Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computational Design Synthesis: Part I Spatial Grammars

Prof. Dr. Kristina Shea

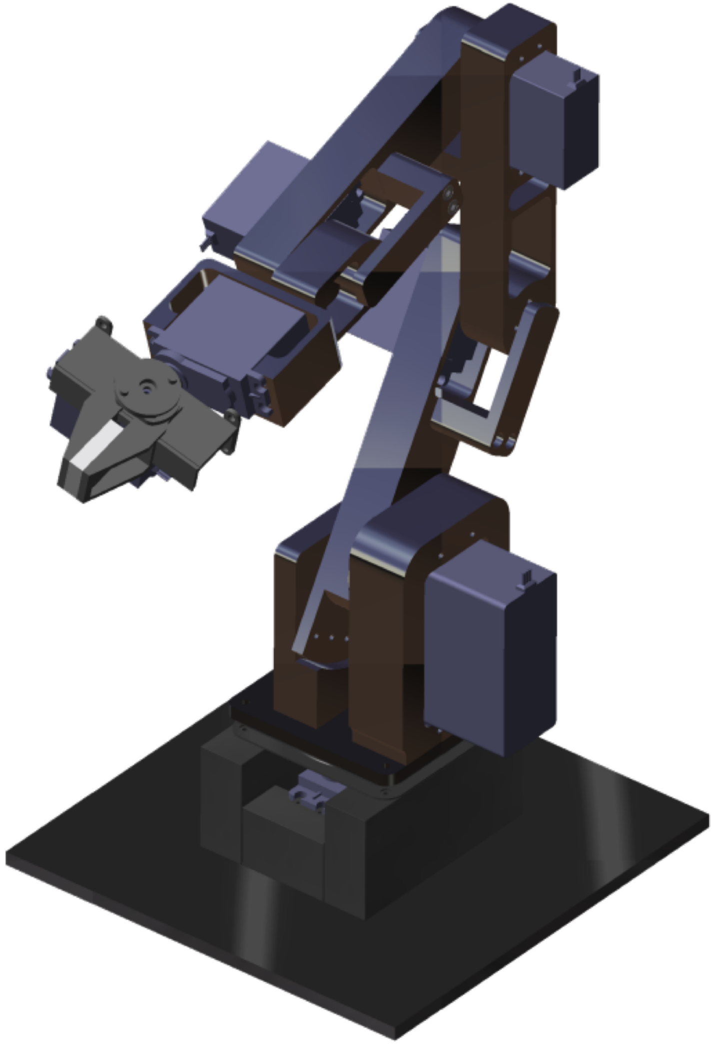
ENGINEERING
DESIGN
AND
COMPUTING

Outline

- Introduction to Computational Design Synthesis
- What is a grammar?
  - Definitions and notation
  - Classes of grammars
  - Languages
  - Spatial grammars
- *Spapper*
  - A Visual, Parametric, 3D Spatial Grammar Interpreter

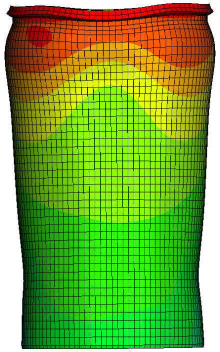
Challenges of Mechanical and Mechatronic Design Synthesis

- Multi-disciplinary involving mechanical, electronic and software components
- A large number of different functional and behavioral elements
- Strong dependencies between geometry, behavior and function
- Large number of different components
- Complex 3D geometry parts and assemblies
- Complex geometric constraints
- Strong dependency between design and fabrication



Synthesis vs. Analysis

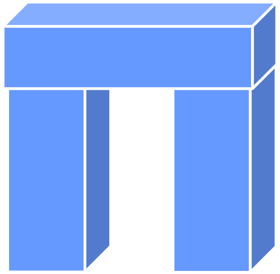
Analysis / Simulation



Resolution of a system into its elements and their interrelationships.

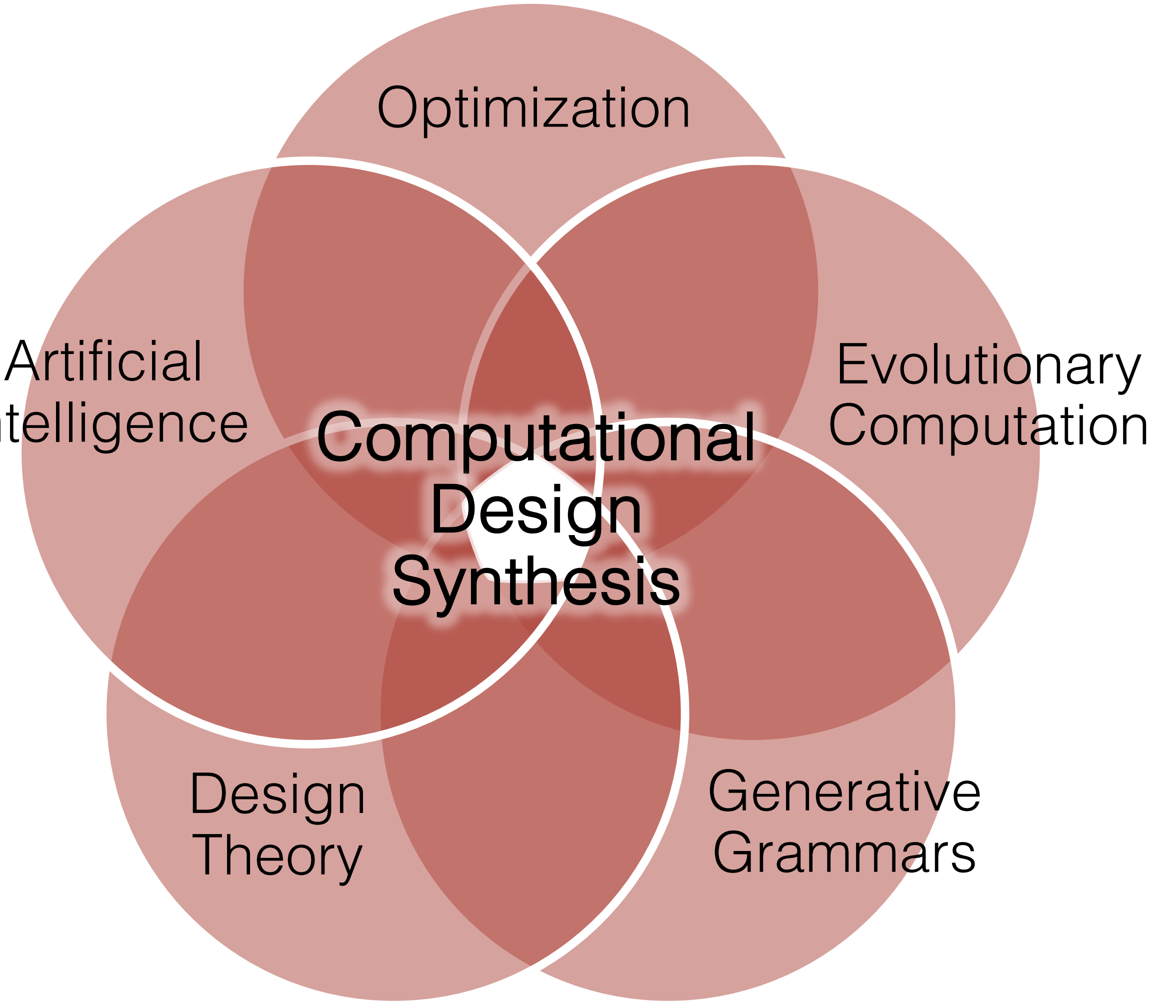
The construction of a mathematical model to reproduce the effects (behavior) of a phenomenon, system, or process.

Synthesis



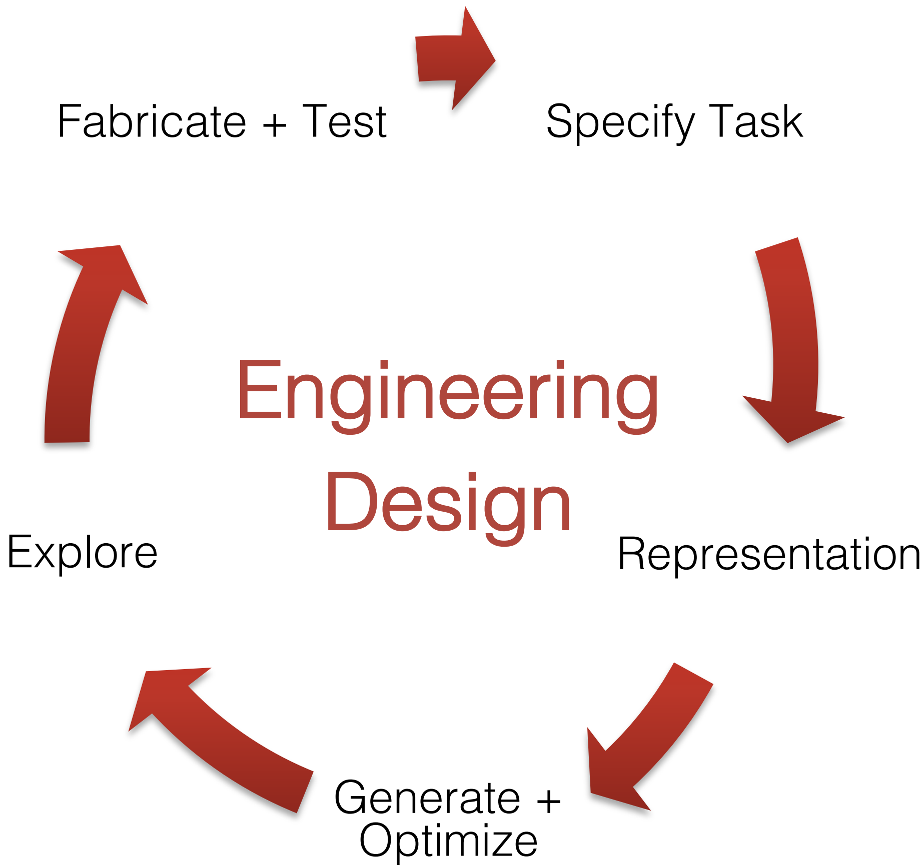
The design and combination of fundamental components, or building blocks, to produce a unified and often complex system that efficiently exhibits at least the required behavior.

A Brief History of Approaches

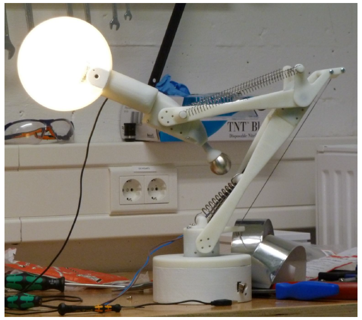


Process Overview

- **Specify** design tasks
- **Formally model and represent** solution spaces
- **Generate** feasible, "good" and optimized designs
- **Explore** complex solution spaces, constraints and preferences
- **Fabricate** and test optimized designs
- **Automate** design and fabrication process steps and processes
- **Spark** creativity and innovation



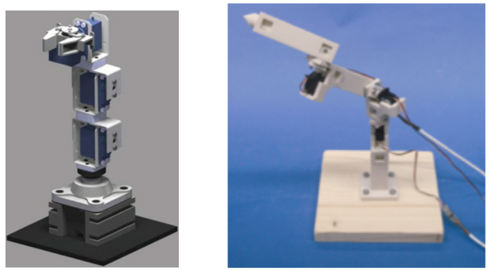
Computational Design Synthesis and Optimization



Fused Deposition Modeling

Fabricate + Test

Specify Task



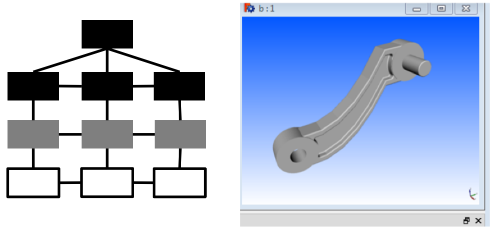
source: mimed, TUM

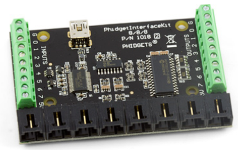
Automated Robot Synthesis and Optimization

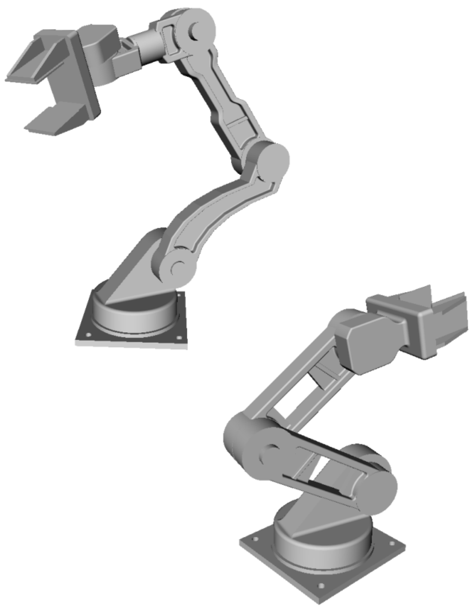
Explore

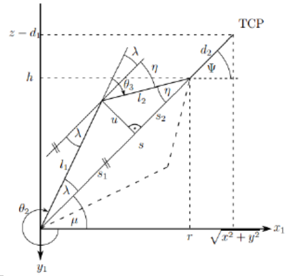
Represent

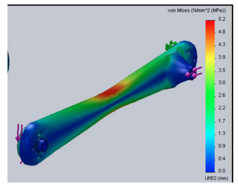
Generate + Optimize







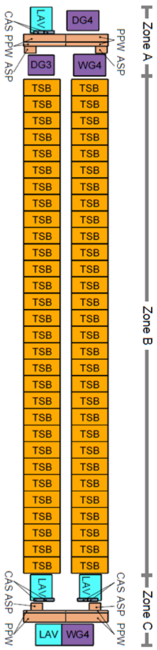


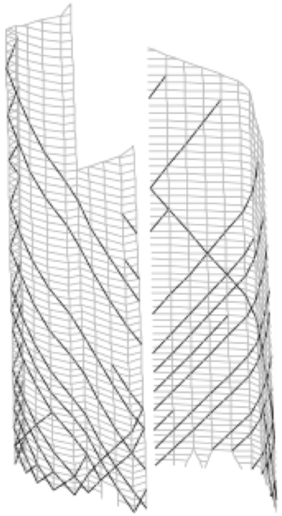


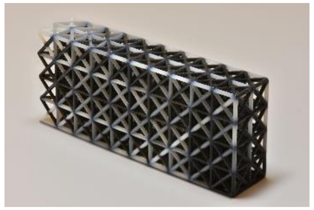
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

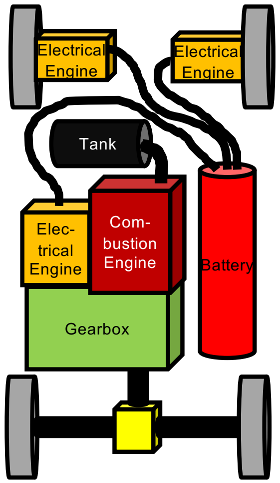
From Static Structures to Machines

"We hope that machines will eventually compete with men in all purely intellectual fields. But which are the best ones to start with? Even this is a difficult decision."
– Alan Turing, "Computing Machinery and Intelligence", 1950









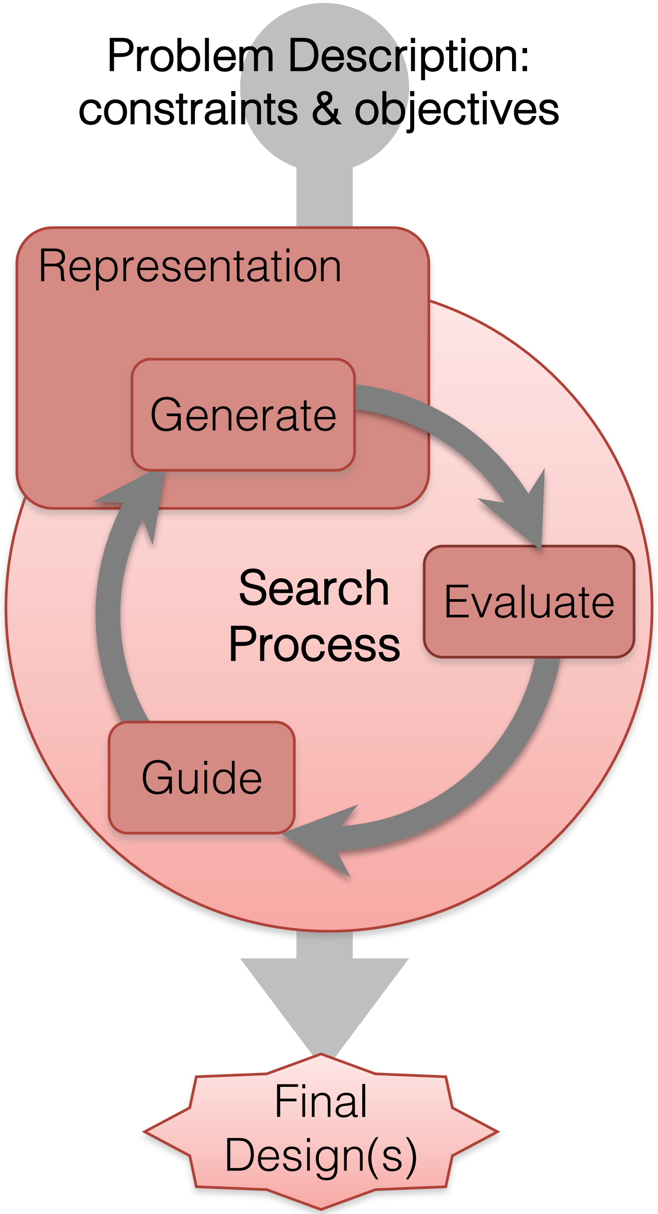
Increasing Variety of Building Blocks and Behaviors

Task Specification Increasingly Abstract and Complex

Computational Design Synthesis and Optimization

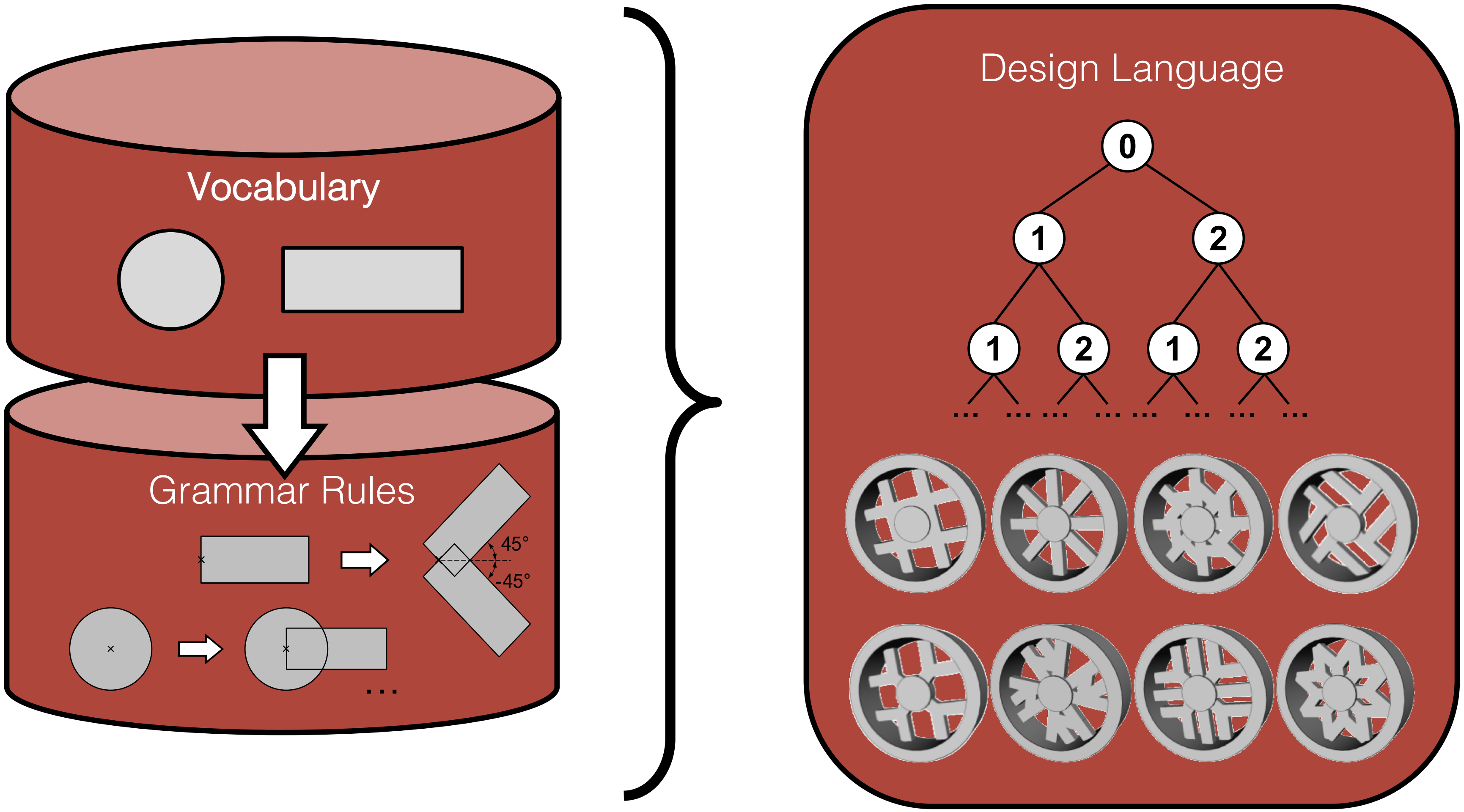
Four Aspects of Computational Design Synthesis

- How do we **represent** the set of all possible designs?
- How do we **generate** candidates based on that representation (problem solving)?
- How do we **evaluate** the quality of each candidate?
- How do we **guide** the search to better solutions?

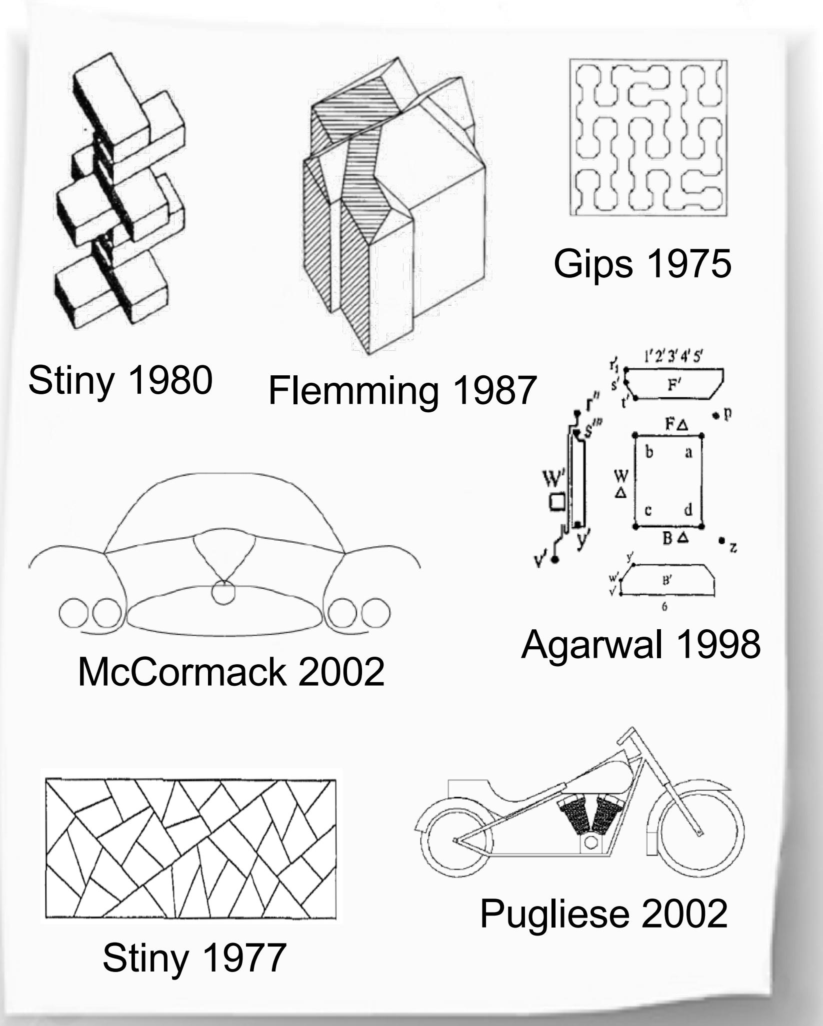


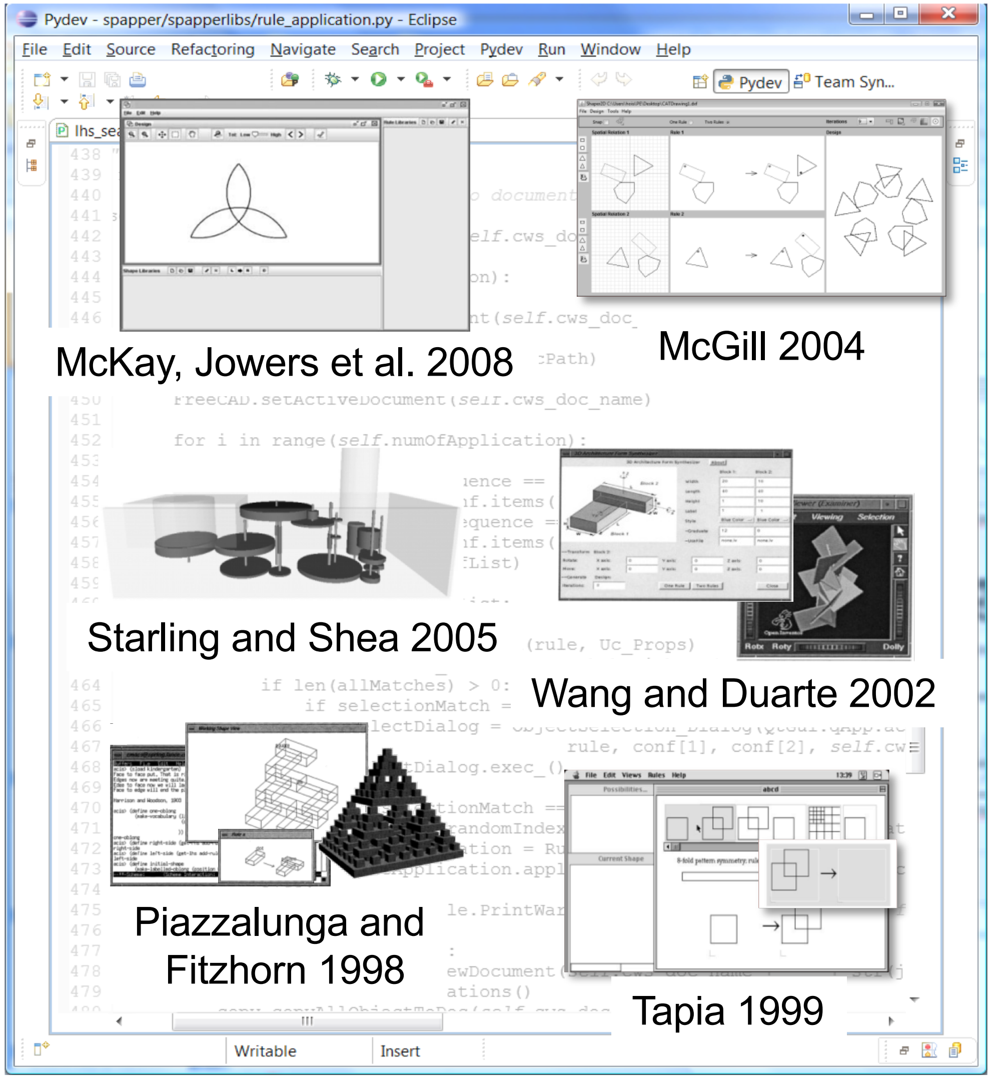
Curtsey of Prof. Matt Campbell

Generate - Engineering Design Grammars



Examples of Spatial Grammars





Arithmetic Expression Grammar

- How can we represent all mathematical expressions concisely?

1. <E> ⇒ number
2. <E> ⇒ (<E>)
3. <E> ⇒ <E> + <E>
4. <E> ⇒ <E> - <E>
5. <E> ⇒ <E> * <E>
6. <E> ⇒ <E> / <E>

Example

- We want to represent (4*3)+2
- start with an initial symbol <E>
- apply rule 3: <E> + <E>
- apply rule 2: (<E>) + <E>
- apply rule 4: (<E> * <E>) + <E>
- apply rule 1:3x: (4*3)+2
- finished: no rule applies

Rules

1. <E> ⇒ number
2. <E> ⇒ (<E>)
3. <E> ⇒ <E> + <E>
4. <E> ⇒ <E> - <E>
5. <E> ⇒ <E> * <E>
6. <E> ⇒ <E> / <E>

Grammar Terminology

- A grammar is defined as G = (N,T,R,I)

- N ≡ non-terminal symbols (metasymbols)
- T ≡ terminal symbols
- R ≡ a set of rules (productions)
- I ≡ initial symbol

Grammar Rule

- (lhs) u $\rightarrow$ v (rhs) where:
  - u is an object containing terminals and non-terminals
  - v is an object containing terminals or non-terminals

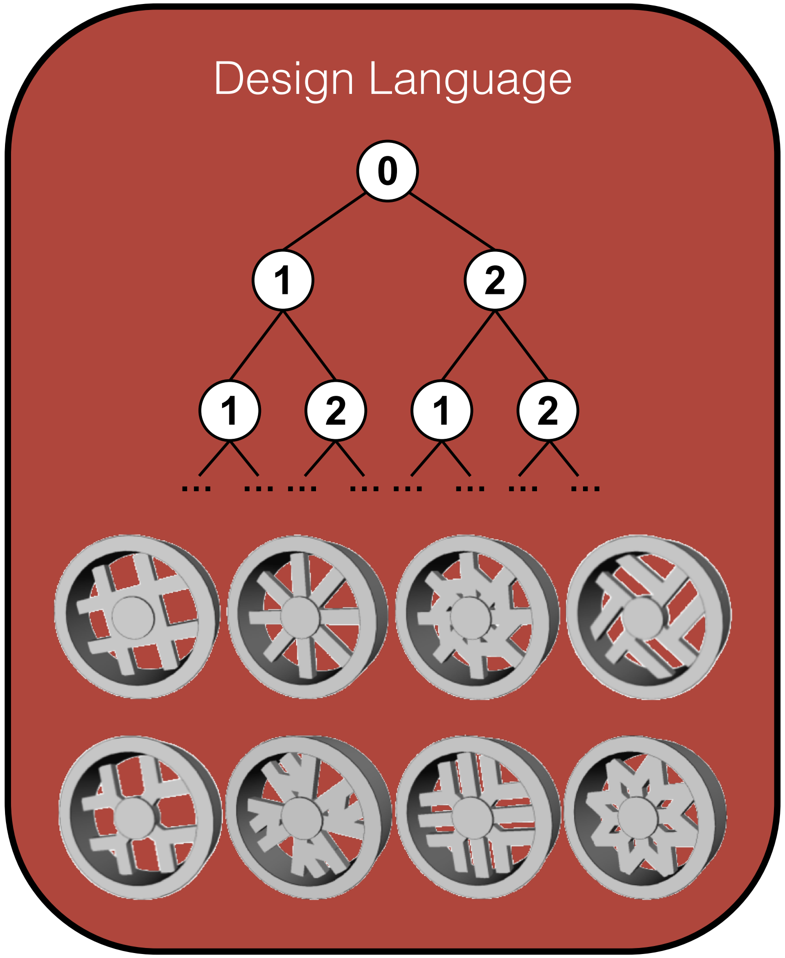
- rule application

  given object w rule u->v applies

  if $f(u) \leq w$ then $w' = [w - f(u)] + f(v)$

Applying Rules

- rule application
  - parallel
  - serial
- interpretive mechanisms
  - variable assignment
  - transformation
- matching relation (≤)
  - generally by sub-object



Defining a Language

- Recursive application of rules to generate all members
  - deterministic
  - non-deterministic

- Finite or Infinite

- Design implications
  - defines a searchable space
  - restricts search space to desired objects

Classes of Grammars

Grammar type	*Matching relation, ≤*	-	+	*Applications*
string	substring	string deletion	string insertion	linguistics, programming languages and compilers, machine design
set	subset	set difference	set union	product design, manufacturing
tree	frontier node	erase node label	add labeled subtree	pattern recognition
graph	subgraph	erase subgraph	insert subgraph	pattern recognition, solid modeling, 3D layout, structural layout, machine design
shape	subshape	shape difference	shape addition	spatial design & architecture

Generation or Parsing

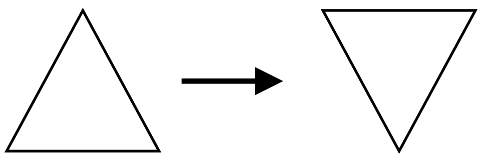
- rules within a grammar can be applied in both directions
- forward application generates members of a language
- reverse application can determine if an object exists within a language

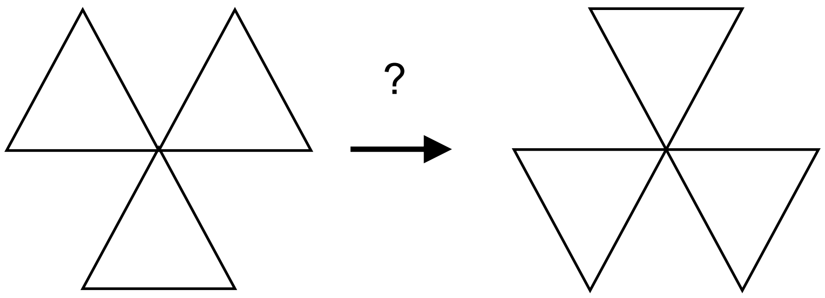
Shape Grammars (Stiny, 1981)

- G = (S,L,R,I)
  - S ≡ a set of shapes
  - L ≡ a set of labels
- matching: subshape
- unique features
  - maximal line representation
  - rule irreversibility
  - emergence

Maximal Lines and Emergence

- only “maximal” lines are represented
- maximal lines can be broken into an infinite number of pieces
- lines that are transformed into one another are re-represented as one line

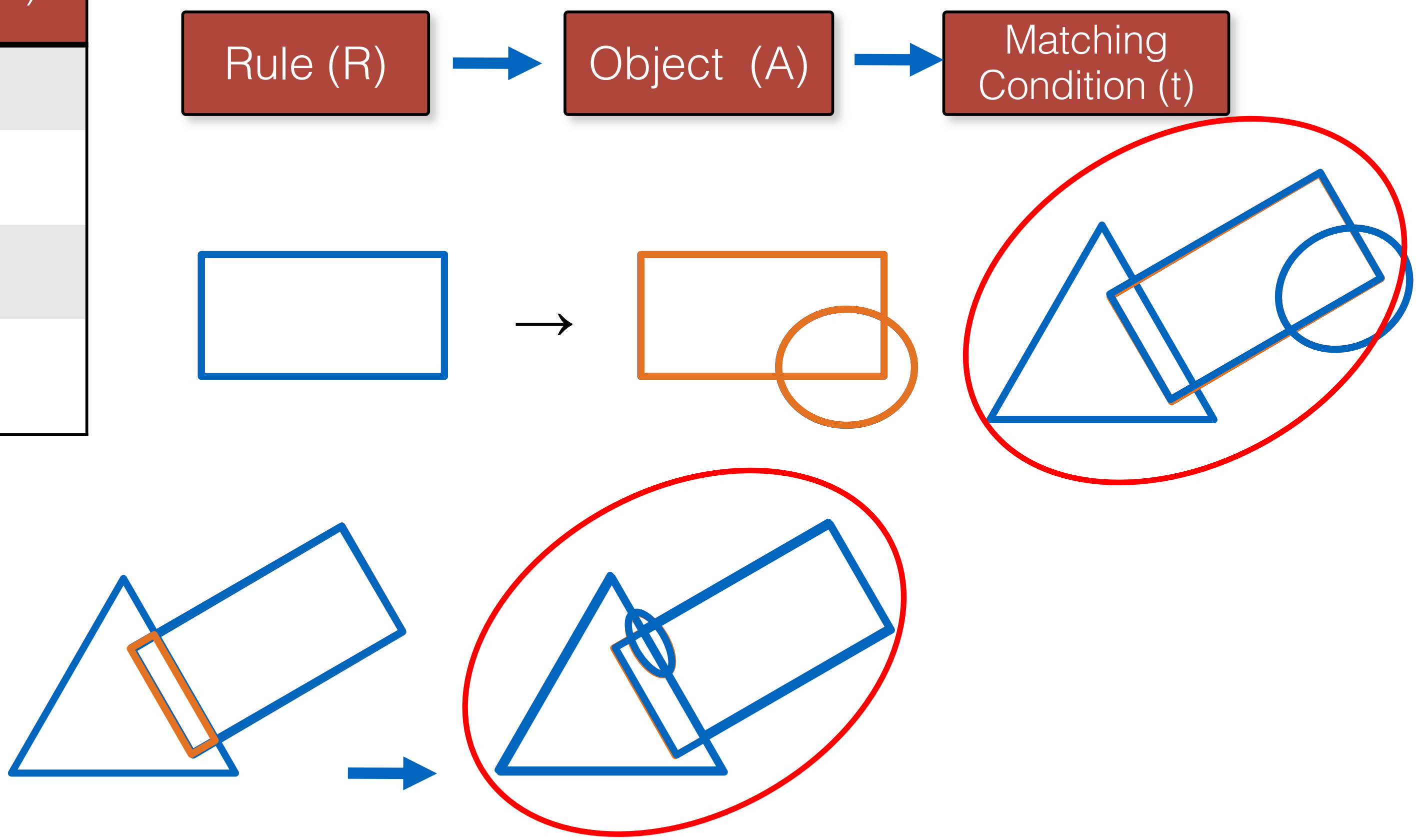




Spatial Grammars

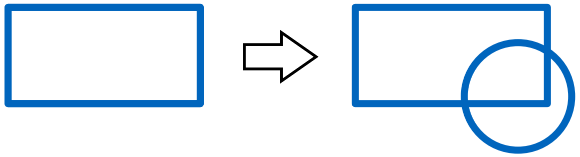
Rule R: $A \rightarrow B$ $C' = C - t(A) + t(B)$

Shape Grammar G = (S, L, R, I)	
S	finite set of shapes
L	finite set of labels
R	finite set of rules
I	the initial shape where I (S,L)^0 (vocabulary)

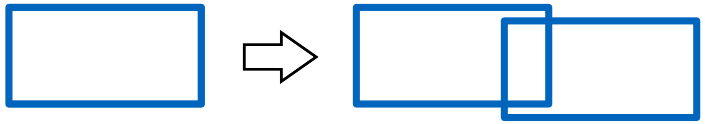


Rule Sequences

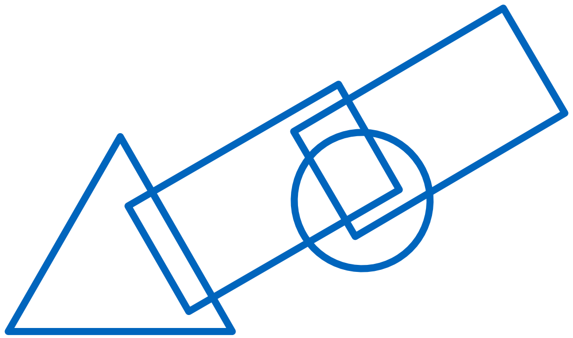
(1)



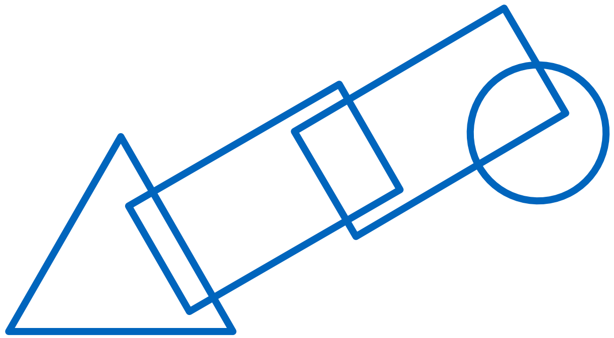
(2)



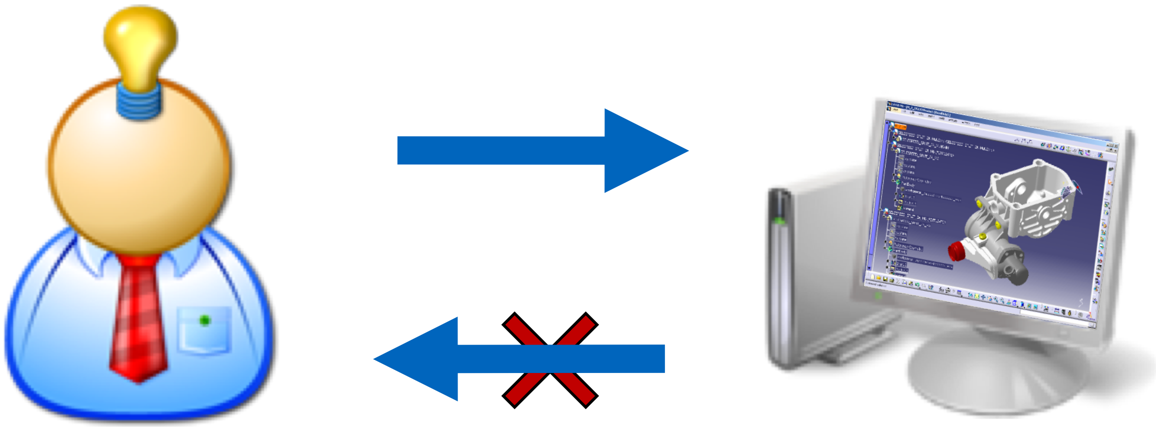
→ (1) → (2)



→ (2) → (1)



A Visual Spatial Grammar Interpreter



human designer

Spatial Grammars

CAD system

„Aided“ = assistance in creation, modification and documentation

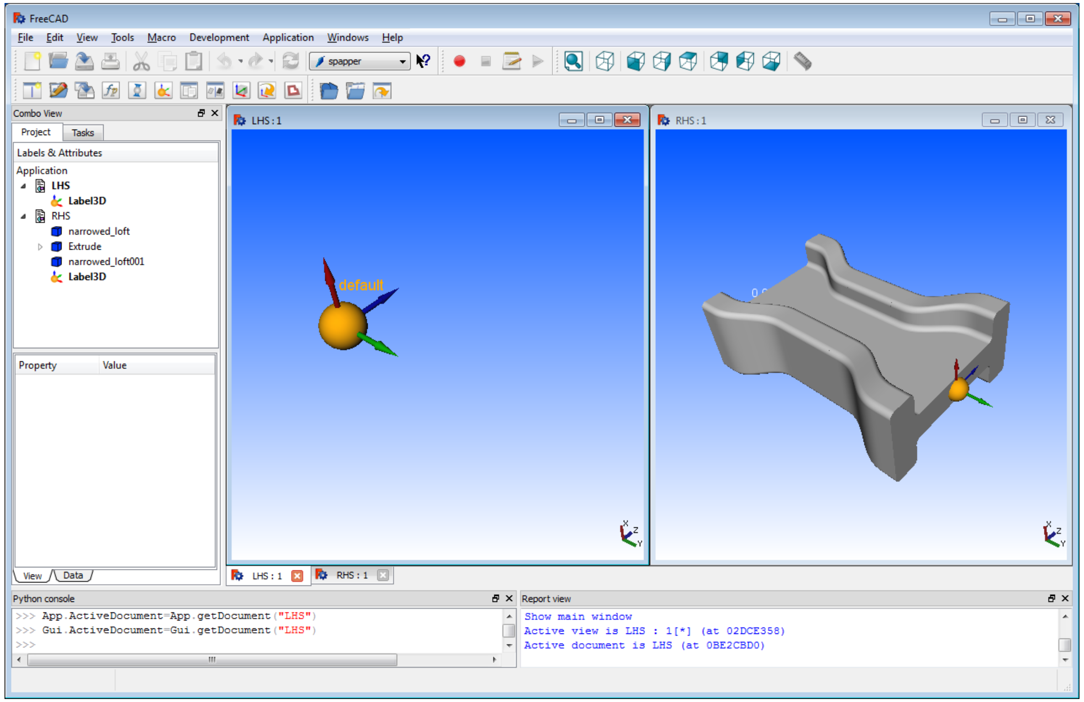
active support

„Aided“ ≠ active support

CAD-Based Generative Shape Design: Spapper

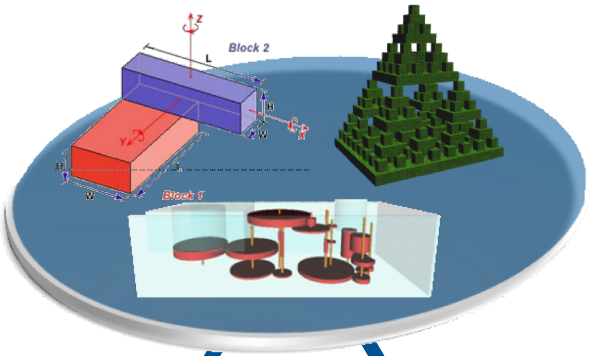
An interactive environment for parametric shape rule definition and generative design.

(source bottom: http://www.fanucrobotics.de/)

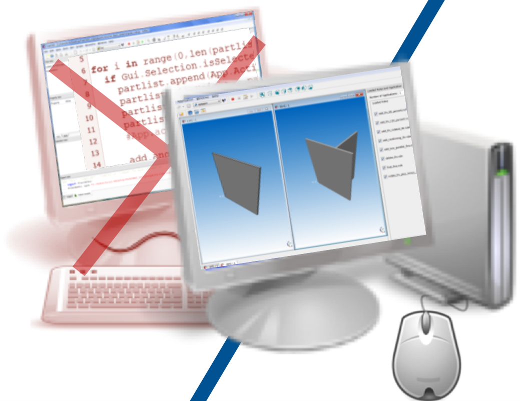


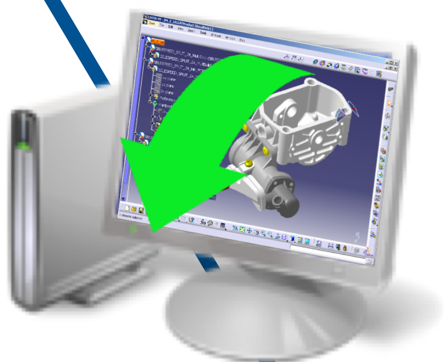
http://sourceforge.net/projects/spapper/

Spapper Goals



approach for creating a general 3D spatial grammar platform





visual rule development, no programming

interactive / automatic rule application => ‘active design partner’

Challenges

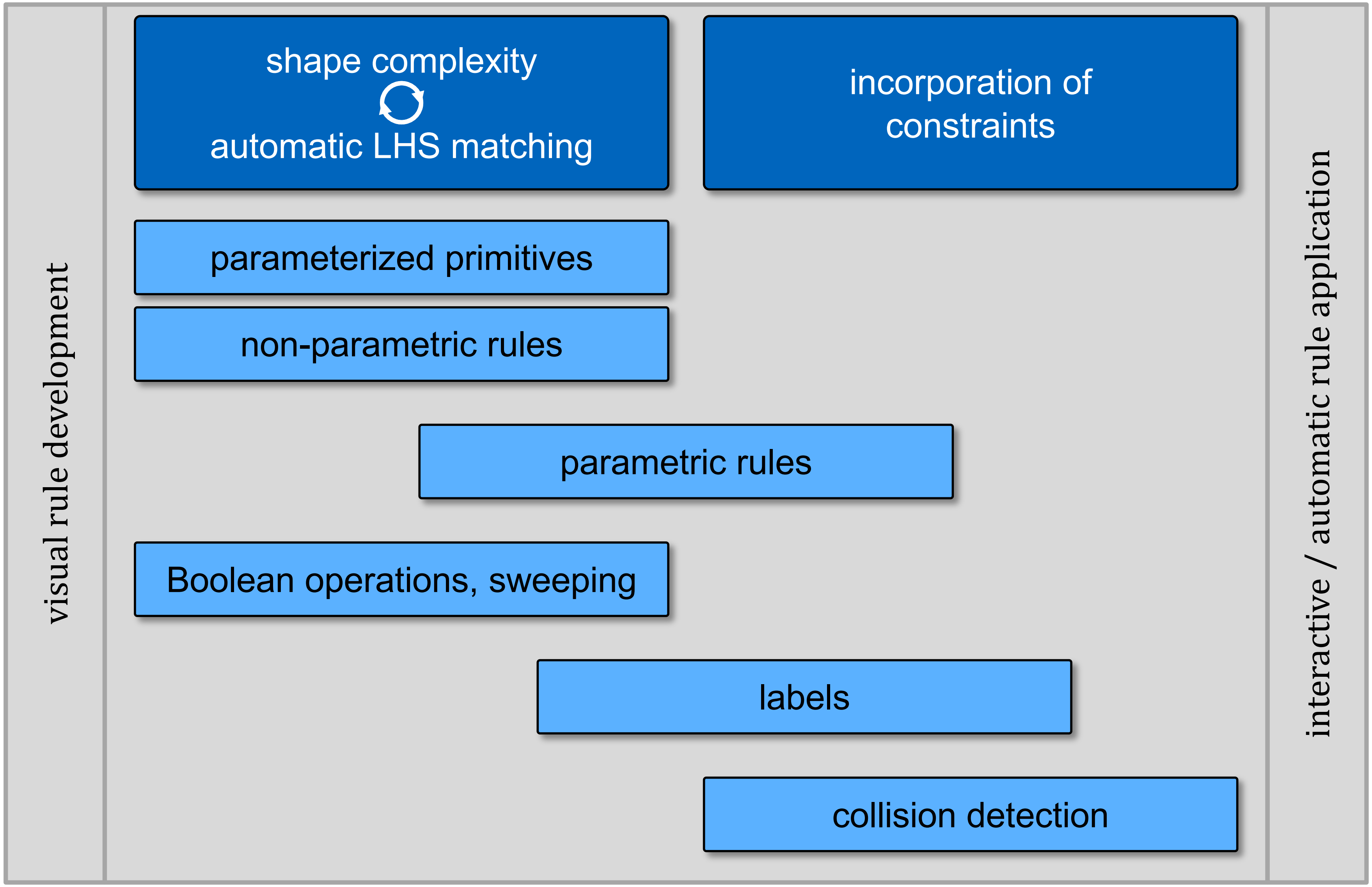
visual rule development

shape complexity
automatic LHS matching

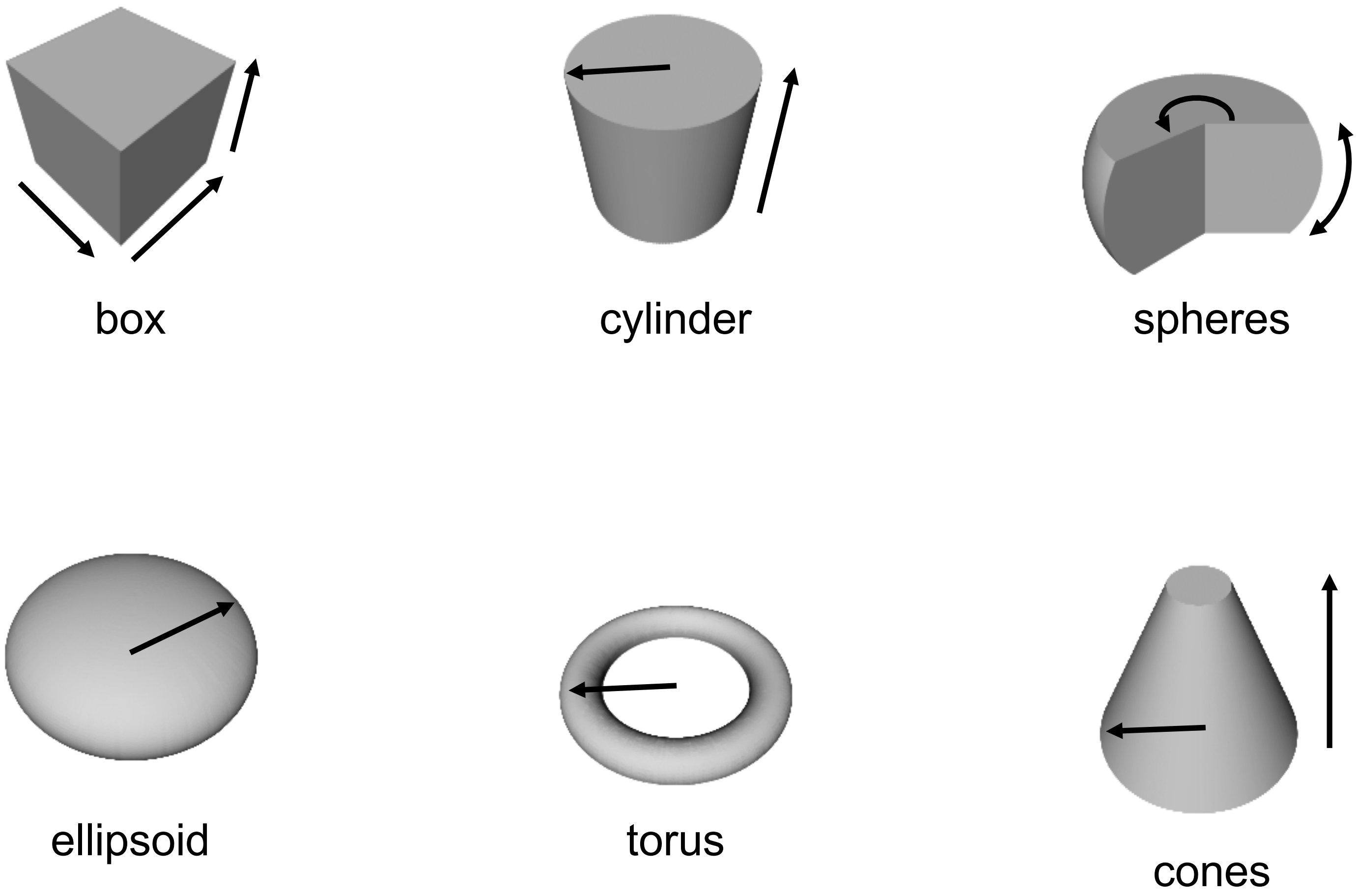
incorporation of constraints

interactive / automatic rule application

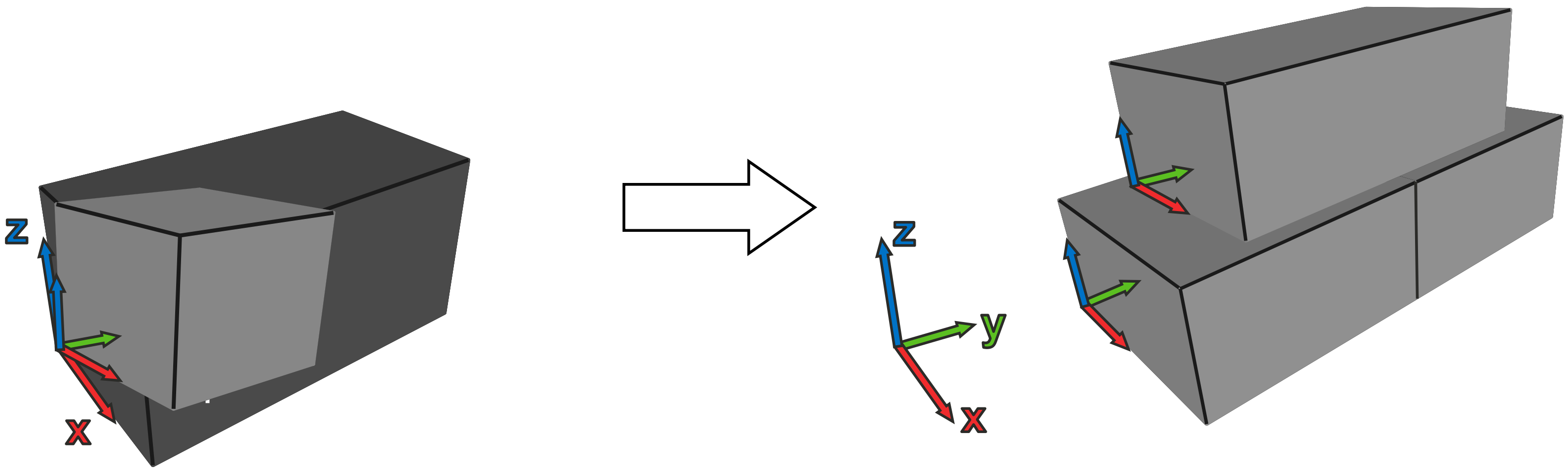
Approach – Overview



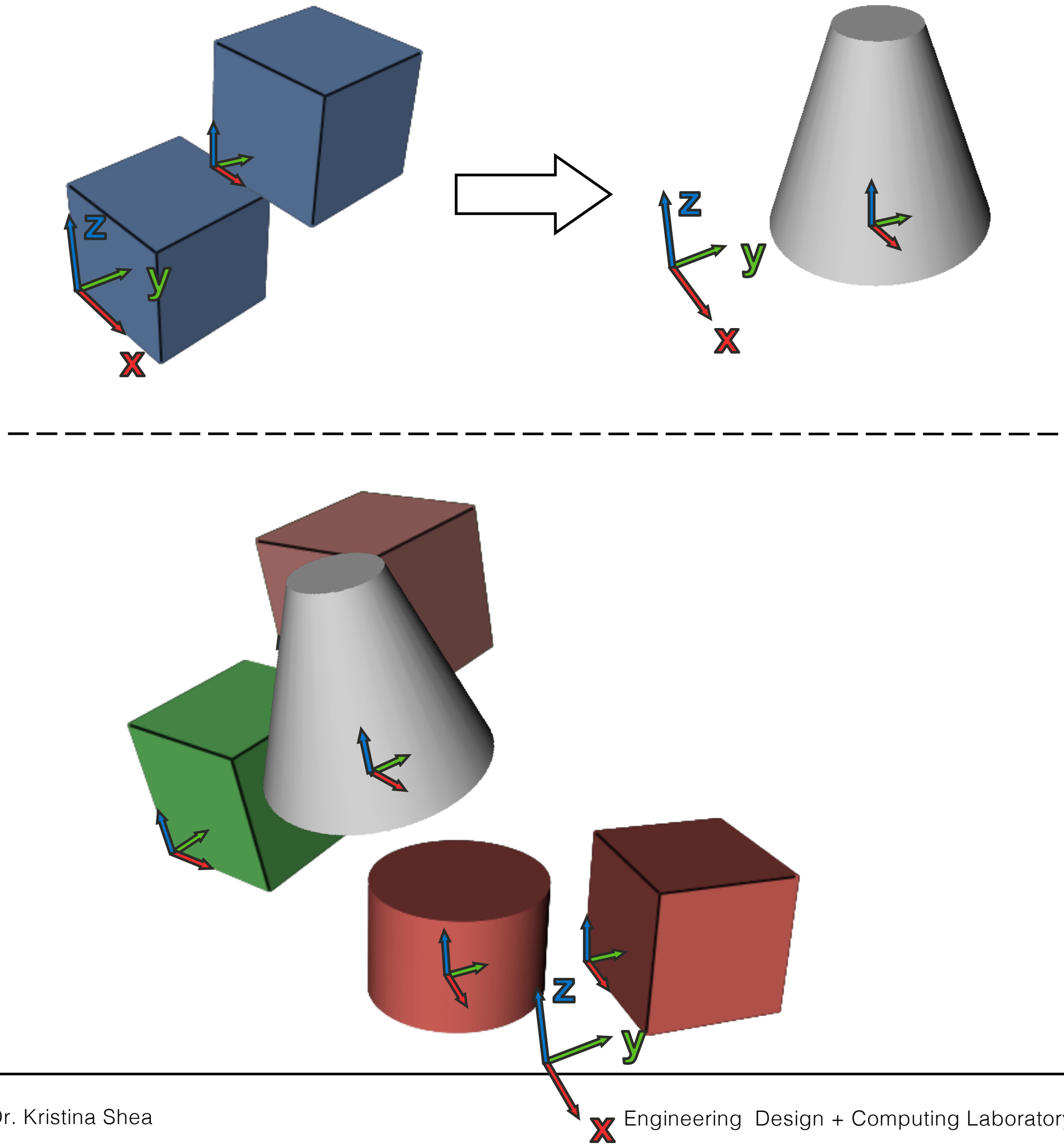
Set Grammar Formulation of Spatial Grammars

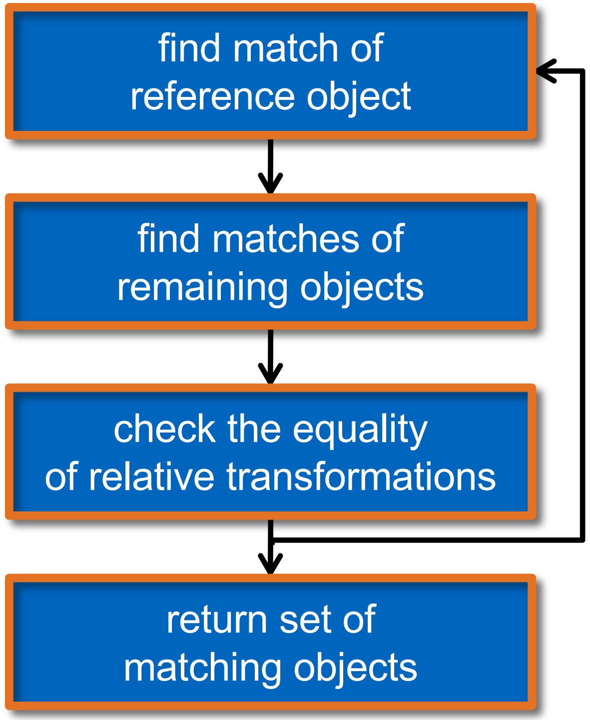


Rule Definition

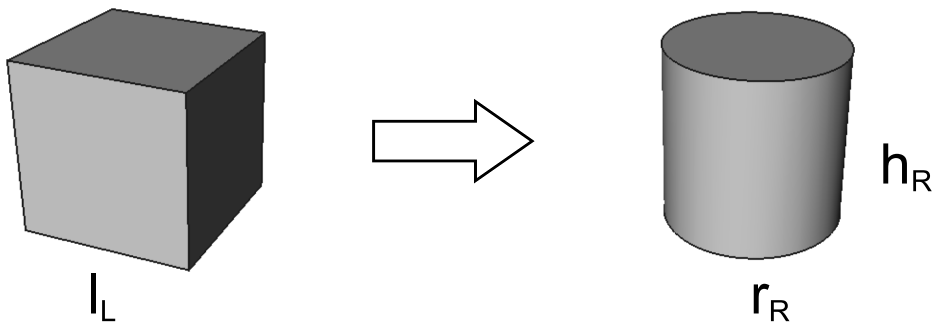


Rule Application



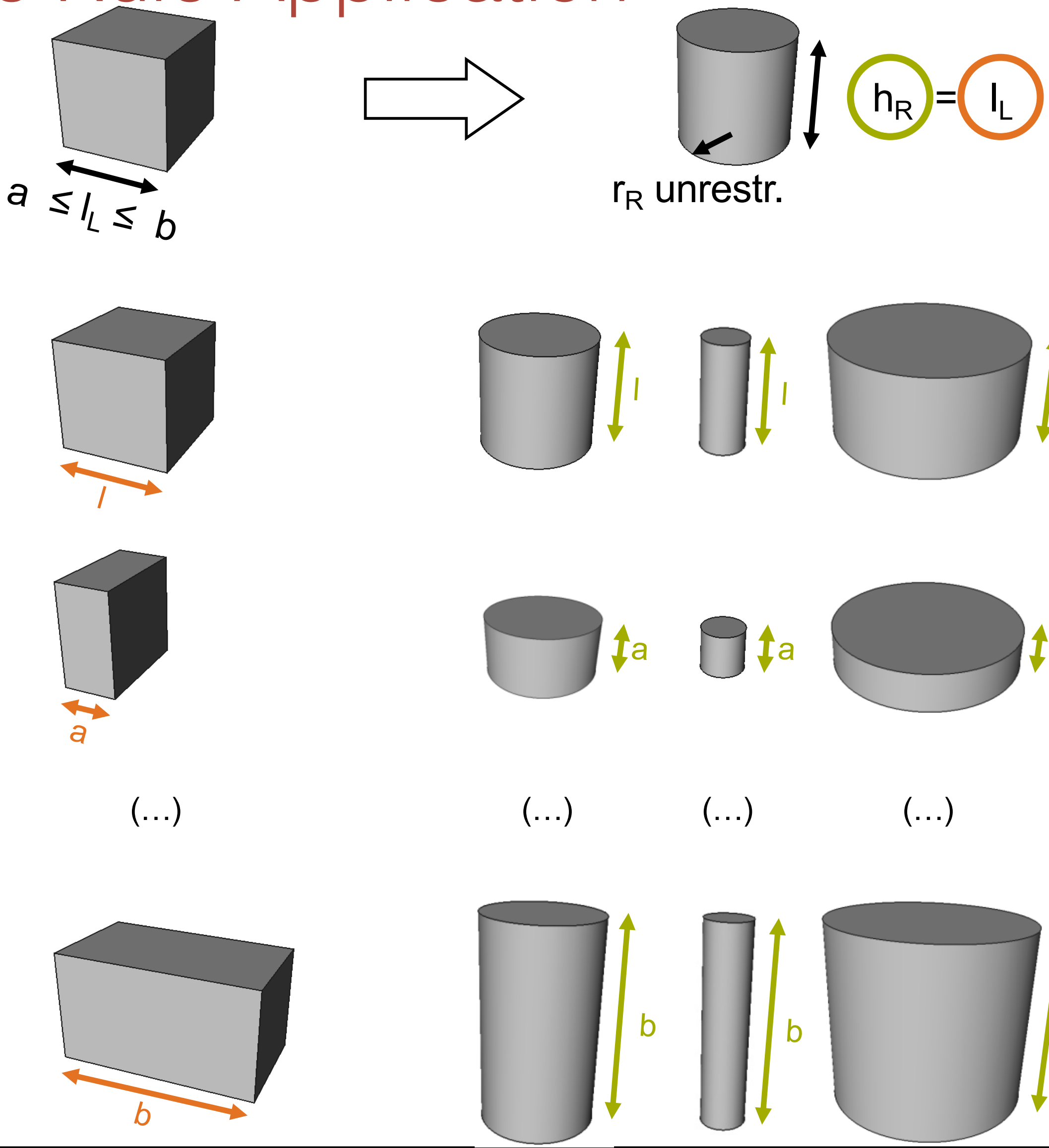


Parametric Rule Definition

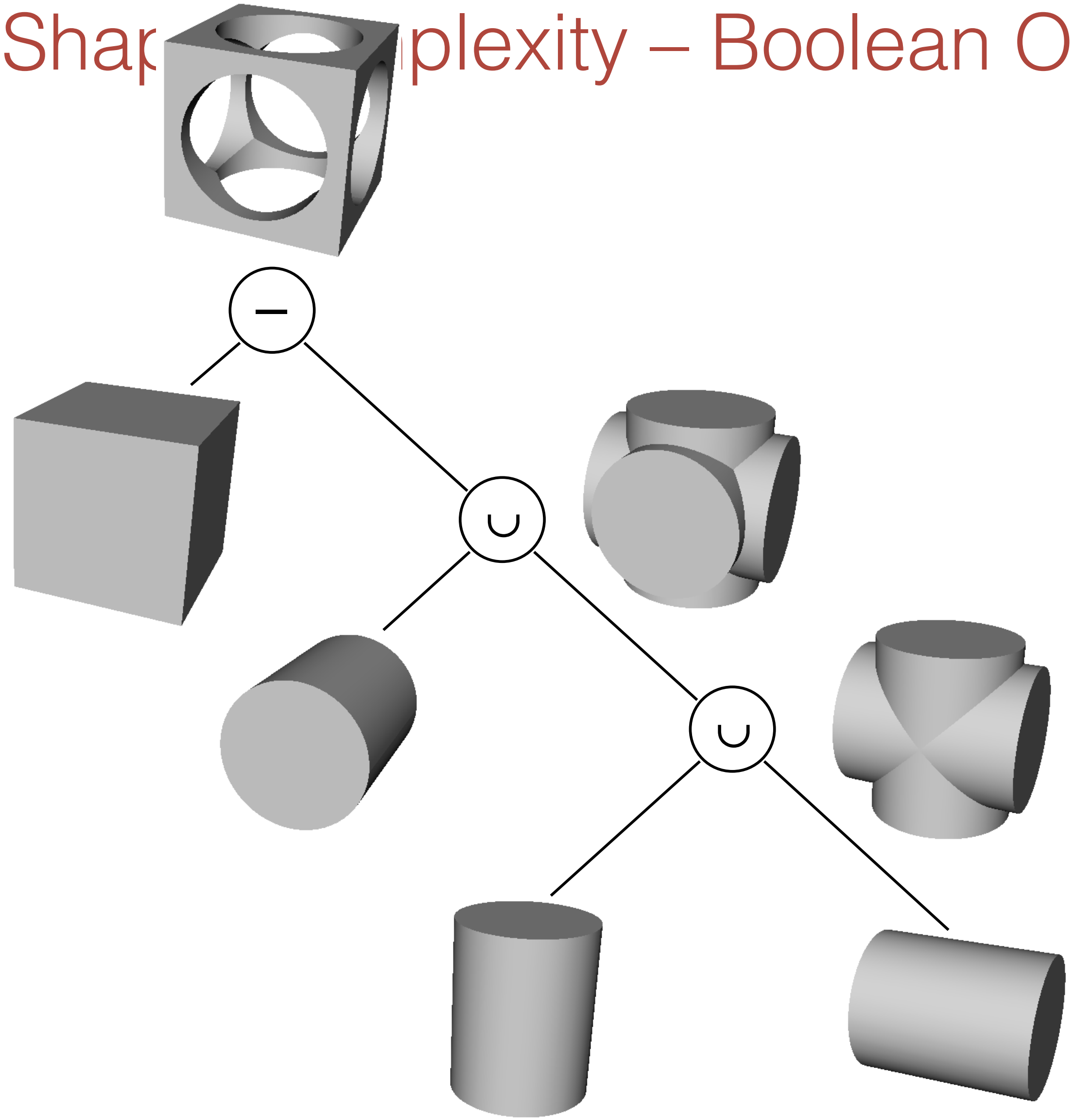


unrestricted	range	parametric relation
r_R arbitrary	$a \leq l_L \leq b$	$h_R = r_R*4 - l_L$

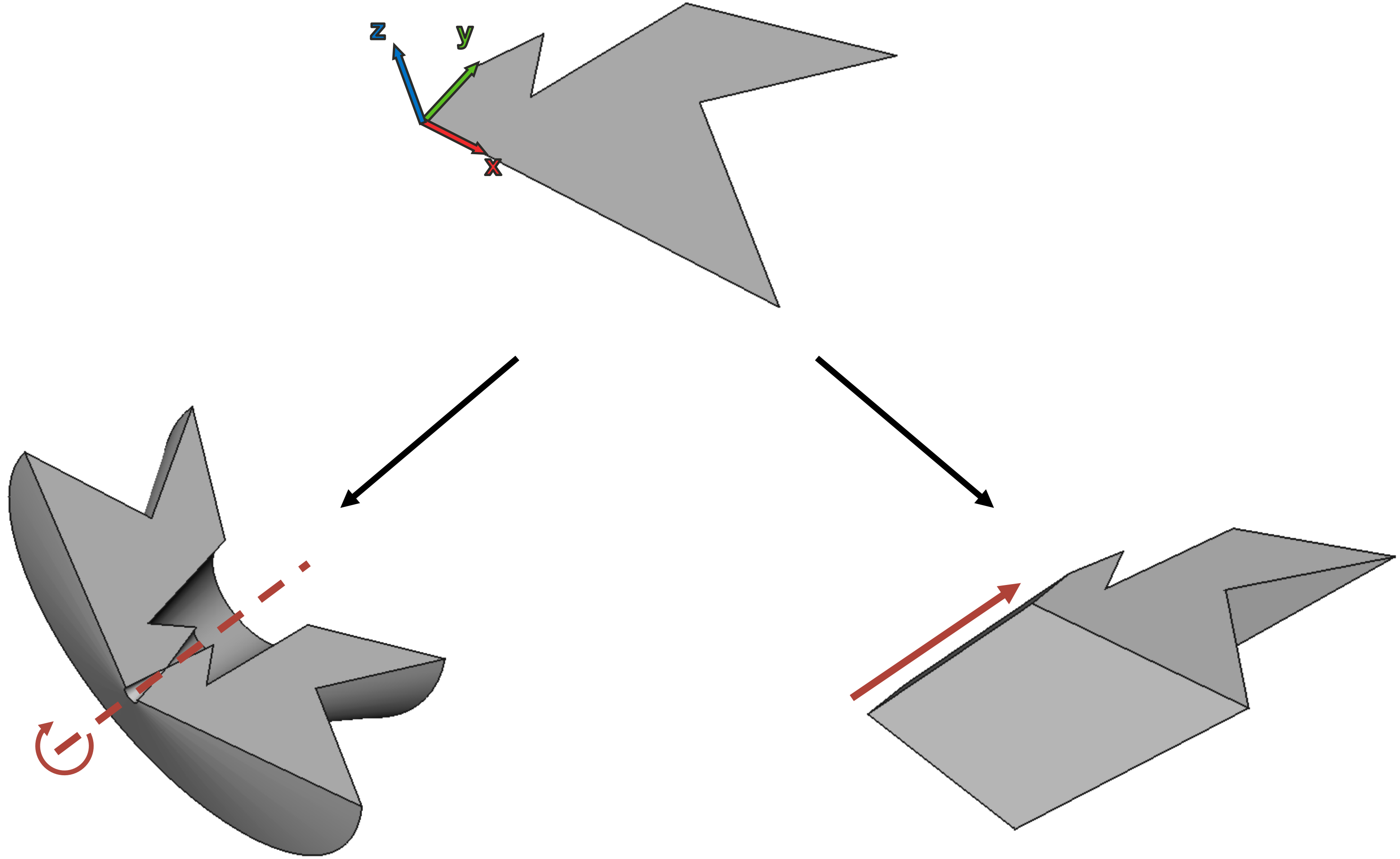
Parametric Rule Application



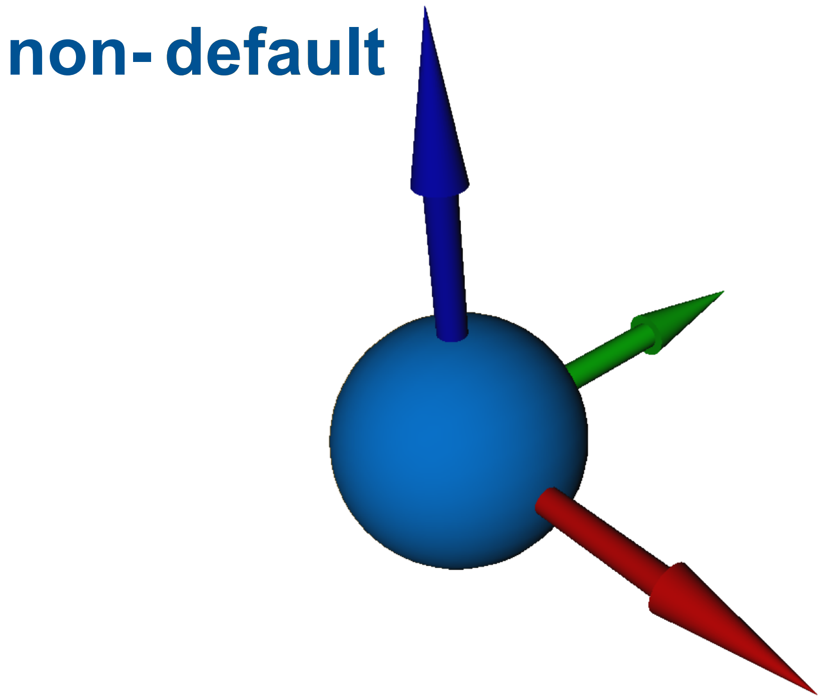
Increased Shap plexity – Boolean Operations



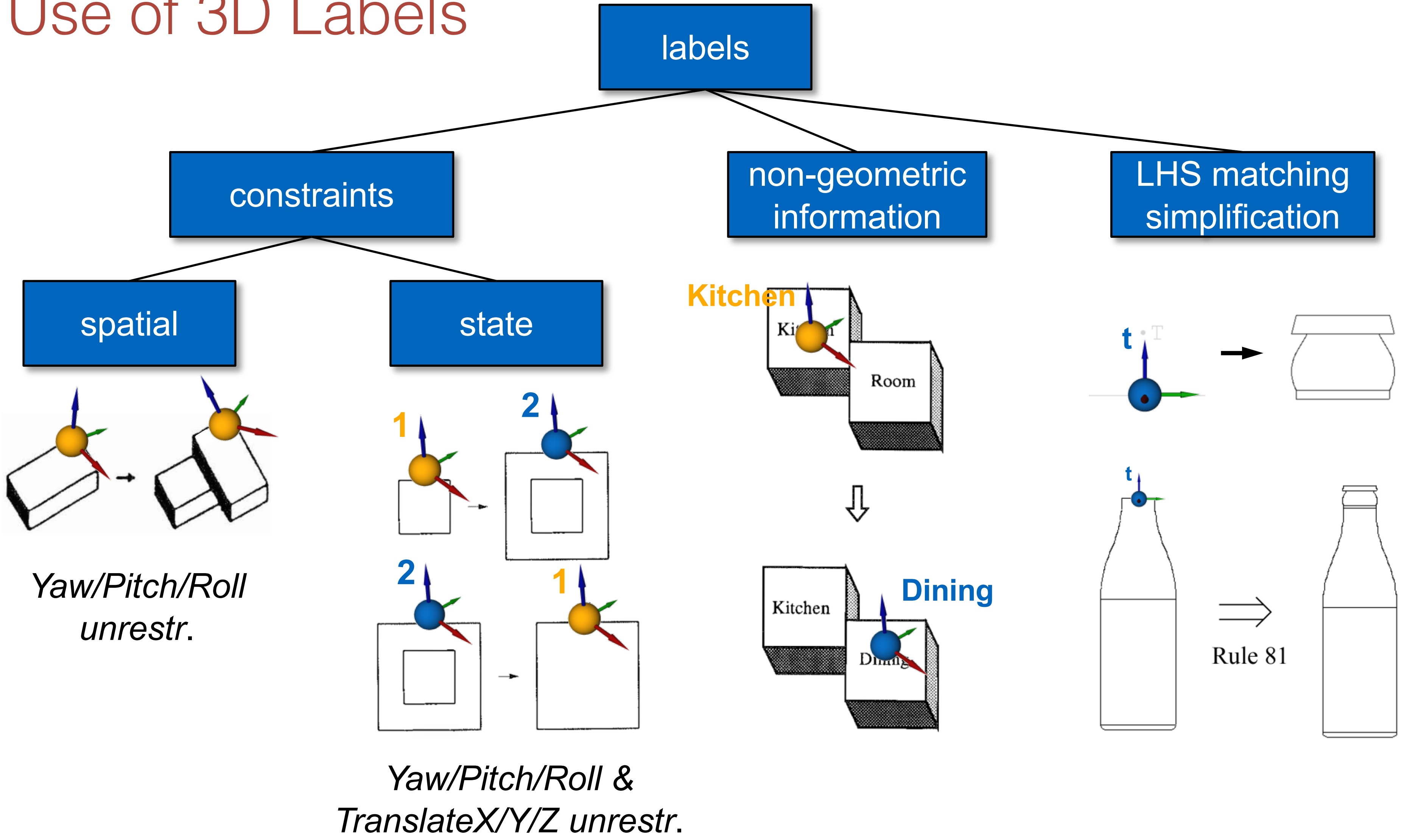
Increased Shape Complexity – Sweeping



3D Labels

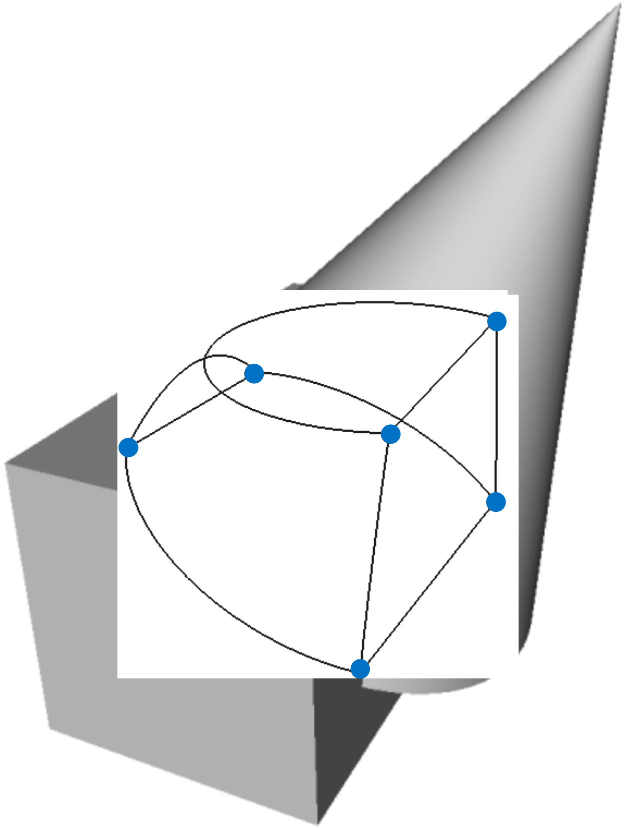


Use of 3D Labels

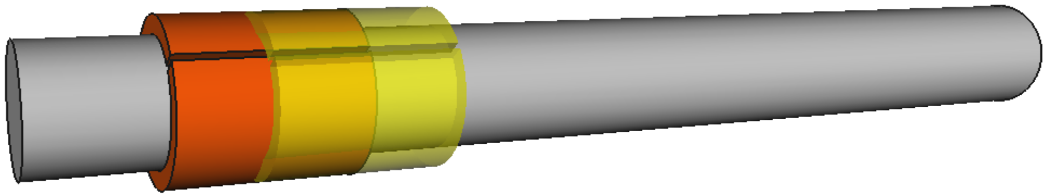


(figures: Stiny 1980, Knight 1994, Heisserman 1994, Chau 2004)

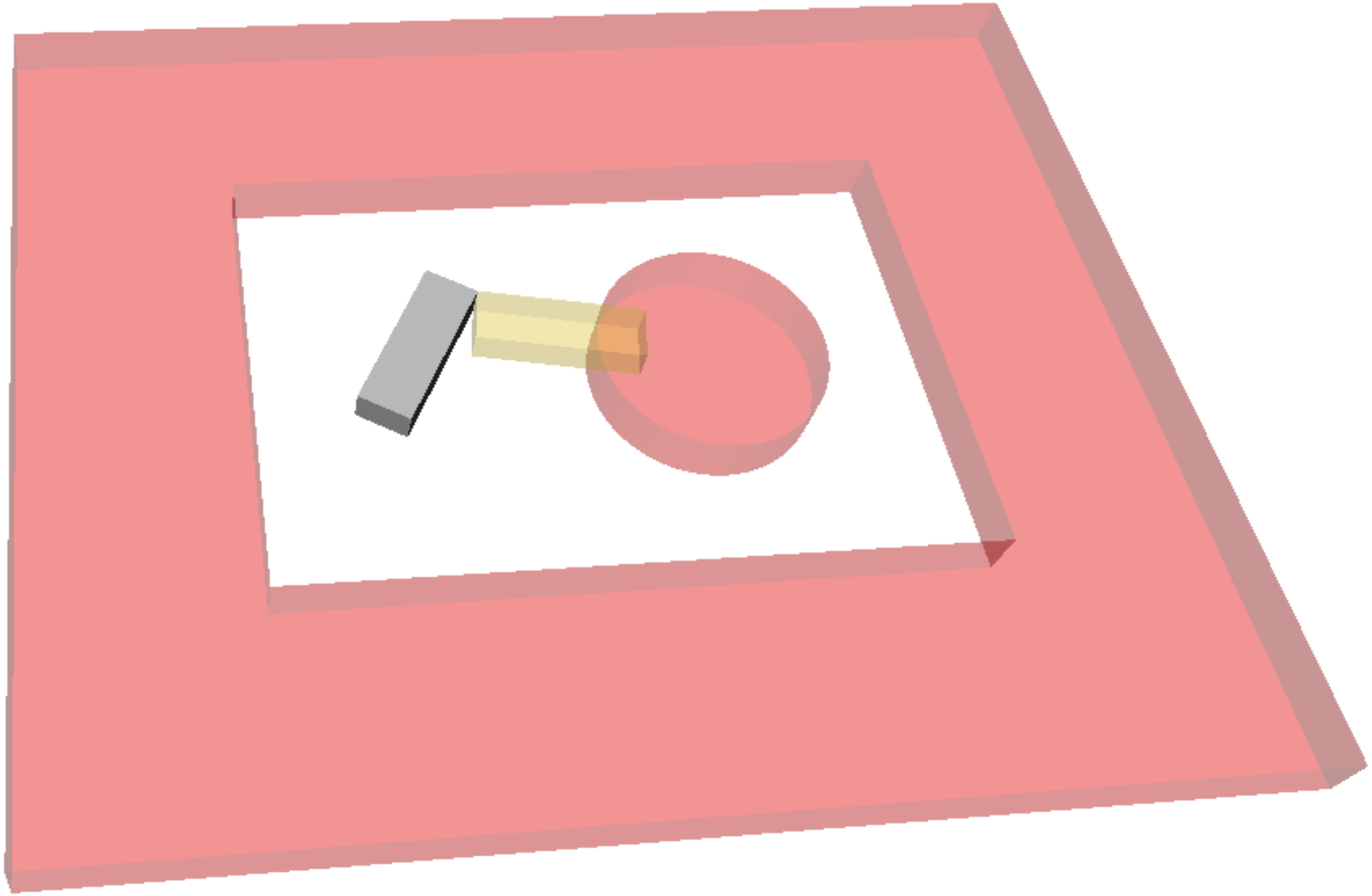
Collision Detection



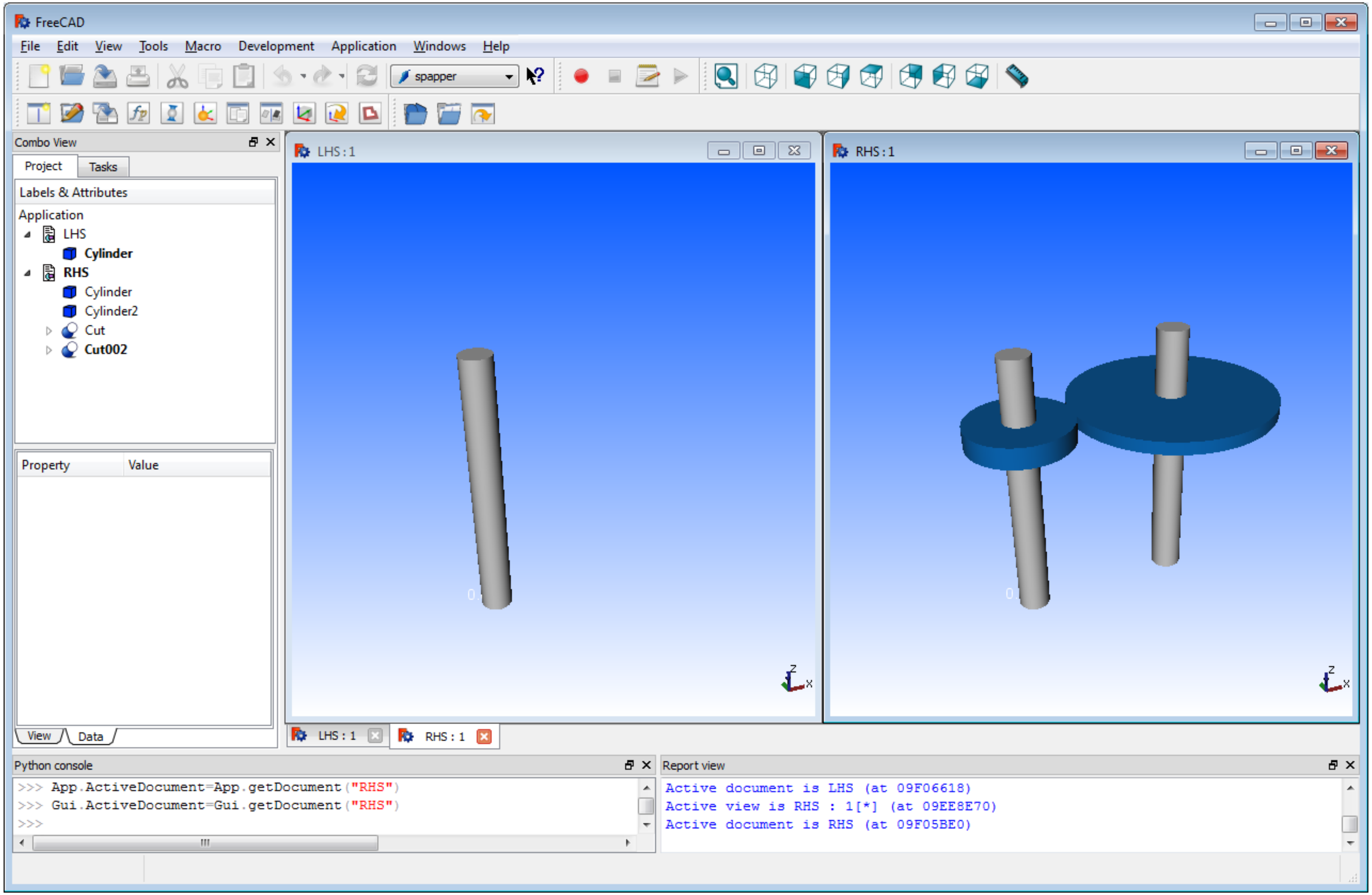
Part Collision Avoidance



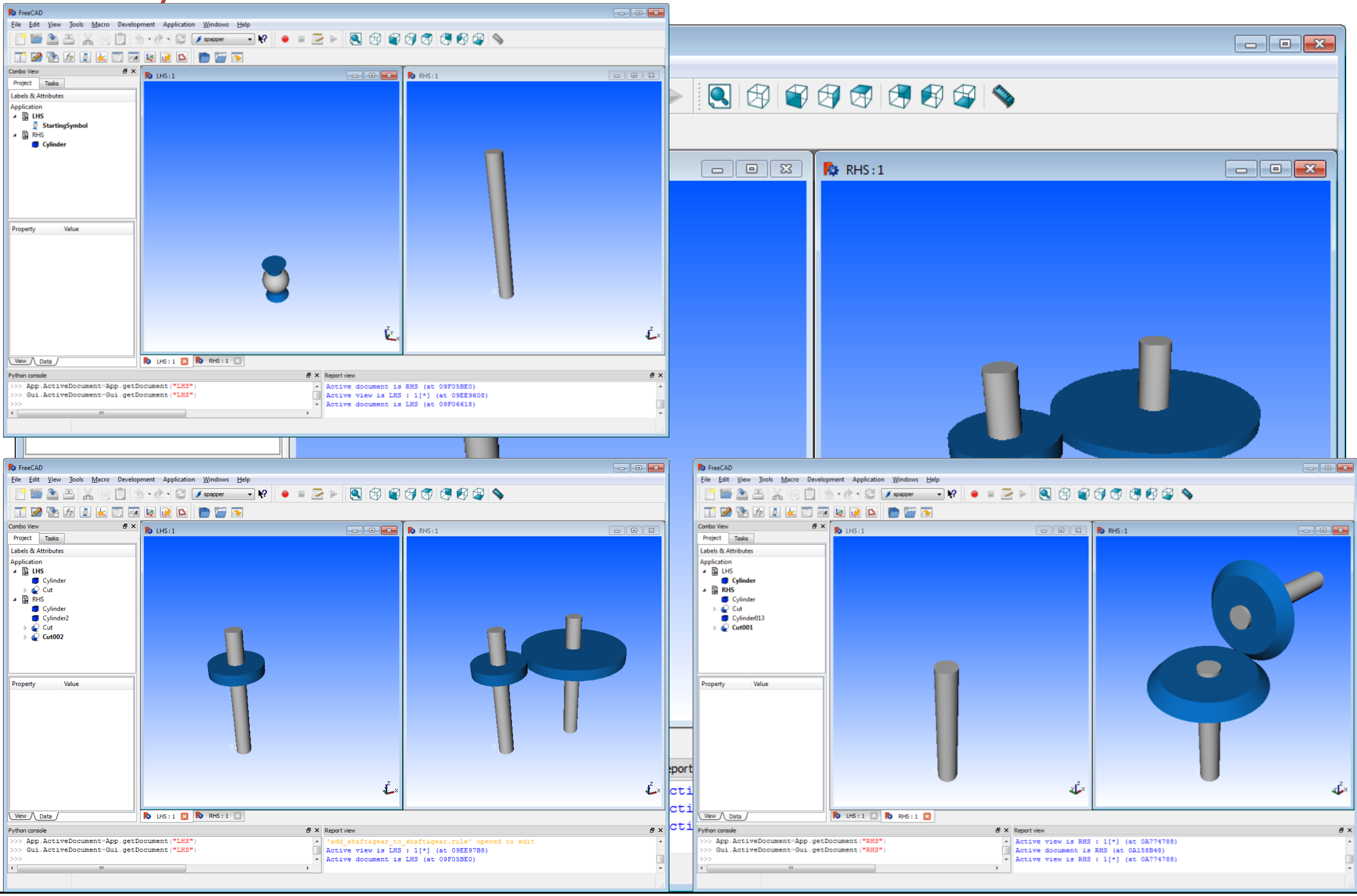
Design Space Restriction



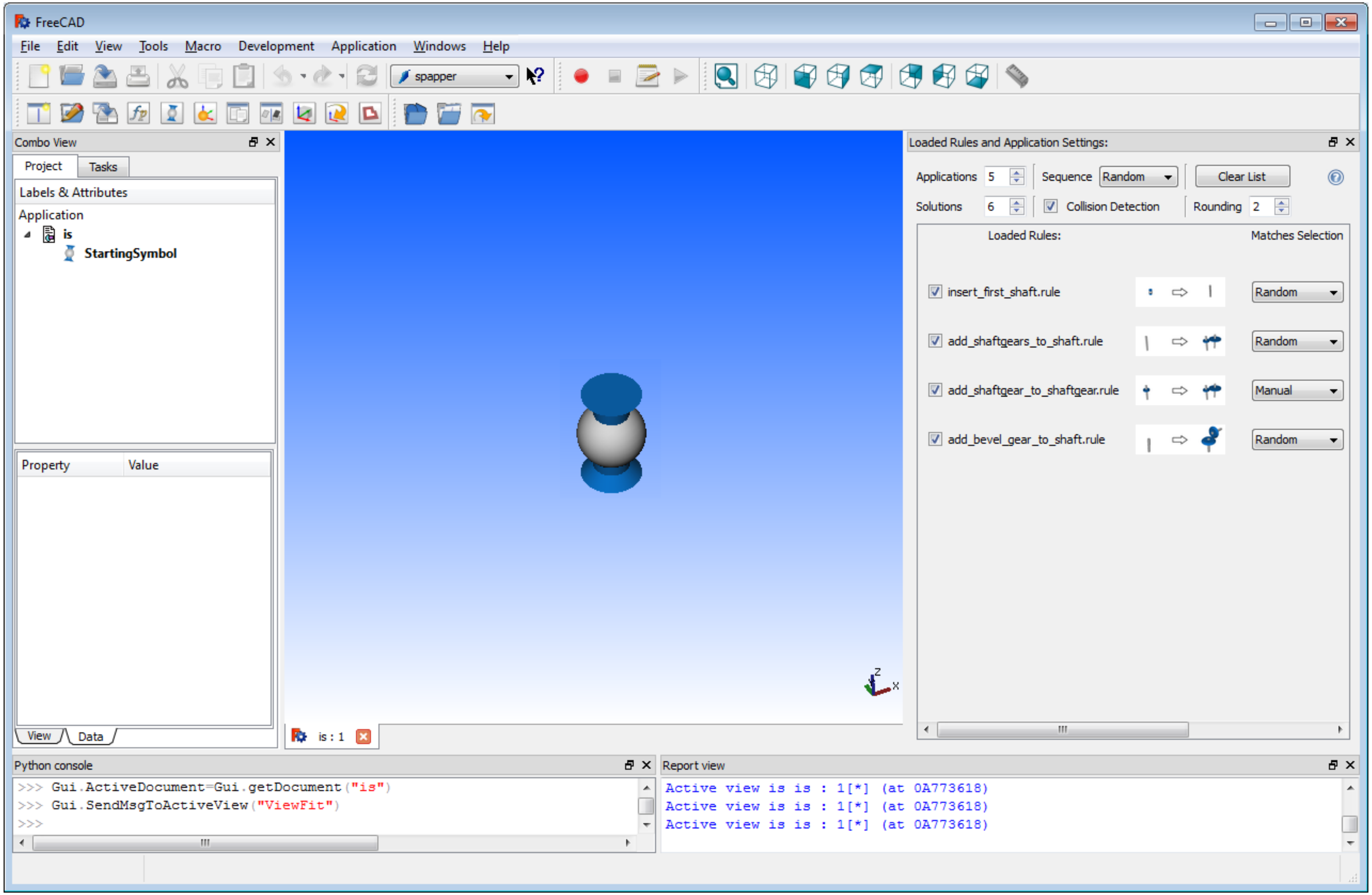
Software Prototype – Rule Definition



Gear Systems – Rules



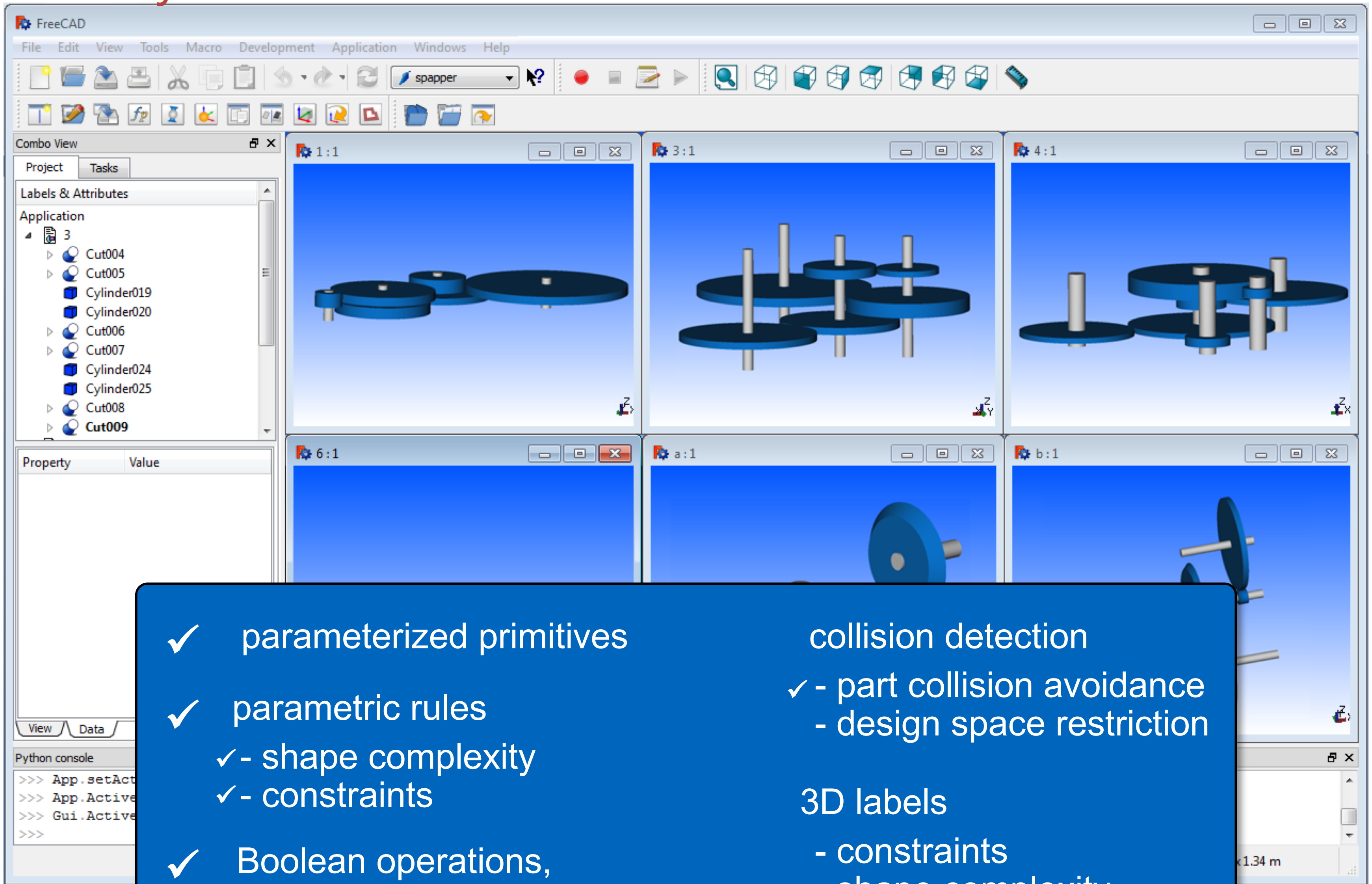
Rule Application – Initial Set



Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

ENGINEERING DESIGN AND COMPUTING

Gear Systems – Solutions



- ✓ parameterized primitives
- ✓ parametric rules
  - ✓ - shape complexity
  - ✓ - constraints
- ✓ Boolean operations, sweeping

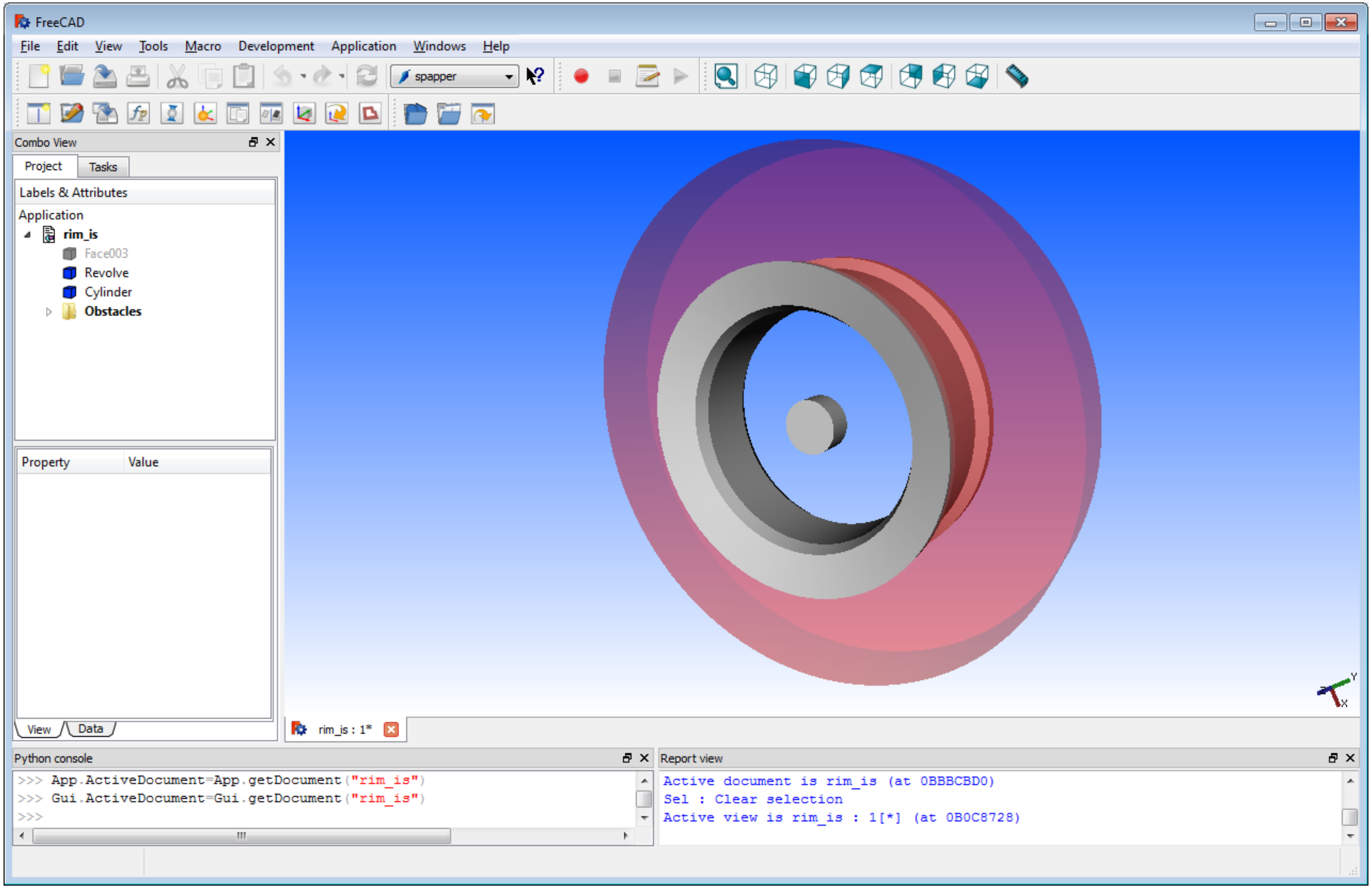
collision detection

- ✓ - part collision avoidance
- - design space restriction

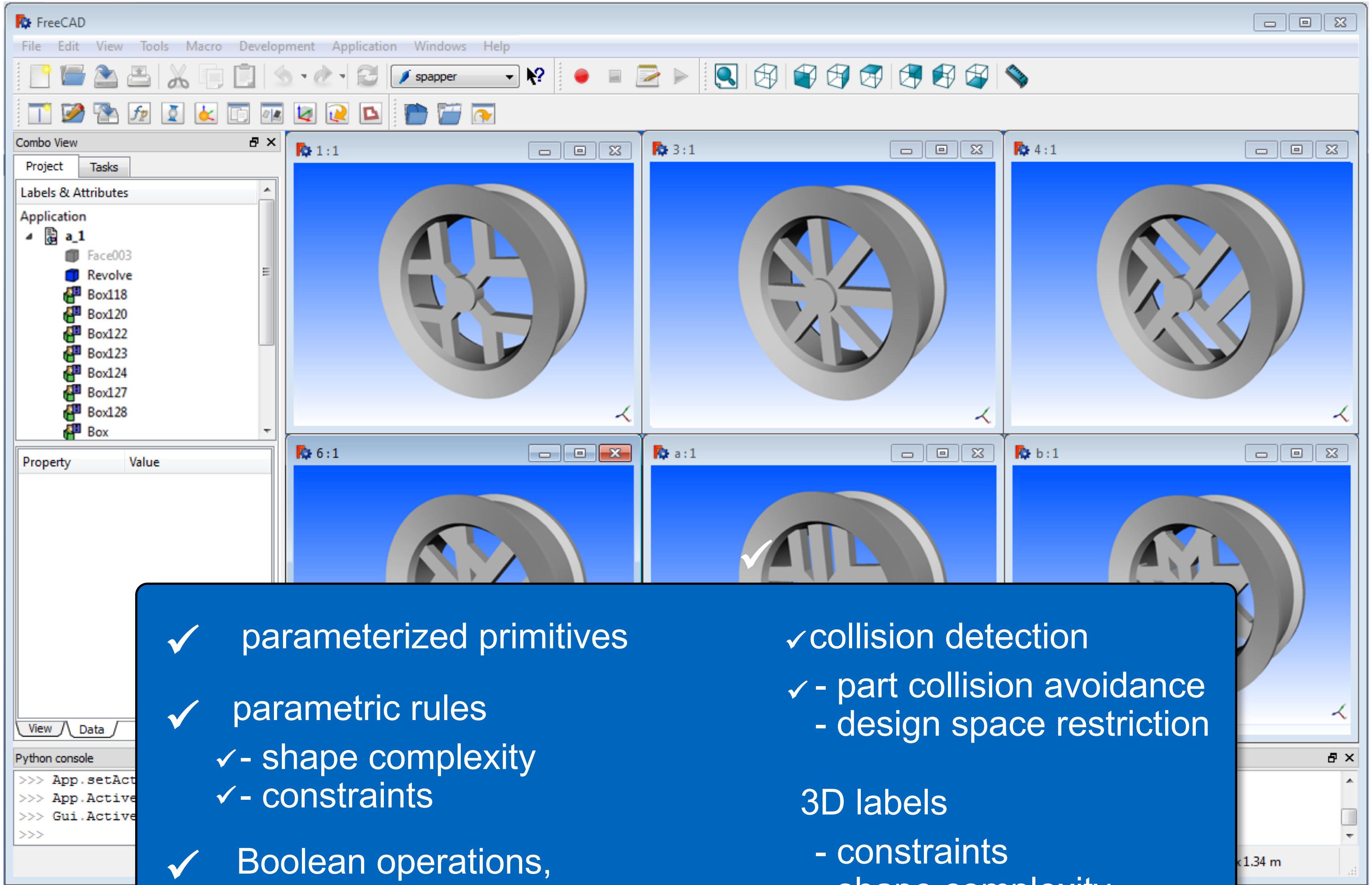
3D labels

- - constraints
- - shape complexity

Vehicle Wheel Rims – Design Space Restriction



Vehicle Wheel Rims – Solutions



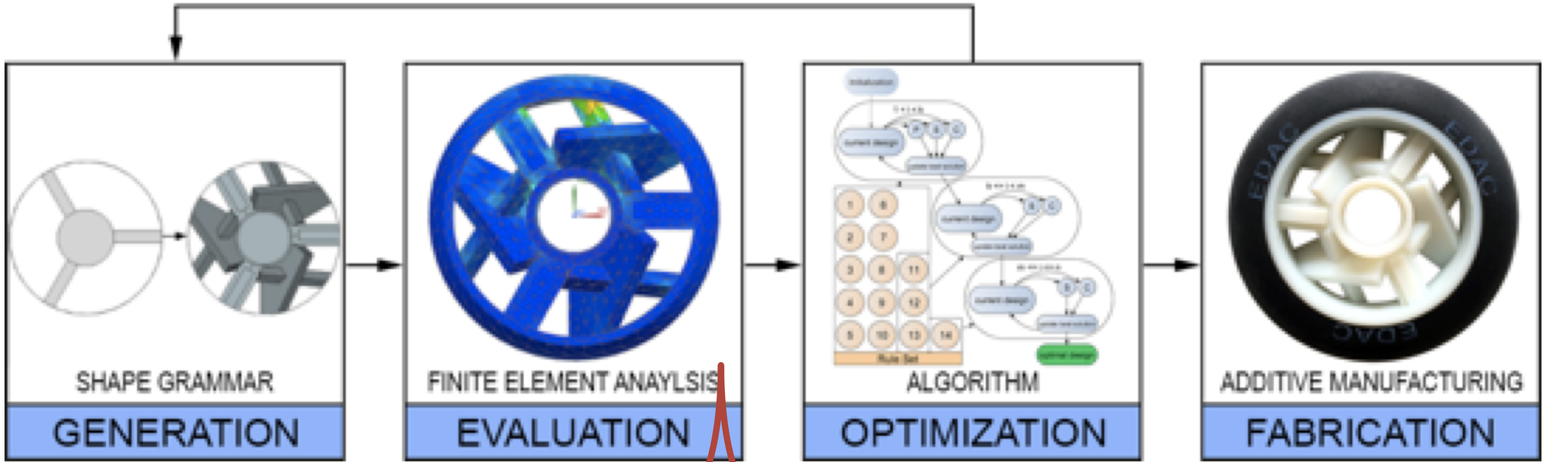
- ✓ parameterized primitives
- ✓ parametric rules
  - ✓ - shape complexity
  - ✓ - constraints
- ✓ Boolean operations, sweeping

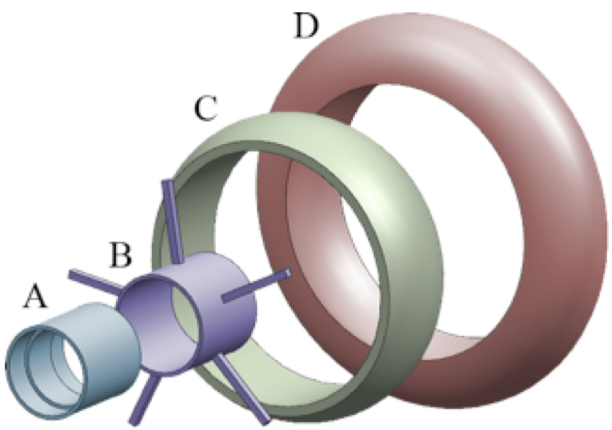
- ✓ collision detection
  - ✓ - part collision avoidance
  - - design space restriction
- 3D labels
  - - constraints
  - - shape complexity

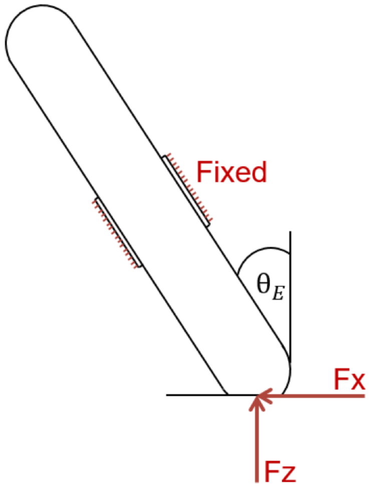
ENGINEERING DESIGN AND COMPUTING

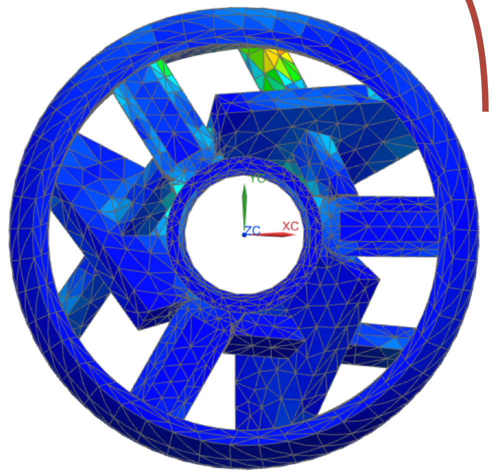
Wheel spoke design generation

A 3D, Performance-Driven Generative Design Framework





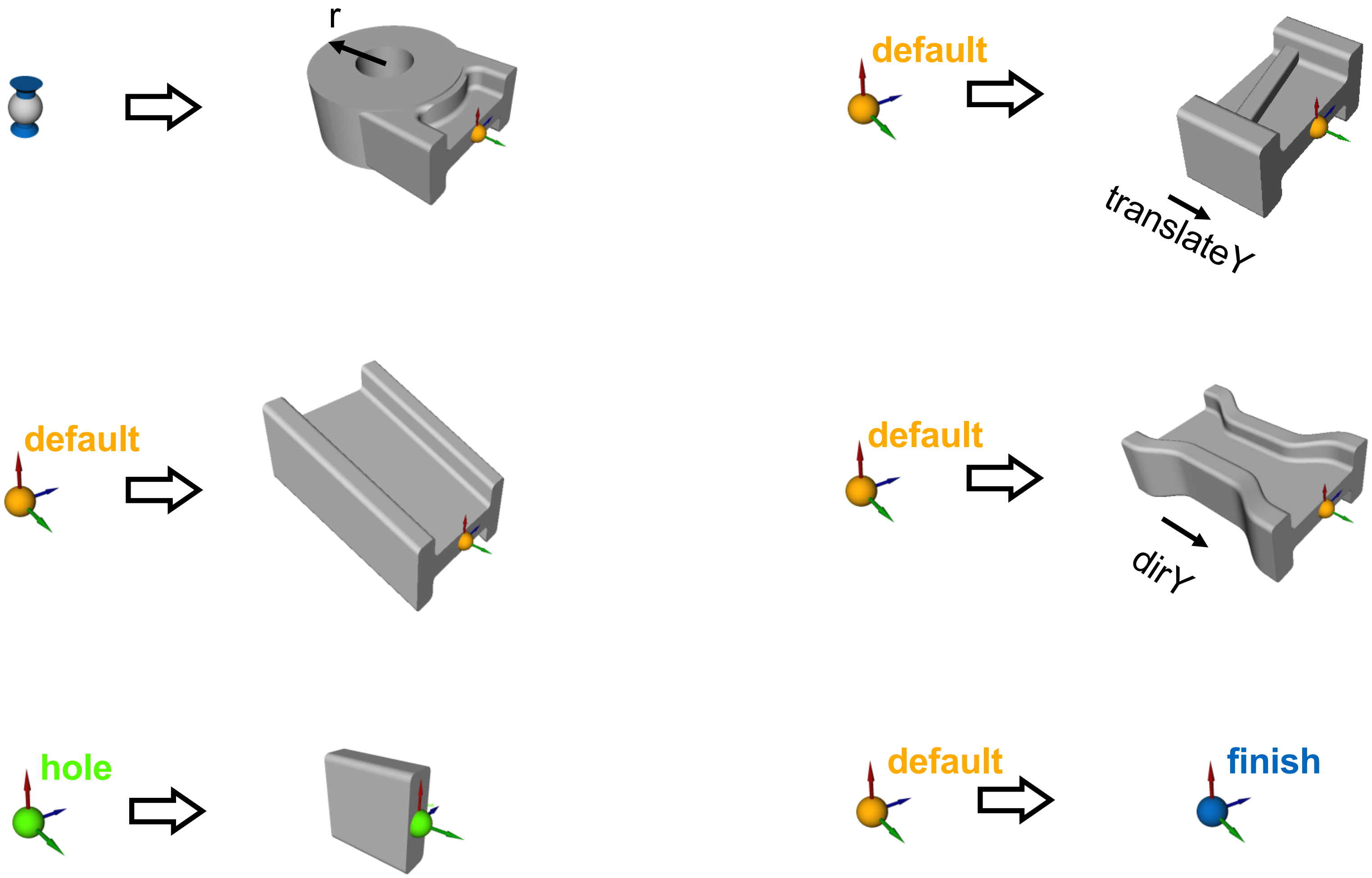




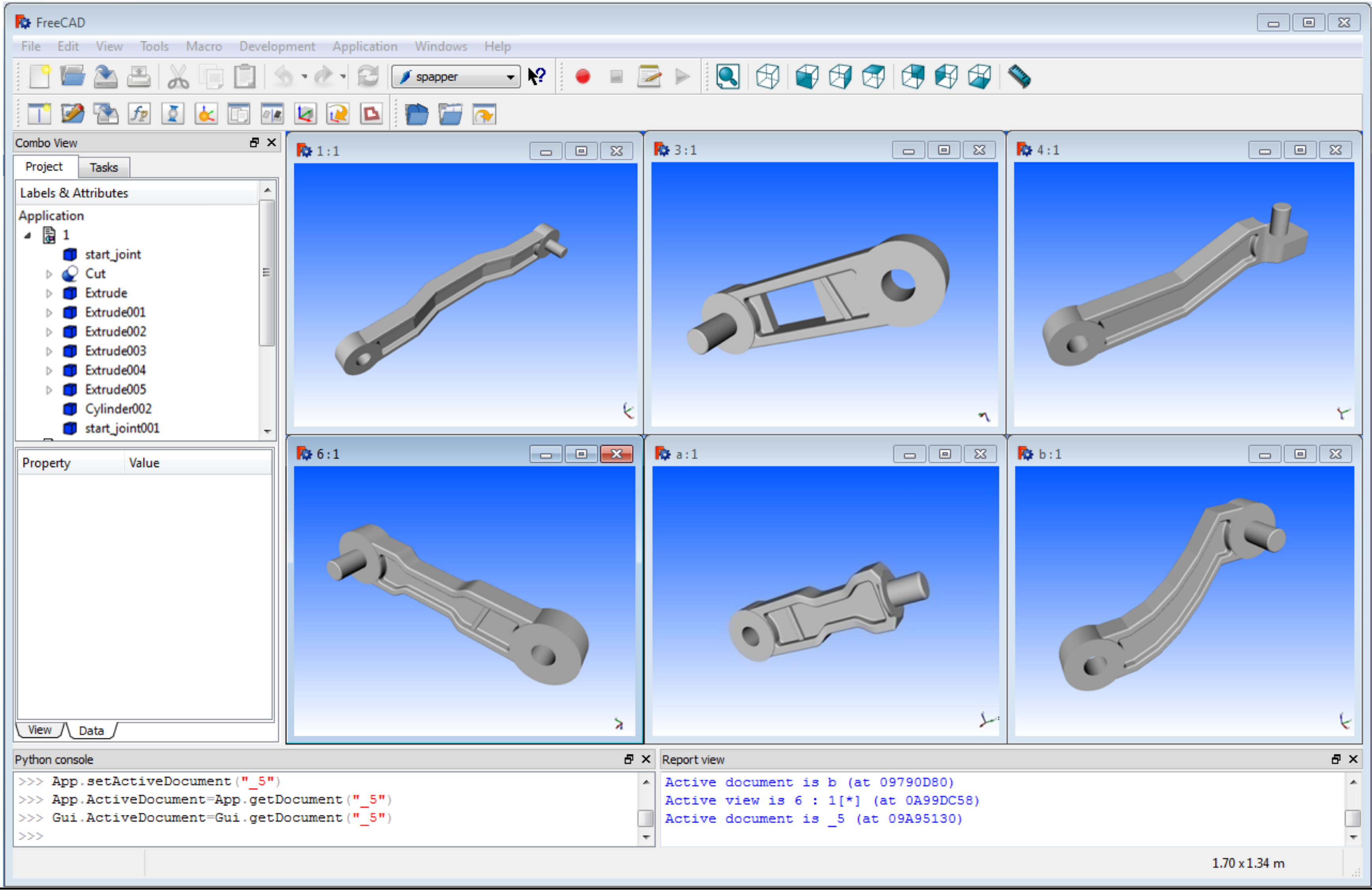
Example Multi-Material 3D Printed Design

L. Zimmermann, T. Chen and K. Shea, "A 3D, Performance-Driven Generative Design Framework: Automating the Link from a 3D Spatial Grammar Interpreter to Structural Finite Element Analysis and Stochastic Optimization", *AIEDAM*, 2018.

Robot Arm Concepts – Rules with 3D Labels

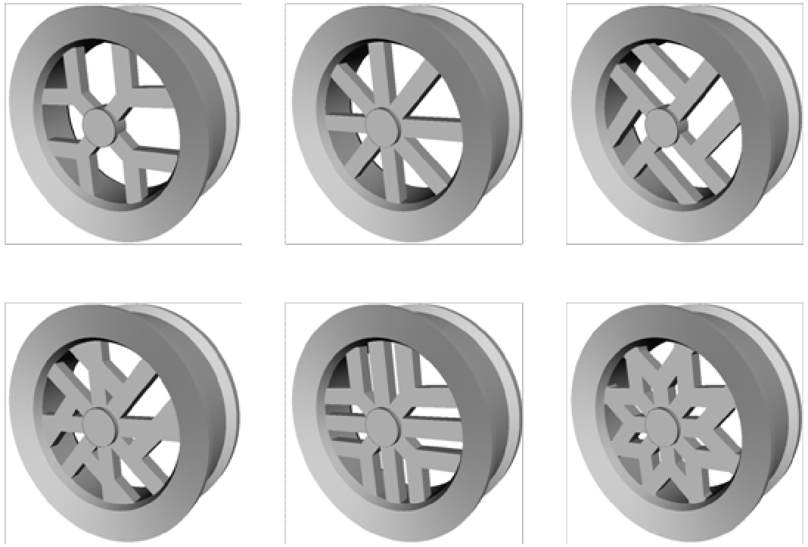


Robot Arm Concepts – Parts

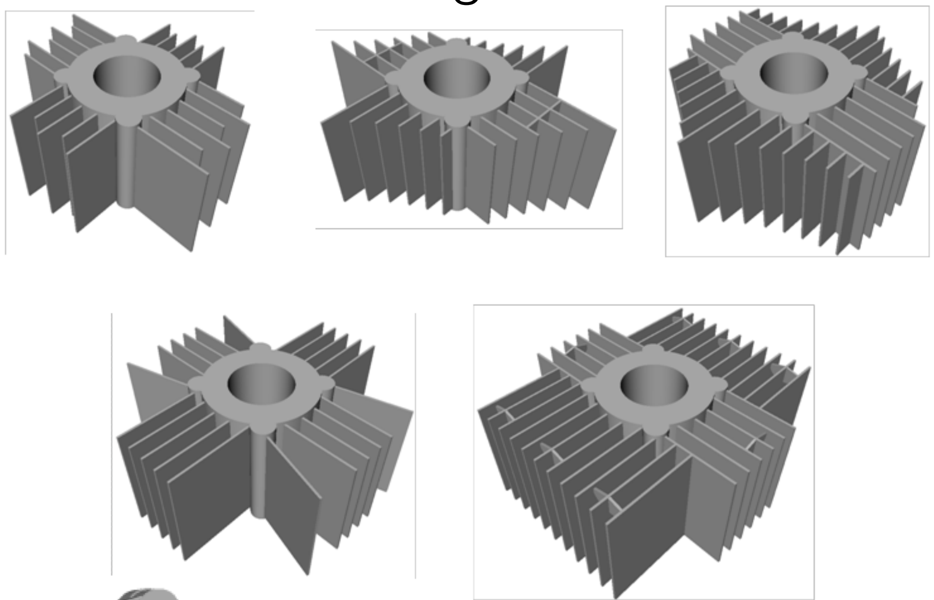


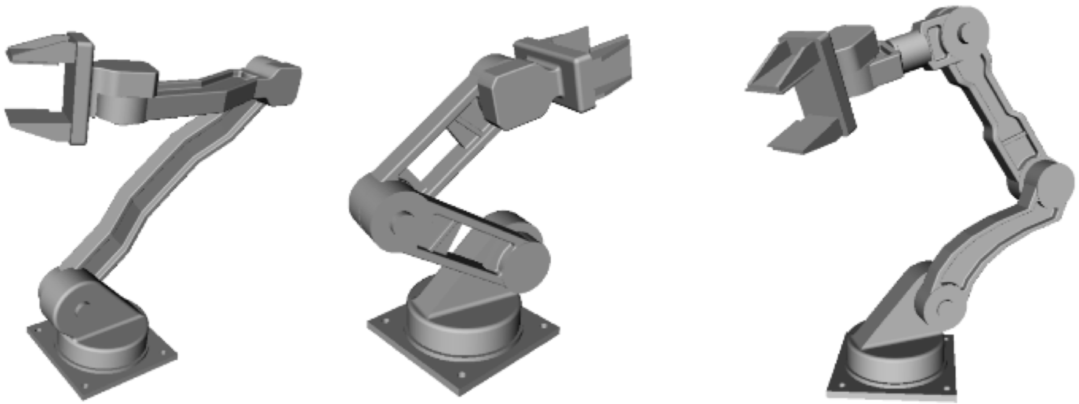
CAD-Based Generative Shape Design - Examples

Vehicle Wheel Rims



Cooling Fins





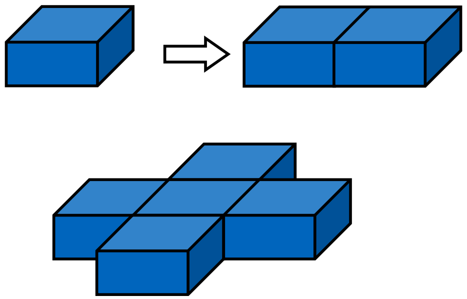
Customized Robot Arm Concepts

Spapper Summary (https://sourceforge.net/projects/spapper/)

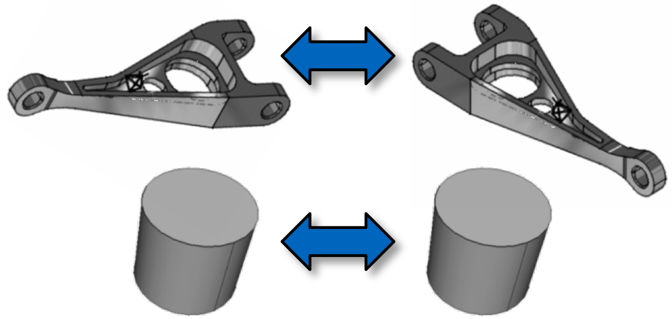
Integrated into one approach for a general 3D spatial grammar platform:
Visual definition and modification of rules
Interactive (automatic/semi-automatic) rule application
Wide range of shapes
Automatic LHS matching
Parametric rules
Consolidated concept for labels
Collision detection
Unrestricted number of rules, shapes in rules and applications of rules
Definition of additive, subtractive and substituting rules
Integration into CAD

Computational Design Synthesis in CAD => 'active design partner'

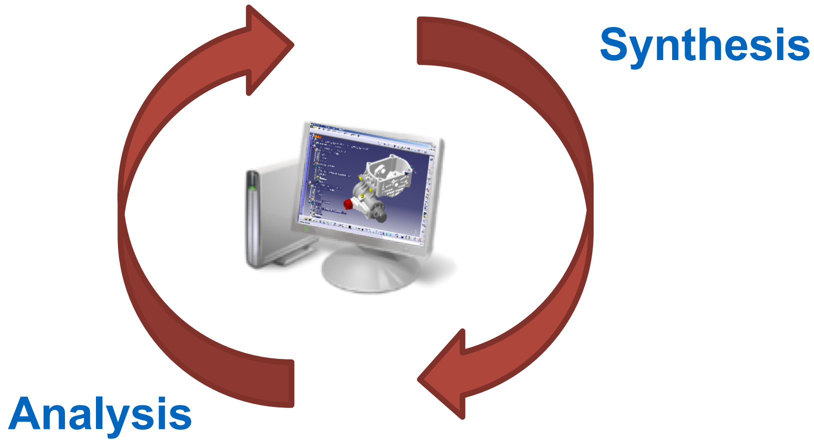
Limitations and Future Extensions



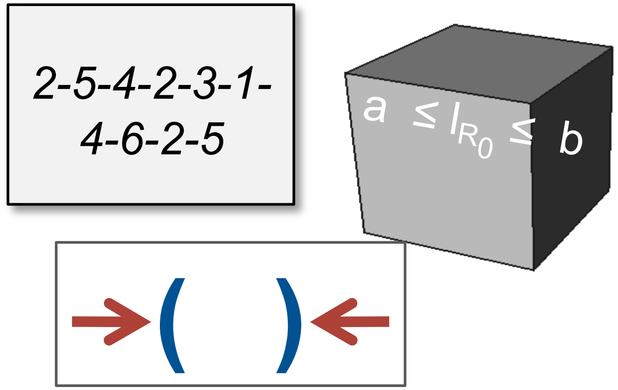
no matching shapes under multiple transformations



no generalized 3D shape matching



combination with automatic analysis of solutions



enhancement: rule selection, parameter definition, constraint specification

Further Reading

- General
  - "Formal Reductions of the General Combinatorial Decision Problems", E. Post, *American Journal of Mathematics*, 65:197-268, 1943.
  - *Syntactic Structures*, N. Chomsky, The Hague:Mouton, 1957.
  - "Production systems and grammars: a uniform characterization", J. Gips and G. Stiny, *Environment and Planning B*, 1980, 7:399-408

- Shape Grammars
  - "Introduction to Shape and Shape Grammars", G. Stiny, *Environment and Planning B*, 7:343-351, 1980.
  - "Spatial Grammars: Motivation, Comparison, and New Results", R Krishnamurti and R. Stouffs, *CAAD Futures '93*, 57-74, 1993.
  - "Spatial grammar implementation: From theory to useable software", McKay et al., *AIEDAM*, 26(02):143-159, 2012.

Further Reading

- Generative Grammars
  - "Optimally Directed Shape Generation by Shape Annealing," J. Cagan and W.J. Mitchell, *Environment and Planning B*, 20:5-12, 1993.
  - "Grammatical Design," K.N Brown, *IEEE Expert/Intelligent Systems and Their Applications*, 12(2):27-33, 1997.
  - "Generative Geometric Design," J. Hiesserman, *IEEE Computer Graphics and Applications*, 14(2):37-45, 1994.

- Spapper
  - "An Interactive, Visual Approach to Developing and Applying Parametric Three-Dimensional Spatial Grammars", F. Hoisl and K. Shea, *AIEDAM*, 25(4): 333-356, 2011.
  - "Three-dimensional labels: A unified approach to labels for a general spatial grammar interpreter", F. Hoisl and K. Shea, *AIEDAM*, 27(4):359-375, 2013.
  - "A 3D, performance-driven generative design framework: automating the link from a 3D spatial grammar interpreter to structural finite element analysis and stochastic optimization", L. Zimmermann, T. Chen and K. Shea, *AIEDAM*, 32(2):189-199, 2018.