technische universität
dortmund

How to Automatically Compose Larger Systems from Repositories of Components

Applications of Combinatory Logic Synthesis to industrial systems

Jakob Rehof

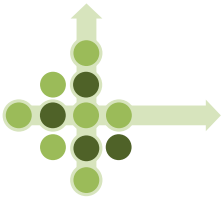
Chair of Software Engineering, TU Dortmund University

Director Research Strategy, Fraunhofer-ISST Dortmund

SummerSoc 2022

16th Symposium and Summer School on Service-Oriented Computing

Hersonissos, Crete, July 7 2022



(CL)S

Combinatory Logic Synthesis

What is CLS?

CLS-Framework

Applications

Program synthesis

- A classical challenge in computer science since Church‘s Problem (1956)
- Classical problem understood as synthesis „from scratch“
  - Given a logical specification, to construct a system satisfying it
- Classical problem faces inherent obstacles in the form of
  - Computational complexity (super-) exponential
  - Specification complexity
- CLS is *component-based* (not from-scratch)
- CLS is *type-based* (not based on complete specifications)

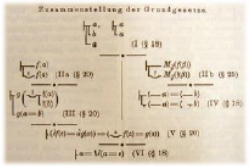
CLS: A type-based approach to component-oriented synthesis

Component-oriented Synthesis

Synthesis *relative* to *library* (repository) of components

Combinatory Logic Synthesis (CLS)

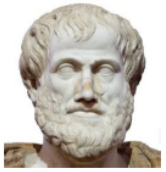
Libraries need *classification systems* to enable *retrieval and composition*



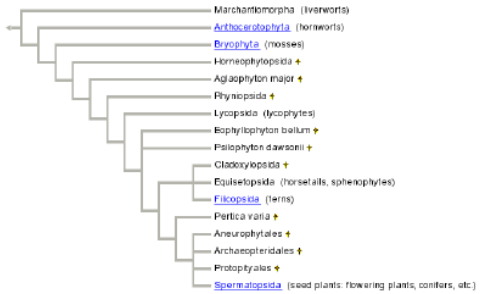
$$(\forall\, x; \forall\, y; \forall\, z; (p(x,y) \wedge p(y,z) \rightarrow p(x,z)) \wedge \\ \forall\, x; \forall\, y; (p(x,y) \rightarrow \neg p(y,x)) \wedge \\ \exists\, x; (p(a,x) \wedge p(x,b)) \\ ) \rightarrow \neg p(b,a)$$

Bottom-up specification
Hoare logic

CLS



Classification
Taxonomy
...
Types



J Rehof, MY Vardi:

Design and synthesis from components (Dagstuhl seminar 14232, 2014)

Combinatory logic

David Hilbert

1862-1943

Moses Schönfinkel

1889-1942

Haskell Curry

1900-1982

Über die Bausteine der mathematischen Logik.

Von

M. Schönfinkel in Moskau[1]).

---

§ 1.

Es entspricht dem Wesen der axiomatischen Methode, wie sie heute vor allem durch die Arbeiten Hilberts zur Anerkennung gelangt ist, daß man nicht allein hinsichtlich der Zahl und des Gehalts der *Axiome* nach möglichster Beschränkung strebt, sondern auch die Anzahl der als undefiniert zugrunde zu legenden *Begriffe* so klein wie möglich zu machen sucht, indem man nach Begriffen fahndet, die vorzugsweise geeignet sind, um aus ihnen alle anderen Begriffe des fraglichen Wissenszweiges aufzubauen. Begreiflicherweise wird man sich im Sinne dieser Aufgabe bezüglich des Verlangens nach Einfachheit der an den Anfang zu stellenden Begriffe entsprechend bescheiden müssen.

Bekanntlich lassen sich die grundlegenden *Aussagenverknüpfungen* der mathematischen Logik, die ich hier in der von Hilbert in seinen Vorlesungen verwendeten Bezeichnungsweise wiedergebe:

---

[1]) Die folgenden Gedanken wurden vom Verfasser am 7. Dez. 1920 vor der Mathematischen Gesellschaft in Göttingen vorgetragen. Ihre formale und stilistische Durcharbeitung für diese Veröffentlichung wurde von H. Behmann in Göttingen übernommen.

Combinators

$$
\begin{aligned}
\mathbf{S}xyz &= (xz)(yz) \\
\mathbf{K}xy &= x \\
\mathbf{I}x &= x
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{S} &: (\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma)) \\
\mathbf{K} &: \alpha \to (\beta \to \alpha) \\
\mathbf{I} &: \alpha \to \alpha
\end{aligned}
$$

SKI-calculus is Turing-complete

Hindley, J. R.; Seldin, J. P. (2008). *λ-calculus and Combinators: An Introduction*. Cambridge University Press.

Barendregt, H. P. (1984). *The Lambda Calculus, Its Syntax and Semantics*. Studies in Logic and the Foundations of Mathematics. Vol. 103. North Holland.

Combinatory logic (simple types)

$F, G ::= X \mid (F\ G)$ combinatory expressions

$\sigma, \tau ::= \alpha \mid \sigma \to \tau$ types

Typing rules

$$\frac{}{\Gamma, (X : \tau) \vdash X : S(\tau)}\,(\text{comb})$$

$$\frac{\Gamma \vdash F : \sigma \to \tau \quad \Gamma \vdash G : \sigma}{\Gamma \vdash (F\ G) : \tau}\,(\text{app})$$

Synthesis as type inhabitation in combinatory logic

Given Γ and σ:

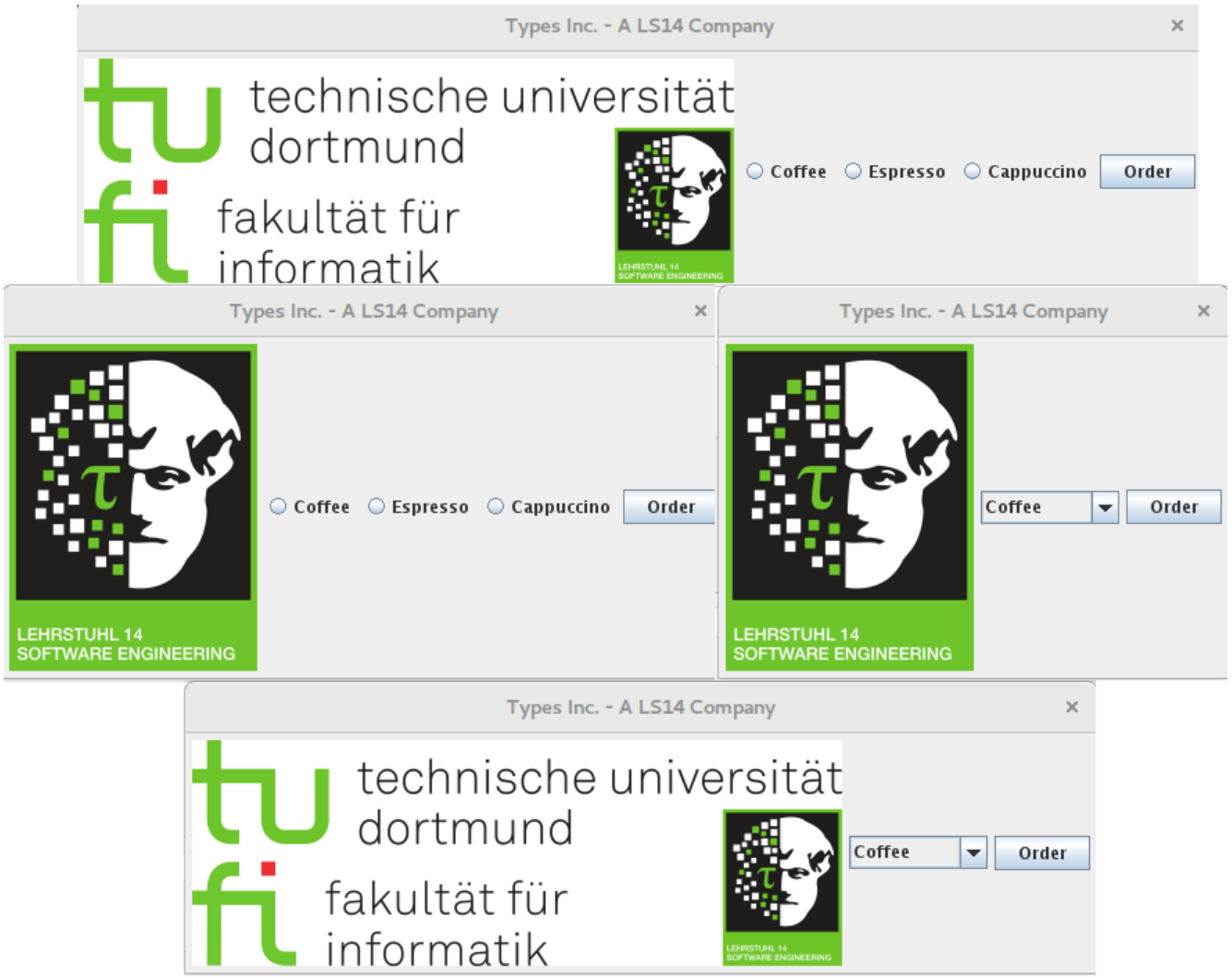
- does there exist a combinatory term F such that $\Gamma \vdash F : \sigma$?

Synthesis is type inhabitation in combinatory logic

$$\Gamma \vdash F : \sigma$$

- Repository of component interfaces $X : \tau$
- Synthesized (meta-)program
- Synthesis goal

Example: Webpage synthesis



Native repository (native types)

$$
\begin{aligned}
\Gamma = \{ \quad & customerForm : (\texttt{String} \rightarrow \texttt{java.net.URL} \rightarrow \texttt{OptionSelection} \rightarrow \texttt{Form}) \\
& dropDownSelector : (\texttt{java.net.URL} \rightarrow \texttt{OptionSelection}) \\
& radioButtonSelector : (\texttt{java.net.URL} \rightarrow \texttt{OptionSelection}) \\
& companyTitle : \texttt{String} \\
& databaseLocation : \texttt{java.net.URL} \\
& logoLocation : \texttt{java.net.URL} \\
& alternateLogoLocation : \texttt{java.net.URL} \quad \}
\end{aligned}
$$

Semantic repository (semantic types)

Intersection type

$$
\begin{aligned}
\Gamma = \{ \quad & customerForm : (\texttt{String} \to \texttt{java.net.URL} \to \texttt{OptionSelection} \to \texttt{Form}) \cap \\
& (Title \to Location(Logo) \to ChoiceDialog(\alpha) \to OrderMenu(\alpha)) \\
& dropDownSelector : (\texttt{java.net.URL} \to \texttt{OptionSelection}) \cap \\
& (Location(Database) \to ChoiceDialog(DropDown)) \\
& radioButtonSelector : (\texttt{java.net.URL} \to \texttt{OptionSelection}) \cap \\
& (Location(Database) \to ChoiceDialog(RadioButtons)) \\
& companyTitle : \texttt{String} \cap Title \\
& databaseLocation : \texttt{java.net.URL} \cap Location(Database) \\
& logoLocation : \texttt{java.net.URL} \cap Location(Logo) \\
& alternateLogoLocation : \texttt{java.net.URL} \cap Location(Logo) \quad \}
\end{aligned}
$$

$$\Gamma = \{ \quad customerForm : (\texttt{String} \to \texttt{java.net.URL} \to \texttt{OptionSelection} \to \texttt{Form})$$
$$dropDownSelector : (\texttt{java.net.URL} \to \texttt{OptionSelection})$$
$$radioButtonSelector : (\texttt{java.net.URL} \to \texttt{OptionSelection})$$
$$companyTitle : \texttt{String}$$
$$databaseLocation : \texttt{java.net.URL}$$
$$logoLocation : \texttt{java.net.URL}$$
$$alternateLogoLocation : \texttt{java.net.URL} \quad \}$$

$$\Gamma = \{ \quad customerForm : (\texttt{String} \to \texttt{java.net.URL} \to \texttt{OptionSelection} \to \texttt{CompilationUnit}) \cap (Title \to Location(Logo) \to ChoiceDialog(\alpha) \to OrderMenu(\alpha))$$
$$dropDownSelector : (\texttt{java.net.URL} \to \texttt{OptionSelection}) \cap (Location(Database) \to ChoiceDialog(DropDown))$$
$$radioButtonSelector : (\texttt{java.net.URL} \to \texttt{OptionSelection}) \cap (Location(Database) \to ChoiceDialog(RadioButtons))$$
$$companyTitle : \texttt{String} \cap Title$$
$$databaseLocation : \texttt{java.net.URL} \cap Location(Database)$$
$$logoLocation : \texttt{java.net.URL} \cap Location(Logo)$$
$$alternateLogoLocation : \texttt{java.net.URL} \cap Location(Logo) \quad \}$$

$$\Gamma \vdash ? : \texttt{CompilationUnit} \cap OrderMenu(\omega)$$

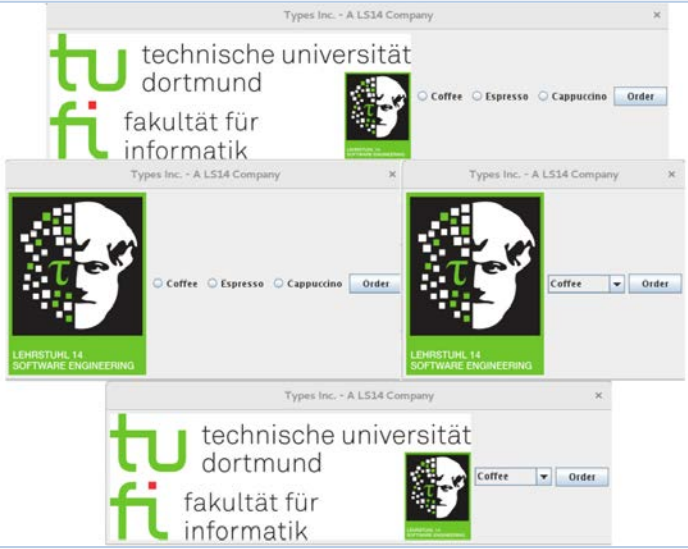
Inhabitation problem

$$\Gamma \vdash ? : \texttt{CompilationUnit} \cap OrderMenu(\omega)$$

Inhbaitants:

$$\{customerForm(companyTitle, logoLocation, radioButtonSelector(databaseLocation)),$$
$$customerForm(companyTitle, alternateLogoLocation, radioButtonSelector(databaseLocation)),$$
$$customerForm(companyTitle, logoLocation, dropDownSelector(databaseLocation)),$$
$$customerForm(companyTitle, alternateLogoLocation, dropDownSelector(databaseLocation))\}$$

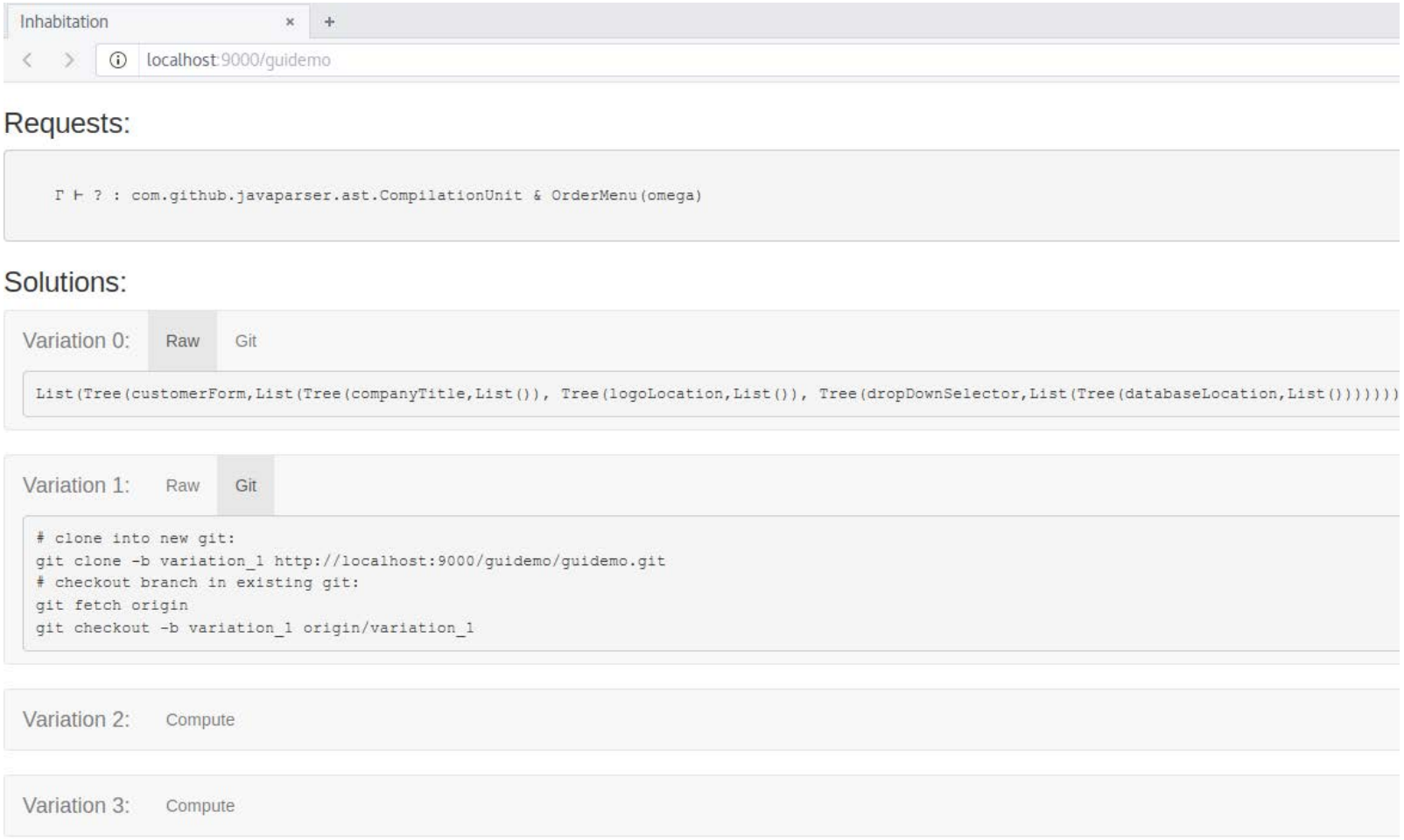
Interpreted as calls to combinators and resulting code is made available via branches in source code versioning system (git).



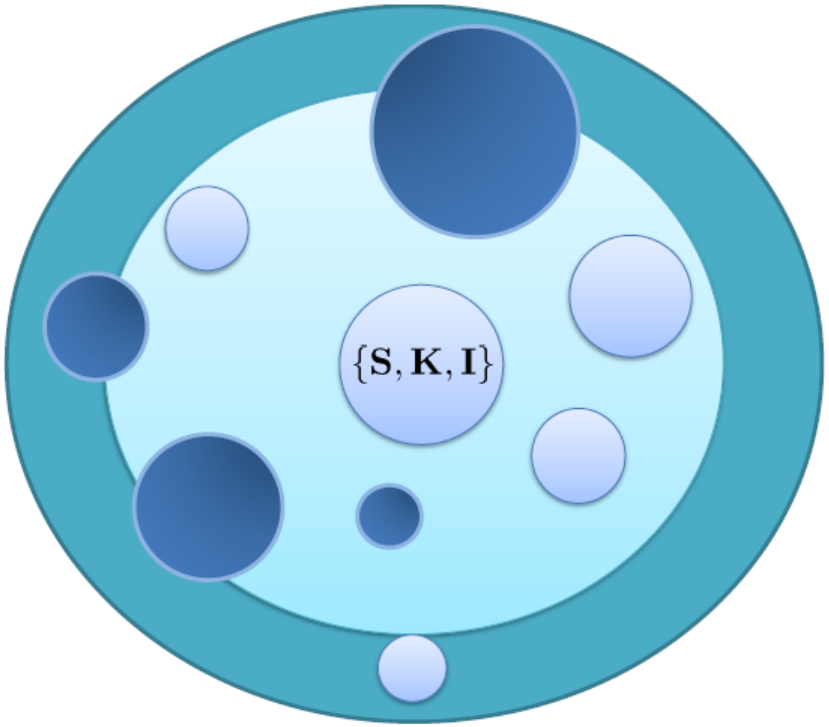
Scala combinator

```
@combinator object customerForm {
        def apply(title: String,
            logoLocation: URL,
            optionSelector: OptionSelection): Form = {
    val form = Java(readFile("CustomerForm.java")).compilationUnit()
    addOptionSelection(form, optionSelector)
    addTitle(form, title)
    addLogo(form, logoLocation)
    form
  }
  val semanticType: Type =
    'Title =>: 'Location('Logo) =>: 'ChoiceDialog(alpha) =>: 'OrderMenu(alpha)
}
```

Inhabitation request



CLS world view: a repository is a combinatory basis



- CL over *arbitrary bases*:

  General theory of component collections (repositories)

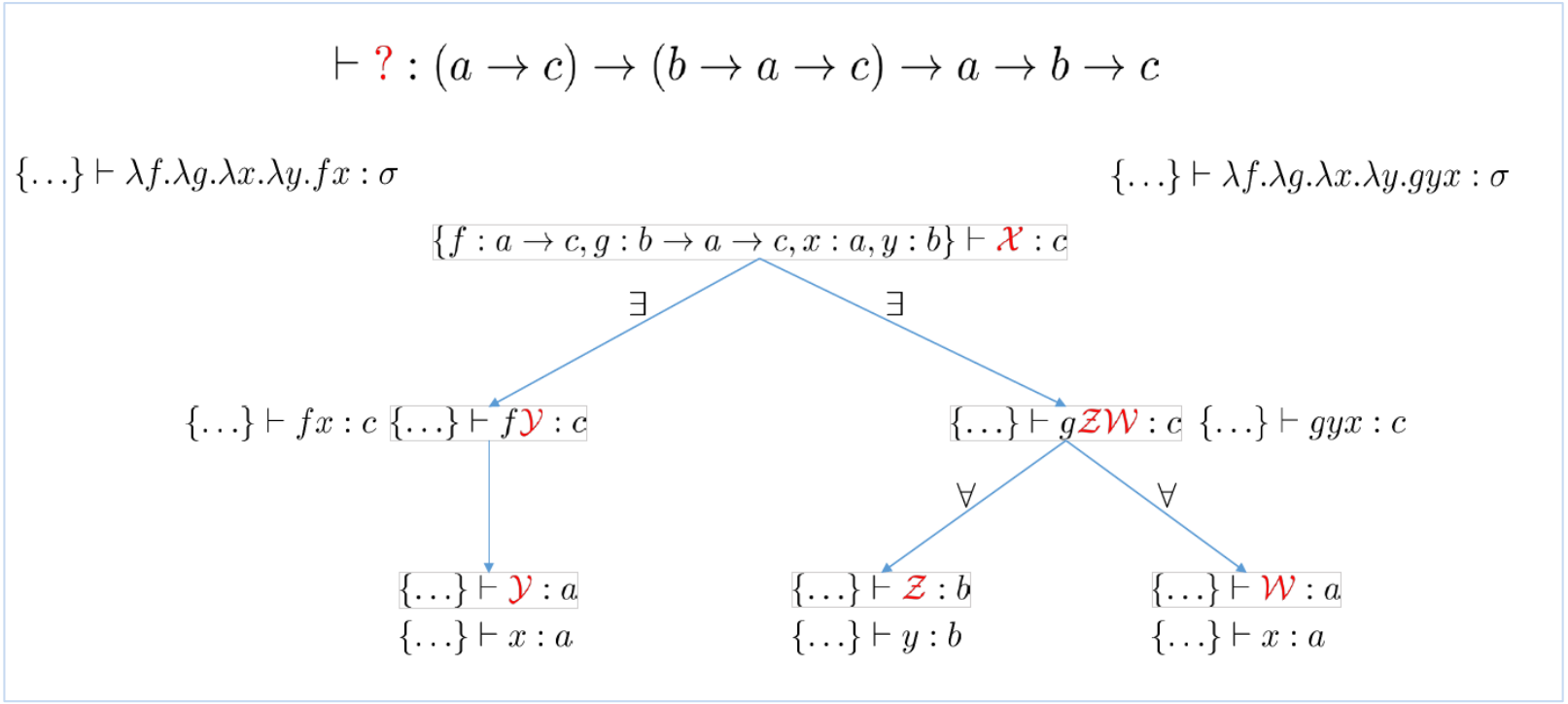
- The implementation theory" $\{\mathbf{S}, \mathbf{K}, \mathbf{I}\}$ is one, very special case

Synthesis is type inhabitation in combinatory logic

- We consider the *relativized inhabitation* problem:
  - **Given a set of typed combinators** Γ **and** τ, *does there exist combinatory expression e such that* $\Gamma \vdash e : \tau$*?*
- Inhabitation for fixed base $\{\mathbf{S}, \mathbf{K}, \mathbf{I}\}$ is PSPACE-complete in simple types (Statman's Theorem [Sta79])
- Relativized inhabitation is much harder
  - *Undecidable in simple types*: **Linial-Post theorems**, 1948ff. [LP49][1]
- *The CLS view:* Already in simple types, relativized inhabitation defines a Turing-complete logic programming language for component composition
  - Reduction from 2-counter automata [Reh13]
  - Similar idea used to prove undecidability for synthesis in ML relative to library of functions [BSWC16]

[1] See also A. Dudenhefner, JR: *Lower End of the Linial-Post Spectrum*, TYPES 2017

Theory and algorithms of inhabitation based on alternating Turing Machines

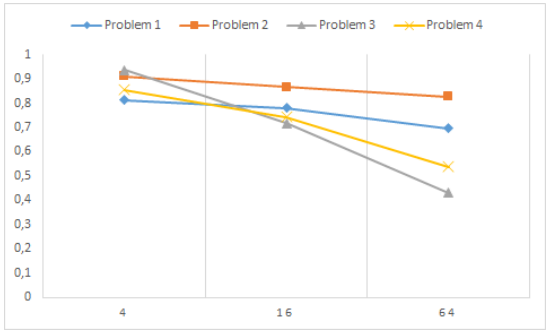


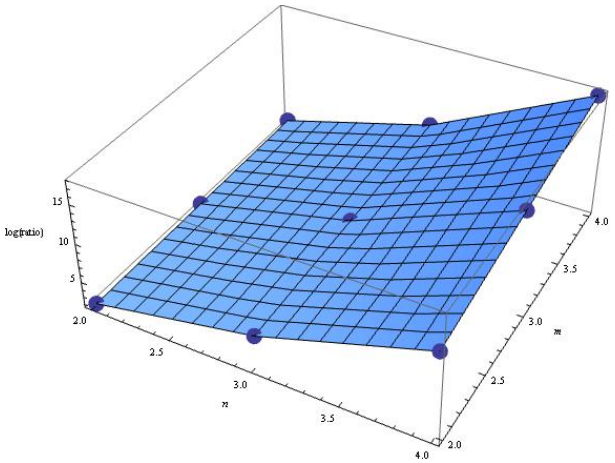
Computational complexity

System	Complexity	Authors
$\mathsf{FCL}(\leq)$	PTIME	RU, TLCA 2011
$\mathsf{FCL}(\cap, \leq)$	EXPTIME	RU, TLCA 2011
$\mathsf{BCL}_k(\leq)$	EXPTIME	DMRU, CSL 2012
$\mathsf{BCL}(\cap,\leq)$	(k+2)-EXPTIME	DMRU, CSL 2012
$\mathrm{CL}(\mathbf{SK}) \rightarrow$	PSPACE	S, 1979
$\lambda(- \cap I)$	EXPSPACE	RU, KF 2012
$\lambda^{r2}\cap$	EXPSPACE	U, TLCA 2009
$\lambda\cap$	∞	U, 1999
$\mathrm{CL}(\cap)$	∞	DH, 1992 + U, 1999

- CLS allows for „controlled combinatorial explosion“, since interface types can be used to program the composition (much like a logic program)

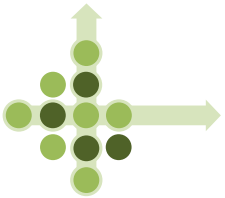
Algorithm engineering





Input : Γ, τ— all types in Γ and $\tau = \bigcap_{i \in I} \tau_i$ organized
loop :

1 CHOOSE $(x : \sigma) \in \Gamma$;
2 write $\sigma \equiv \bigcap_{j \in J} \sigma_j$
3 FOR EACH $i \in I, j \in J, m \leq \|\sigma\|$ DO
4 $\quad$ $\texttt{candidates}(i, j, m) := \texttt{Match}(tgt_m(\sigma_j) \leq \tau_i)$
5 $M := \{m \leq \|\sigma\| \mid \forall_{i \in I} \exists_{j \in J} : \texttt{candidates}(i, j, m) = \textbf{true}\}$
6 CHOOSE $m \in M$;
7 FOR EACH $i \in I$ DO
8 $\quad$ CHOOSE $j_i \in J$ with $\texttt{candidates}(i, j_i, m) = \textbf{true}$
9 $\quad$ CHOOSE S_i a substitution
10 $\quad$ CHOOSE $\pi_i \in \mathbb{P}_m(\overline{S_i(\sigma_{j_i})})$ with $tgt_m(\pi_i) \leq \tau_i$ and
11 $\quad$ $\forall 1 \leq l \leq m \; \forall \pi' \in \overline{arg_l(\pi_i)} \; \exists (y : \rho) \in \Gamma \; \exists$ a path ρ'
12 $\quad$ in $\rho \; \exists k \; : \; \texttt{Match}(tgt_k(\rho') \leq \pi') = \textbf{true}$
13 IF $(m = 0)$ THEN ACCEPT;
14 ELSE FORALL$(l = 1 \ldots m)$
15 $\quad\quad$ $\tau := \bigcap_{i \in I} arg_l(\pi_i)$;
16 $\quad\quad$ GOTO *loop*;

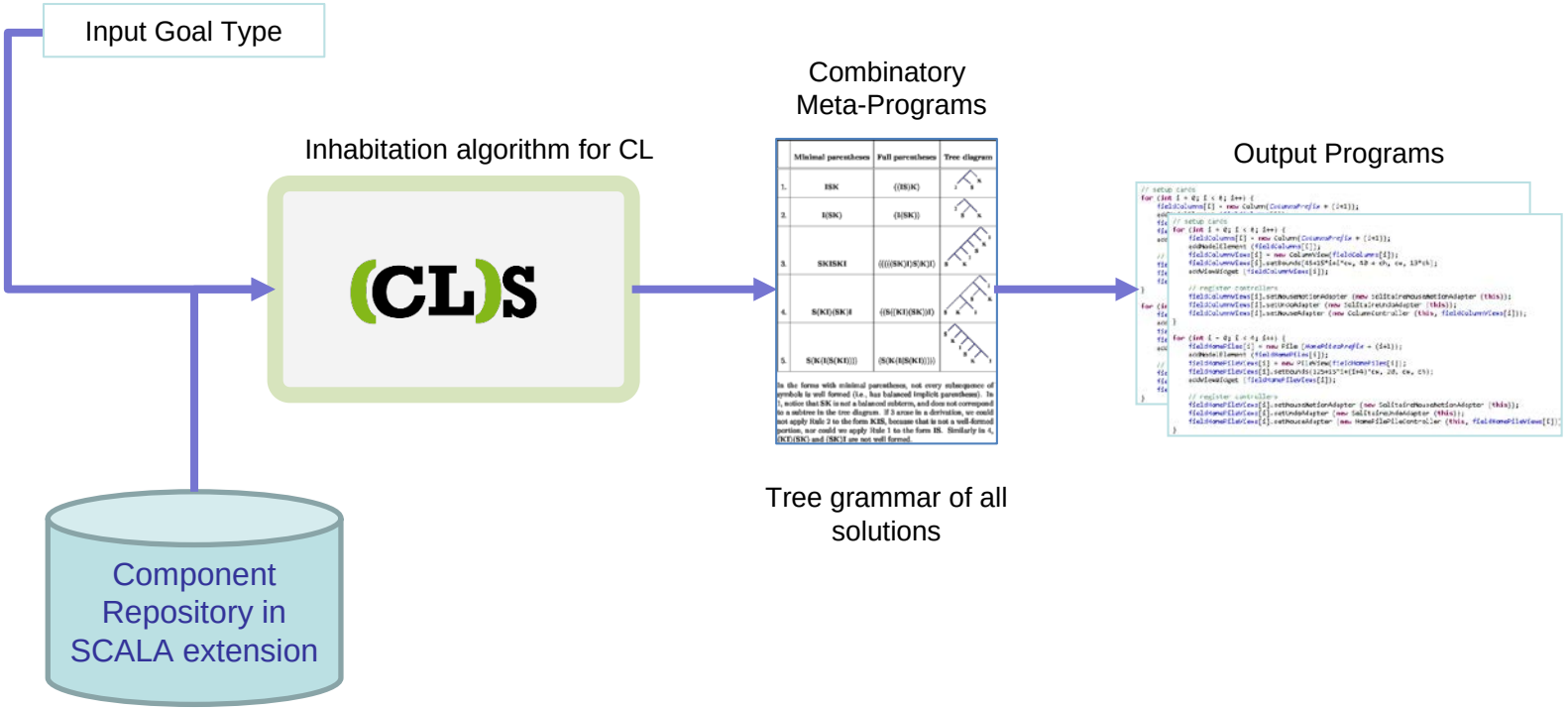


(CL)S

Combinatory Logic Synthesis

- What is CLS?
- CLS-Framework
- Applications

CLS-Framework

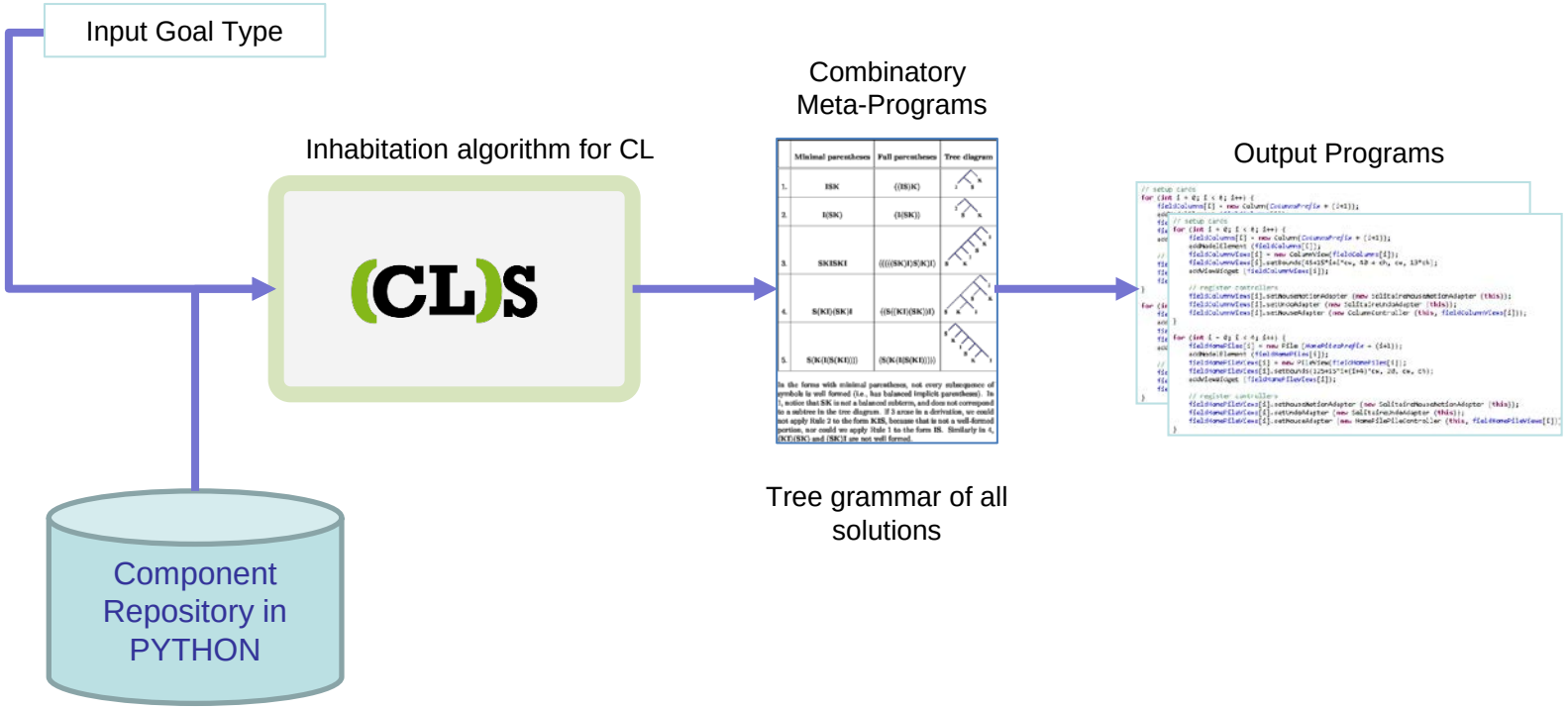


Current implementation (GitHub) www.combinators.org, https://github.com/combinators

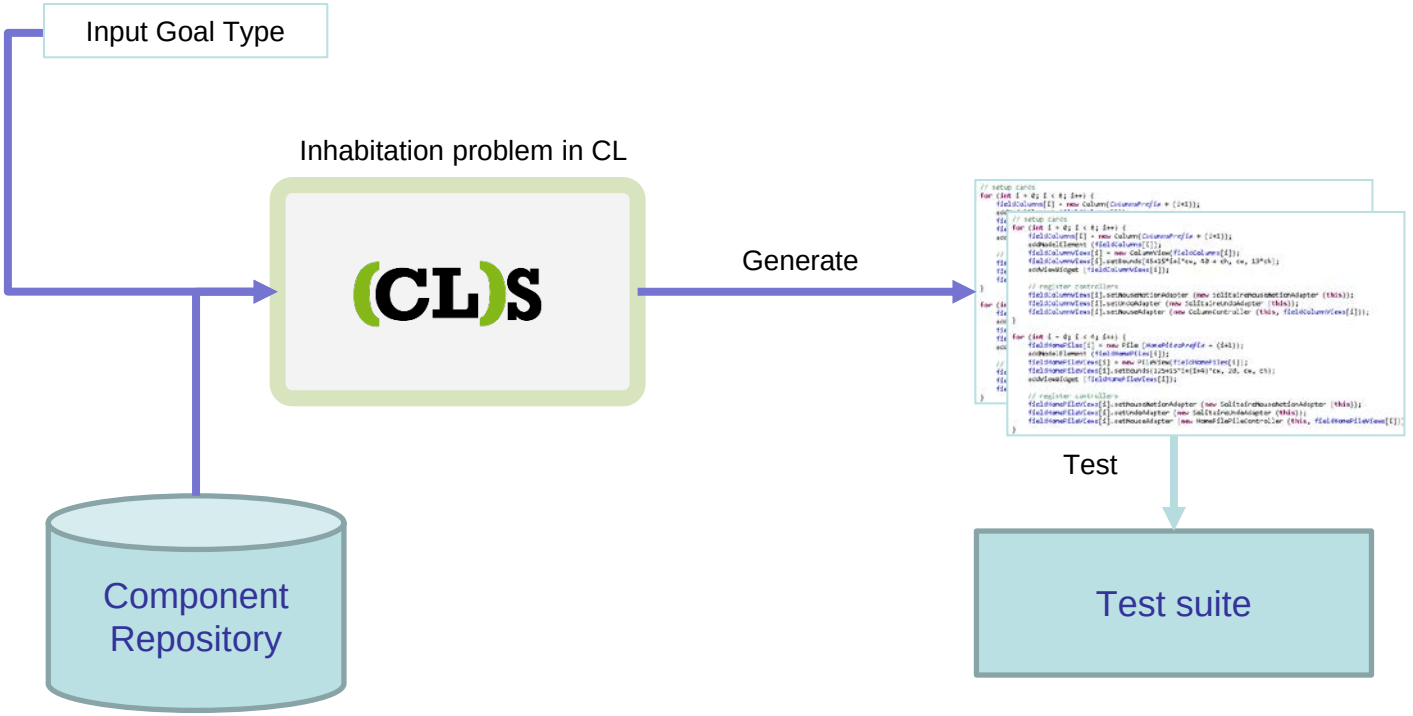
Jan Bessai: *A Type-Theoretic Framework for Software Component Synthesis*.

Dissertation TU Dortmund 2019.

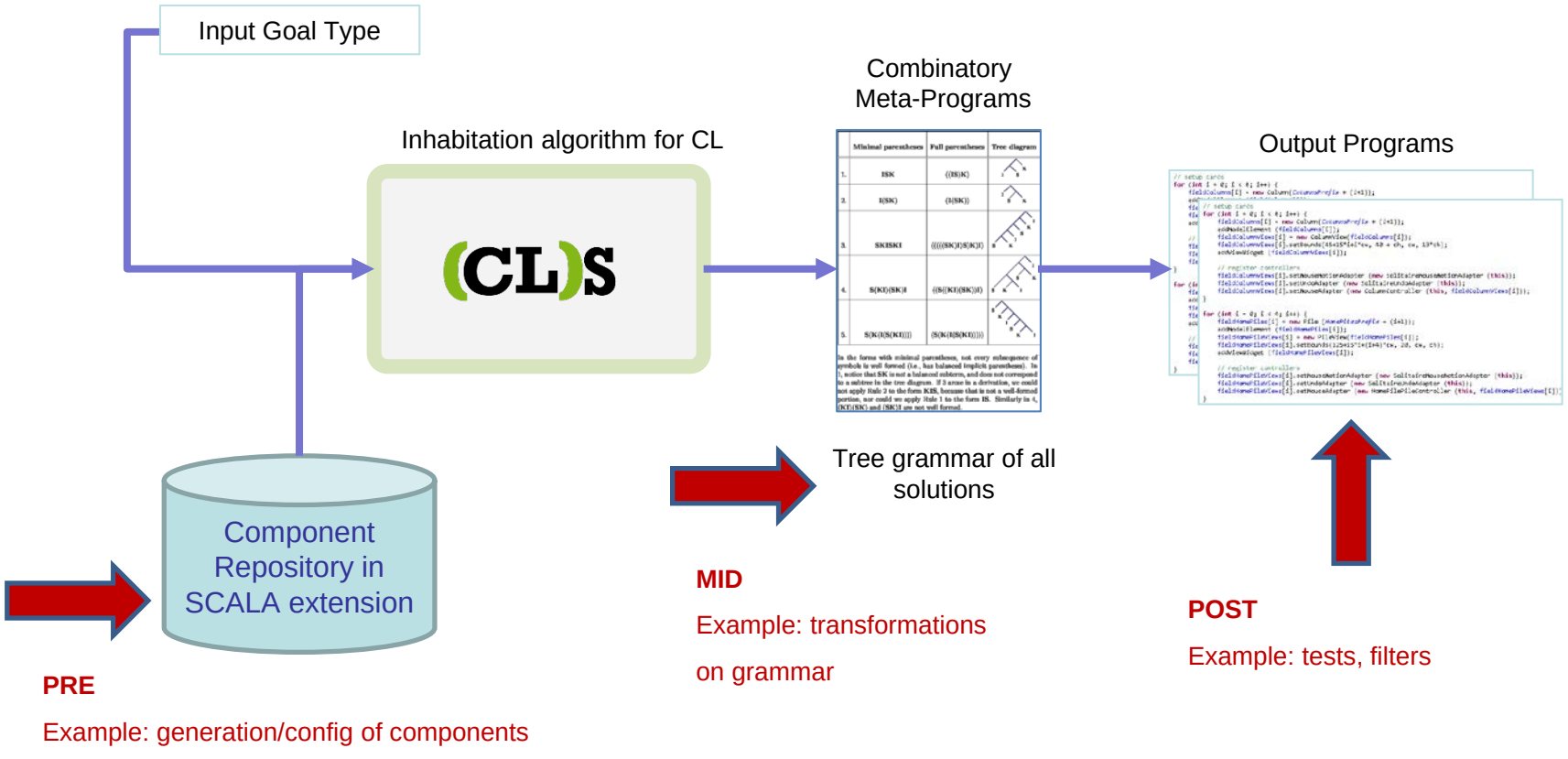
CLS-Framework



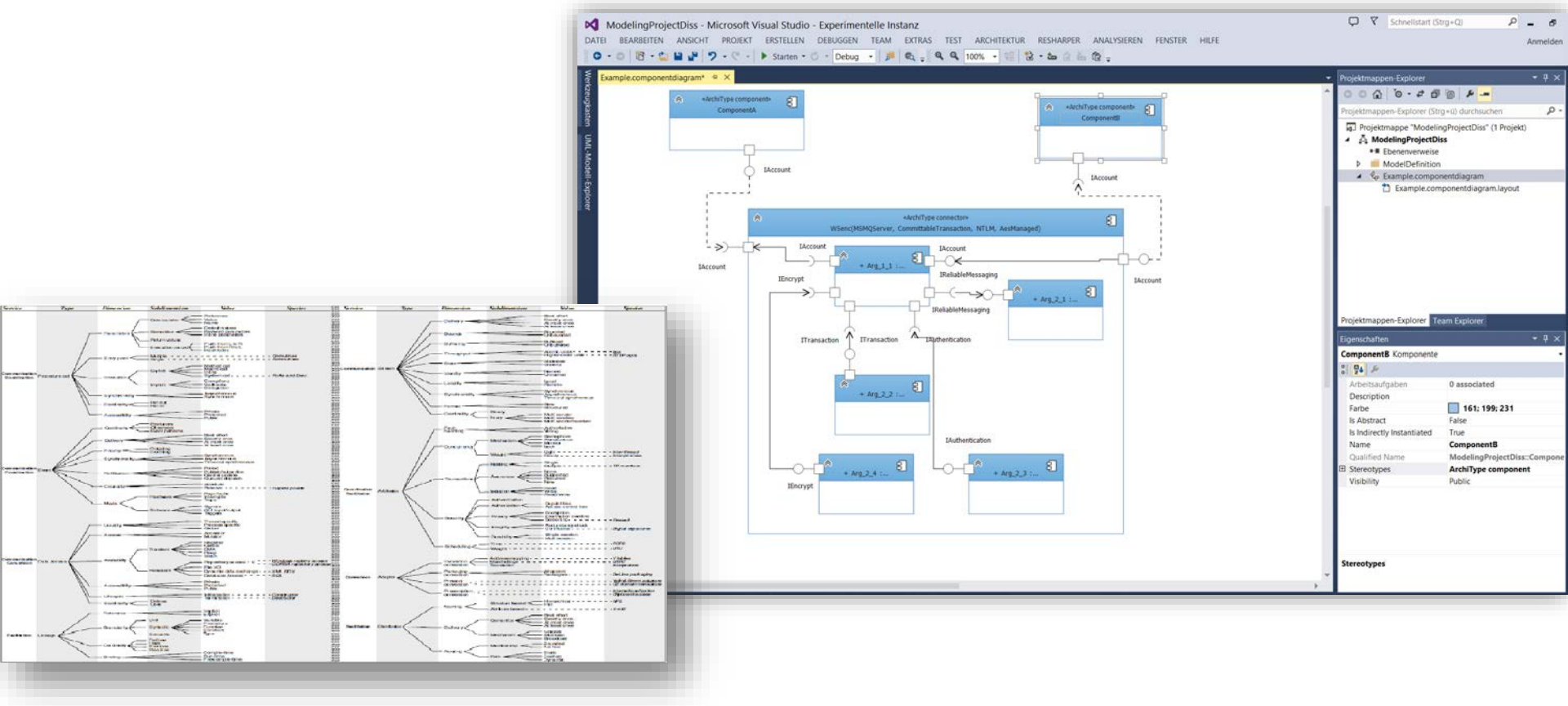
Generate-and-test



Extensions to CLS-framework



Example: Component & Connector Synthesis

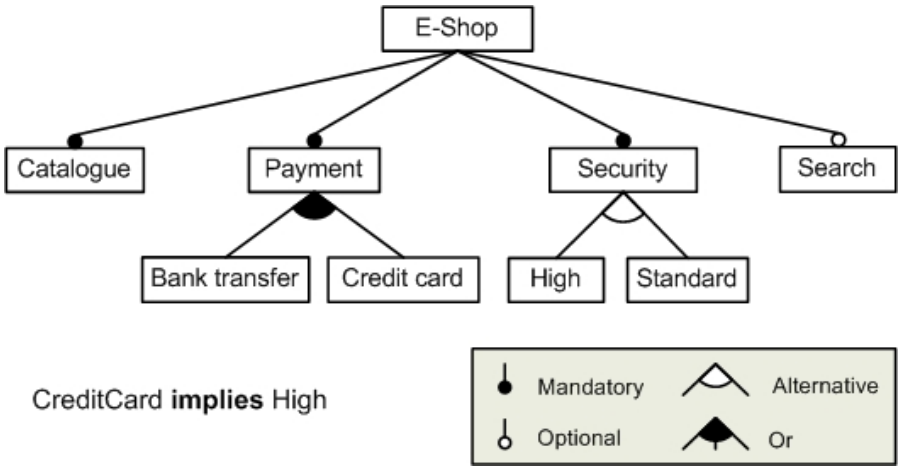


Boris Düdder: *Automatic Synthesis of Component & Connector-Software Architectures with Bounded Combinatory Logic.* Dissertation TU Dortmund, August 2014.

CLS strength: Automated software product lines

Software product lines (SPLs), or software product line development, refers to software engineering methods, tools and techniques for creating a collection of similar software systems from a shared set of software assets using a common means of production

https://en.wikipedia.org/wiki/Software_product_line

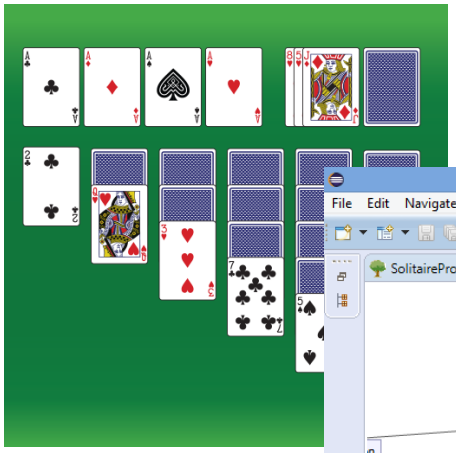


https://en.wikipedia.org/wiki/Feature_model

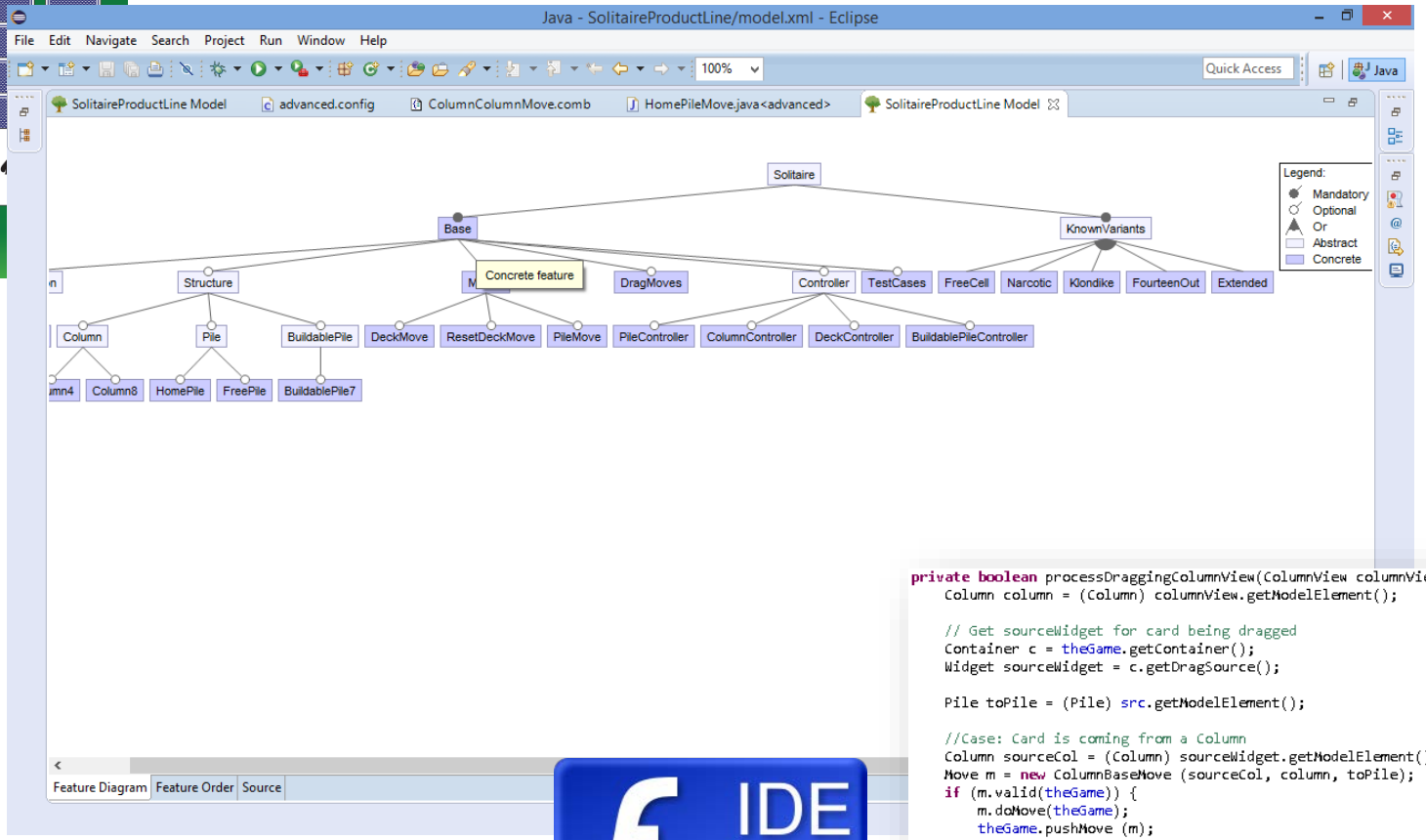
G Heineman, A Hoxha, B Düdder, J Rehof:

Towards migrating object-oriented frameworks to enable synthesis of product line members.

Proceedings of the 19th International Conference on Software Product Line, 56-60



Example: LaunchPad – Feature-oriented Programsynthesis by George T. Heineman using (CL)S



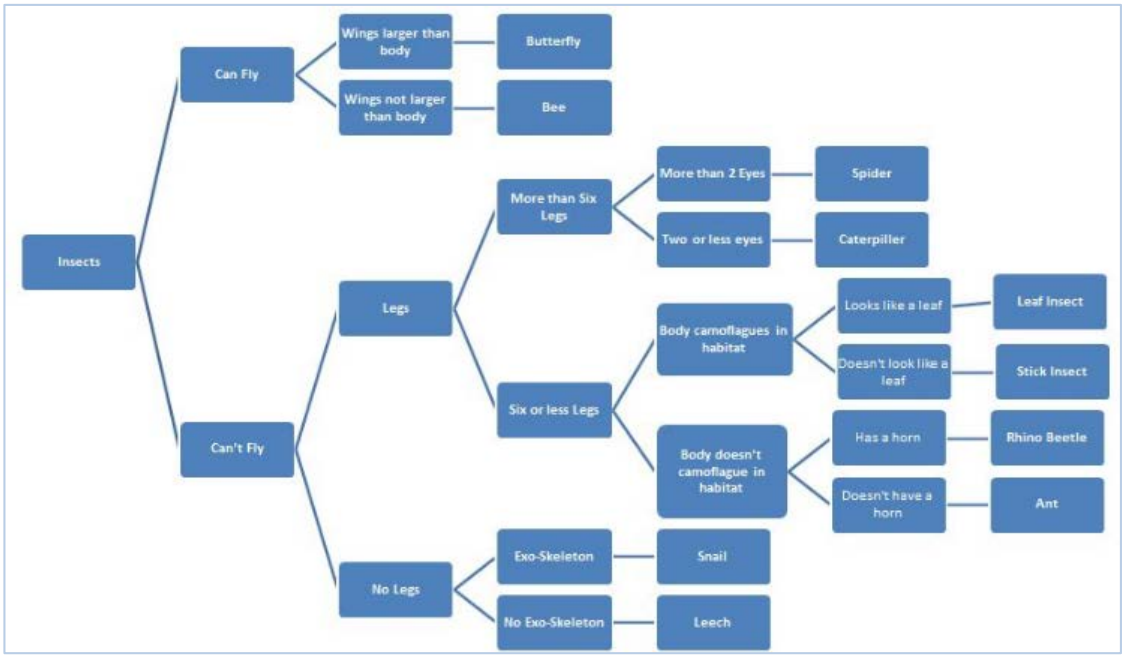

```
private boolean processDraggingColumnView(ColumnView columnView) {
    Column column = (Column) columnView.getModelElement();

    // Get sourceWidget for card being dragged
    Container c = theGame.getContainer();
    Widget sourceWidget = c.getDragSource();

    Pile toPile = (Pile) src.getModelElement();

    //Case: Card is coming from a Column
    Column sourceCol = (Column) sourceWidget.getModelElement();
    Move m = new ColumnBaseMove (sourceCol, column, toPile);
    if (m.valid(theGame)) {
        m.doMove(theGame);
        theGame.pushMove (m);
    } else {
        sourceWidget.returnWidget (columnView);
    }

    return true;
}
```

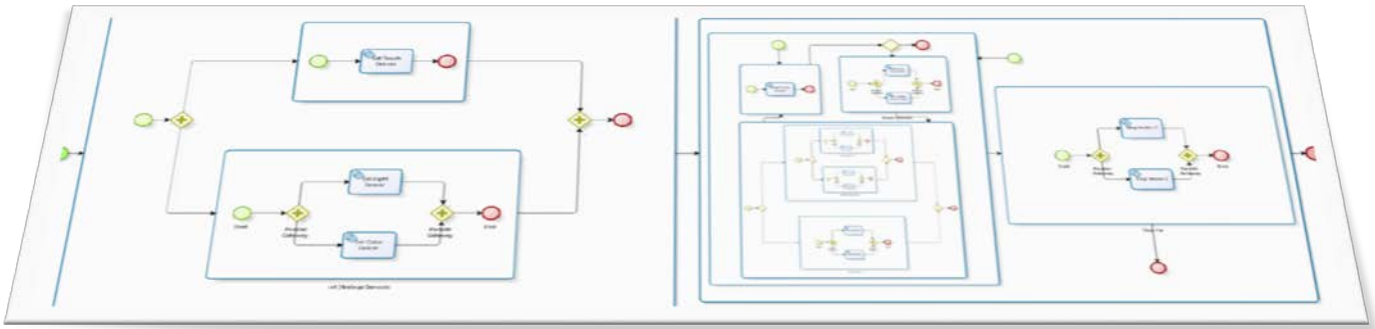
Example:

LEGO NXT robot control synthesis

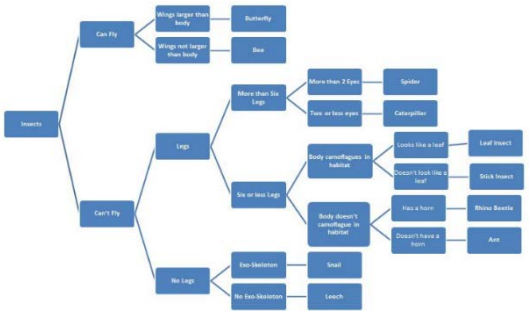
GOAL:

fstproc ∩ car ∩ followsLine ∩ twoLightSensors ∩ stopsOnTouch ∩ robotProgram





CLS pro‘s

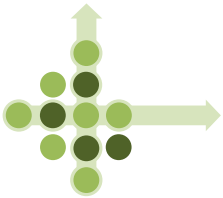


- **Specifications**
  - Simple and intuitive (type-based classification)
  - Integrated in code (component wrapping and types)
- **Automation**
  - Search for components
  - Formalized rules for use of components
  - Composition of components (program generation)
  - Logical dependencies expressible
  - Nondeterminism (multiple solutions)
- **Migration strategies** from code (system) to repository
  - Generalization of existing application to a product line
- **Flexibililty**
  - Extendability

    e.g. transformations, optimizers, constraint-solvers, ML-components
  - Language agnostic, code templating via metalanguage and staging

CLS con‘s

- **Construction of repository and type specifications**
  - Presupposes component structure in the application domain
  - Up-front cost (componentization, specification, wrapping),
    only amortizable with certain level of complexity of product line
  - Knowledge of CLS
  - CLS debugger still open to improvements



(CL)S

Combinatory Logic Synthesis

- What is CLS?
- CLS-Framework
- Applications

technische universität dortmund

GRK 2193 Anpassungsintelligenz von Fabriken im dynamischen und komplexen Umfeld

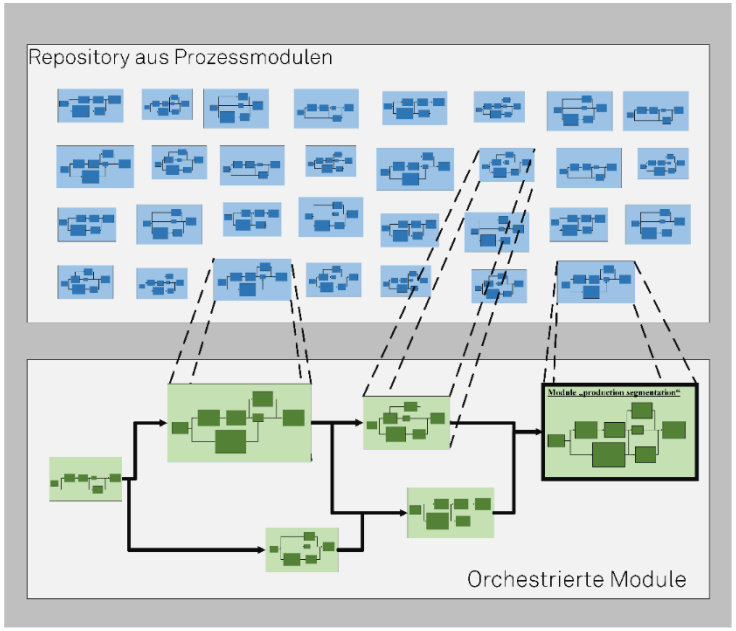
Adaption Intelligence of Factories

DFG Graduiertenkolleg 2016 - 2025

Gefördert durch
DFG Deutsche Forschungsgemeinschaft

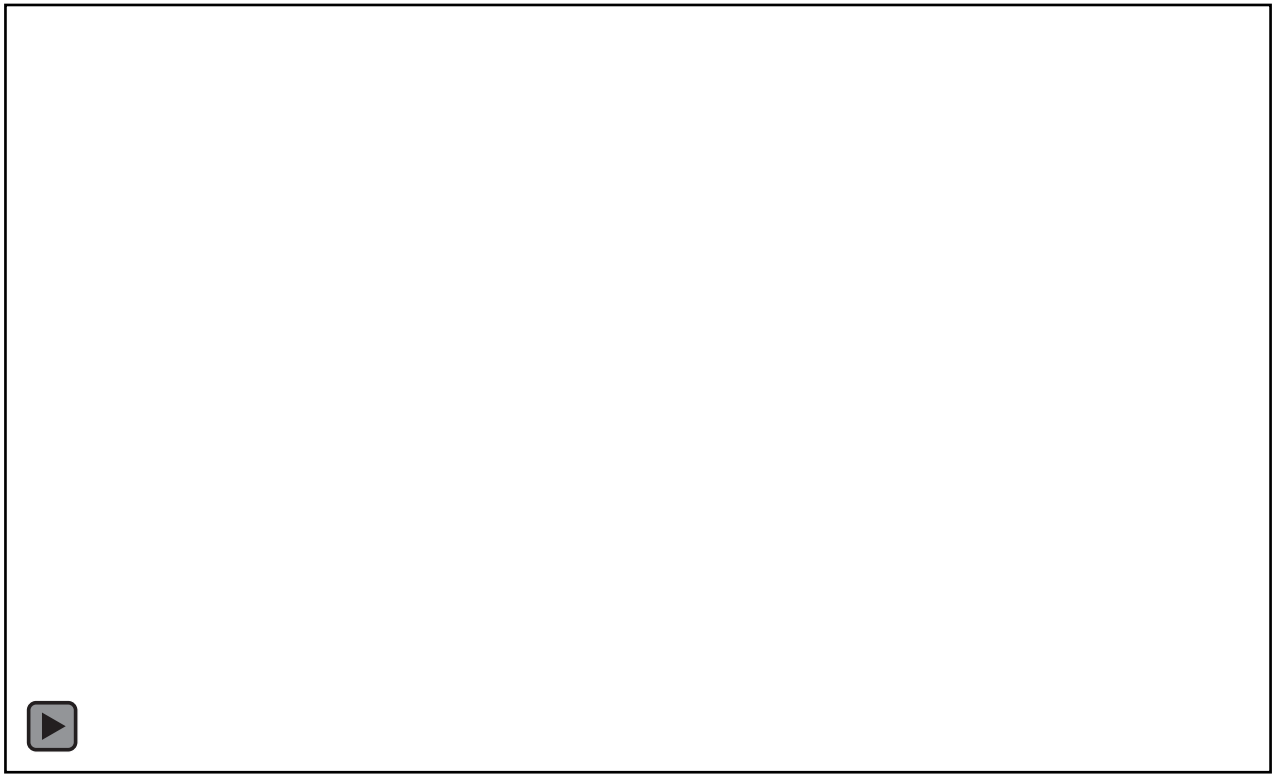
technische universität dortmund

PDM synthesis

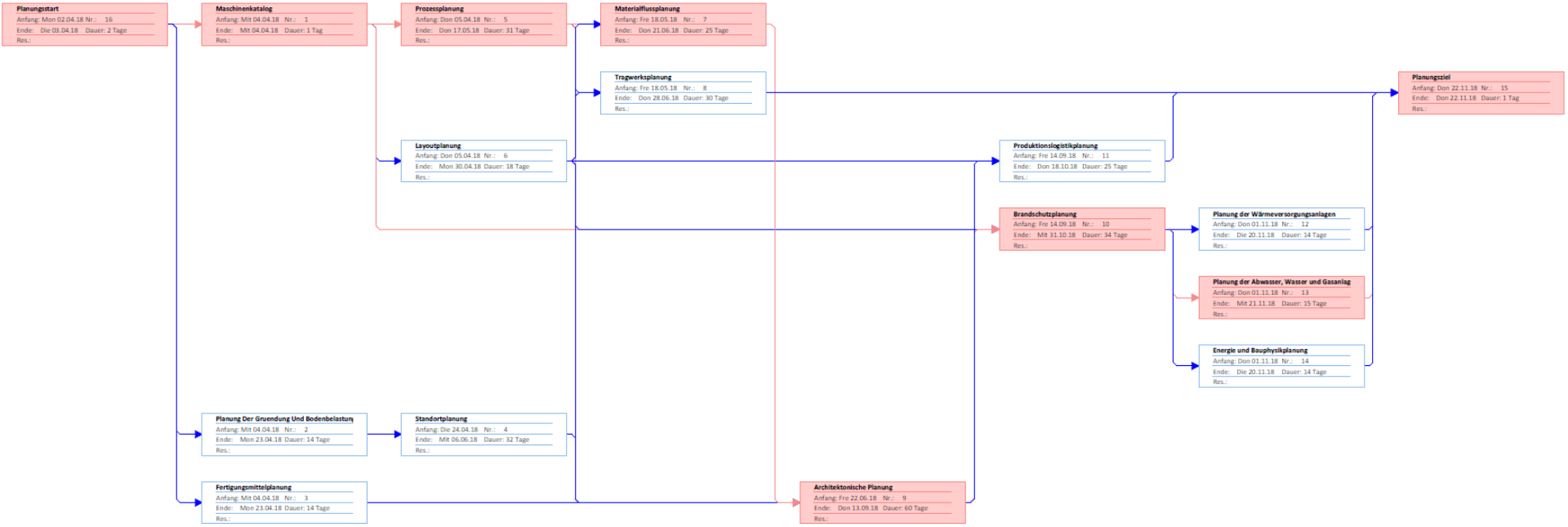


Jan Winkels: *Automatisierte Komposition und Konfiguration von Workflows zur Planung mittels kombinatorischer Logik*. Dissertation (GRK 2193) TU Dortmund 2019.

PDM synthesis



PDM synthesis



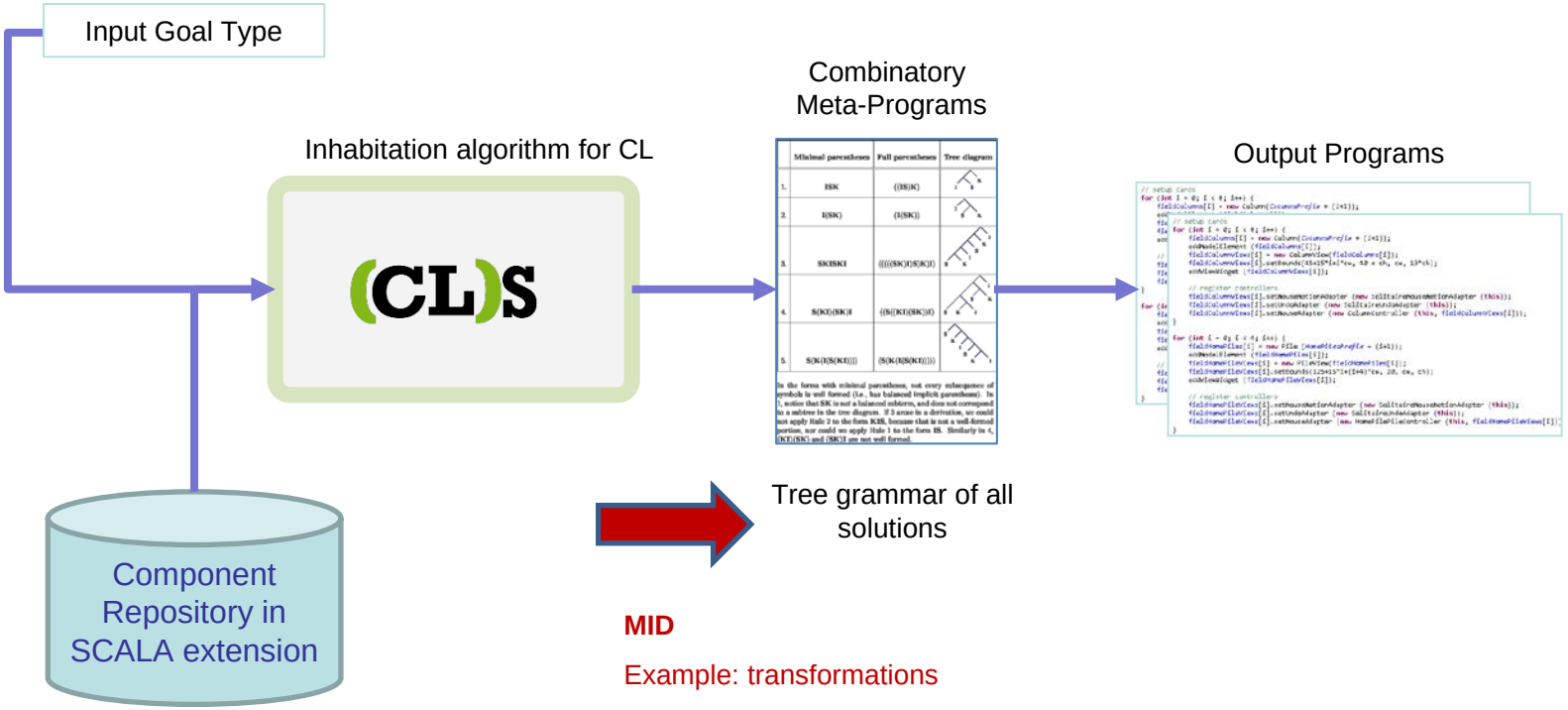
Generation of manufacturing simulation models

Using Component-based Software Synthesis and Constraint Solving to generate Sets of Manufacturing Simulation Models

Fadil Kallat*
Carina Mieth
Jakob Rehof
Anne Meyer

GRK 2193
technische universität dortmund
TRUMPF

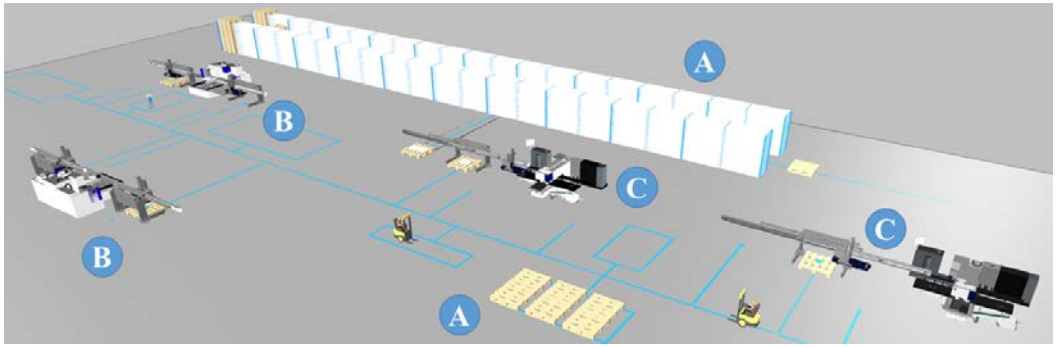
Extensions to CLS-framework



J Bessai, L Czajka, F Laarmann, J Rehof: *Restricting Tree Grammars with Term Rewriting.* FSCD 2022

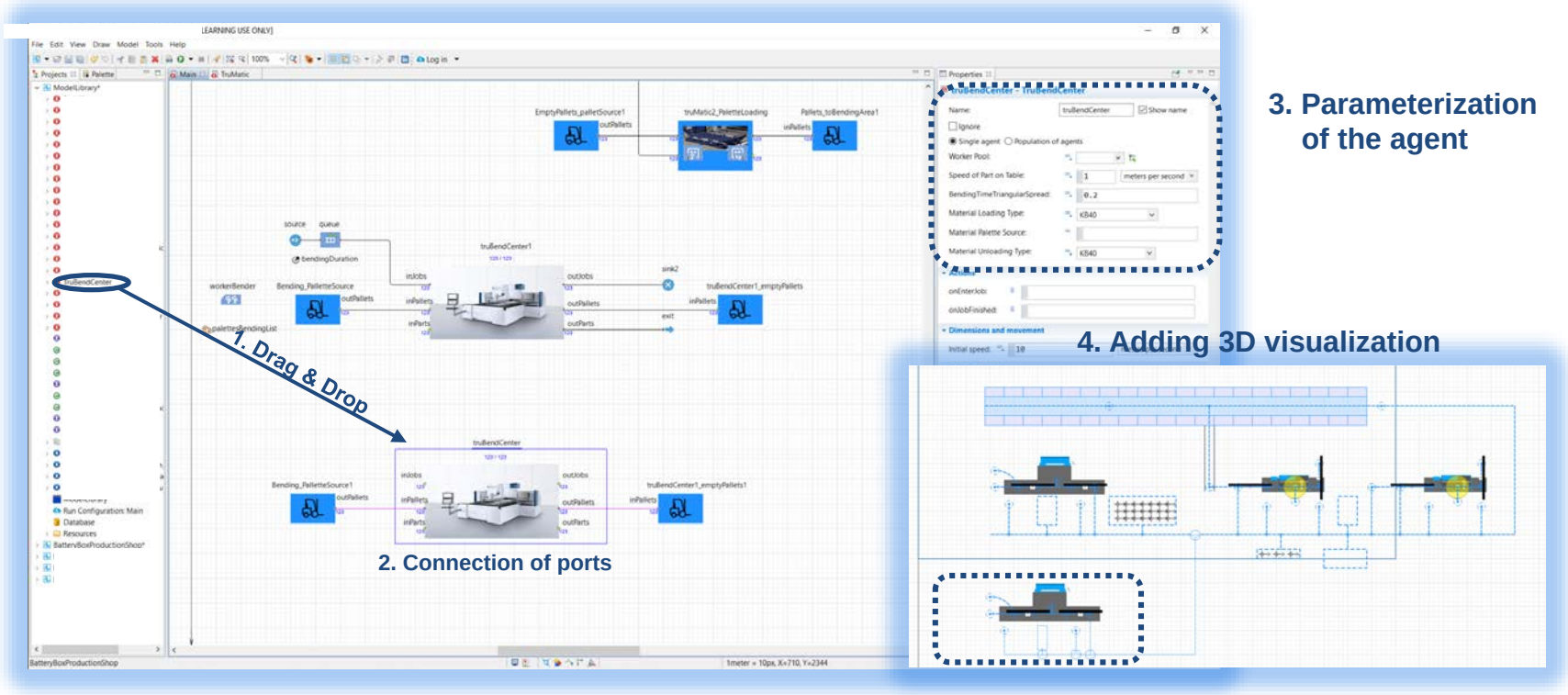
Industrial use case

- Real-world example of a sheet metal box production
- Implemented as discrete event simulation application using simulation environment AnyLogic 8
- It consists of
  A. An inbound warehouse and an intermediate storage for the sheets
  B. Two bending machines
  C. Two cutting machines



- The simulation model is composed manually from the TRUMPF model library

TRUMPF Model Library



Variability and approach

- Planning results in the choice of two cutting and two bending machines
- Cutting Machines
  - Differ in their cutting speed, assembly space and type of loading and unloading
  - ➢ **18** possible configurations
- Bending Machines
  - Differ in their Bending Speed and assembly space
  - ➢ **6** possible configurations
- Continuous parameters are grouped into intervals to avoid uncountable number of possible combinations
- In total, 18 * 18 * 6 * 6 = **11664** different system variants

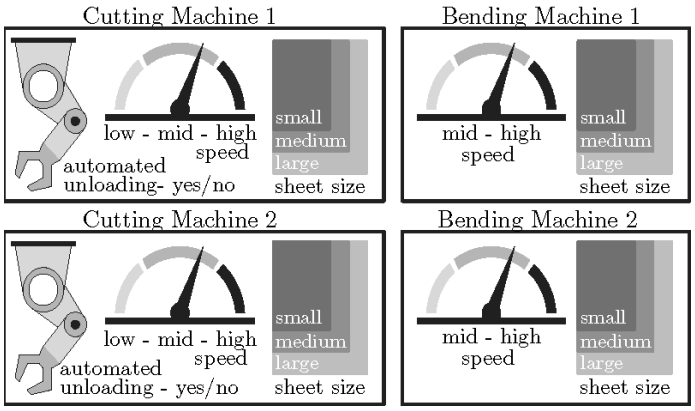
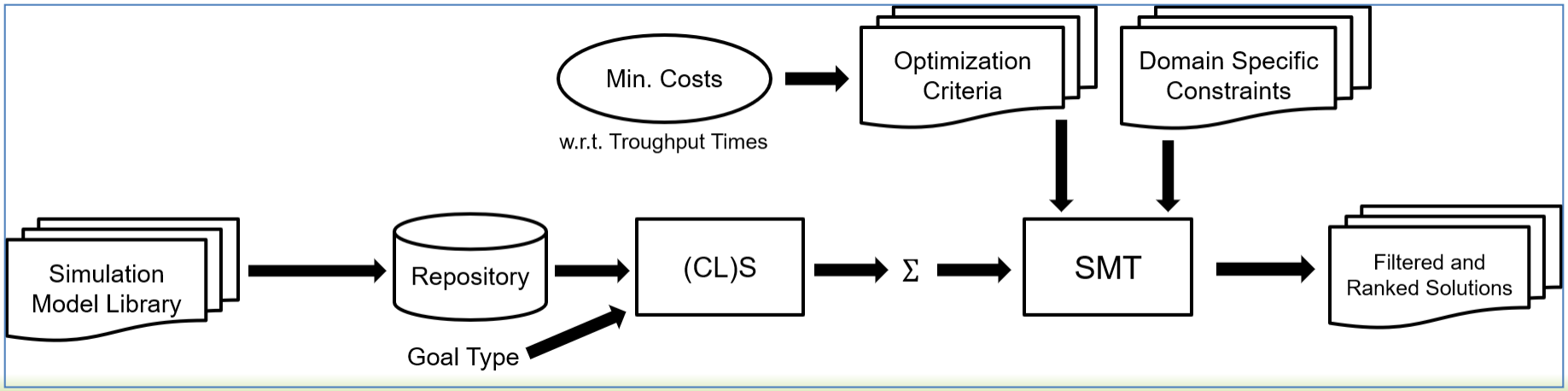


Fig. 1. Overview of the decision-making scope of the industrial use case



Migration of a simulation model into a productline

- The scenario is converted into a feature model
- A feature model represents all occurences of a software product line
- It allows to map the variability and different configurations of the given simulation model

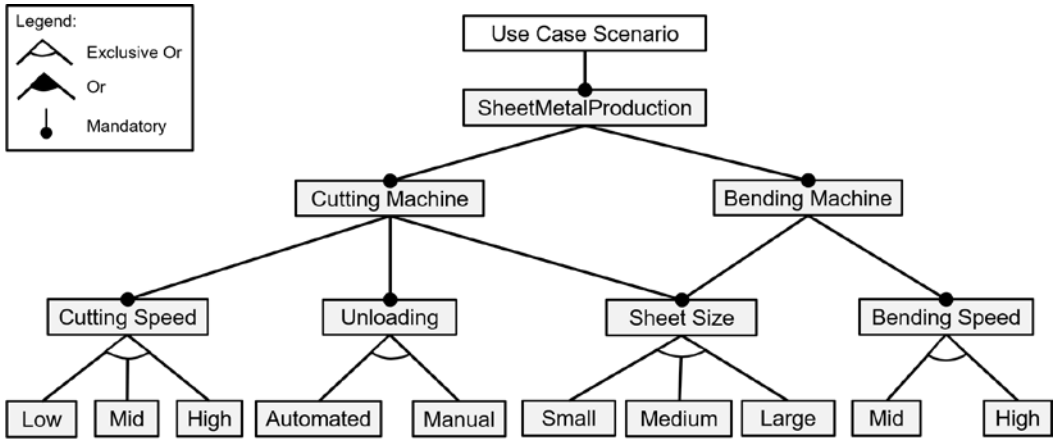


Fig. 2. Feature model

- Given repository Γ and a target goal
- (CL)S tries to find a set of combinators that meet the target type and cover all input parameters
- The output of the synthesis are XML files, which are directly executable by AnyLogic
- In our case, we obtain 11664 well-prepared and valid simulation models
- But!
  - Not all of the generated simulation models are useful
  - Various configurations violate against general project constraints

➢ Solution: Using constraint solving techniques for filtering simulation models

Migration of a simulation model into a productline

- The feature model is transformed into combinators for synthesis
- The combinators are held in a repository Γ
- Each combinator is classified by
  - A combinator name
  - A native type (e.g. integer or string)
  - A semantic type
  - Input parameters, which are classified in the same way (optional)
- Combinators also have implementation details such functions manipulating XML fragments

$$
\begin{aligned}
\Gamma = \{\ & sheetProduction : (E \to E \to E \to E \to E) \cap \\
& (CuttingMachine(\alpha_1, \beta_1, \gamma_1) \to \\
& CuttingMachine(\alpha_2, \beta_2, \gamma_2) \to \\
& BendingMachine(\alpha_3, \gamma_3) \to \\
& BendingMachine(\alpha_4, \gamma_4) \to \\
& SheetProduction), \\
& cuttingMachine : (E \to E \to E \to E) \cap \\
& (CuttingTime(\alpha) \to \\
& Unloading(\beta) \to \\
& SheetSize(\gamma) \to \\
& CuttingMachine(\alpha, \beta, \gamma)), \\
& bendingMachine : (E \to E \to E) \cap \\
& (BendingTime(\alpha) \to \\
& SheetSize(\gamma) \to \\
& BendingMachine(\alpha, \gamma)), \\
& cuttingLowEnd : E \cap CuttingTime(Low), \\
& cuttingMidEnd : E \cap CuttingTime(Mid), \\
& cuttingHighEnd : E \cap CuttingTime(High), \\
& bendingMidEnd : E \cap BendingTime(Mid), \\
& bendingHighEnd : E \cap BendingTime(High), \\
& manualUnloading : E \cap Unloading(Manual), \\
& automaticUnloading : E \cap Unloading(Automated) \\
& smallSheetSize : E \cap SheetSize(Small) \\
& mediumSheetSize : E \cap SheetSize(Medium) \\
& largeSheetSize : E \cap SheetSize(Large)\ \}
\end{aligned}
$$

$$
\begin{aligned}
WF = \{\ & (\alpha \to Low), (\alpha \to Mid), (\alpha \to High), \\
& (\beta \to Automated), (\beta \to Manual), \\
& (\gamma \to Small), (\gamma \to Medium), (\gamma \to Large)\ \}
\end{aligned}
$$

Experimental results

- Exemplary job that requires processing of 40 small-sized and 60 medium-sized sheets
- Assumptions
  - Defined threshold for the allowed throughput time of 400 minutes
  - Large-sized configurated machine can also handle medium-sized and small-sized sheets
- Without considering threshold for the throughput time, 5184 different configurations are generated

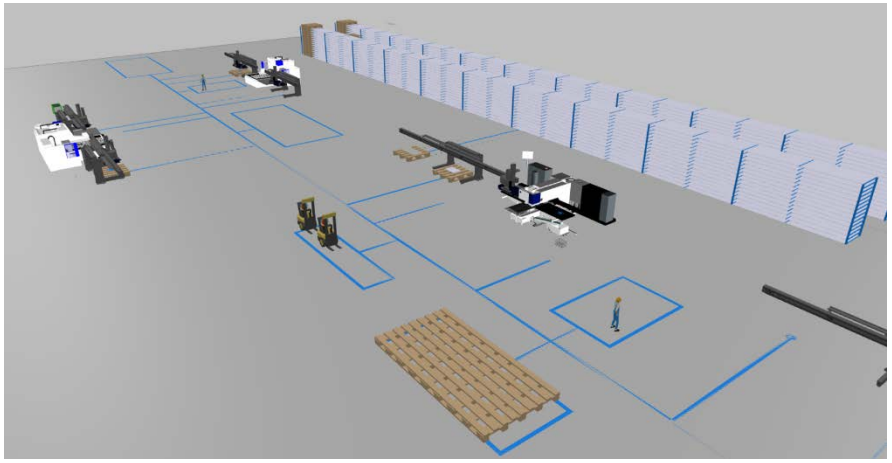


Fig. 4. Real-world example of a sheet metal production

Experimental results

- After filtering 1332 solutions are remaining
- The following configurations are the best economic solutions:

Costs	Time (in min)	Cutting Machine 1			Cutting Machine 2			Bending Machine 1		Bending Machine 2	
		Speed	Sheet Size	Unloading	Speed	Sheet Size	Unloading	Speed	Sheet Size	Speed	Sheet Size
2240	380	L	S	A	L	M	A	M	S	M	M
2360	380	L	S	A	L	M	A	M	S	M	L
2400	380	L	M	A	L	M	A	M	S	M	M

Table 1. Best configurations for considered use case example

	Cutting Speed			Bending Speed		Unloading		Sheet Size		
	Low	Mid	High	Mid	High	Auto	Manual	Small	Medium	Large
Time (in sec)	72	60	48	120	96	30	180	0	10	20
Costs	5	10	15	10	15	5	1	2	4	6

Table 2. Parameters used in use case example

Generalizing an existing simulation model via CLS-retrofitting

Proceedings of the 2022 Winter Simulation Conference
B. Feng, G. Pedrielli, Y. Peng, S. Shashaani, E. Song, C.G. Corlu, L.H. Lee, E.P. Chew, T. Roeder, and P. Lendermann, eds.

AUTOMATIC COMPONENT-BASED SYNTHESIS OF USER-CONFIGURED MANUFACTURING SIMULATION MODELS

Alexander Mages
Carina Mieth

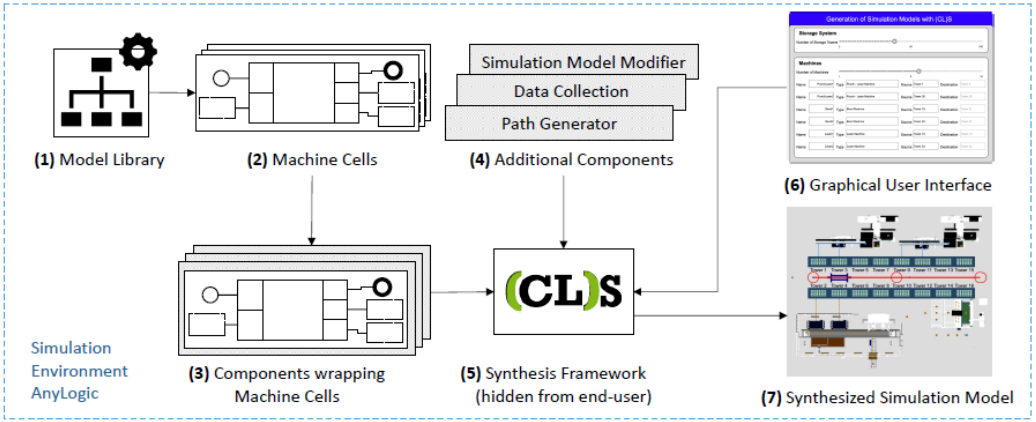
TRUMPF Werkzeugmaschinen SE & Co. KG
Johann-Maus-Str. 2
Ditzingen, 71254, GERMANY

Jens Hetzler

ITK Engineering GmbH
Im Speyerer Tal 6
Rülzheim, 76761, GERMANY

Jakob Rehof
Fadil Kallat
Christian Riest
Tristan Schäfer

Department of Computer Science
Chair for Software Engineering
TU Dortmund University
Otto-Hahn-Str. 12
Dortmund, 44227, GERMANY



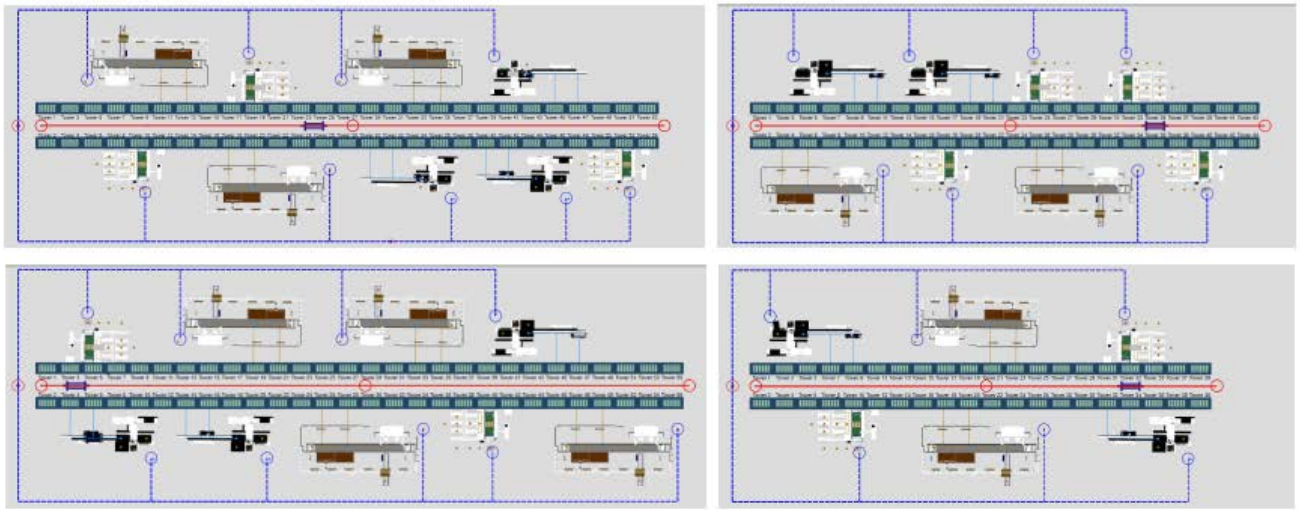


Figure 5: Screenshots of the 2D visualization of generated simulation model variants which represent shop floors with various instances of each machine type and a storage system with a varying number of towers. The paths for the workers are also synthesized.

Design space exploration with active learning

Design Space Exploration for Sampling-Based Motion Planning Programs with Combinatory Logic Synthesis

Tristan Schäfer, Jan Bessai, Constantin Chaumet, Jakob Rehof, and Christian Riest

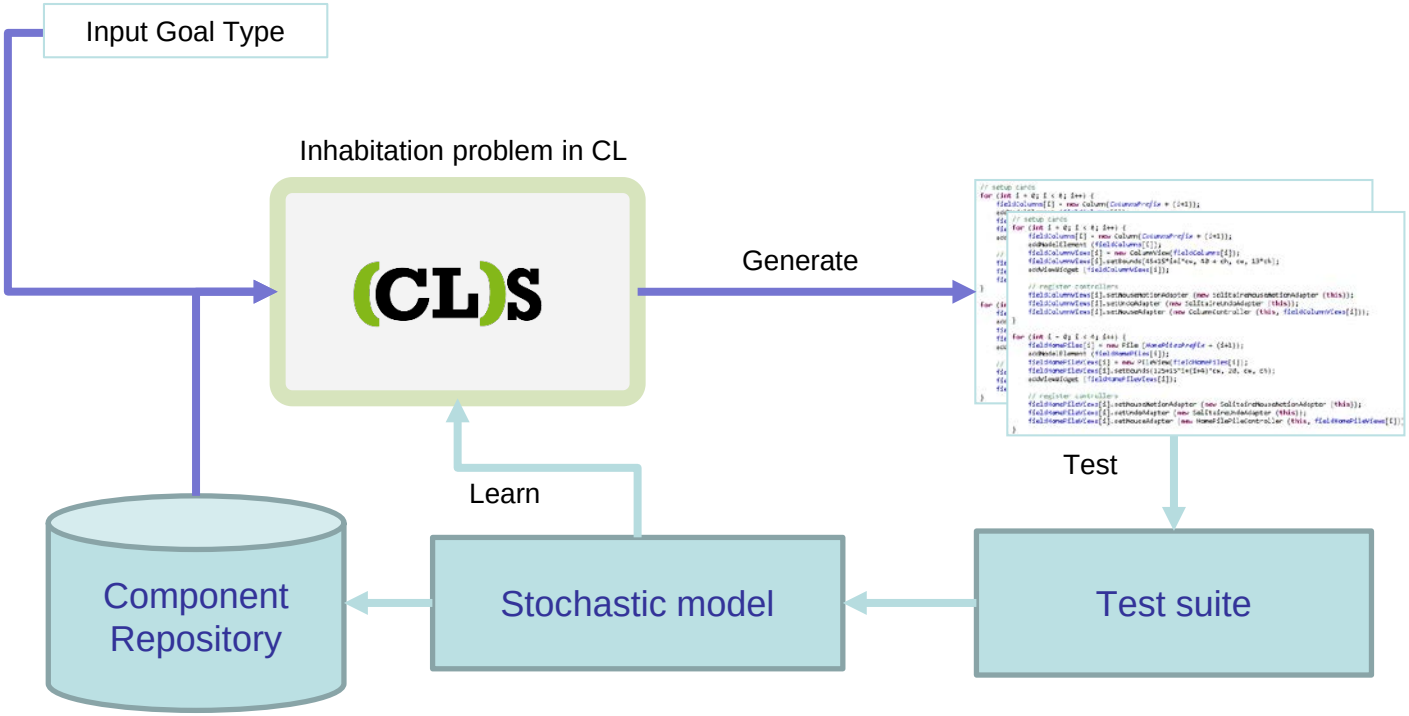
TU Dortmund University, 44227 Dortmund, Germany
{tristan.schaefer, jan.bessai, constantin.chaumet, jakob.rehof, christian.riest}@tu-dortmund.de

WAFR 2022

The 15th International Workshop on the Algorithmic Foundations of Robotics

June 22-24, 2022 at the University of Maryland, College Park

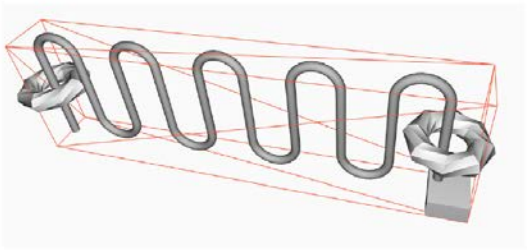
Extension: generate-and-test loop for data analysis and learning



Feature space exploration for sample-based motion planning

OMPL Example [1]

"Escape": Result path along narrow passage



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

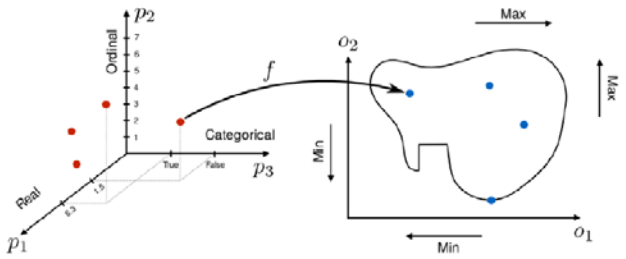


Figure: Multi-objective optimization [2]

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

Feature space exploration for sample-based motion planning

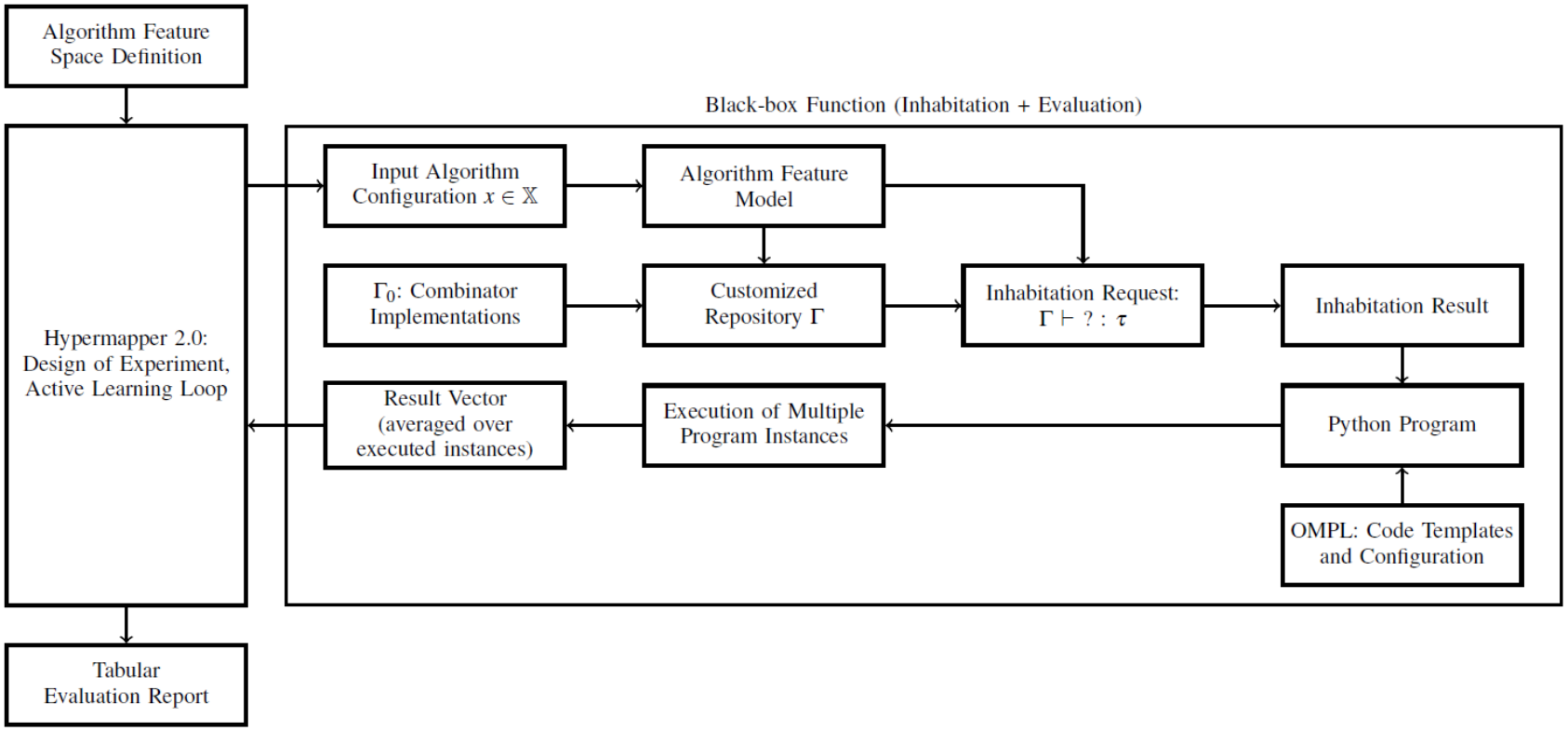
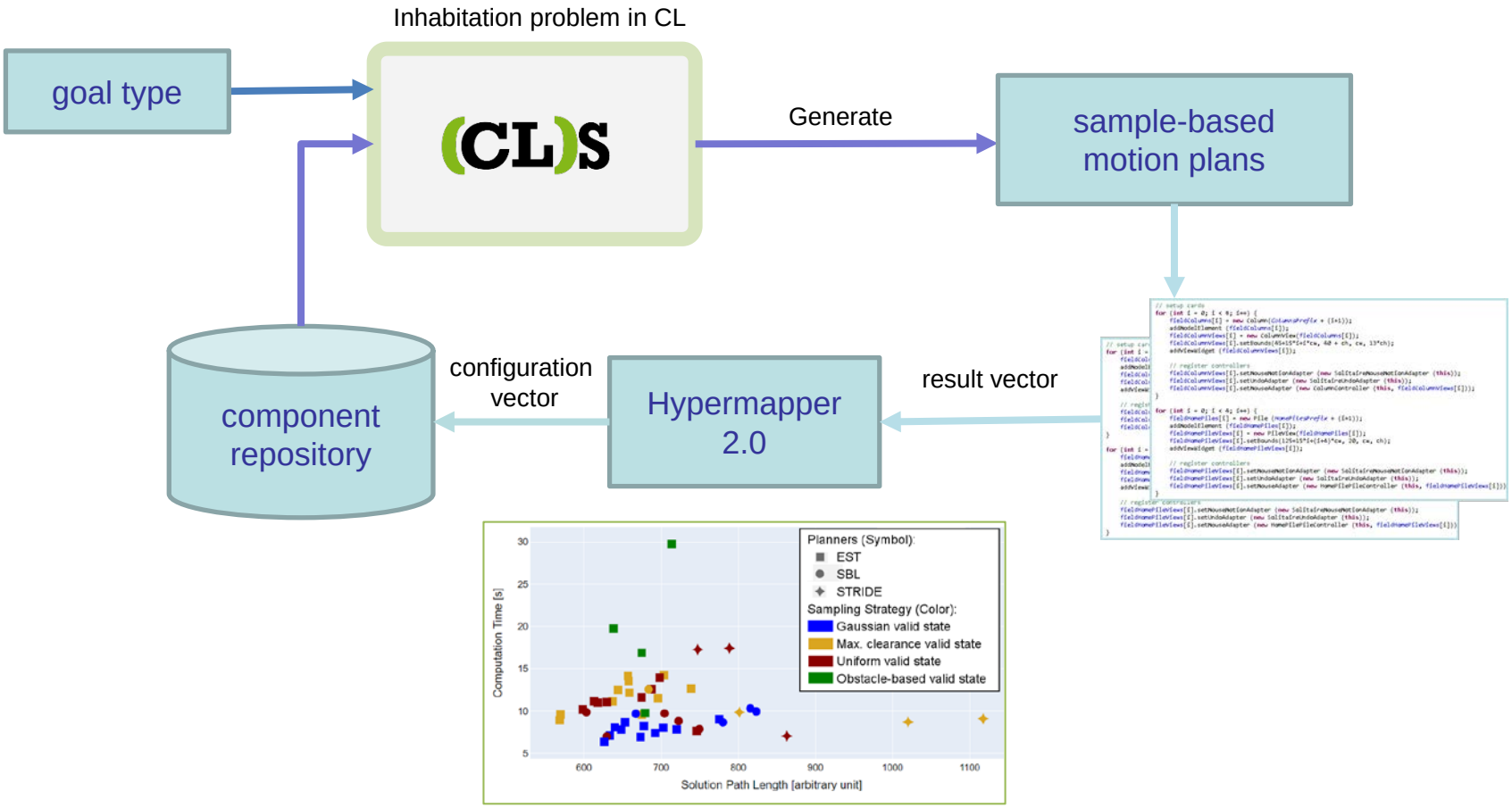


Fig. 2. Overview of the approach: Hypermapper selects samples from the algorithm feature space. Samples are forwarded into the black-box function, which generates Python programs that are executed to obtain evaluation information that is fed back to close the learning-loop.



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

Feature space exploration for sample-based motion planning

$$
\begin{aligned}
&\Gamma_s = \{ \\
&\text{PlannerAssembly:} \\
&any\_planner \rightarrow any\_state\_validator \rightarrow \\
&any\_motion\_validator \rightarrow any\_simplification \rightarrow \\
&sbmp\_input \rightarrow sbmp\_program, \\
&\\
&\text{PRMStarSchema :} \\
&(any\_opt\_obj \rightarrow sampler\_space \rightarrow PRMStar) \cap \\
&(any\_opt\_obj \rightarrow sampler\_valid\_state \rightarrow PRMStar) \cap \\
&(any\_opt\_obj \rightarrow sampler\_informed \rightarrow PRMStar), \\
&\\
&\text{ESTSchema :} \\
&obj\_path \rightarrow sampler\_valid\_state \rightarrow EST, \\
&[...] \\
&\}
\end{aligned}
$$

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

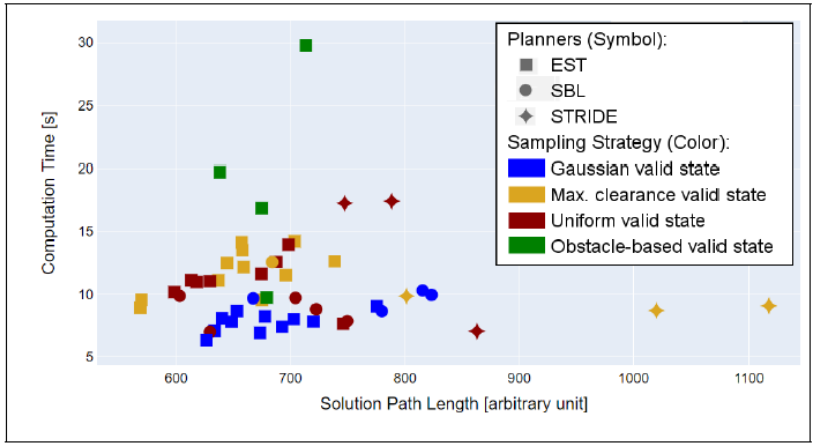


Fig. 4. Results for the motion planning problem "Abstract"

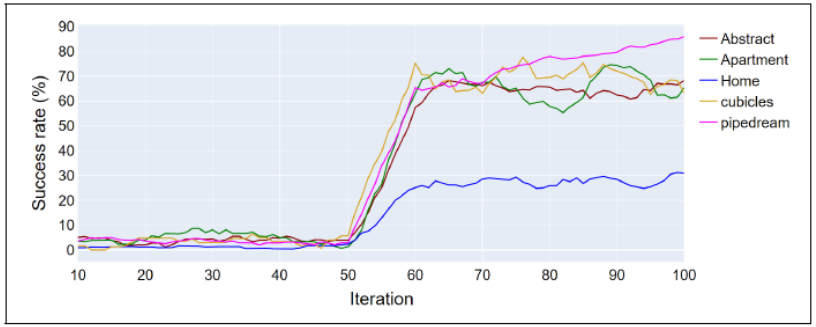


Fig. 5. Success rate for various problem instances

Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 00 (2021) 000–000

www.elsevier.com/locate/procedia

54th CIRP Conference on Manufacturing Systems

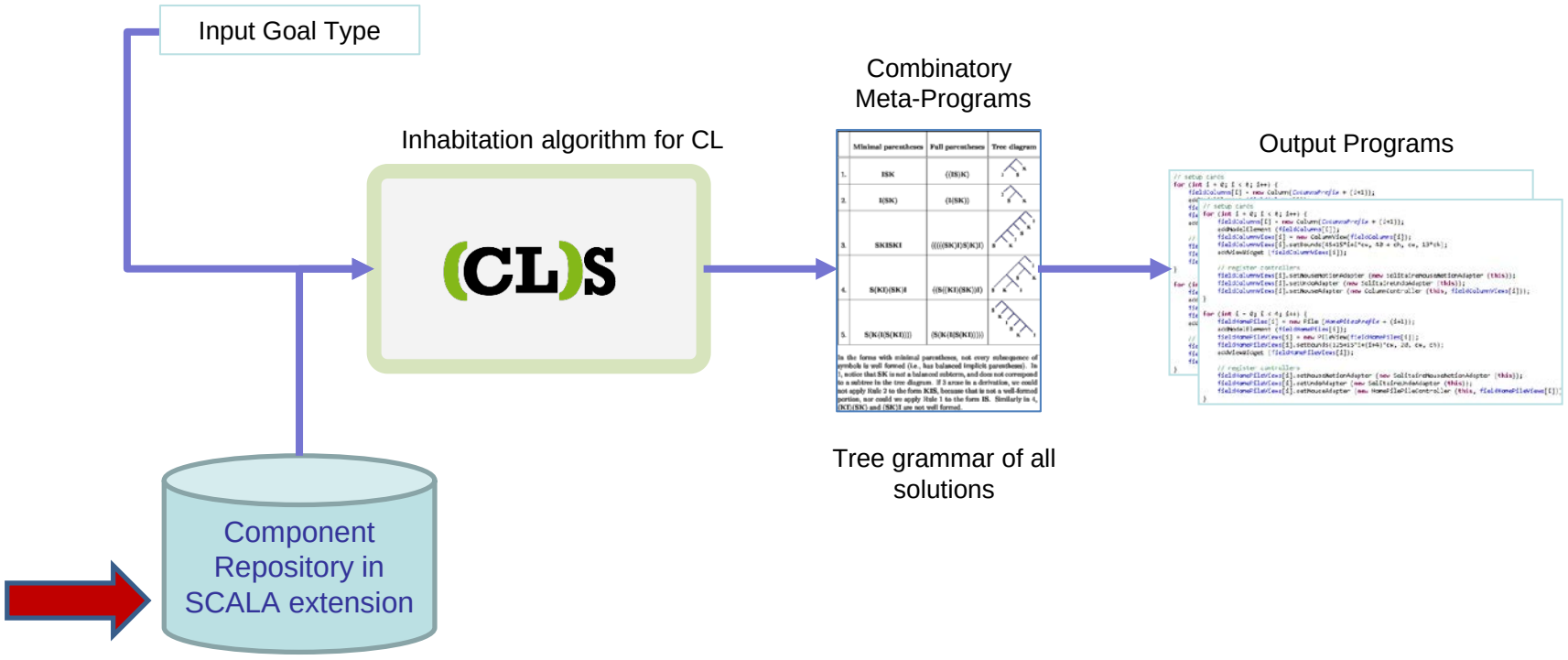
Automatic Building of a Repository for Component-based Synthesis of Warehouse Simulation Models

Fadil Kallat[a,*], Jakob Pfrommer[b], Jan Bessai[a], Jakob Rehof[a], Anne Meyer[c]

Motivation

- Decision-making process in factory and warehouse planning is often supported by simulation studies
- But creating simulation models is a time-consuming and costly task
- Due to budget and time restrictions, planners often must leave out variants
- Research field “Automatic Simulation Model Generation” brought out many successful approaches, but …
  - it’s still challenging to generate structural variants and
  - corresponding complex control strategies, and
  - many approaches are tailored to specific use-cases

Extensions to CLS-framework



PRE

Example: generation/config of components

ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 00 (2021) 000–000

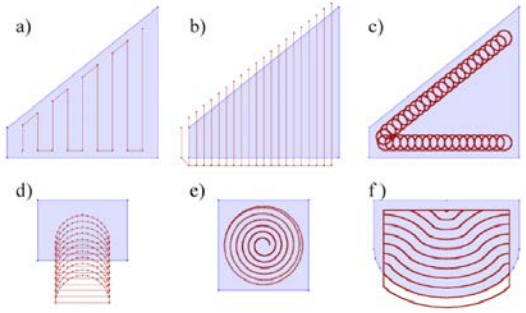
www.elsevier.com/locate/procedia

54th CIRP Conference on Manufacturing Systems

A Synthesis-based Tool Path Planning Approach for Machining Operations

Tristan Schäfer[a,*], Jim A. Bergmann[b], Rafael Garcia Carballo[c], Jakob Rehof[a], Petra Wiederkehr[b]

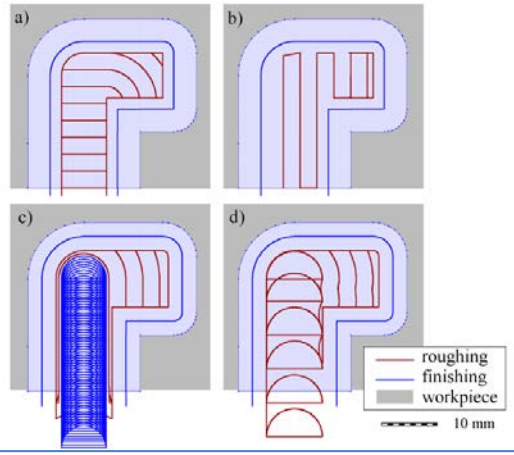
Motion Planning Primitives



Repository Overview

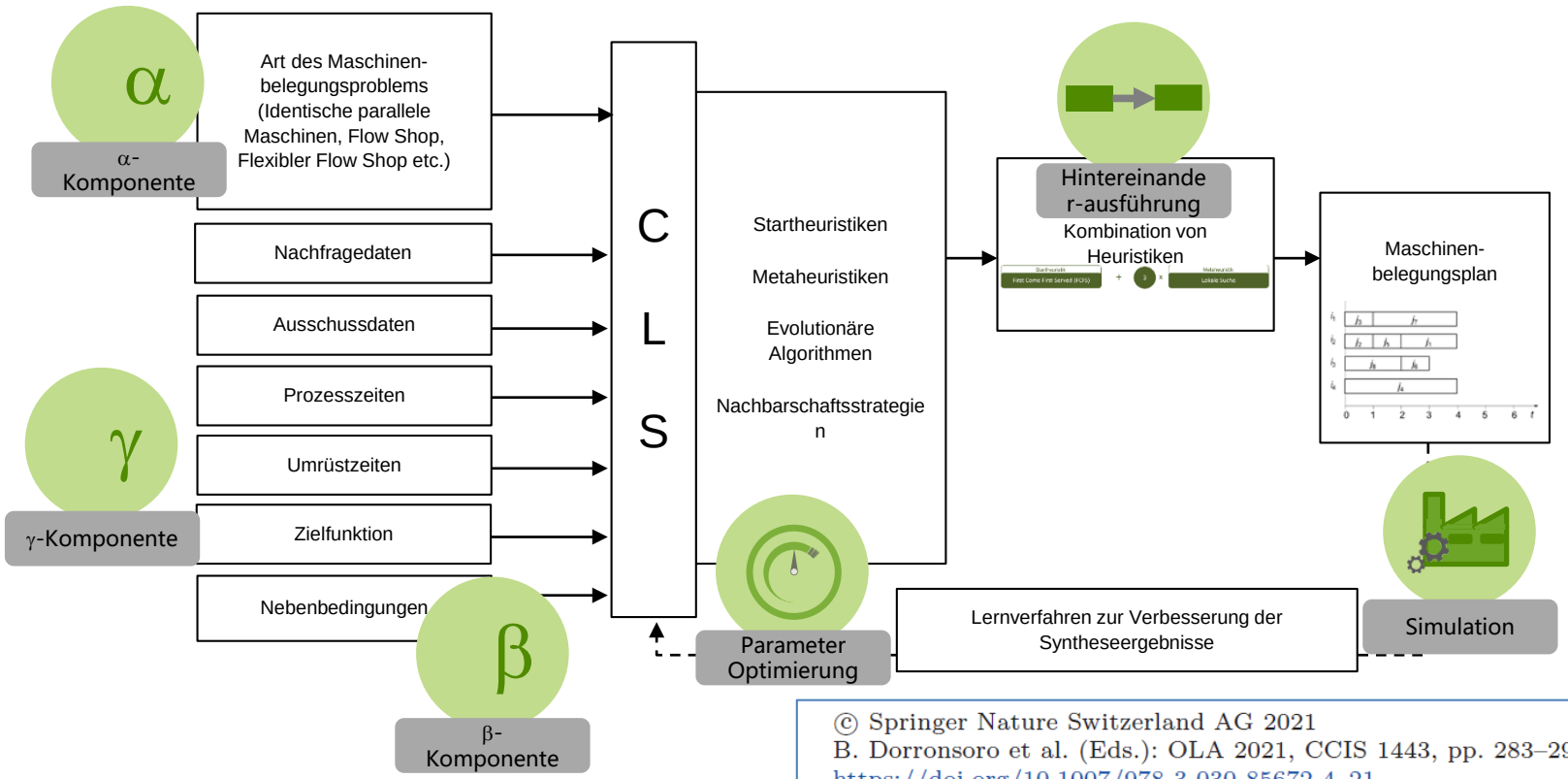
Combinator Name	*Type*
AluminumRoughing	$CncTool \cap roughing \cap aluminum$
AluminumFinishing	$CncTool \cap finishing \cap aluminum$
SteelRoughing	$CncTool \cap roughing \cap steel$
SteelFinishing	$CncTool \cap finishing \cap steel$
GenericCompositionPcStep	$PathCoverageStep \cap \alpha \rightarrow PathCoverageStep \cap \alpha \rightarrow PathCoverageStep \cap \alpha$
TrochoidalSlot	$CncTool \cap \alpha \rightarrow PathCoverageStep \cap \alpha \rightarrow PathCoverageStep \cap \alpha \cap pcRoot$
SpiralRoughing	$CncTool \cap \alpha \rightarrow PathCoverageStep \cap \alpha \rightarrow PathCoverageStep \cap \alpha \cap pcRoot$
ContourSteel	$CncTool \cap steel \rightarrow PathCoverageStep \cap steel$
ContourMultiTool	$CncTool \cap roughing \cap aluminum \rightarrow CncTool \cap finishing \cap aluminum \rightarrow PathCoverageStep \cap aluminum$
ZigSteel	$CncTool \cap steel \rightarrow PathCoverageStep \cap steel$
ZigZagAlu	$CncTool \cap aluminum \rightarrow PathCoverageStep \cap aluminum$
SteelTrochoidal	$CncTool \cap steel \rightarrow PathCoverageStep \cap steel$

Results



Synthesis of Scheduling Heuristics by Composition and Recombination

Dominik Mäckel[1], Jan Winkels[1], and Christin Schumacher[2(✉)]



B. Dorronsoro et al. (Eds.): OLA 2021, CCIS 1443, pp. 283–293, 2021.
https://doi.org/10.1007/978-3-030-85672-4_21

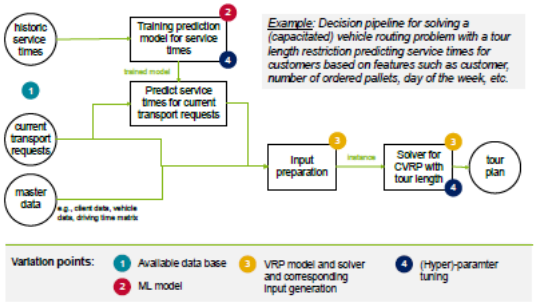
Decision Pipeline Synthesis

Anne Meyer[1] und Jan Bessai[2]

technische universität dortmund

Starting point

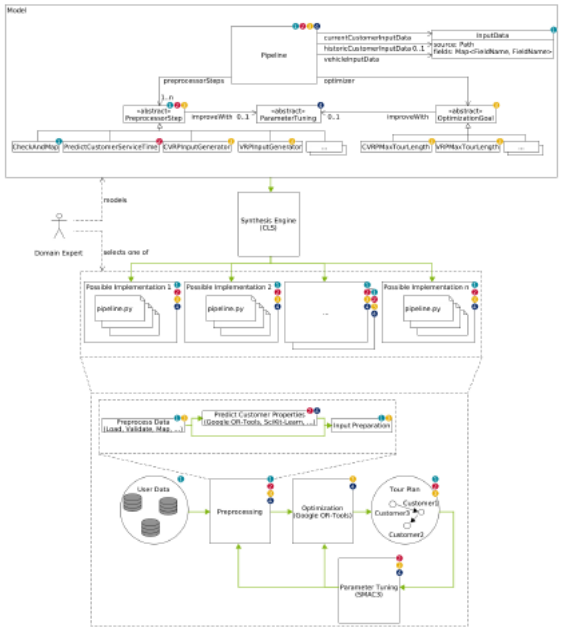
Decision-making in logistics usually requires decision pipelines combining ML and optimization



Hypothesis 1: To automate the selection, configuration, and deployment of decision pipelines combining software synthesis and algorithm selection and configuration is beneficial!

Hypothesis 2: Configuration of algorithms present in pipeline components should consider the overall pipeline performance instead of the isolated performance of components!

Architecture decision pipeline synthesis



Current work

Hypothesis 1:

- Working code generator for Python via synthesis from abstract domain model
- In a next step, the parameter space of the algorithms present in a pipeline is also generated to prepare algorithm configuration

Hypothesis 2:

- At the moment we integrate SMAC3 as parameter tuning tool
- We develop performance measures considering the whole pipeline performance
- In a next step, we use real world data from industry to test our system

Open questions

- Do you know any similar work in this context?
- Do you have ideas for evaluation?
- Which conferences and journals fit well?

Related Work

Kallat, F., Pfrommer, J., Bessai, J., Rehof, J., Meyer, A. (2021). Automatic Building of a Repository for Component-based Synthesis of Warehouse Simulation Models. 54th CIRP Conference on Manufacturing Systems.

Schäfer, T., Bergmann J., Garcia Carballo, R., Rehof, J., Wiederkehr P. (2021) A Synthesis-based Tool Path Planning Approach for Machining Operations. 54th CIRP Conference on Manufacturing Systems.

Bessai, J. (2019). A Type-Theoretic Framework for Software Component Synthesis. Technical University of Dortmund, Germany. DOI: http://dx.doi.org/10.17877/DE290R-20320

Meyer, A., Zander, S., Knapper, R., Setzer, T. (2018). Decision support pipelines – Durchgängige Datenverarbeitungsinfrastrukturen für die Entscheidungen von morgen. In: S. Wischmann, E. A. Hartmann (Hrsg.), Zukunft der Arbeit: Eine praxisnahe Betrachtung (S. 207–220). doi: 10.1007/978-3-662-49266-6_15

[1] anne2.meyer@tu-Dortmund.de | [2] jan.bessai@tu-dortmund.de

Proceedings of the 2019 Winter Simulation Conference
N. Mustafee, K.-H.G. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son, eds.

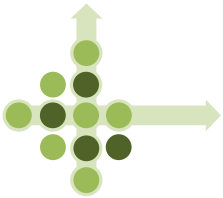
TRENDS IN AUTOMATIC COMPOSITION OF STRUCTURES FOR SIMULATION MODELS IN PRODUCTION AND LOGISTICS

Sigrid Wenzel
Jana Stolipin

Institute of Production Technology and Logistics
Dept. of Production Org. and Factory Planning
University of Kassel
Kurt-Wolters-Str. 3
Kassel, 34125, GERMANY

Jakob Rehof
Jan Winkels

Department of Computer Science
Chair for Software Engineering
TU Dortmund University
Otto-Hahn-Str. 12
Dortmund, 44227, GERMANY



Conclusion

Many industrial applications. Some are surprising.

CLS approach works well in complex, real scenarios.

CLS-Framework is easy to use and easily exstensible.

Future directions around model-based training for AI.