GENERATING SIMULATION MODELS FROM COMPONENTS:

Industrial use cases in design space exploration and machine learning

NEXT GENERATION COMBINATORY LOGIC SYNTHESIS: CLSP

Jakob Rehof

TU Dortmund University

Lamarr Institute for Machine Learning and Artificial Intelligence

SummerSoc 2024, Crete, Greece
June 25, 2024

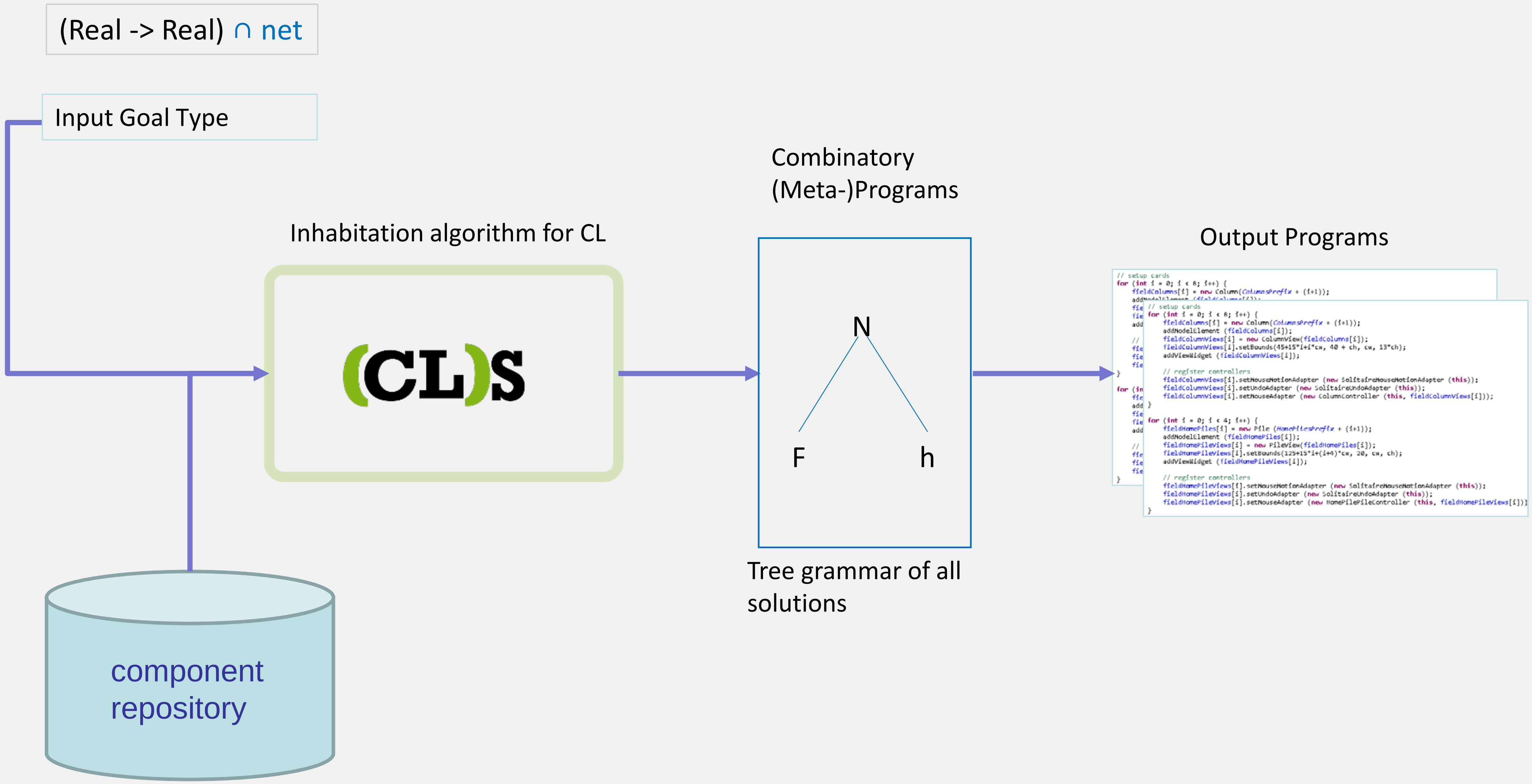
WWW.GITHUB.COM/TUDO-SEAL/BCLS-PYTHON

(CL)S

COMBINATORY LOGIC SYNTHESIZER recap

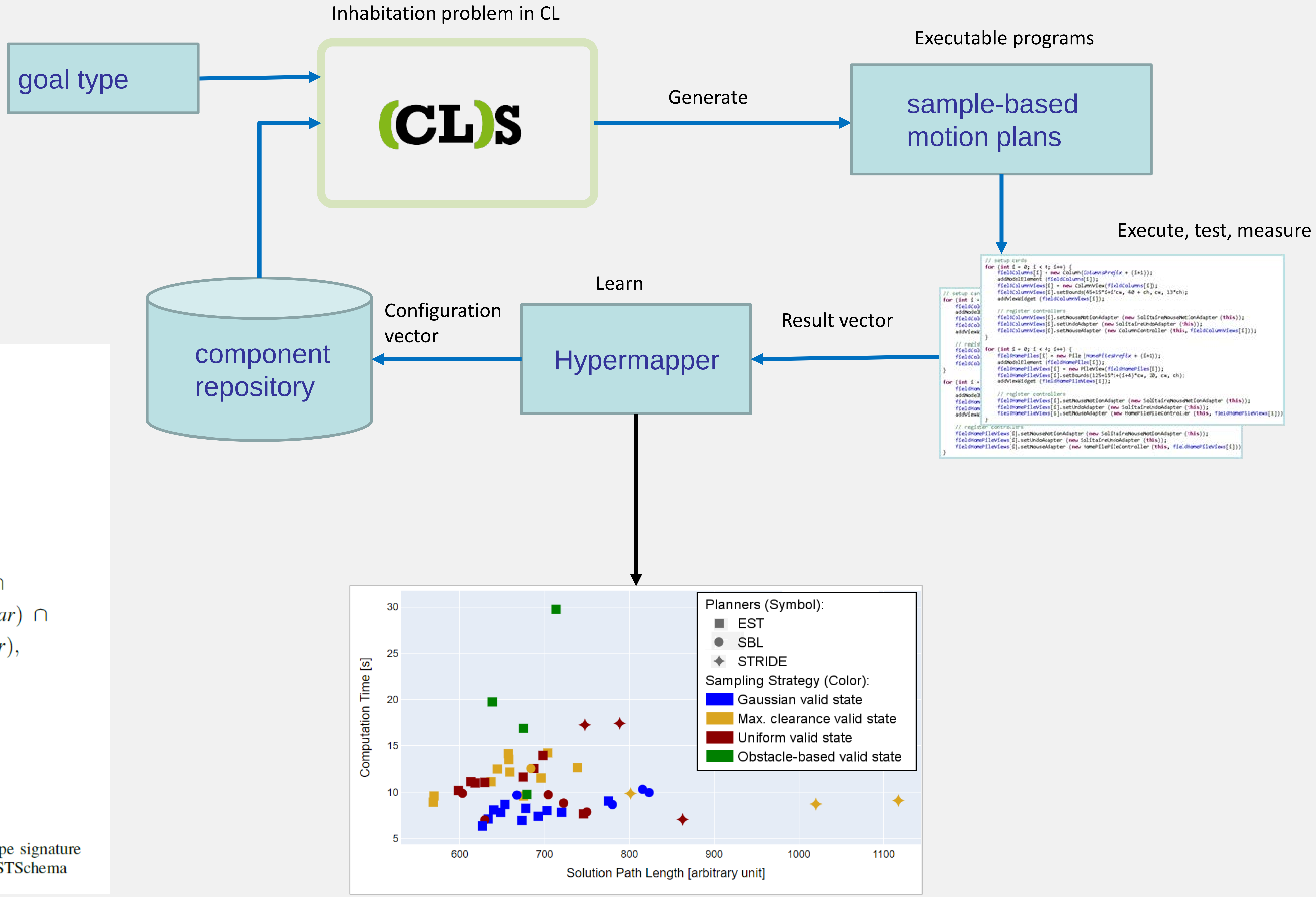
THE (CL)S FRAMEWORK IS A LANGUAGE-AGNOSTIC AND FORMALLY VERIFIED FRAMEWORK THAT IS ABLE TO GENERATE ALL COMBINATIONS OF MODULAR COMPONENTS THAT SATISFY A PARTICULAR REQUEST/SPECIFICATION.

Basic synthesis pipeline of CLS-framework



F: (Real -> Real) ∩ layer
h: (Real -> Real) ∩ activation
N: (Real -> Real) -> (Real -> Real) -> (Real -> Real) ∩ net
…

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 :

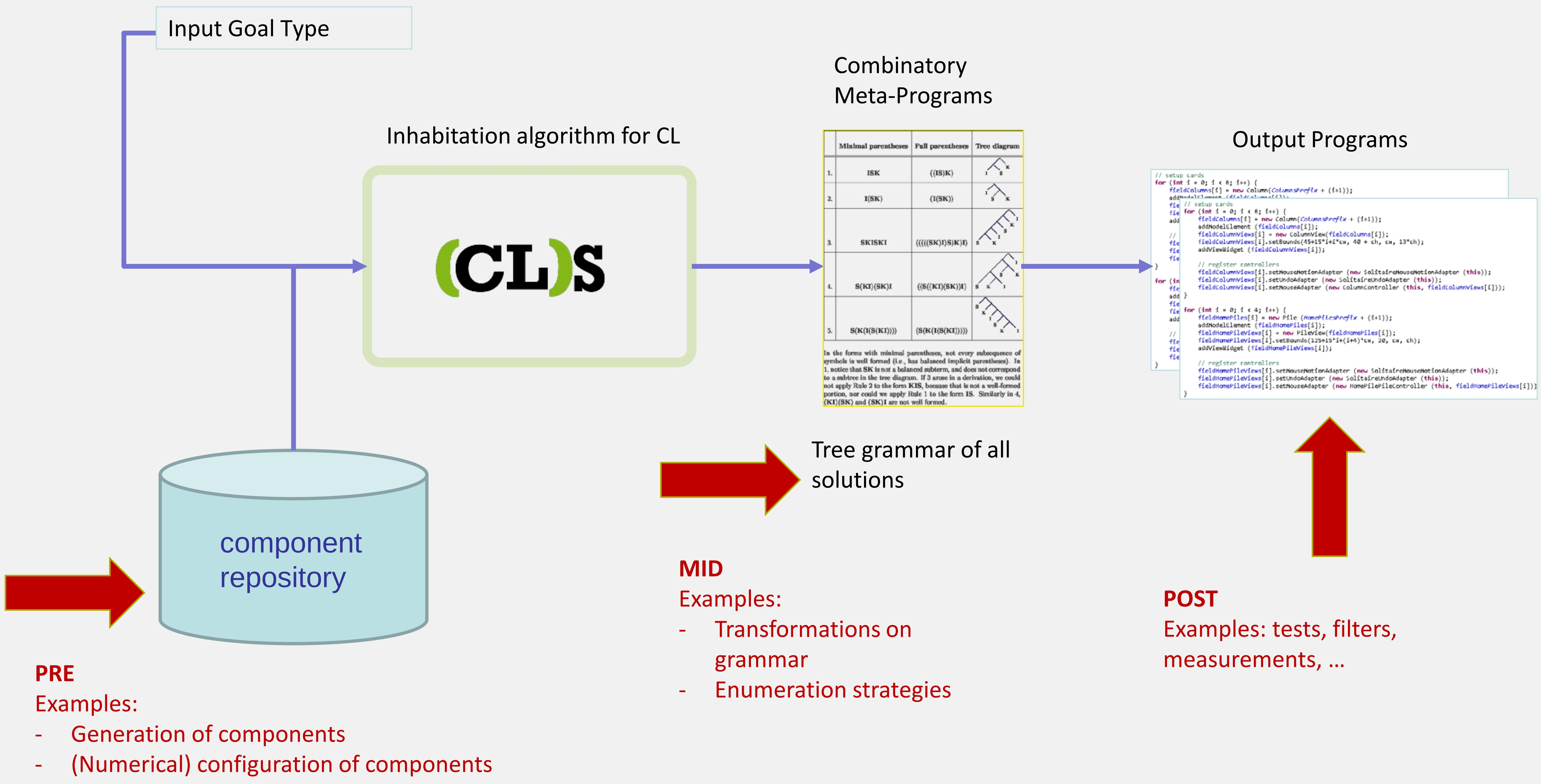
$obj\_path \to sampler\_valid\_state \to EST,$

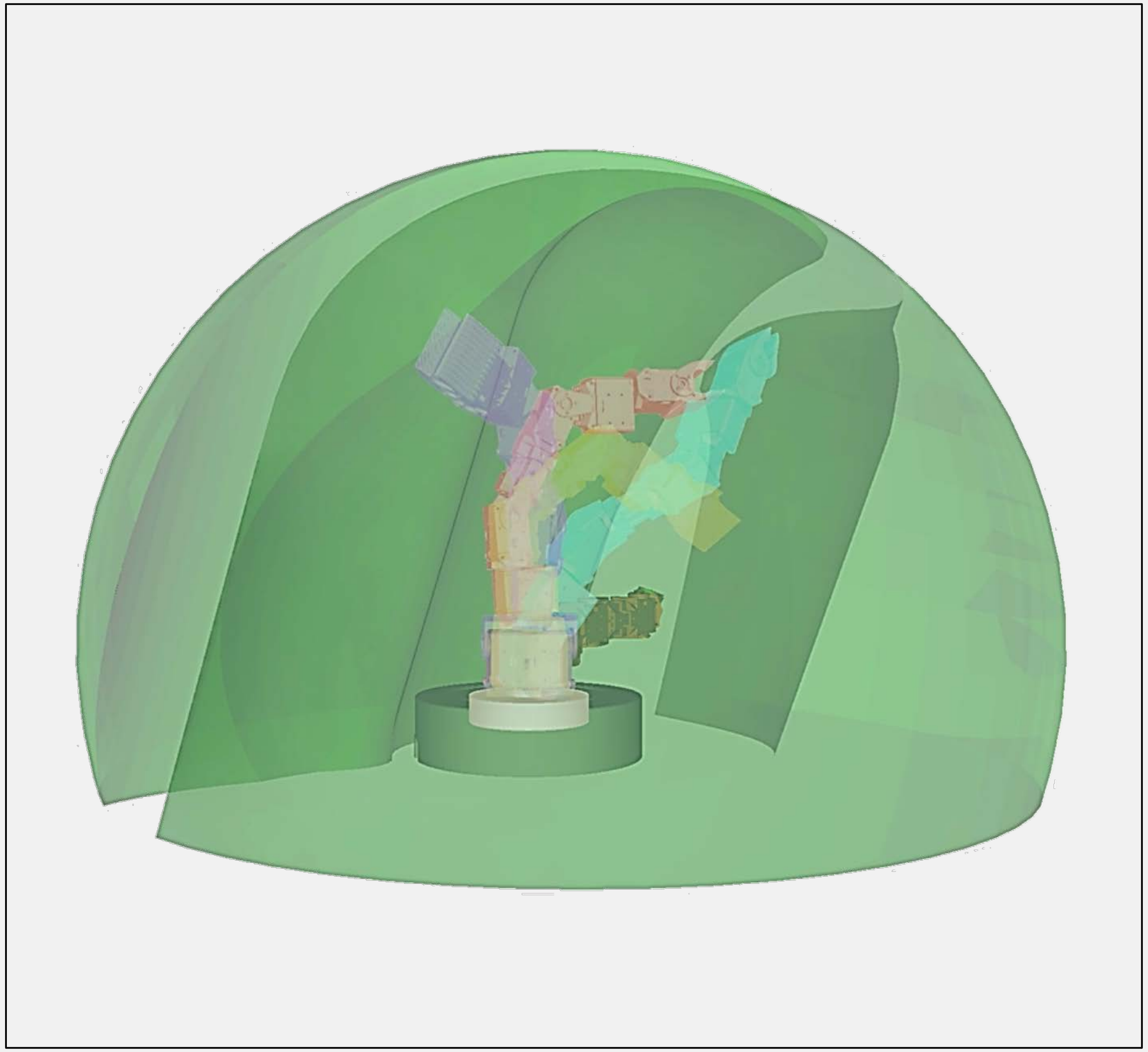
[...]

$\}$

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

Extensions to CLS-framework



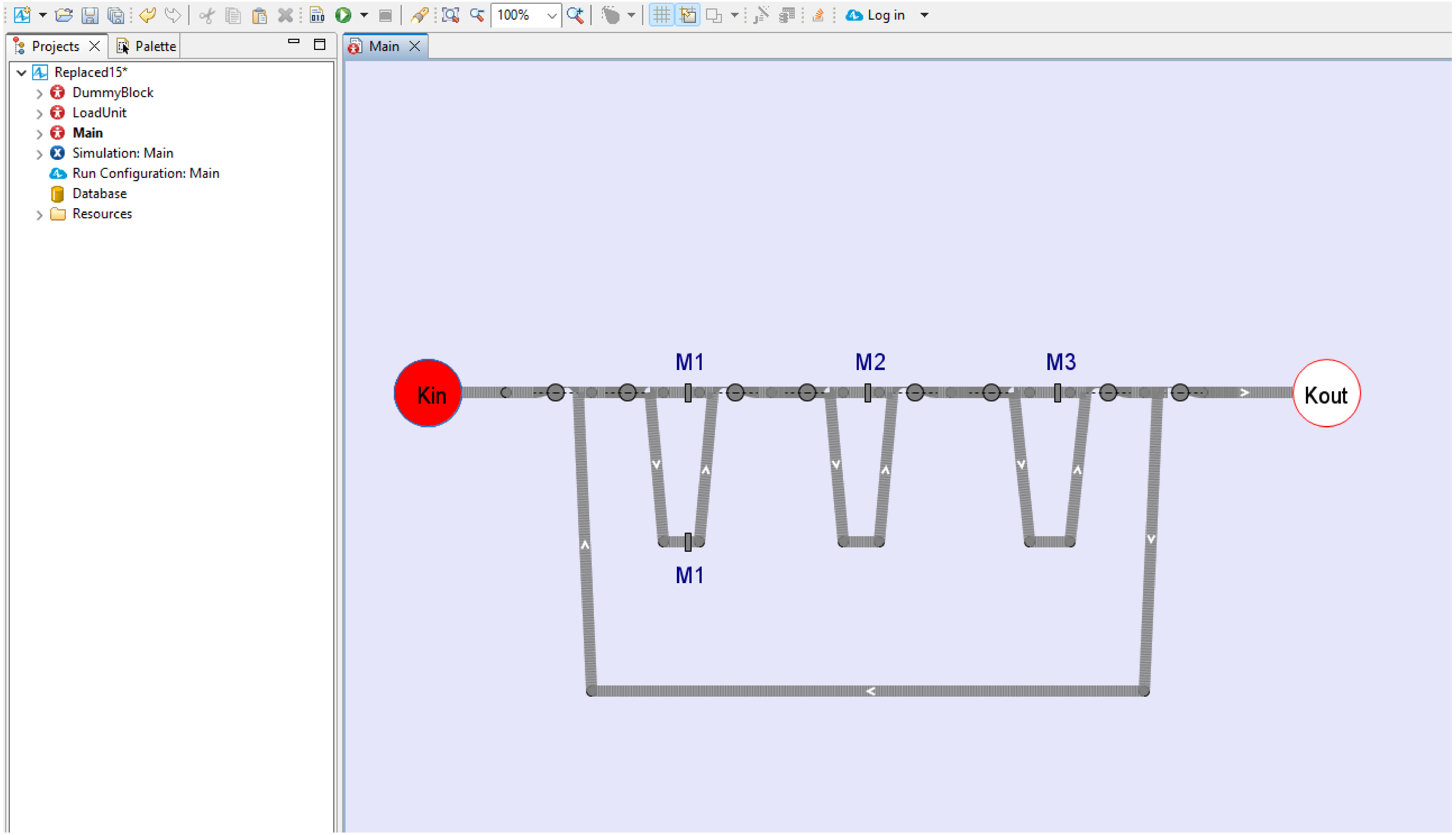


MOTION PLANNING

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

Executable Simulation Models



- All generated solutions can be compiled into an executable simulation model in **AnyLogic**
- See solution #15 from previous step

```
##########--15--############

||(1)|  -> |(2)|  -> |(3)||
||(1)|     |-->|     |-->||
|<----------------------|
```


Deutsche Forschungsgemeinschaft
Projektnummer: 511349842

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, Thomas Schuster: *A knowledge-driven framework for synthesizing designs from modular components*.
34th CIRP Design Conference, 2024

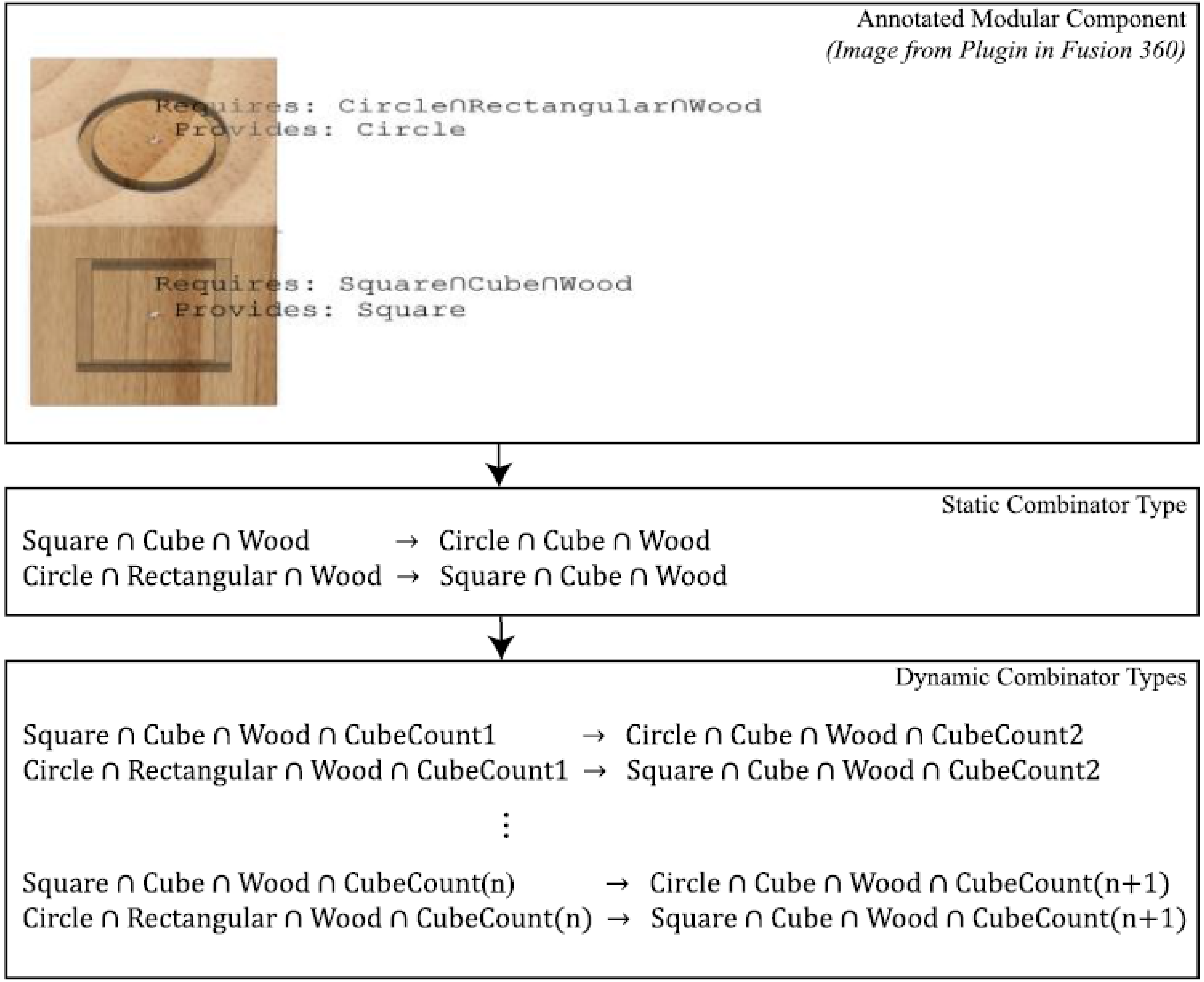
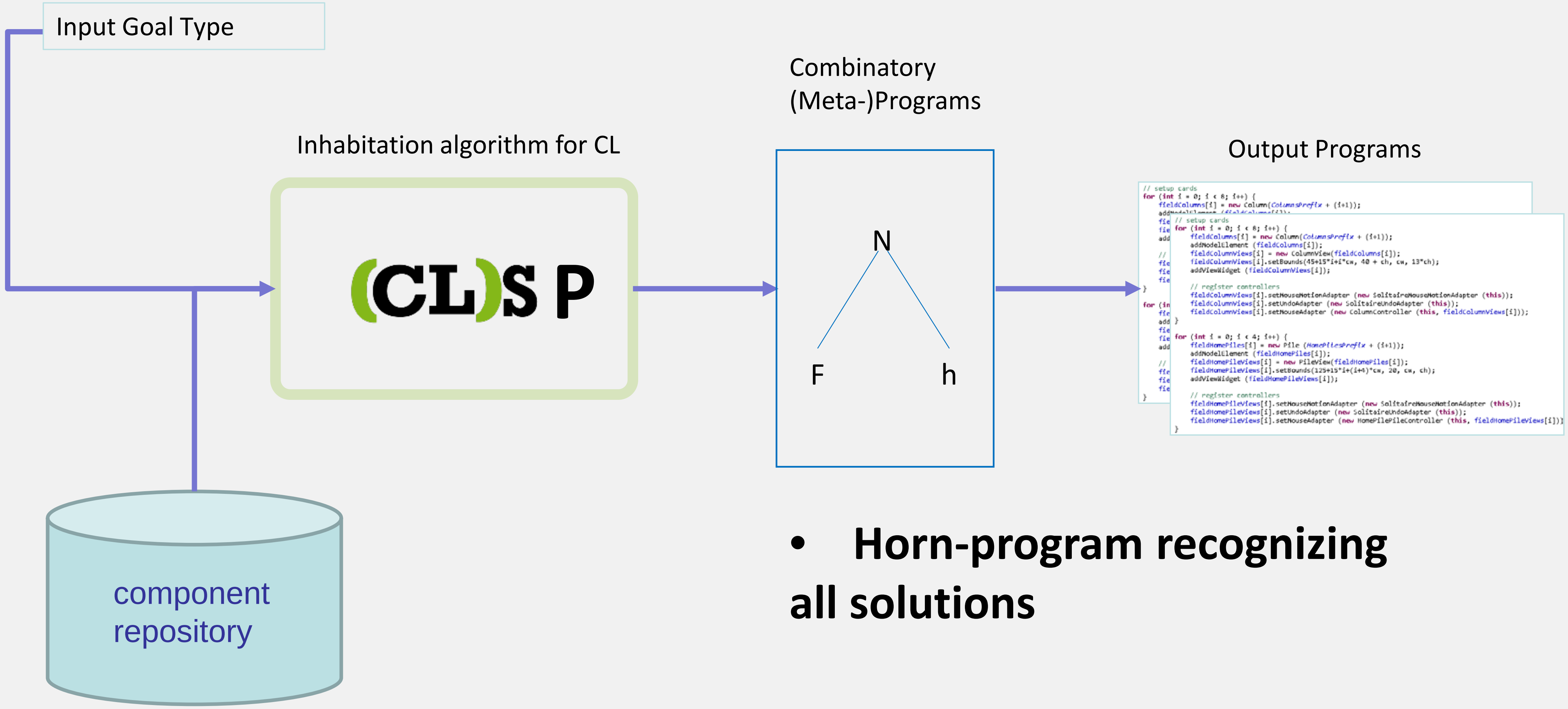


Fig. 3. Dynamic type generation for repository.

Next generation CLS: CLS with predicates (CLSP)



- Horn-program recognizing all solutions

- Dependent type specifications with computable predicates

Dudenhefner, Stahl, Chaumet, Laarmann, Rehof: *Finite Combinatory Logic with Predicates*. TYPES `23 post-proceedings. LIPIcs 2024.

Size	$\Gamma^{\mathcal{M}}_{\text{FCL}}$	$\Gamma^{\mathcal{M}}_{\text{FCLP(lit)}}$	$\Gamma^{\mathcal{M}}_{\text{FCLP(pos)}}$	$\Gamma^{\mathcal{M}}_{\text{FCLP(pred)}}$	$\Gamma^{\mathcal{M}}_{\text{FCLP}}$
10×10	$1.3\,s$	$0.5\,s$	$0.3\,s$	$0.1\,s$	$0.1\,s$
20×20	$21.9\,s$	$8.0\,s$	$6.4\,s$	$2.4\,s$	$1.9\,s$
30×30	$125.2\,s$	$41.0\,s$	$30.7\,s$	$12.8\,s$	$9.8\,s$
40×40	$464.7\,s$	$130.2\,s$	$97.9\,s$	$42.3\,s$	$32.4\,s$
50×50	$1279.8\,s$	$322.2\,s$	$239.5\,s$	$103.2\,s$	$78.5\,s$
60×60	$3038.5\,s$	$645.4\,s$	$486.3\,s$	$214.2\,s$	$160.2\,s$
70×70	—	$1195.6\,s$	$893.5\,s$	$384.9\,s$	$299.4\,s$

Figure 3 Benchmarks for different maze sizes and different type environments

Finite Combinatory Logic with Predicates (see slides part 2)

Ongoing and future work with CLSP

- Further integration with
  - constraint solving
  - optimizers
  - ML

- Simulation-based machine learning
- Controlled experiments and statistics
- Categorical framework for synthesizing learners

- Further industrial applications

Finite Combinatory Logic with Predicates

Jakob Rehof
Slides by A. Dudenhefner

TU Dortmund University, Dortmund, Germany

SummerSoc 2024
Crete

Intersection Types with Covariant Constructors and Literals

Definition

$$\sigma, \tau \quad ::= \quad \omega \mid \sigma \to \tau \mid \sigma \cap \tau \mid c(\sigma) \mid l$$

σ, τ	:	intersection type
c	:	unary type constructor
l	:	literal

Example

$(x(3) \cap y(1)) \to (x(4) \cap y(1))$	move right on a grid
$\mathrm{pos}((3, 1)) \to \mathrm{pos}((4, 1))$	any (python) object as literal
$(\mathrm{True} \to \mathrm{False}) \cap (\mathrm{False} \to \mathrm{True})$	finite function type

Parameterized Types

Definition

$$\varphi, \psi \quad ::= \quad \sigma \mid \langle \alpha : t \rangle \Rightarrow \varphi \mid \langle\langle x : \sigma \rangle\rangle \Rightarrow \varphi \mid P \Rightarrow \varphi$$

φ, ψ	:	parameterized type
σ	:	intersection type
α	:	literal variable
t	:	collection identifier
x	:	term variable
P	:	decidable predicate

Example (move right on a grid)

$$\langle \alpha : \mathbb{N}^2 \rangle \Rightarrow \langle \beta : \mathbb{N}^2 \rangle \Rightarrow \big(\beta = (\pi_1(\alpha) + 1, \pi_2(\alpha))\big) \Rightarrow \mathsf{pos}(\alpha) \to \mathsf{pos}(\beta)$$

Example (count motors)

$$\langle \alpha : \mathbb{N} \rangle \Rightarrow \langle\langle x : \mathsf{Part} \rangle\rangle \Rightarrow \big(\alpha = \mathsf{numMotors}(x)\big) \Rightarrow \mathsf{Assembly}(\alpha)$$

Combinatory Terms and Arguments

Definition

$$\begin{aligned} M, N &::= A \mid M\ T \\ T &::= M \mid l \end{aligned}$$

M, N	:	combinatory term
A	:	combinator
T	:	combinatory argument
l	:	literal

Example (move right, then down on a grid)

$$\text{Down}\ (1,0)\ (1,1)\ \big(\text{Right}\ (0,0)\ (1,0)\ \text{Start}\big)$$

Environments

Definition (Type Environment)

$$\Gamma \quad ::= \quad \{A_1 : \varphi_1, \ldots, A_n : \varphi_n\}$$

Γ	:	type environment
A_i	:	combinator
φ_i	:	parameterized type

Definition (Literal Environment)

$$\Delta \quad ::= \quad \{l_1 : t_1, \ldots, l_n : t_n\}$$

Δ	:	literal environment
l_i	:	literal
t_i	:	collection identifier

Example (Maze)

$$\Gamma = \{\text{Right} : \langle\alpha : \mathbb{N}^2\rangle \Rightarrow \langle\beta : \mathbb{N}^2\rangle \Rightarrow \text{isRight}(\beta, \alpha) \Rightarrow \text{pos}(\alpha) \to \text{pos}(\beta),$$
$$\text{Left} : \ldots, \text{Down} : \ldots, \text{Up} : \ldots, \text{Start} : \text{pos}((0,0))\}$$

$$\Delta = \{(x, y) : \mathbb{N}^2 \mid \text{isFree}((x, y)) \text{ where } x, y \in \{0, \ldots, 10\}\}$$

Finite Combinatory Logic with Predicates (FCLP)

Definition

FCL

$$\frac{(A : \varphi) \in \Gamma}{\Gamma; \Delta \vdash A : \varphi} \text{(Var)}$$

$$\frac{\Gamma; \Delta \vdash M : \sigma \qquad \sigma \leq \tau}{\Gamma; \Delta \vdash M : \tau} (\leq)$$

$$\frac{\Gamma; \Delta \vdash M : \sigma \to \tau \qquad \Gamma; \Delta \vdash N : \sigma}{\Gamma; \Delta \vdash M\, N : \tau} (\to\text{E})$$

extra

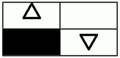
$$\frac{\Gamma; \Delta \vdash M : P \Rightarrow \varphi \qquad P \text{ holds}}{\Gamma; \Delta \vdash M : \varphi} (P\text{E})$$

$$\frac{\Gamma; \Delta \vdash M : \langle \alpha : t \rangle \Rightarrow \varphi \qquad (I : t) \in \Delta}{\Gamma; \Delta \vdash M\, I : \varphi[\alpha := I]} (\langle\rangle\text{E})$$

$$\frac{\Gamma; \Delta \vdash M : \langle\langle x : \sigma \rangle\rangle \Rightarrow \varphi \qquad \Gamma; \Delta \vdash N : \sigma}{\Gamma; \Delta \vdash M\, N : \varphi[x := N]} (\langle\langle\rangle\rangle\text{E})$$

Maze

Example



$$\mathbf{\Gamma} = \{\text{Right} : \langle \alpha : \mathbb{N}^2 \rangle \Rightarrow \langle \beta : \mathbb{N}^2 \rangle \Rightarrow \text{isRight}(\beta, \alpha) \Rightarrow \text{pos}(\alpha) \rightarrow \text{pos}(\beta),$$
$$\text{Left} : \ldots, \text{Down} : \ldots, \text{Up} : \ldots, \text{Start} : \text{pos}((0,0))\}$$

$$\mathbf{\Delta} = \{(0,0) : \mathbb{N}^2, (1,0) : \mathbb{N}^2, (1,1) : \mathbb{N}^2\}$$
$$\text{isRight}((1,0),(0,0)) = \text{True}, \ldots$$

$$\mathbf{\Gamma}, \mathbf{\Delta} \vdash \mathbf{?} : \text{pos}((1,1))$$
$$\downdownarrows$$

- $\text{Down}\ (1,0)\ (1,1)\ \big(\text{Right}\ (0,0)\ (1,0)\ \text{Start}\big)$
- $\text{Down} \ldots \Big(\text{Right} \ldots \big(\text{Left} \ldots (\text{Right} \ldots \text{Start})\big)\Big)$
- $\ldots$

Properties

Lemma

The following rule is derivable $\dfrac{\Gamma; \Delta \vdash M : \sigma \qquad \Gamma; \Delta \vdash M : \tau}{\Gamma; \Delta \vdash M : \sigma \cap \tau}$ $(\cap\mathrm{I})$

Lemma

The following is equivalent

- $\{A_1 : \sigma_1, \ldots, A_n : \sigma_n\}; \varnothing \vdash M : \tau$
- $\{A_1 : \sigma_1, \ldots, A_n : \sigma_n\} \vdash M : \tau$ *in* $\mathrm{FCL}(\cap, \leq)$

Lemma

Intersection type checking "$\Gamma; \Delta \vdash M : \sigma ?$*" is decidable.*

Lemma

Intersection type inhabitation "$\Gamma; \Delta \vdash ? : \sigma$*" is semi-decidable.*

Decidable Fragment

Arity

Definition

$$
\begin{aligned}
\mathbf{ar}(\langle \alpha : t \rangle \Rightarrow \varphi) = \mathbf{ar}(\langle x : \sigma \rangle \Rightarrow \varphi) &= 1 + \mathbf{ar}(\varphi) \\
\mathbf{ar}(P \Rightarrow \varphi) &= \mathbf{ar}(\varphi) \\
\mathbf{ar}(\omega) = \mathbf{ar}(c(\sigma)) = \mathbf{ar}(l) = \mathbf{ar}(\alpha) &= 0 \\
\mathbf{ar}(\sigma \to \tau) &= 1 + \mathbf{ar}(\tau) \qquad \text{where } \tau \neq \omega \\
\mathbf{ar}(\sigma \cap \tau) &= \max\{\mathbf{ar}(\sigma), \mathbf{ar}(\tau)\}
\end{aligned}
$$

Lemma (Maximal Arity)

If $(A : \varphi) \in \Gamma$, $n > \mathbf{ar}(\varphi)$, *and* $\Gamma; \Delta \vdash A\, T_1 \ldots T_n : \tau$, *then* $\tau = \omega$.

$\rightsquigarrow$ maximal arity of typed combinator $(A : \varphi)$ is $\mathbf{ar}(\varphi)$

Example (move right on a grid)

$$\mathbf{ar}\left(\langle \alpha : \mathbb{N}^2 \rangle \Rightarrow \langle \beta : \mathbb{N}^2 \rangle \Rightarrow \left(\beta = \left(\pi_1(\alpha) + 1, \pi_2(\alpha)\right)\right) \Rightarrow \mathrm{pos}(\alpha) \to \mathrm{pos}(\beta)\right) = 3$$

Literal and Term Constraints

Definition (Term Constraint)

A *term constraint* is either $x = M$ or $x \neq M$.

Example (Term Constraint)

$$x \neq S\,(S\,y)$$

Definition (Literal Constraint)

A *literal constraint* is a predicate P referencing only literal variables.

Example (Literal Constraint)

$$\beta = \left(\pi_1(\alpha) + 1, \pi_2(\alpha)\right)$$

Inhabitation with Literal and Term Constraints

Problem (Inhabitation with Literal and Term Constraints)

Given

- *type environment* $\boldsymbol{\Gamma}$ *using literal and term constraints only*
- *literal environment* $\boldsymbol{\Delta}$
- *intersection type* τ

is there a combinatory term M *such that*

- M *respects arities in* $\boldsymbol{\Gamma}$
- $\boldsymbol{\Gamma}; \boldsymbol{\Delta} \vdash M : \tau$ *holds?*

Theorem

Intersection type inhabitation with literal and term constraints is decidable.

Proof.

Reduce inhabitation to decidable emptiness of bottom-up tree automata with term constraints [Reuß and Seidl 2010]. □

- Maze uses literal (and term) constraints only

FCL vs. FCLP Workflows

FCLP isn't just an update for CLS, it's faster, sleeker, snazzier – a whole new level! (ChatGPT on FCLP)

FCL Workflow

1. Compute "dynamic" environment $\boldsymbol{\Gamma}$
2. Compute goal type $\boldsymbol{\tau}$
3. Ask $\boldsymbol{\Gamma} \vdash \mathbf{?} : \boldsymbol{\tau}$

$\rightsquigarrow$ Tree grammar $\mathcal{G}$

4. Enumerate terms of $\mathcal{G}$: $\boldsymbol{M_1}, \boldsymbol{M_2}, \ldots$
5. Filter enumerated terms
6. Interpret filtered terms: $[\![\boldsymbol{M_{i_1}}]\!], [\![\boldsymbol{M_{i_2}}]\!], \ldots$

FCLP Workflow

1. Fix *parametric* environment $\boldsymbol{\Gamma}$
2. Compute literal environment $\boldsymbol{\Delta}$
3. Compute goal type $\boldsymbol{\tau}$
4. Ask $\boldsymbol{\Gamma};\boldsymbol{\Delta} \vdash \boldsymbol{?} : \boldsymbol{\tau}$

$\rightsquigarrow$ Logic program $\boldsymbol{\mathcal{H}}$ (list of Horn clauses)

5. Enumerate terms in the model of $\boldsymbol{\mathcal{H}}$: $\boldsymbol{M_1}, \boldsymbol{M_2}, \ldots$
6. Interpret terms: $[\![\boldsymbol{M_1}]\!], [\![\boldsymbol{M_2}]\!], \ldots$

Benefits

- Parametric environment is fixed (5 combinators for Maze)
- Literal predicates hold in $\boldsymbol{\mathcal{H}}$ (no generate-and-test)
- Non-literal predicates are evaluated in the model construction of $\boldsymbol{\mathcal{H}}$

Horn Clauses

Example (count motors)

$$A : \langle \alpha : \mathbb{N} \rangle \Rightarrow \langle\langle x : \text{Part} \rangle\rangle \Rightarrow \big(\alpha = \text{numMotors}(x)\big) \Rightarrow \text{Assembly}(\alpha)$$

$$\Downarrow$$

$$Q_{\text{Assembly}(0)}(A\,0\,x) \leftarrow Q_{\text{Part}}(x), (0 = \text{numMotors}(x))$$
$$Q_{\text{Assembly}(1)}(A\,1\,x) \leftarrow Q_{\text{Part}}(x), (1 = \text{numMotors}(x))$$
$$Q_{\text{Assembly}(2)}(A\,2\,x) \leftarrow Q_{\text{Part}}(x), (2 = \text{numMotors}(x))$$
$$\ldots$$

Specification Comparison

FCL stands trembling as FCLP rockets in, a supersonic revolution rewriting the rules! (Google Bard on FCLP)

Parameterized vs. Dynamic Specification

Example (Parameterized Specification)

$$\mathsf{Right} : \langle \alpha : \mathbb{N}^2 \rangle \Rightarrow \langle \beta : \mathbb{N}^2 \rangle \Rightarrow \mathsf{isRight}(\beta, \alpha) \Rightarrow \mathsf{pos}(\alpha) \to \mathsf{pos}(\beta)$$

- fixed
- small specification $\rightsquigarrow$ fast subtyping $\rightsquigarrow$ fast synthesis
- inhabitants contain parameters: Right $(0, 0)$ $(1, 0)$ $(\ldots)$

Example (Dynamic Specification)

$$\mathsf{Right}_{\mathsf{FCL}} : \bigcap_{x=0}^{n-1} \bigcap_{y=0}^{n} \big(\mathsf{pos}((x, y)) \to \mathsf{pos}((x+1, y))\big)$$

- depends on n
- formed by computation
- large specification $\rightsquigarrow$ slow subtyping $\rightsquigarrow$ slow synthesis
- instance-oblivious inhabitants: $\mathsf{Right}_{\mathsf{FCL}}\ \big(\mathsf{Down}_{\mathsf{FCL}}\ (\ldots)\big)$

Parameters vs. Combinators vs. Substitutions

Example (Parameter Propagation)

$$A : \langle \alpha : \mathbb{N}^2 \rangle \Rightarrow a(\alpha)$$

$$B : \langle \alpha : \mathbb{N}^2 \rangle \Rightarrow b(\alpha)$$

$$F : \langle \alpha : \mathbb{N}^2 \rangle \Rightarrow \left(\pi_1(\alpha) + \pi_2(\alpha) = 2\right) \Rightarrow a(\alpha) \to b(\alpha) \to f$$

$$\Gamma ; \Delta \vdash ? : f$$

$$\Downarrow$$

$$F\,(0,2)\,\big(A\,(0,2)\big)\,\big(B\,(0,2)\big)$$
$$F\,(1,1)\,\big(A\,(1,1)\big)\,\big(B\,(1,1)\big)$$
$$F\,(2,0)\,\big(A\,(2,0)\big)\,\big(B\,(2,0)\big)$$

- Δ can become large
- terms contain parameter information
- configuration can change subterms
- configuration can be changed in subterms
- configuration predicates part of specification

Parameters vs. Combinators vs. Substitutions

Example (Configuration Combinator)

$$A : \text{config} \to a$$
$$B : \text{config} \to b$$
$$C : \text{config}$$
$$F : \text{config} \to a \to b \to f$$

$$\Gamma ; \emptyset \vdash ? : f$$
$$\Downarrow$$
$$F\ C\ (A\ C)\ (B\ C)$$

- interpretation of C instantiates configuration
- configuration can not change subterms
- configuration can not be changed in subterms
- configuration predicates not part of specification

Parameters vs. Combinators vs. Substitutions

Example (Application in Context)

$$\boldsymbol{A} : \boldsymbol{a}$$
$$\boldsymbol{B} : \boldsymbol{b}$$
$$\boldsymbol{F} : \boldsymbol{a} \to \boldsymbol{b} \to \boldsymbol{f}$$

$$\Gamma; \varnothing \vdash \,?\, : \boldsymbol{f}$$
$$\downdownarrows$$
$$\boldsymbol{F}\ \boldsymbol{A}\ \boldsymbol{B}$$

$$[\![\boldsymbol{A}]\!] = \lambda \text{config}.\textbf{doA}\ \text{config}$$
$$[\![\boldsymbol{B}]\!] = \lambda \text{config}.\textbf{doB}\ \text{config}$$
$$[\![\boldsymbol{F}]\!] = \lambda \text{config}\ \boldsymbol{a}\ \boldsymbol{b}.\textbf{doF}\ \text{config}\ (\boldsymbol{a}\ \text{config})\ (\boldsymbol{b}\ \text{config})$$
$$[\![\boldsymbol{F}\ \boldsymbol{A}\ \boldsymbol{B}]\!] =_{\beta} \lambda \text{config}.\textbf{doF}\ \text{config}\ (\textbf{doA}\ \text{config})\ (\textbf{doB}\ \text{config})$$

- interpretation as function from config
- configuration not part of specification
- configuration does not appear in terms
- combinator interpretation is “eierlegende Wollmilchsau”

Restriction Comparison

FCLP empowers the CLS framework to operate at unprecedented levels of efficiency, scalability, and precision. (Llama on FCLP)

Generate-and-Test vs. Invariant

Example (Filter Looping Maze Solutions)

$\text{Right} : \langle \alpha : \mathbb{N}^2 \rangle \Rightarrow \langle \beta : \mathbb{N}^2 \rangle \Rightarrow \text{isRight}(\beta, \alpha) \Rightarrow \text{pos}(\alpha) \to \text{pos}(\beta)$,

$\text{Left} : \ldots, \text{Down} : \ldots, \text{Up} : \ldots,$

$\text{Start} : \text{pos}((0,0))$

$$\Gamma; \{(0,0), \ldots, (10,10)\} \vdash ? : \text{pos}((10,10))$$

$$\Downarrow$$

candidates

$$\Downarrow$$

filter looping solutions

$$\Downarrow$$

filtered results

- many unwanted candidates
- arbitrary time between results
- non-termination: infinitely many candidates / finitely many results

Generate-and-Test vs. Invariant

Example (Avoid Looping Maze Solutions)

$$\begin{aligned}\mathsf{Right} &: \langle \alpha : \mathbb{N}^2 \rangle \Rightarrow \langle \beta : \mathbb{N}^2 \rangle \Rightarrow \langle\!\langle x : \mathsf{pos}(\alpha) \rangle\!\rangle \Rightarrow \\ &\quad \big(\mathsf{isRight}(\beta, \alpha) \wedge \mathsf{notIn}(\beta, x)\big) \Rightarrow \mathsf{pos}(\beta), \\ \mathsf{Left} &: \ldots, \mathsf{Down} : \ldots, \mathsf{Up} : \ldots, \\ \mathsf{Start} &: \mathsf{pos}((0,0))\end{aligned}$$

$$\Gamma; \{(0,0), \ldots, (10,10)\} \vdash ? : \mathsf{pos}((10,10))$$

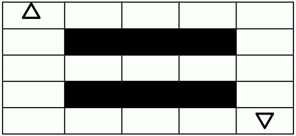
$$\Downarrow\!\Downarrow$$

loop-free results

- no unwanted candidates
- improved performance
- termination for finite result set

Generate-and-Test vs. Invariant

Example (Loop-free Maze Solutions)



	FCL	FCLP
time	20s	0.5s
behavior	non-termination	termination

Imagine a maze. FCL bumbles around like a lost puppy, while FCLP zooms through like a laser beam. (Claude on FCLP)

Bibliography I

Reuß, Andreas and Helmut Seidl (2010). "Bottom-Up Tree Automata with Term Constraints." In: *Logic for Programming, Artificial Intelligence, and Reasoning - 17th International Conference, LPAR-17, Yogyakarta, Indonesia, October 10-15, 2010. Proceedings*. Ed. by Christian G. Fermüller and Andrei Voronkov. Vol. 6397. Lecture Notes in Computer Science. Springer, pp. 581–593. DOI: 10.1007/978-3-642-16242-8_41. URL: https://doi.org/10.1007/978-3-642-16242-8_41.