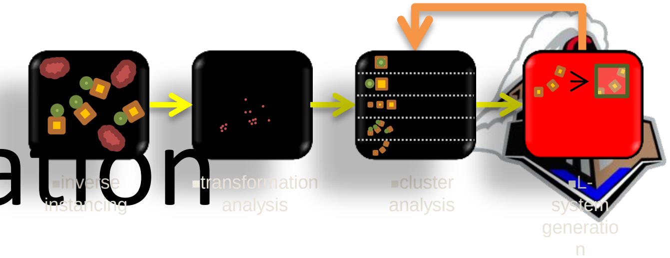






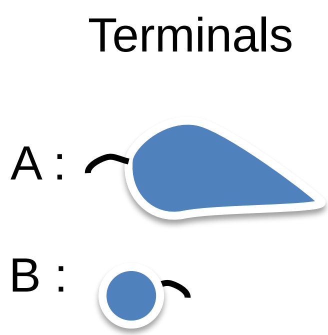
Non-Terminals

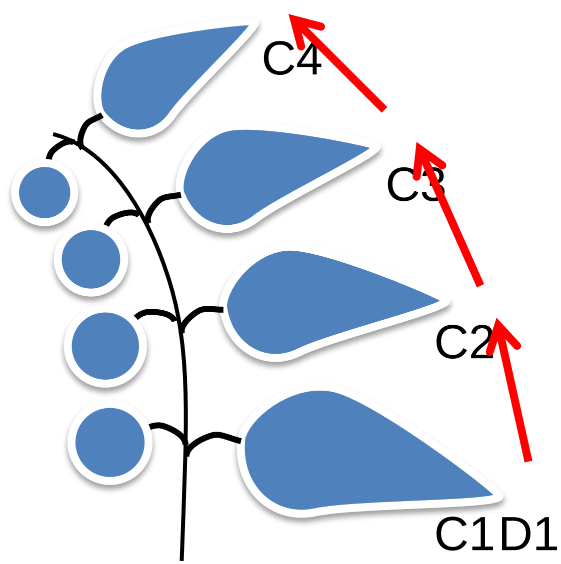


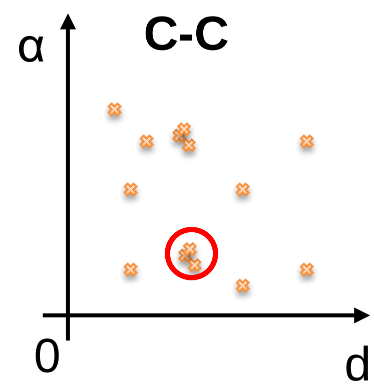


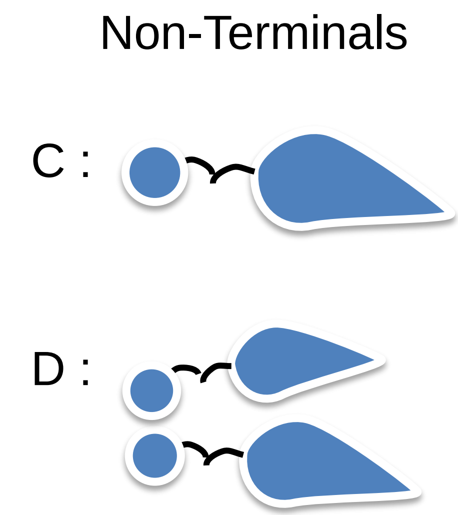
L-system Generation

- Generate new rules until there are no clusters
  - Axiom → Last non-terminal



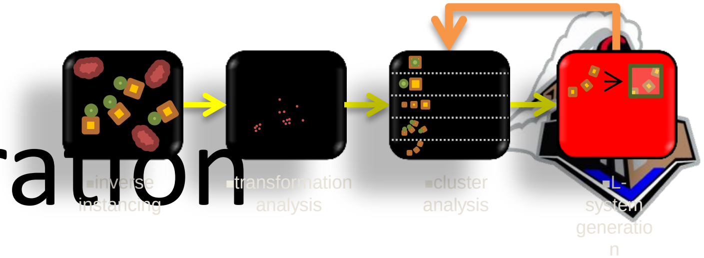






Axiom: $S \rightarrow T_S \, D(3)$

L-system Generation

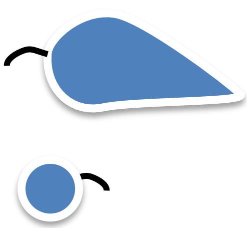


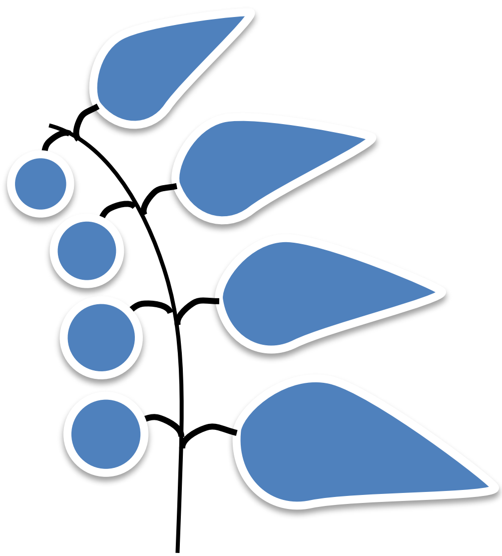
- Final L-system

Terminals

A :

B :





L-system

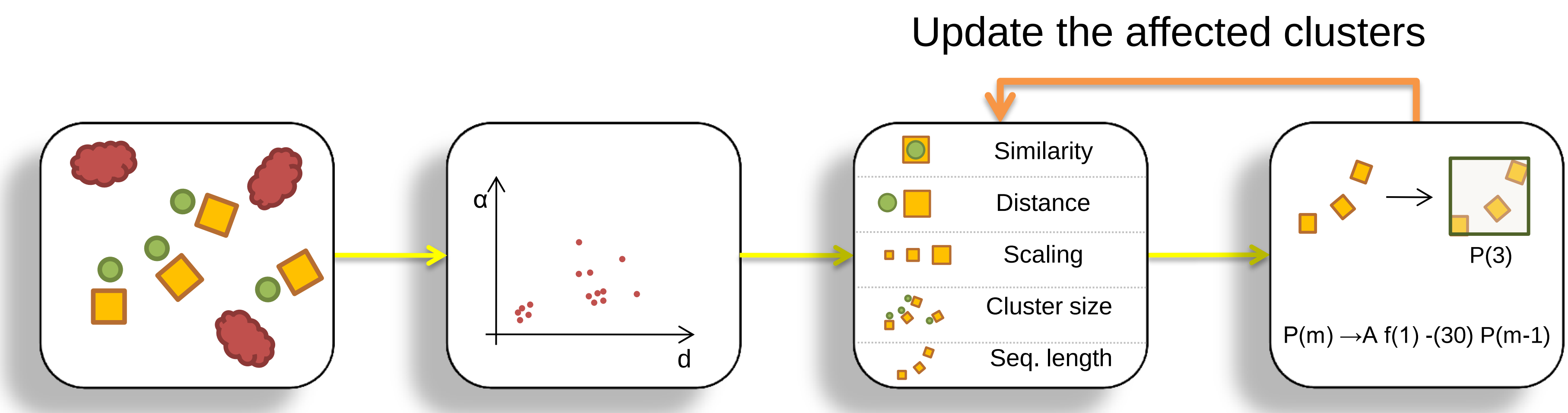
$$C(m) \rightarrow [A]\ T_1\ [B]$$

$$D(m) : m>0 \rightarrow [C]\ T_2\ D(m-1)$$

$$m=0 \rightarrow [C]$$

$$S \rightarrow T_s\ [D(3)]$$

Summary



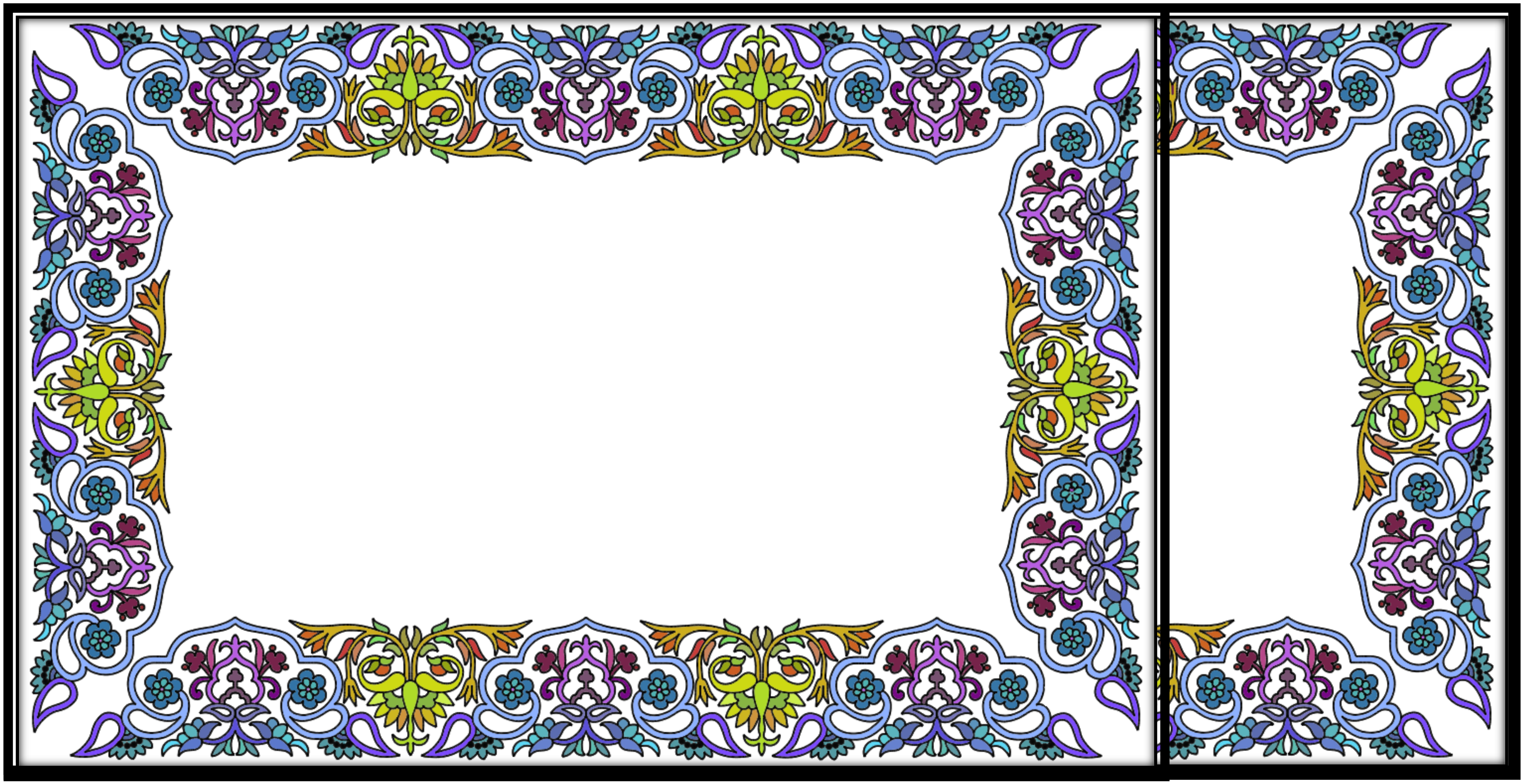
1. Calculate terminals by inverse instancing

2. Calculate transformations, fill transformation spaces, and perform clustering

3. Analyze clusters and calculate cluster significance

4. Create non-terminal symbols and L-system rules

Results



Results

Conclusion

- Important step towards the solution of the problem of inverse procedural modeling
- Key concepts
  - Multiple transformation spaces
  - Cluster analysis

Future Work

- General rules
  - More complex expression in the L-system rules
    - Polynomials
  - Context sensitivity
- 3D structures