GENERATING SIMULATION MODELS FROM COMPONENTS:

Industrial use cases in design space exploration and machine learning

Jakob Rehof, TU Dortmund University

SummerSoc 2023

CONSTANTIN CHAUMET

Researcher at SEAL Research Group
Focus on Robotics and Design Automation
Developer of CLS-CAD
Maker of Slides

WWW.GITHUB.COM/TUDO-SEAL

AGENDA.

A short overview of presented topics.

01

(CL)S FRAMEWORK

Intent of- and Information about the (CL)S framework

02

USE-CASE: MOTION PLANNING

Application of the (CL)S framework and the HyperMapper to explore the motion planning design space.

03

USE-CASE: CAD SYNTHESIS

Application of the (CL)S framework to synthesize complete CAD assemblies and in future optimize key metrics.

04

WRAP-UP

Short summary of presented contents.

WWW.GITHUB.COM/TUDO-SEAL/BCLS-PYTHON

COMBINATORY LOGIC SYNTHESIZER

THE (CL)S FRAMEWORK IS A LANGUAGE-AGNOSTIC AND FORMALLY VERIFIED FRAMEWORK THAT IS ABLE TO GENERATE ALL COMBINATIONS OF MODULAR COMPONENTS THAT SATISFY A PARTICULAR REQUEST/SPECIFICATION.

HOW IT WORKS.

INHABITATION REQUEST → ENUMERATION → INTERPRETATION

01

REPOSITORY

A set of combinators with types. The types define how the combinators can be applied to each other and thus combined.

02

TREE GRAMMAR

The grammar that describes all terms that are inhabitants of the requested type.

03

TERMS

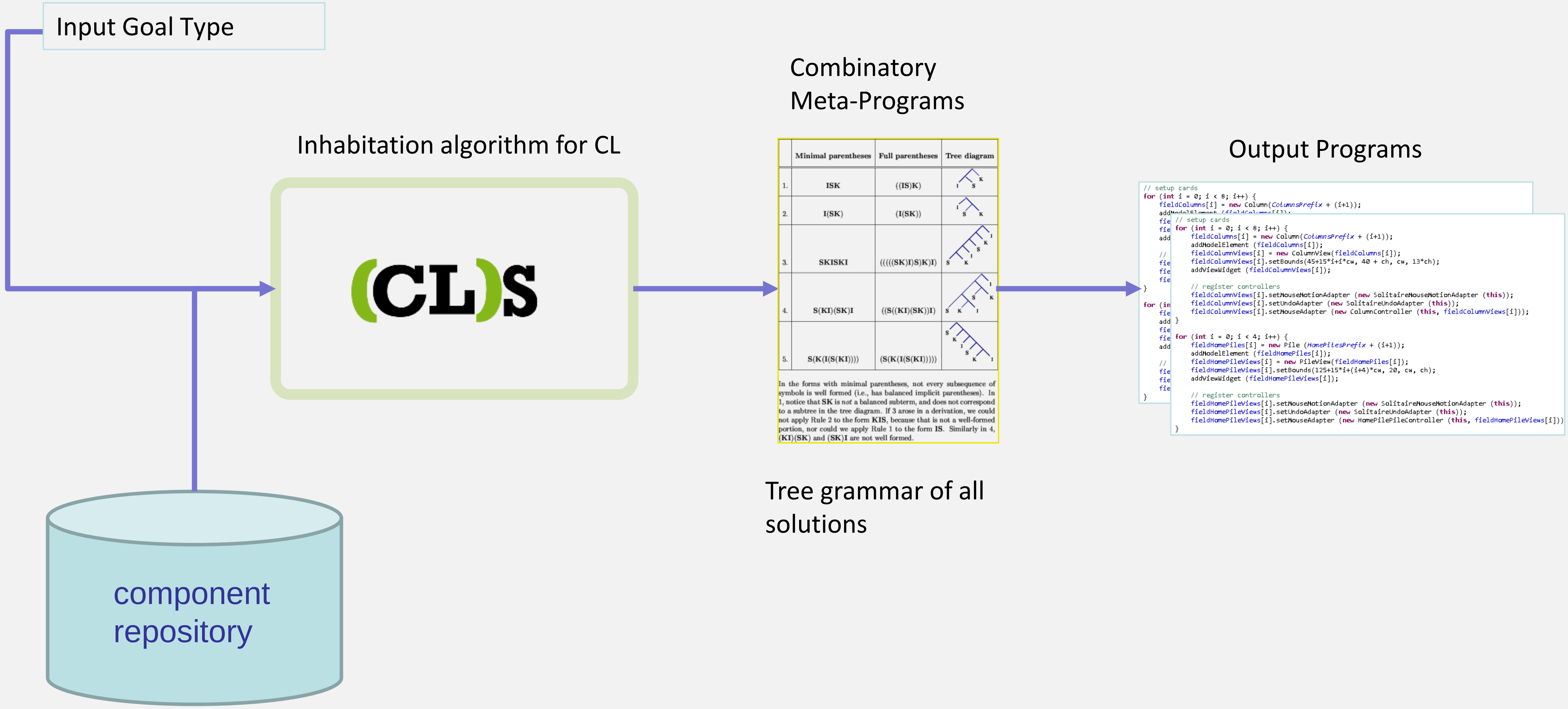
An enumeration of the terms present in the grammar. Multiple different approaches are possible.

04

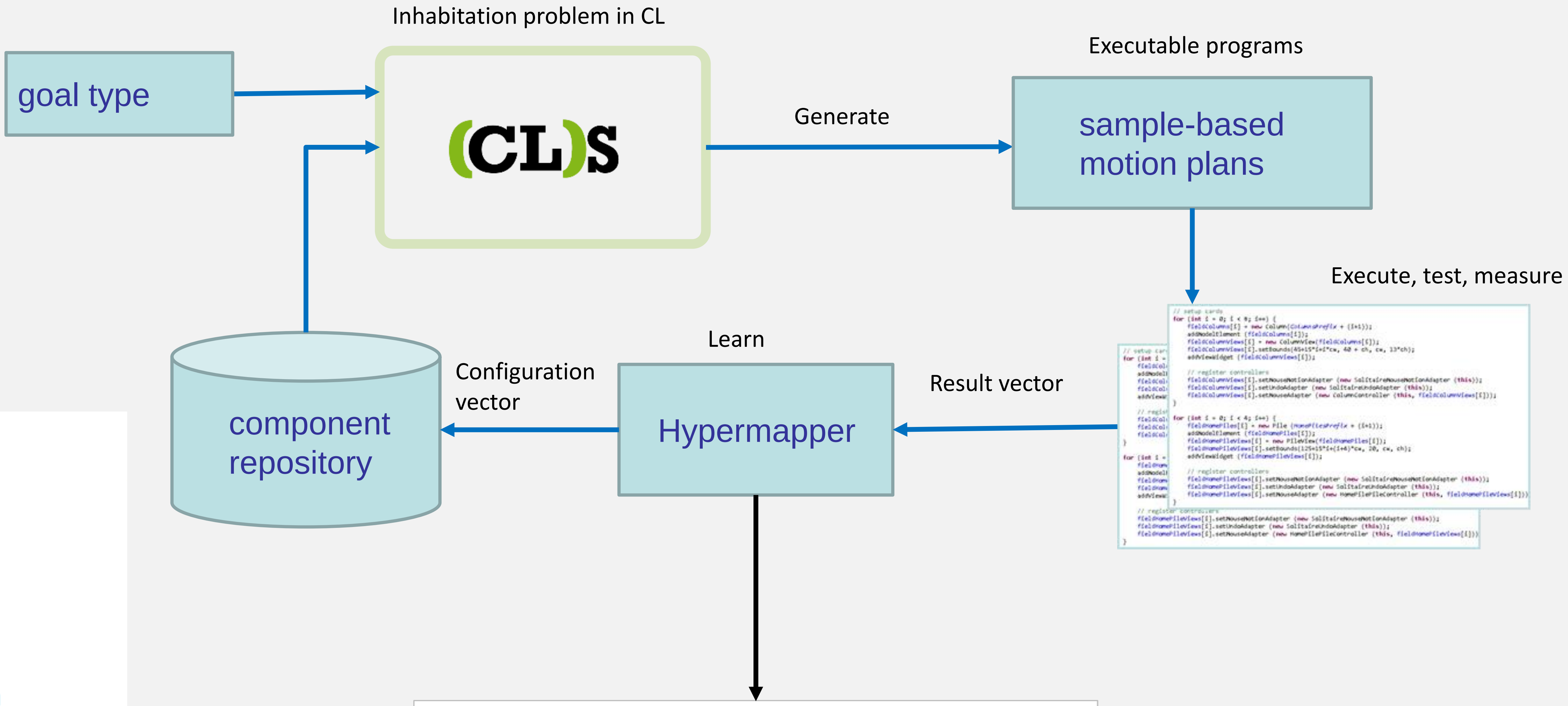
ARTIFACT

The interpretation of the terms resolves the applications etc. to concrete artifacts.

Basic synthesis pipeline of CLS-framework



Design space exploration and learning with CLS-framework



$\Gamma_s = \{$

PlannerAssembly:

$any\_planner \to any\_state\_validator \to$
$any\_motion\_validator \to any\_simplification \to$
$sbmp\_input \to sbmp\_program,$

PRMStarSchema :

$(any\_opt\_obj \to sampler\_space \to PRMStar) \cap$
$(any\_opt\_obj \to sampler\_valid\_state \to PRMStar) \cap$
$(any\_opt\_obj \to sampler\_informed \to PRMStar),$

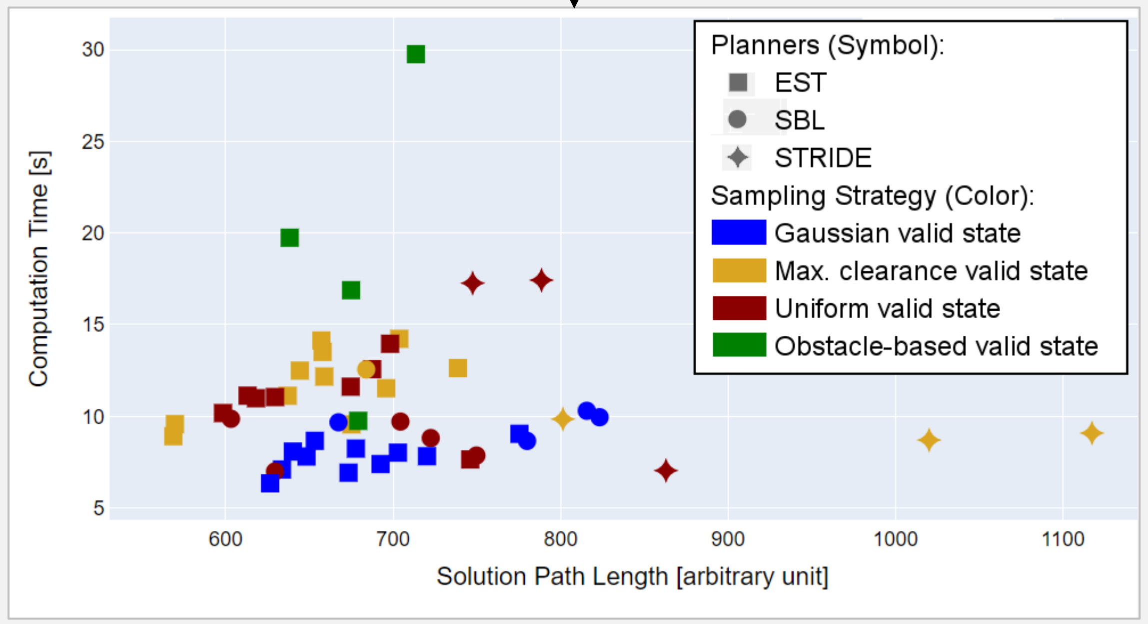
ESTSchema :

$obj\_path \to sampler\_valid\_state \to EST,$

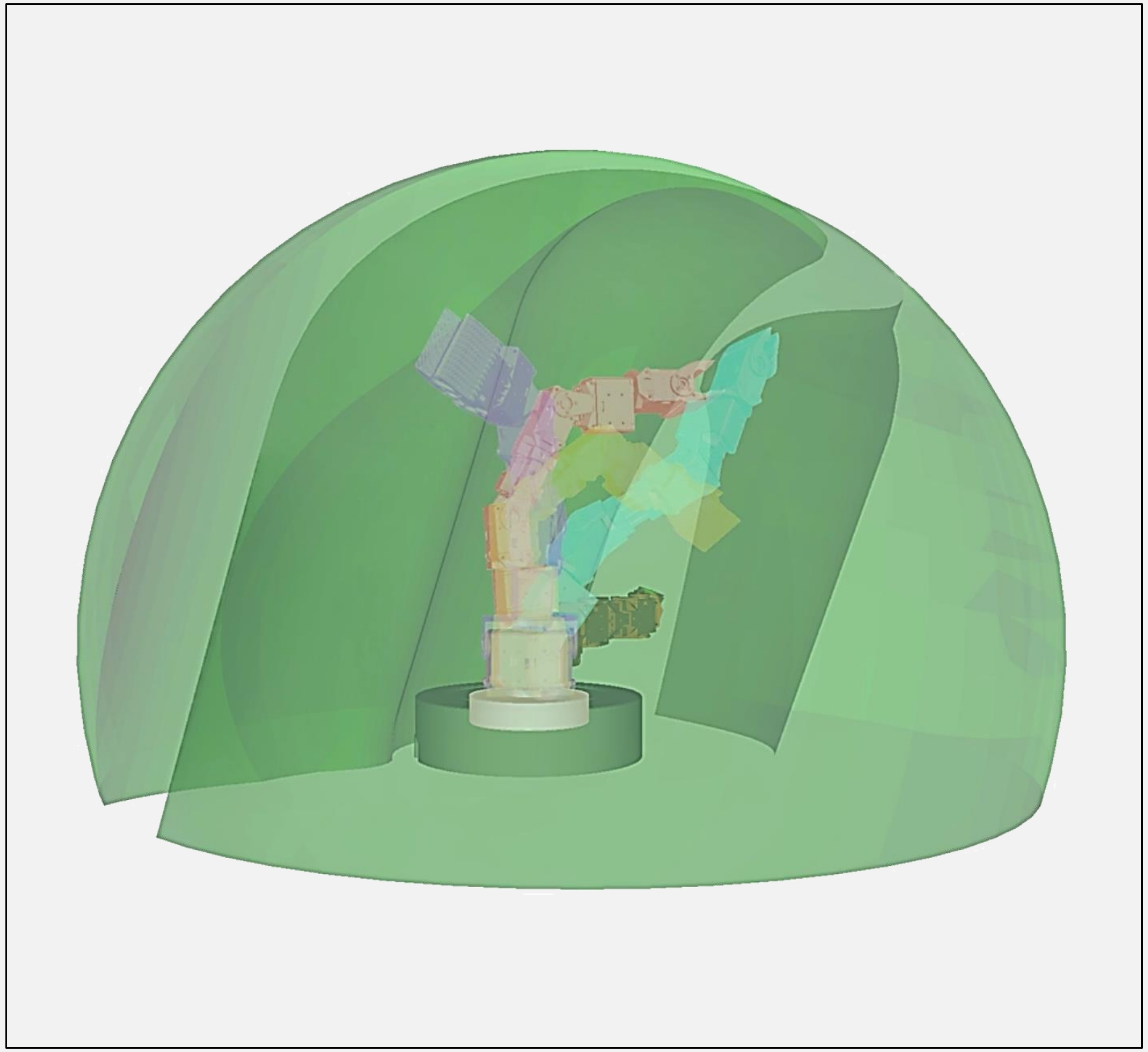
[...]

$\}$

Fig. 3. Excerpt of the semantic repository Γ_s showing the type signature of the combinators PlannerAssembly, PRMStarSchema, and ESTSchema



WWW.GITHUB.COM/TUDO-SEAL/BCLS-PYTHON



MOTION PLANNING

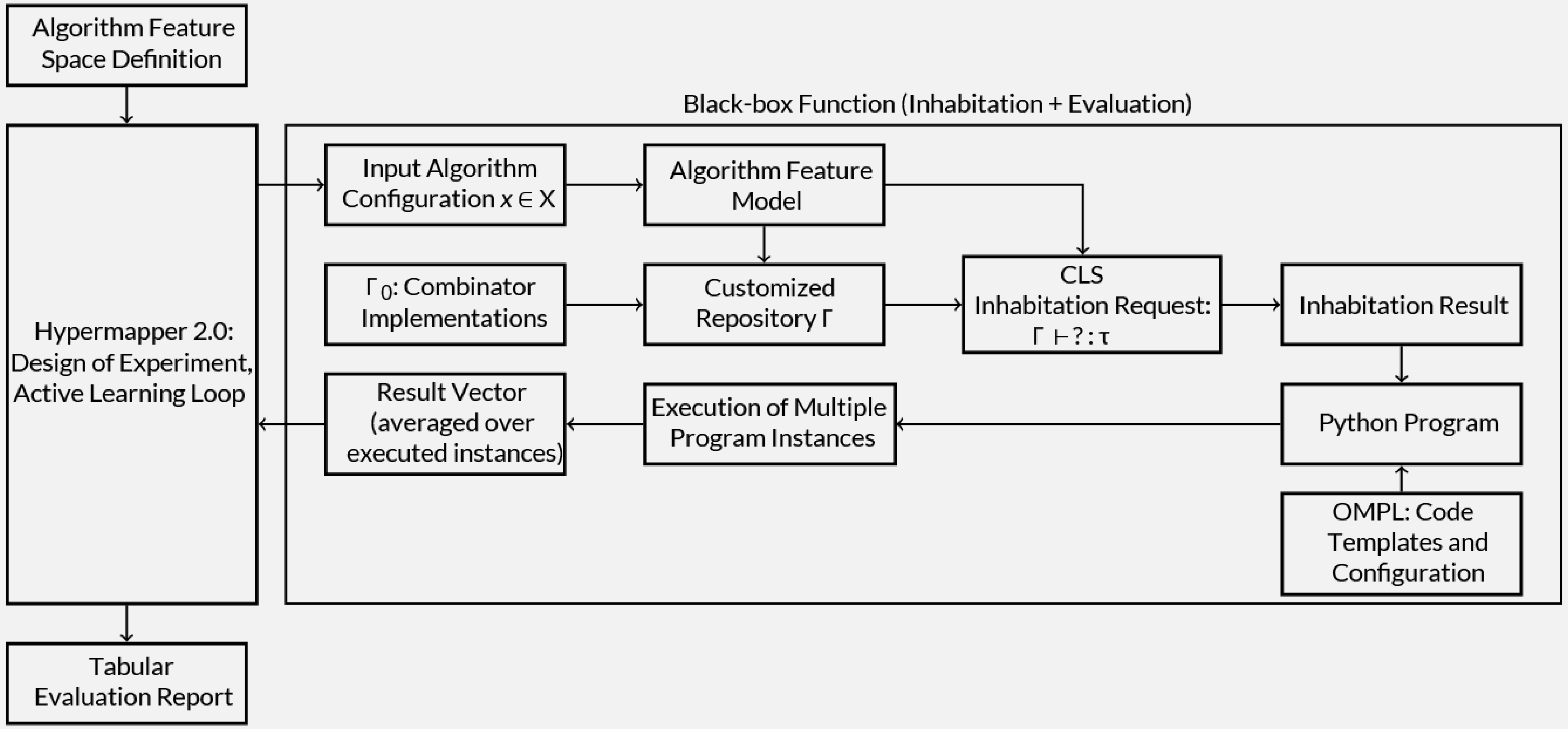
TACKLING A FUNDAMENTAL PROBLEM OF ROBOTICS WITH (CL)S:
FINDING A COMPROMISE BETWEEN PERFORMANCE METRICS.

Schäfer, Bessai, Chaumet, Rehof, Riest*: Design Space Exploration for Sampling-based Motion Planning Programs with Combinatory Logic Synthesis*, WAFR 2022

APPROACH.

Motion Planning Program Design Space:

- Planner
- Sampler
- Motion Validator
- Maximum Planning Time



PRODUCES GENERATORS FOR PYTHON PLANNING PROGRAMS,

WHICH IN TURN PRODUCE RESULT VECTORS IN $\mathbb{R}^n$.

THESE RESULTS GET EXTRACTED, COMPARED, AND BECOME PART OF THE HYPERMAPPER LEARNING LOOP.

WWW.DOI.ORG/10.1109/MASCOTS.2019.00053

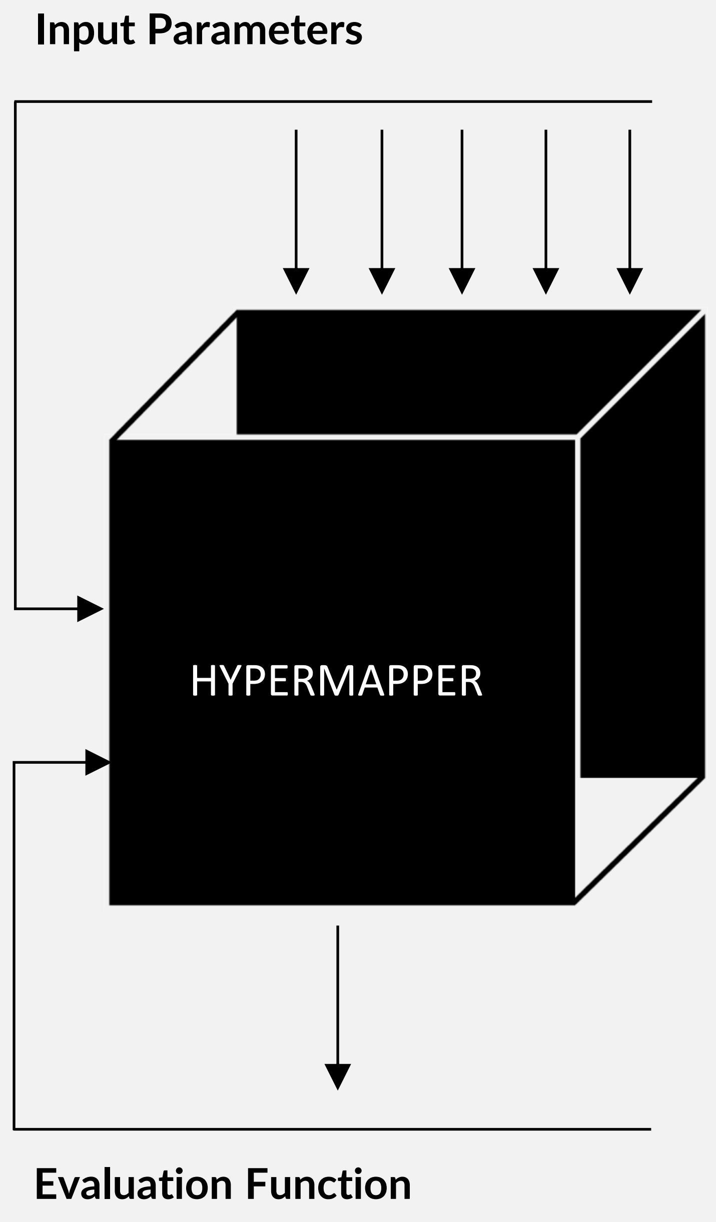
HYPER MAPPER

Black-Box Optimizer using Bayesian Optimization

Often a design space will not have a clearly defined notion of a derivative, making the pareto front impossible to determine analytically.

The *HyperMapper* can solve multi-objective optimization problems in such cases, requiring only a set of *input parameters* and an *evaluation function (multi-objective)*.
The input parameters are then optimized to minimize the evaluation functions objectives.

An initial pareto front of the objectives is found in the warm-up phase.
The following active learning phase learn a model that approximates the true MOP function, iteratively improving model and pareto front.



WWW.GITHUB.COM/LUINARDI/HYPERMAPPER

Nardi, L., Koeplinger, D., Olukotun, K.: *Practical design space exploration*. In: MASCOTS, pp. 347–358. IEEE Computer Society (2019).

WWW.DOI.ORG/10.1109/MASCOTS.2019.00053

HYPER MAPPER

Black-Box Optimizer using Bayesian Optimization

Randomized decision forests

Regression random forests

Injection of prior knowledge (distributions)

Sampling with categorical and discrete parameters

Constrained Bayesian optimization

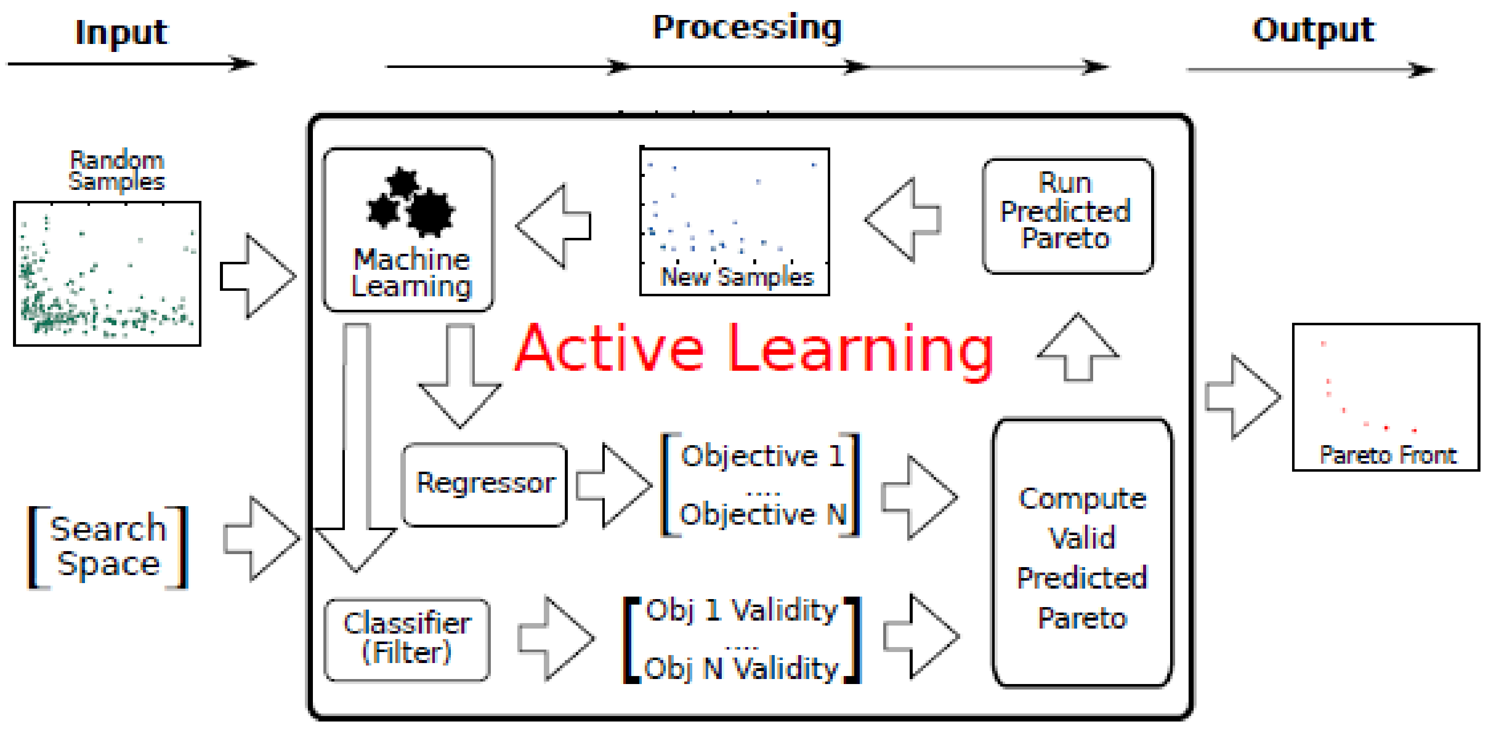


Fig. 3: Active learning with unknown feasibility constraints.

WWW.GITHUB.COM/LUINARDI/HYPERMAPPER

Nardi, L., Koeplinger, D., Olukotun, K.: *Practical design space exploration*. In: MASCOTS, pp. 347–358. IEEE Computer Society (2019).

Feature space exploration for sample-based motion planning

Optimization problem

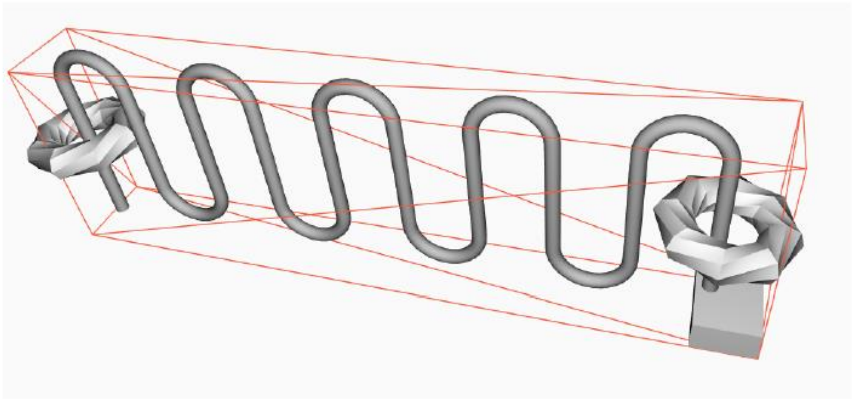
Formulation

Function f: $\mathbb{X} \rightarrow \mathbb{R}$, $\mathbb{X}$ denotes domain of interest (i.e. design space)
Optimization problem for mono objective : $x_{min} = argmin\ f(x),\ x \in \mathbb{X}$

- Design space $\mathbb{X}$ for planning programs: Planner, sampler, state validator, motion validator
- Solution vector in $\mathbb{R}^n$: Maximal computation time, path length, computation time, number of failures (i.e. planner failed to return a result)
- Multi-objective optimization: explore pareto front

OMPL Example [1]

"Escape": Result path along narrow passage



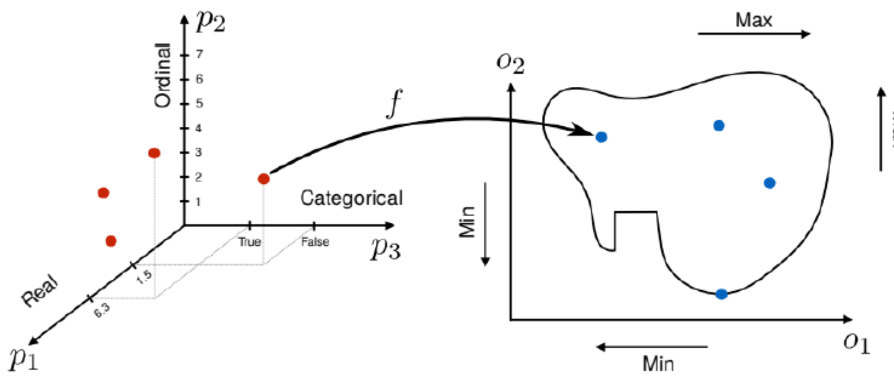


Figure: Multi-objective optimization [2]

Optimization problem

- Optimization problem is derivative-free due to ordinals, i.e. derivative of f is not available
- Given problem can be solved via design space exploration (DSE) (also referred to as derivative-free optimization (DFO), black-box optimization)
- Hypermapper 2.0 [2]
  - Design of Experiment (DoE) followed by optimization
  - Assumes that f is deterministic
  - Bayesian optimization with prior injection
  - Random search
  - Local search

VALIDATION

To validate the approach, we combined the approach with different ongoing research.

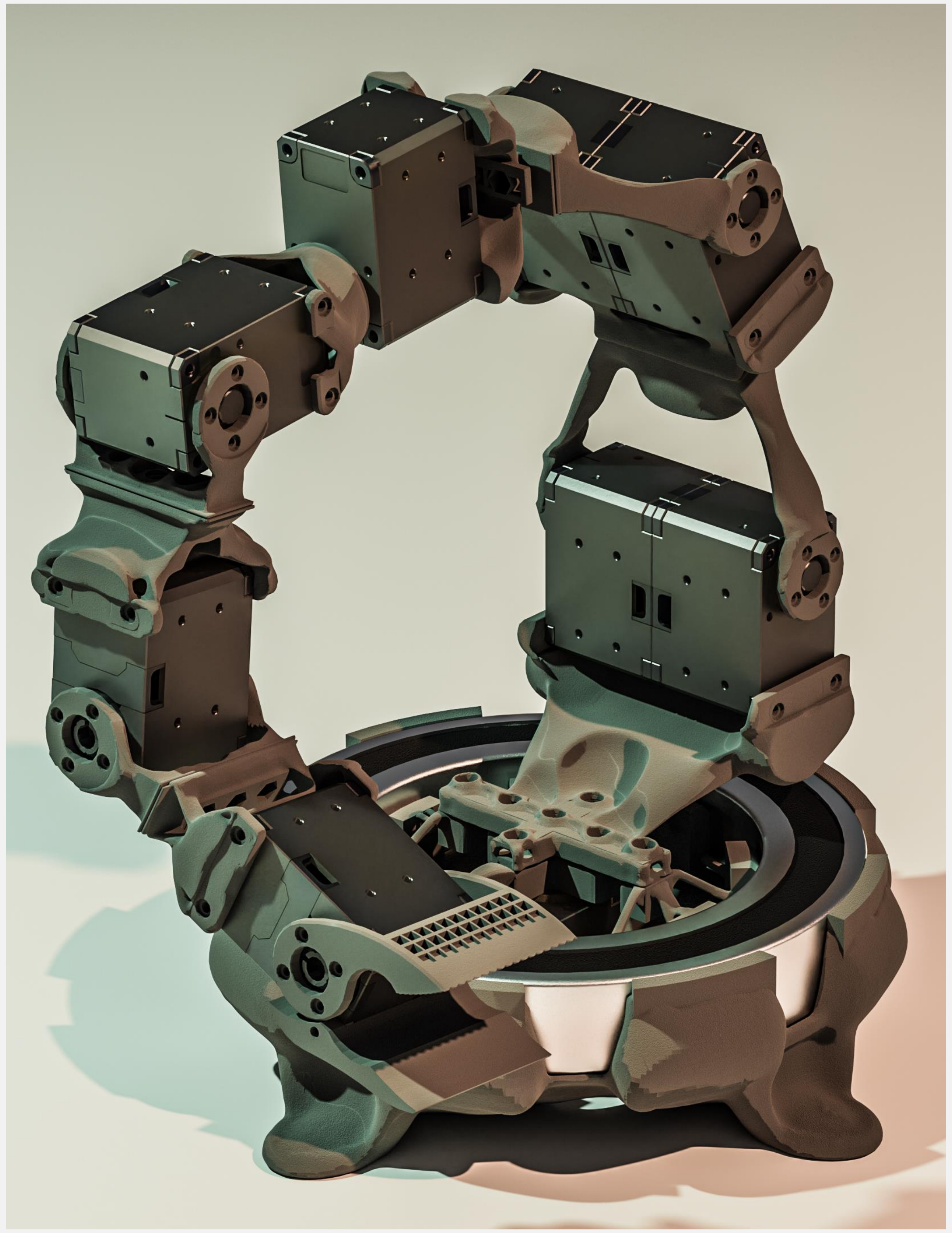
ROBOTIC ARM AUTOGENERATION

The (CL)S framework is also able to generate fully functioning robotic arms.

We utilised these to validate the approach on a

- Multitude of planning situations,
- With different robotic arms.

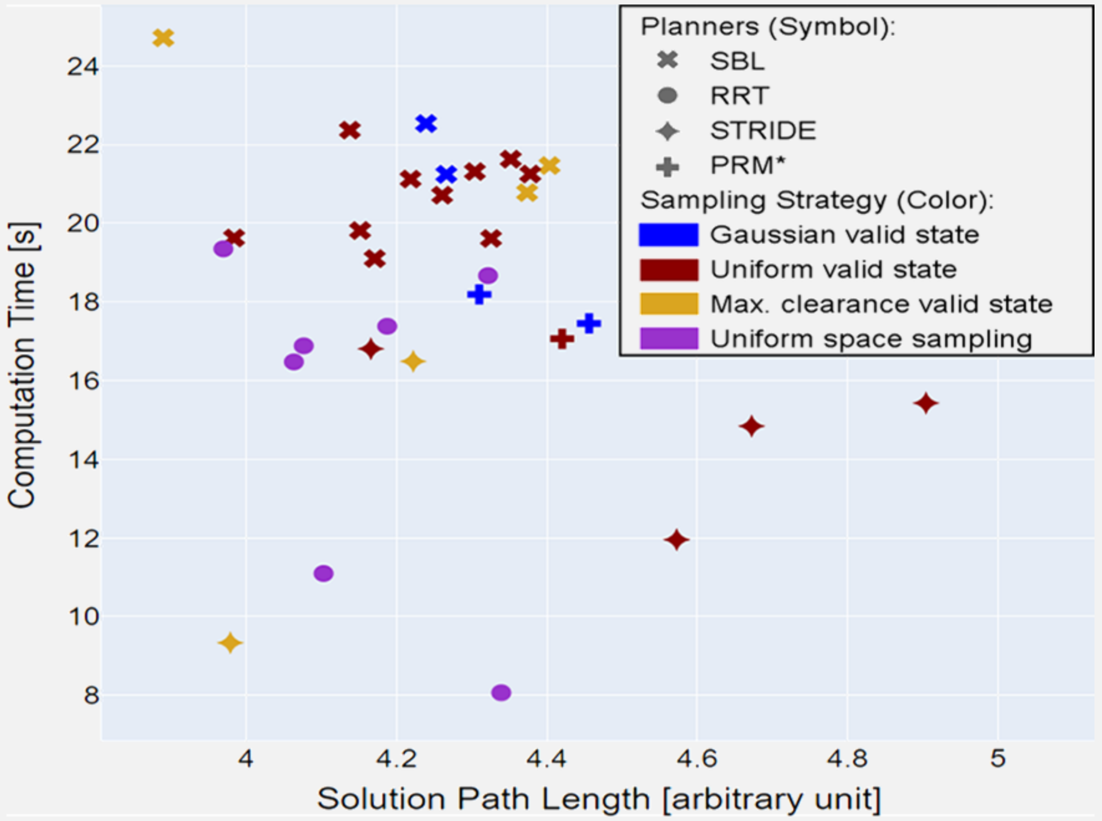
This allowed us to compare the performance of different planner setups for different scenarios, obtaining sets of pareto-optimal planning configurations.



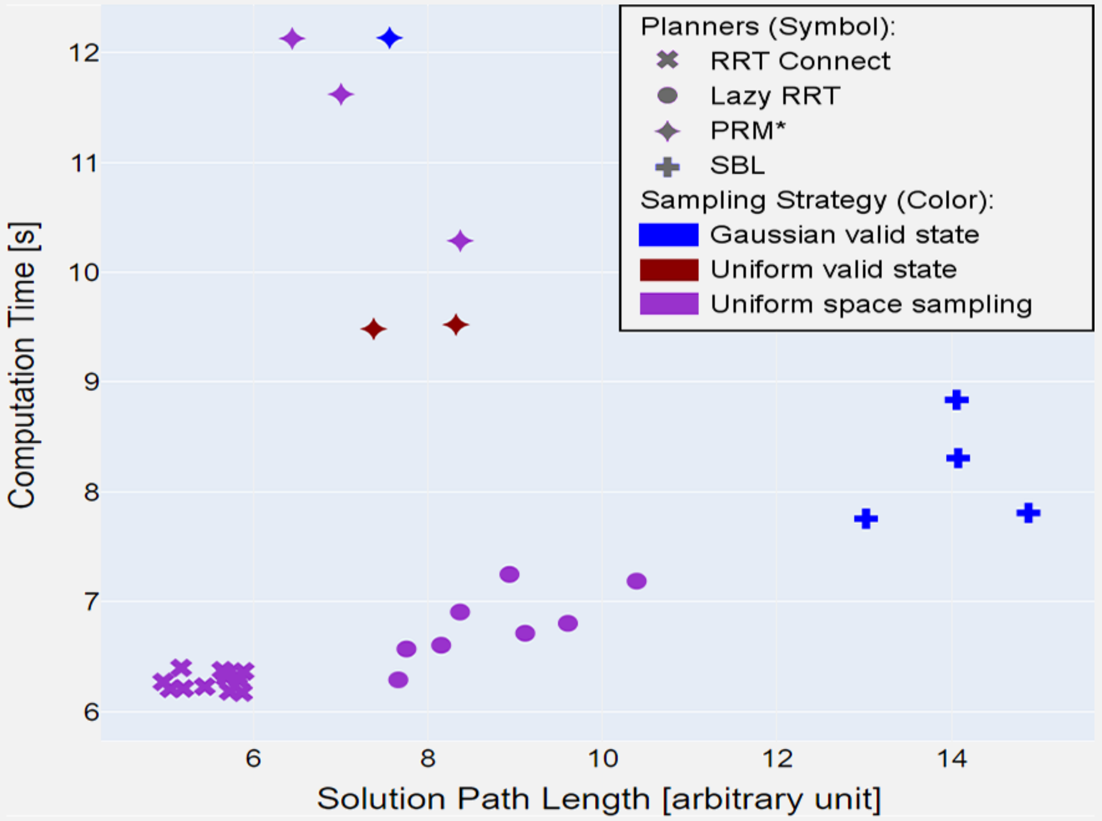
Synthesized robotic arm with attached simple gripper.

PARETO FRONTS

Scene with path simplification



Scene without path simplification



Pareto fronts for computation time and path length (shorter is better).

LEFT

Shortest path easily missed by humans:

- Long computation time
- Unusual combination

RIGHT

Lack of path simplification greatly favours RRT Connect with Uniform Sampling.

If human designer doesn't test specifically that combination, results are very sub-optimal.

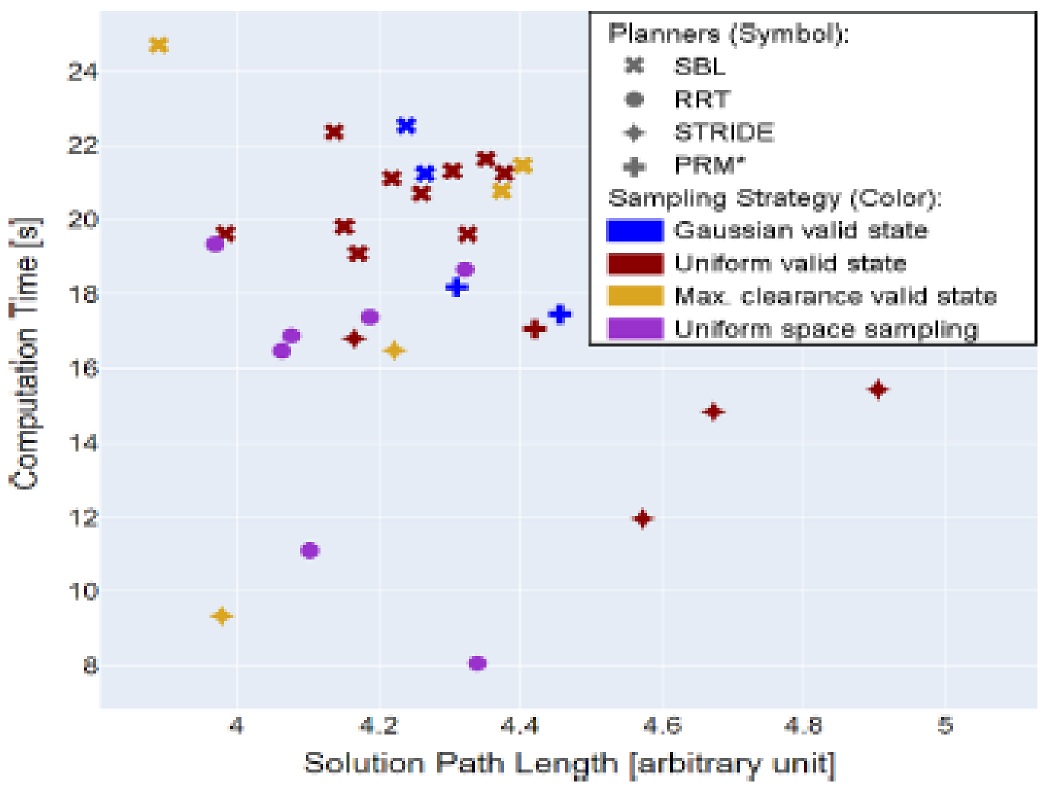


Fig. 6. cylinder with path simplification

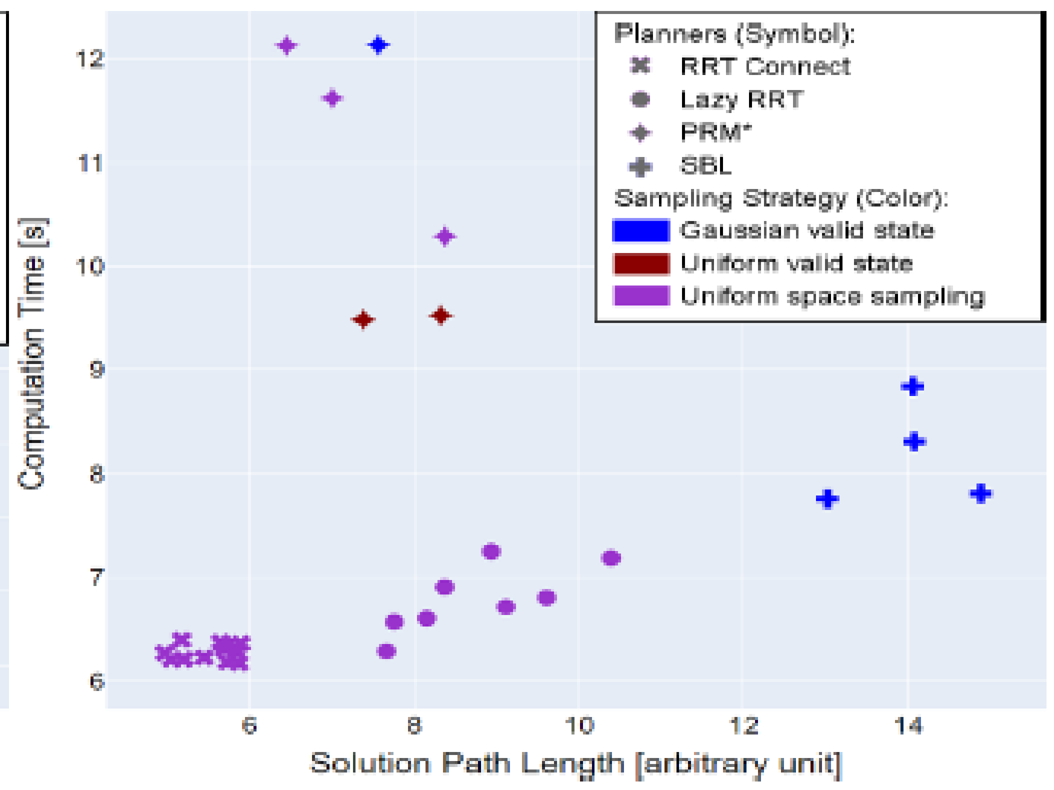


Fig. 7. cylinder without path simplification

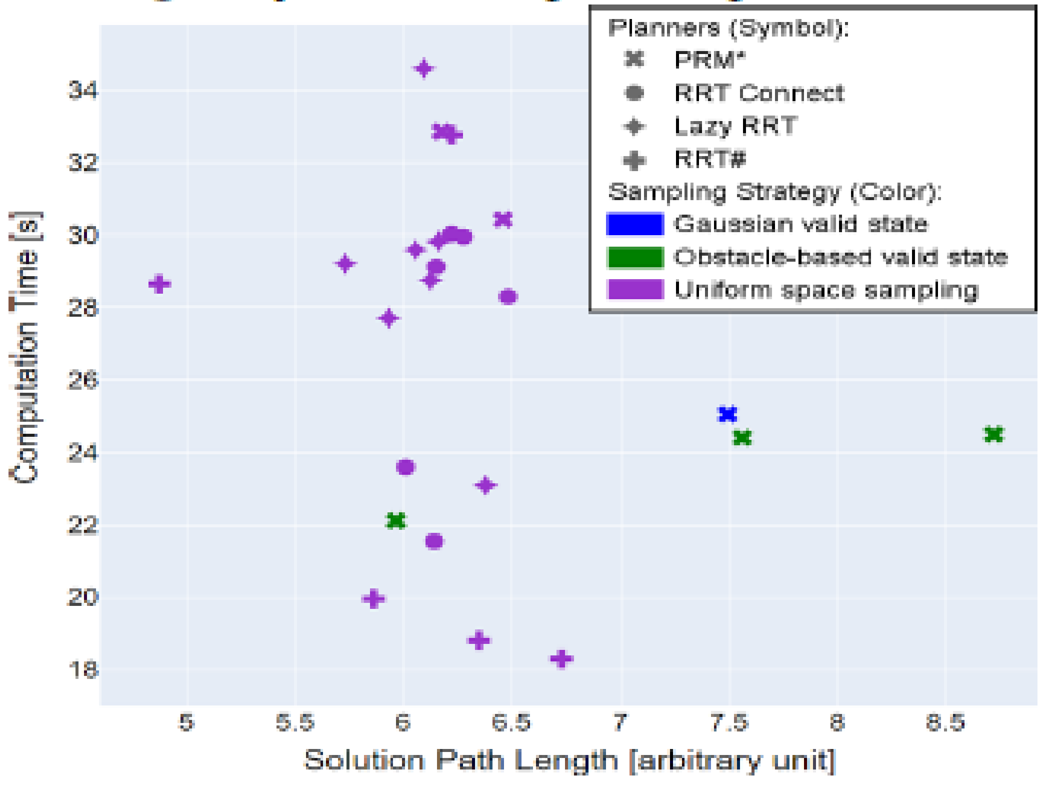


Fig. 8. pillars with 5 DoF

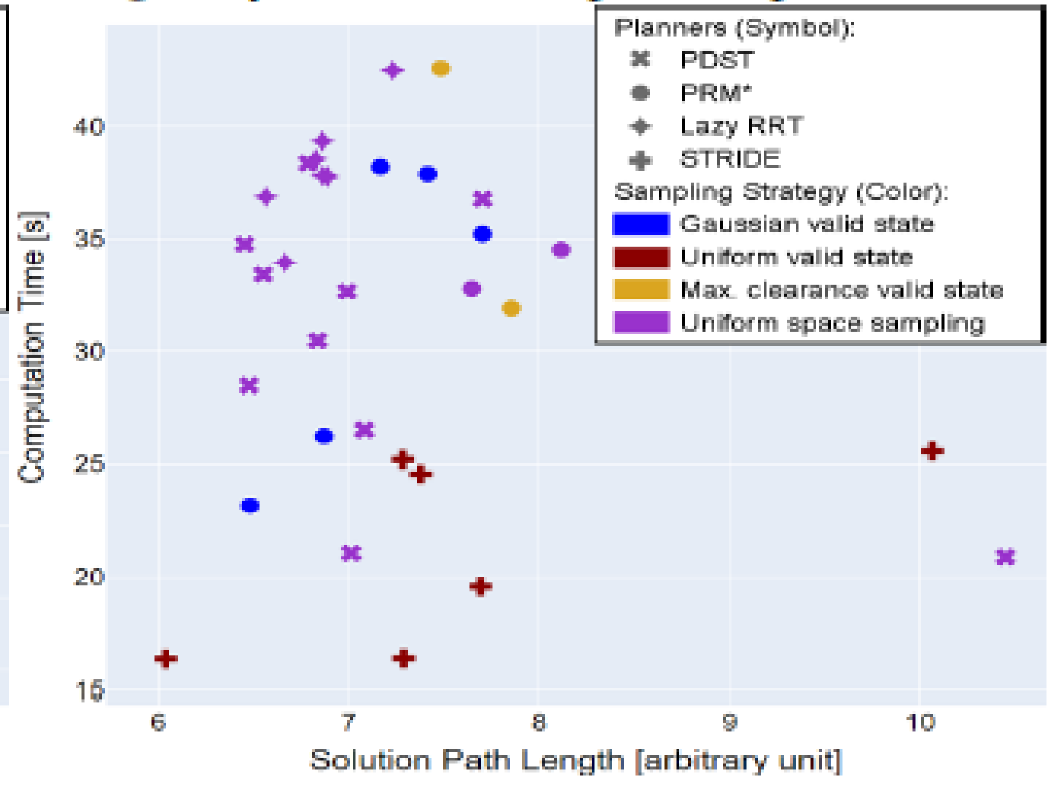
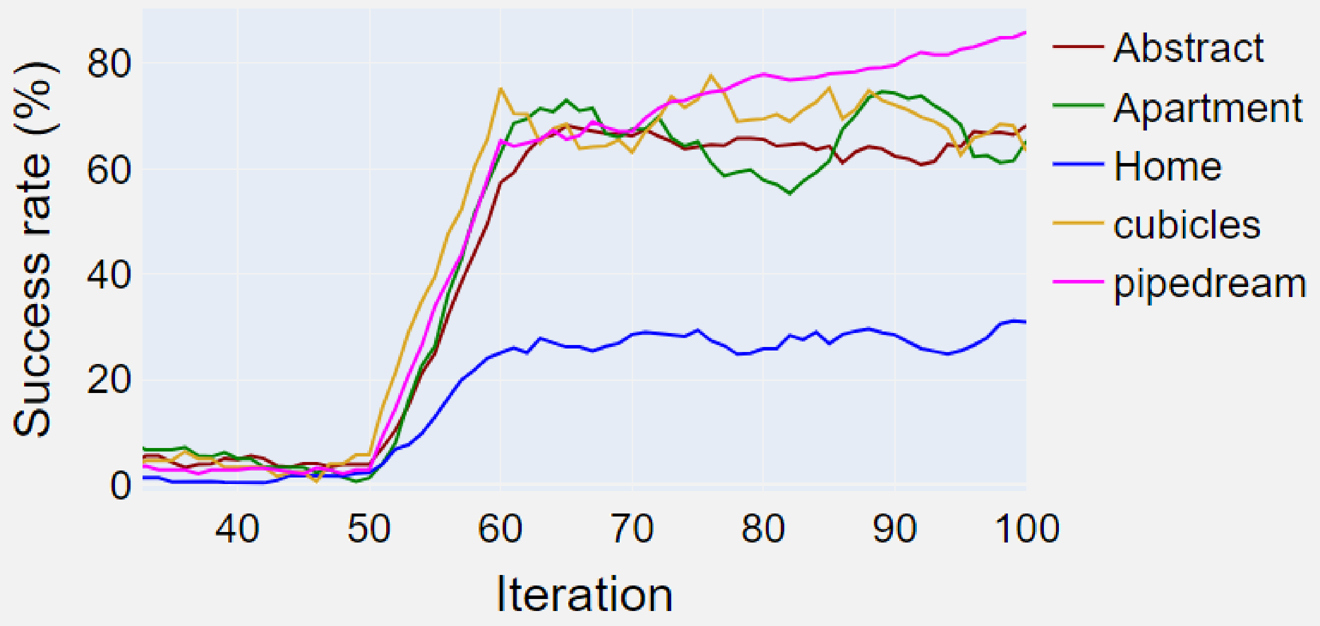
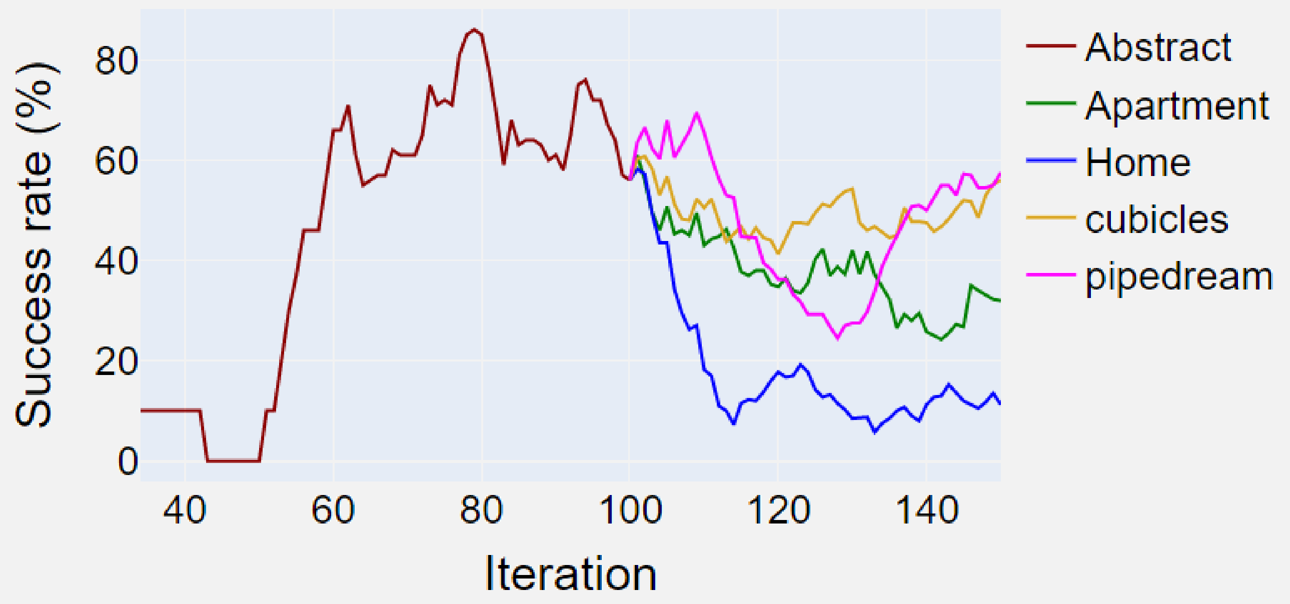


Fig. 9. pillars with 6 DoF

WWW.DOI.ORG/10.1007/978-3-031-21090-7_3





MODEL TRANSFER

We investigated if the learned models of what makes a good planner can be transferred to other models.

The plots shown are from path finding problems, e.g. a robot navigating through an environment.

The model transferred from the problem "Abstract" performed acceptably well for most of the other problem instances.

The models learned on robotic arms were not transferrable, due to them being specific to the robot and the robots being inherently very different.

WWW.GITHUB.COM/TUDO-SEAL/CLS-CAD

SYNTHESIS OF CAD ASSEMBLIES

RELIEVING CAD SOFTWARE ENGINEERS FROM REPEATING THE SAME BASIC TASKS OVER AND OVER;
AUTO-GENERATING ASSEMBLIES AND IMPROVING CREATIVITY.

Constantin Chaumet, Jakob Rehof: *CLS-CAD: Synthesizing CAD Assemblies in Fusion 360*. Subm. ASE 2023

REPETITION

Products nowadays are rarely one-offs, but usually part of a larger product line.
Members are usually similar and share many identical modular parts.

While CAD software is ubiquitously used and is at the core of nearly all product design, repetitive tasks are poorly automated.

Synthesizing CAD assemblies can get rid of that repetition and explore the design space more thoroughly, enhancing **creativity**, **efficiency**, giving **formal guarantees**, and enabling **data-driven methods**.

REPETITION

Auto-generated by CLS-CAD

LEFT

Put in Perspective:

- 91 Screws
- 364 Clicks

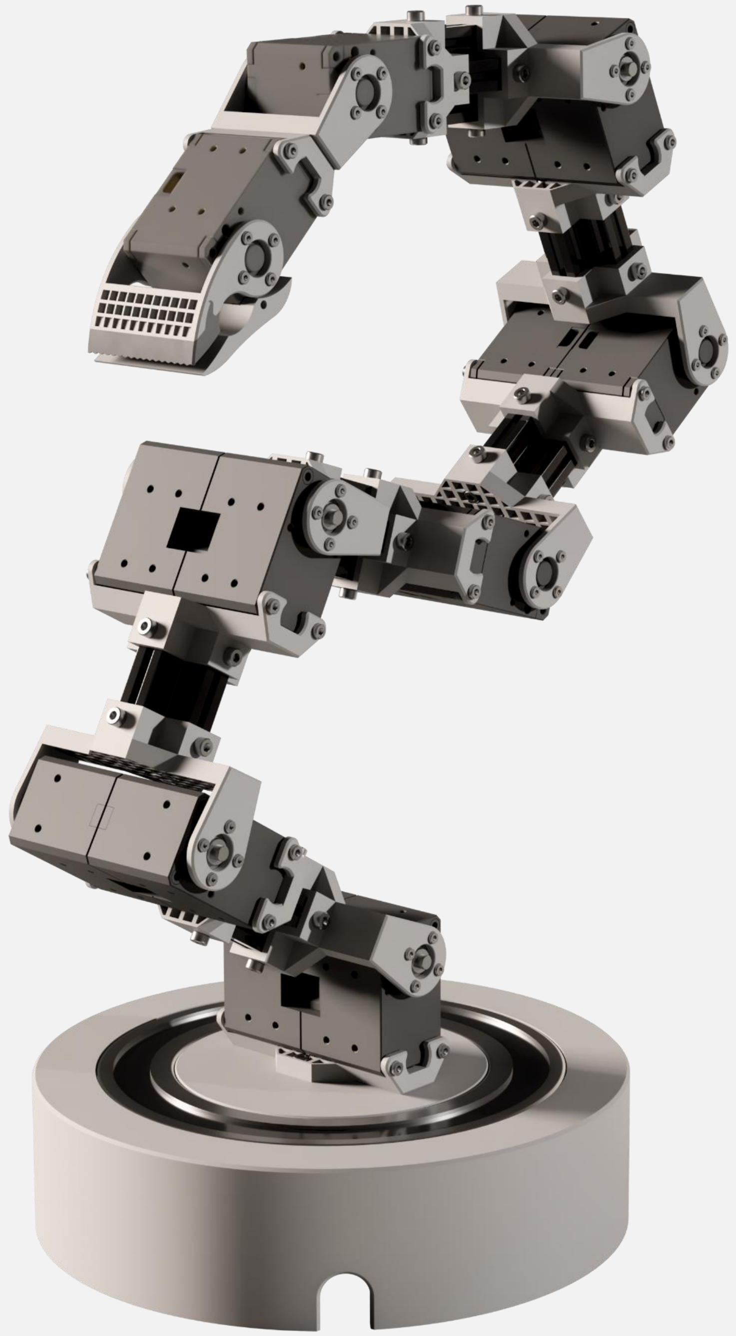
Just for inserting and connecting screws.

RIGHT

Put in Perspective:

- 209 Screws
- 836 Clicks

Just for inserting and connecting screws.



Auto-generated by CLS-CAD

SOLUTION

FUSION 360

Latest CAD package from Autodesk:

- Free for Academia
- Growing Popularity
- Actual Industry Usage *(Mid-Sized Companies)*

CLS-CAD

Add-In for Fusion 360:

- Managing of taxonomies/subtype hierarchies
- Annotating of geometry and documents
- Requesting and assembling products

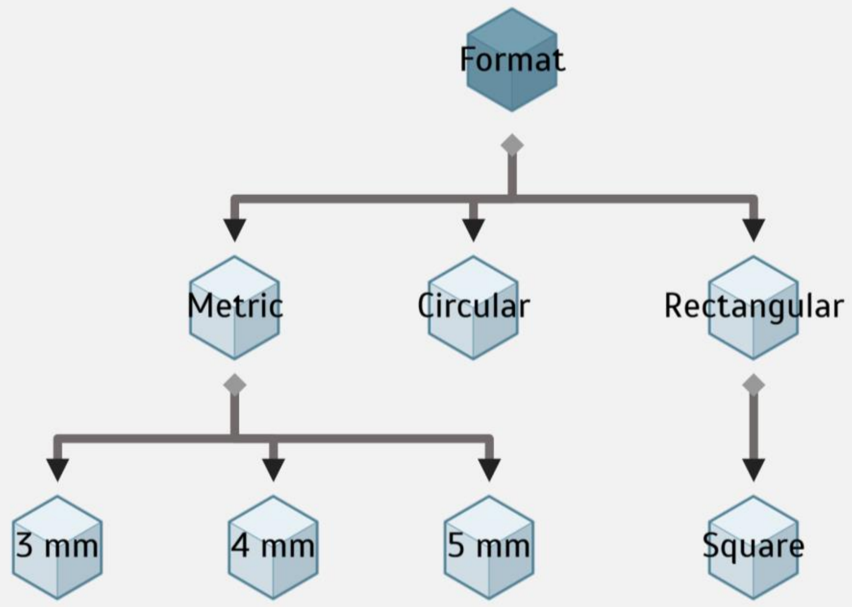
CLS-CAD ENRICHES CAD DOCUMENTS WITH TYPE INFORMATION, ALLOWS REQUESTING PRODUCTS, AND AUTOMATIC ASSEMBLY OF THE SET OF SOLUTIONS.

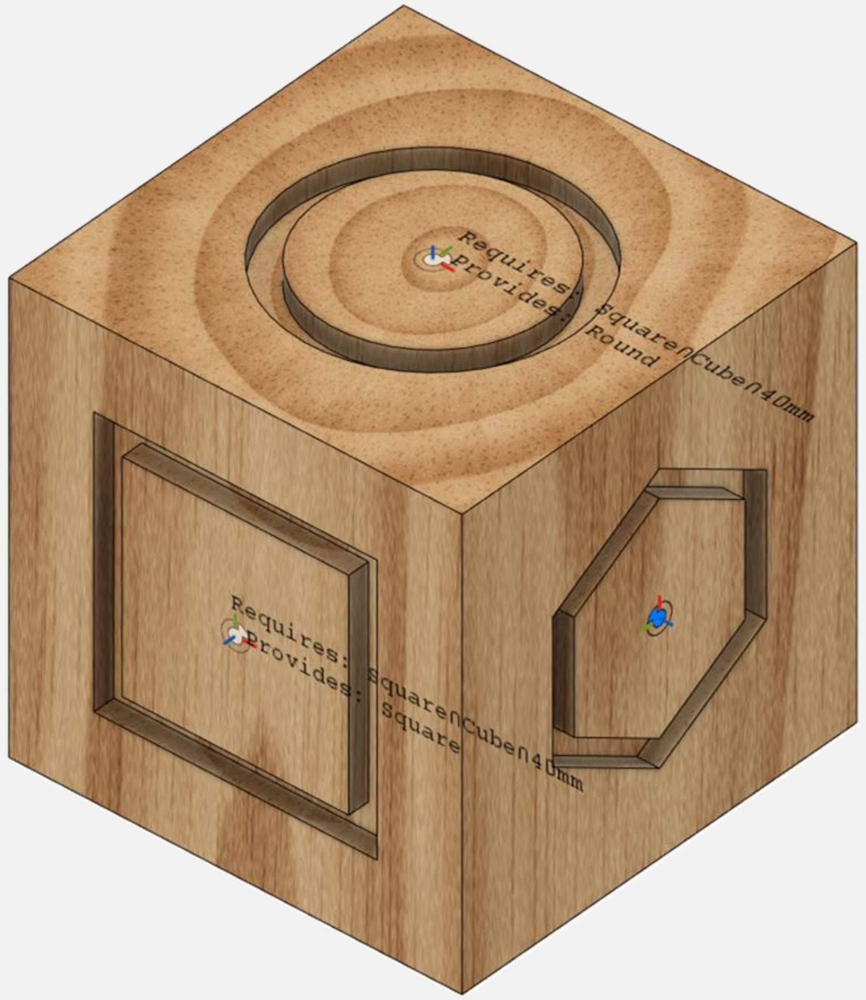
FULLY INTEGRATED INTO FUSION 360 AS AN ADD-IN.

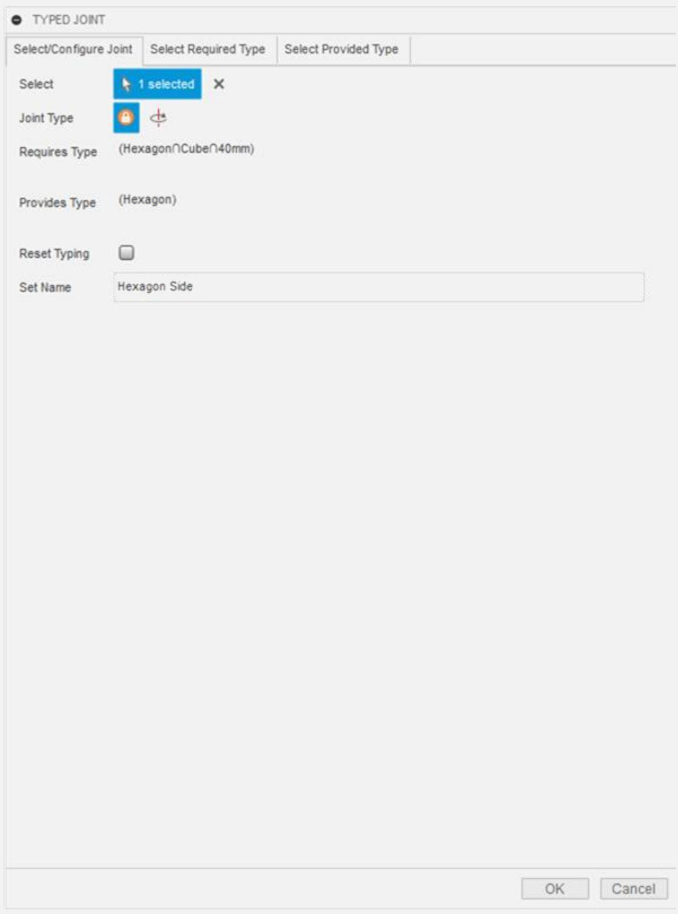
DEMO: WWW.YOUTUBE.COM/WATCH?V=GK00STSAXUK

TAXONOMY EDITOR

The taxonomy editor allows building and managing multiple large taxonomies in an interactive fashion.







TYPE ANNOTATION

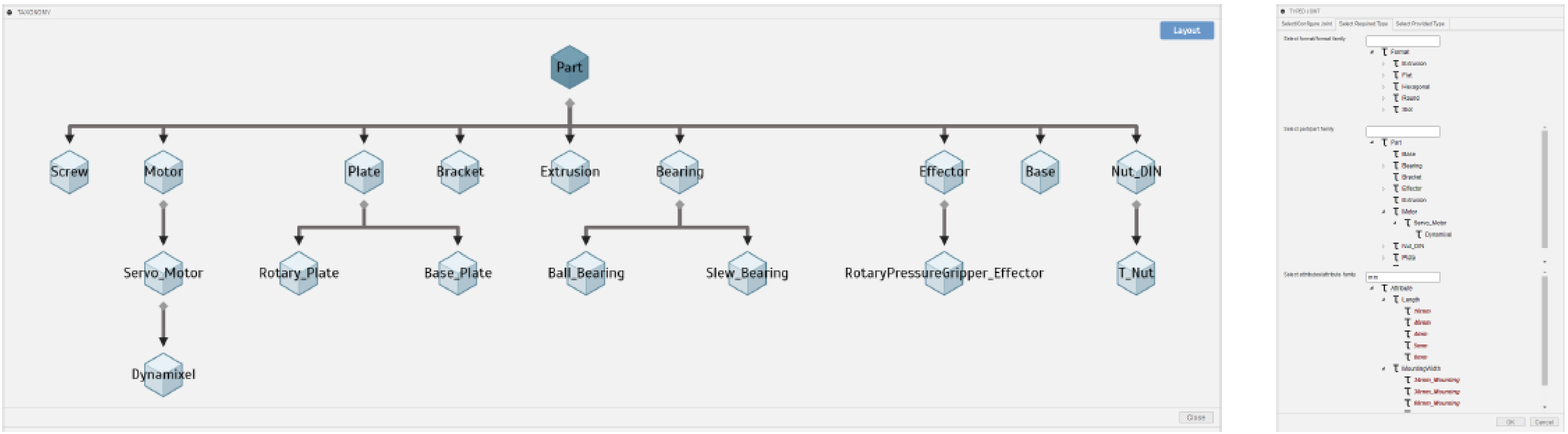
The annotation tools then allow typing the part with intersection types, thus defining connection possibilities that formally encode geometric restrictions, intent of connections, as well as material restrictions etc.

INHABITATION REQUEST

Assemblies are then requested based on an intersection type, with optional type propagation to restrict results further.

TAXONOMY EDITOR

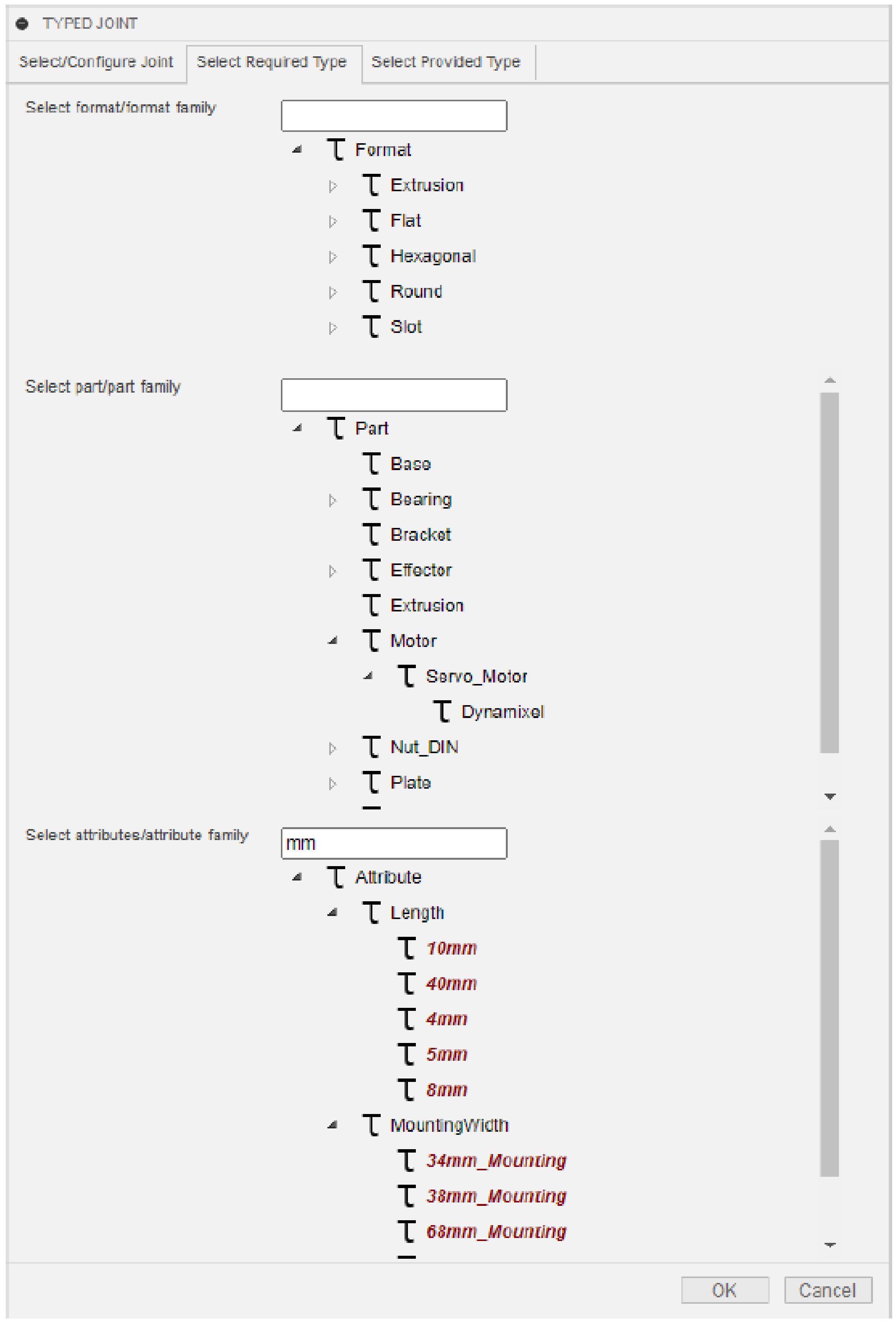
The taxonomy editor allows building and managing multiple large taxonomies in an interactive fashion.



(a) Display of subtype hierarchy for editing.

(b) Display of subtype hierarchy for selecting.

Fig. 1: Windows to manage the subtype hierarchy, created by the plugin natively in Fusion 360.



SUBOPTIMALITY

XL-430 SERVOMOTOR - $

XM-430 SERVOMOTOR - $$$

Sometimes parts get used (whether manually designed or not) that are suboptimal.

This can have many reasons:

- Unexpected downstream changes from usage
- Legacy part, used since forever
- Overly cautious dimensioning

These can be difficult to identify, either due to:

- habitual reasons when manually designing

or

- due to the complexity of the assembly, where it is not clear that using one part affects other parts of the assembly negatively.

GETTING RID OF SUBOPTIMAL PARTS

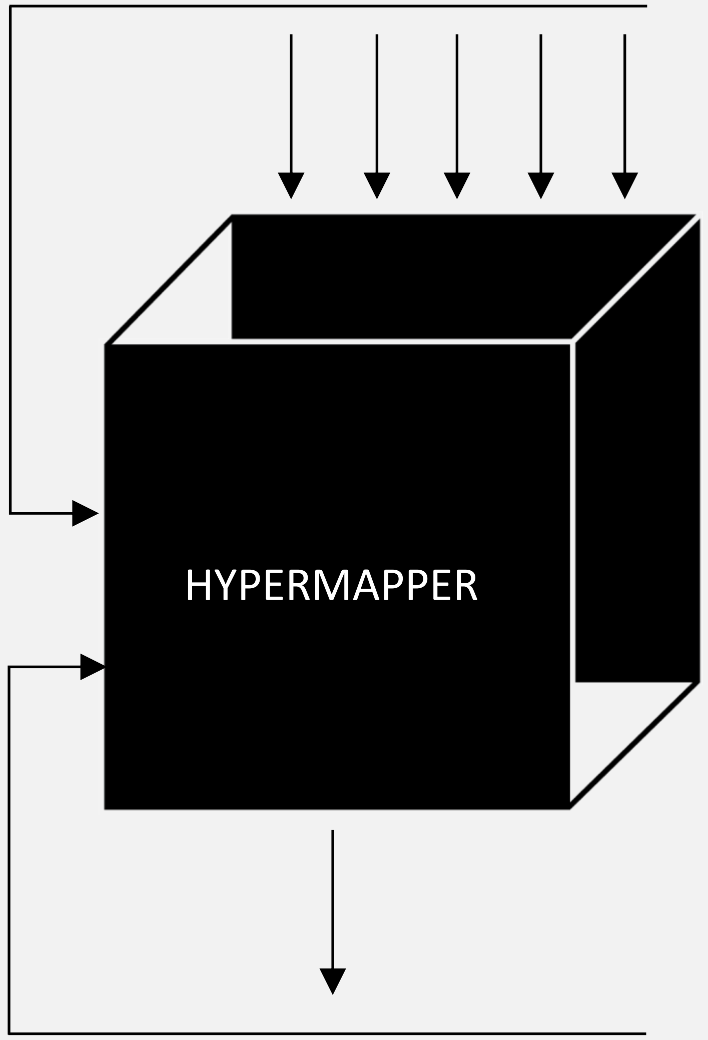
Only basic experimental validation completed thus far

We can view the set of all parts present in the repository as an input parameter vector. The learned model then gives insight into which parts should be avoided.

The *HyperMapper* iteratively keeps improving the model and the pareto front. This allows us to find CAD assemblies (products) that are pareto optimal w.r.t. the defined assembly metrics. Designers thus need to manually evaluate only the assemblies that are part of this pareto front.

The learned model can be utilized to improve the repository of parts. The lowest scoring entries in the input parameter vector can be vetted, allowing badly designed, inefficient or difficult to source parts to be identified and removed/avoided, leading to better products down the line,

Repository of Parts



Assembly Metrics:
- **Complexity**
- **Cost**
- **Part Availability**

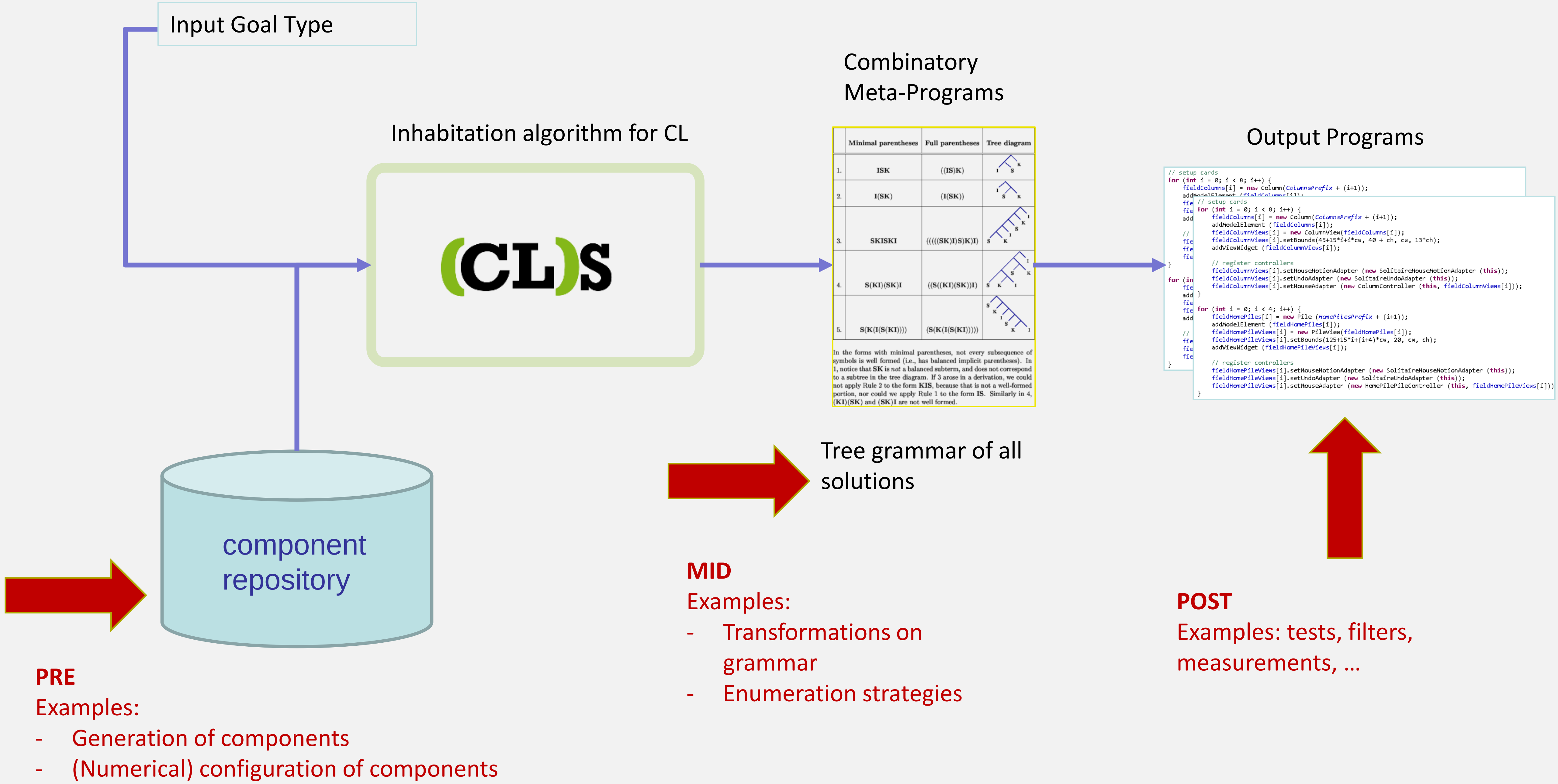
SUMMARY

(CL)S framework: Formally verified and language-agnostic synthesis of artefacts.

Synthesis and learning can be used to pareto-optimally solve motion planning problems and optimize planning parameters.

Synthesis and learning can be used to auto-generate CAD assemblies, reducing redundancy in product line engineering, and identify suboptimal parts.

Extensions to CLS-framework



ONGOING WORK

- Extension of type specifications with boolean connectives (TYPES 2023)
- Using rewriting and algebraic transformations on three grammar (FSCD 2022)
- Further work on CAD-integration (subm. ASE 2023)
- Systematic integration of numerical configuration spaces
- Enumeration strategies for controlled experiments and reinforcement learning
- Automatic configuration of algorithms (algorithmic families)
- Synthesis of NN-architectures, hyperparameter tuning, applications in AutoML …