Proceedings of the Design Society, Volume 6: DESIGN 2026
https://doi.org/10.1017/pds.2026.10437

Automated quantitative functional decomposition in product design

Philipp Rosenthal✉, René Heesch and Oliver Niggemann

Helmut Schmidt University, Germany

✉ philipp.rosenthal@hsu-hh.de

ABSTRACT: Functional decomposition shapes early design decisions but is largely qualitative, leaving units and measures implicit. This work introduces the Quantitative Functional Decomposition Problem, which formalizes functions and interfaces with measurable quantities, making decomposition solvable as a quantified planning problem. Two case studies show that the approach gives immediate feedback on the admissibility of functions and their connections. Design engineers get consistent quantified structures, which speed up iteration, reduce work and set targets for subsequent steps in the design process.

KEYWORDS: functional modelling, functional reasoning, design automation, early design phase, artificial intelligence (AI)

1. Introduction

The early design phase has great influence on the development process of new products (Ehrlenspiel & Meerkamm, 2017). This conceptual phase not only provides the ground for the design process itself, it also has a major impact on the finalized product (Bender & Gericke, 2021; Verein Deutscher Ingenieure, 2019). Functional decomposition and the consequent Function Structure are central in this stage of the design process. The overall function of a new product is derived from its requirements. By decomposing this overall function into base functions and their connections, the purpose of the product becomes accessible for further steps in the design process (Roth, 2000). The resulting structure clarifies the flows of materials, energy and information through the product. The focus here is a static specification for the conceptual design phase, rather than a dynamic behaviour model.

However, most of the Roth-style structures only provide information about the routing of the three entities between functions and not their magnitudes and units. This leaves design engineers with significant uncertainty, since they cannot check on conservation laws, detect infeasibilities or compare competing Function Structures on measurable grounds.

The decomposition process is a tedious and complex task, usually depending on the experience of specialized design engineers. At the same time, it is susceptible to inconsistencies at the interfaces between functions.

Nevertheless, recent research by Rosenthal et al. (2024) and Haddad and Seibel (2025) has shown that decomposition can be automated using Artificial Intelligence (AI) planning and Large Language Model (LLM) assisted approaches. These approaches operate on qualitative graphs, ensuring logical connectivity, but they cannot check unit consistency or conservation.

Formal-methods results show that properties of symbolic system models can be verified from formal, solver-ready specifications (Cimatti et al., 2022). At the same time, emerging automation for functional modelling already produces such machine-readable models in the conceptual design phase. Yet, quantitative consistency checks are still missing. This combination motivates a problem framing that also carries quantitative aspects and consistency criteria into the conceptual design phase.

The resulting challenge consists of three deficits. First, a problem framing deficit: An extended problem formalization is needed, which enables quantitative checks without giving up abstraction advantages of

https://doi.org/10.1017/pds.2026.10437 Published online by Cambridge University Press

the early design phase. Second, a representation deficit: Functions and their structures need to be represented in a different way in order to cater to quantitative measures and conservation laws. The final deficit is an algorithmic one. Automated functional decomposition must still be possible by AI algorithms, when including quantitative measures.
This challenge can be referred to as the Quantitative Functional Decomposition Problem (QFDP), the task of deriving a working Function Structure that, in addition to logical consistency, makes quantitative consistency checkable in conceptual design phase. For practitioners this means earlier plausibility of concepts with units and ranges and clearer requirements for downstream work.
This leads to the following research questions (RQ).
RQ 1: How should the QFDP be formalized to incorporate quantitative consistency and make it accessible to AI algorithms?
RQ 2: How does extending the functional decomposition from qualitative to quantitative measures change the representation and composition of functions and their structures?
RQ 3: Which algorithmic approach is suited to find correct solutions for a given QFDP?
This work introduces the Quantitative Function Structurer (QFS). This new algorithm is able to find meaningful quantified solutions for a given QFDP. A formalized framework supports the approach by defining the problem and making it processable for symbolic AI algorithms. The QFS is evaluated in two real world case studies by assessing its correctness in building quantified solutions for the problems.

2. Related work

The VDI 2222-1 (1997) defined functions as relations between input entities, output entities and their internal mapping. The emphasis is on abstract and solution neutral formulations to avoid specific assignments and ensure comparability. Pahl and Beitz (Bender & Gericke, 2021) identify Function Structures as the central working artefact in the conceptual design phase. Resulting from the requirements, the overall function gets systematically decomposed into subfunctions and their relations. The VDI 2221-1 (2019) embeds the Function Structure in the development process in the same way and gives the procedure a methodical foundation. These references set the standard. However, they focus primarily on qualitative aspects and omit explicit unit and dimension checks and quantitative measures. In addition, VDI/VDE 3682 (2005) formalizes process descriptions in terms of states and process operators, which is conceptually close to functional modelling. Yet, it likewise remains qualitative and does not enforce unit consistency.
Roth (2000; 2001) introduced a general set of 30 functions for the three entities material, energy and information. He provides a uniform vocabulary for the functional decomposition in the early design phase. These Roth Functions enable the systematic and consistent modelling of Function Structures, but they do not prescribe magnitudes, units or parameterized interfaces.
Koller and Kastrup (1998) anchor functions in an engineering methodology emphasizing the cause-and-effect relationship as the foundation of technical functions. This supports the systematic decomposition, yet this approach remains qualitative and omits any quantitative structure.
Gero (1990) describes functions as the link between purpose description i.e. requirements and the actual physical behaviour. He thereby emphasises the role of the function concept for decisions in the early stages of the development process, focusing on conceptual clarity rather than quantitative consistency. Over time different approaches have been elaborated in order to automate and systematize the process of modelling a Function Structure in the early design phase.
Notably, the function-behavior-state (FBS) modeler proposed by Umeda et al. (1996) is a knowledge-based tool and shows an early example of computer supported functional modelling. However, functional decomposition remains largely expert driven and qualitative with no unit and dimension enforcement.
Kitamura and Mizoguchi (2003) present an ontology-based systematization of function knowledge. They separate functions, i.e. what must be achieved, with ways, i.e. how it is achieved. While organizing functions, their approach does not automatically decompose the overall product function into a structure.
Tensa et al. (2019) use association rules in design repositories to extract component-function-flow associations. The data driven approach relies on existing data to derive functional chains and partial structures, supporting the decomposition process but without taking quantifiability into account. Complementing this strand, Mikes et al. (2021) introduce AutoFunc as a practical tool for semi-automated generation and verification of functional models from repositories. Their approach addresses

https://doi.org/10.1017/pds.2026.10437 Published online by Cambridge University Press

consistency and reuse. However, it remains primarily qualitative and does not perform quantitative consistency checks.

Mohammed and Shammari (2021) propose a rule-based procedural algorithm to construct Function Structures. They explicitly address the derivation of these structures from customer requirements. While their approach pays attention to consistency and physical correctness it also does only take qualitative measures into account and is limited to its reconciled functional basis.

Rosenthal et al. (2024) have shown that the problem of finding a working Function Structure can be expressed as a planning problem. In this formulation, Roth Functions are modelled as actions and a partial-order planner (POP) is used to derive a structure from specified boundary conditions. The authors demonstrate their approach by using POP to solve decomposition problems, but the representation remains qualitative and it cannot enforce units/conservation or compute feasible quantitative interfaces.

Haddad and Seibel (2025) propose to frame the decomposition problem as a knowledge-graph construction. They make use of an LLM and further improve the results with a Monte Carlo tree search (MCTS) loop. While this approach shows improvements over the plain usage of LLMs, the authors still report substantial structural errors and the need for expert review. In addition, the representation remains qualitative.

Across standards, textbooks and foundational work, Function Structures are established as a central solution neutral artefact. Yet, they are qualitative. Model-based, rule-based, data-driven, planning-based and LLM-based approaches demonstrate that functional decomposition can be supported and automated but likewise operate on qualitative graphs without explicit unit and dimension consistency or conservation checks.

3. Solution

The previous section pointed out that existing approaches rely on qualitative measures to ensure logical correctness within the Function Structure only, entirely neglecting any kind of quantity. Inspired by the approach of Rosenthal et al. (2024), this paper aim to extend automated decomposition to quantifiable Function Structures. Therefore, the initial problem formalization needs to be adapted and extended.

3.1. Formalization

Definition 1. Index Sets (finite domains):
$\Phi = \{solid, liquid, gas\}$
$E = \{chemical, electrical, kinetic, potential, thermal\}$
$I = \{analog, digital, quantum\}$
U_M *a countable universe of material identifiers*
$M \subseteq U_M$ *a finite material set for a given problem*
$S = \{MFlow, EFlow, IFlow\}$
U_P *a countable universe of Properties*
$P \subseteq U_P$ *a finite Property set for a given problem*
Definition 2. Problem spaces:
Material space: $V_{mat}(M) = \{x_M | x_M : M \times \Phi \to \mathbb{R}\}$
Energy space: $V_{en} = \{x_E | x_E : E \to \mathbb{R}\}$
Information space: $V_{info} = \{x_I | x_I : I \to \mathbb{R}\}$
Stream space: $V_{stream} = \{x_S | x_S : S \to \mathbb{R}\}$
Property space: $V_{prop} = \{x_P | x_P : P \to \mathbb{R}\}$
Definition 3. Entity space for a given problem:
$V = V_{mat} \bigoplus V_{en} \bigoplus V_{info} \bigoplus V_{stream} \bigoplus V_{prop}$
$\dim V = |\Phi| \cdot |M| + |E| + |I| + |S| + |P| = 3 \cdot |M| + 5 + 3 + 3 + |P|$
Definition 4. Entity vector (state of one entity):
Each entity is uniquely defined by a vector $x \in V$, assigning specific values to all components of V, with
$x = (x_M, x_E, x_I, x_S, x_P)$ and $x_M : M \times \Phi \to \mathbb{R}, x_E : E \to \mathbb{R}, x_I : I \to \mathbb{R}, x_S : S \to \mathbb{R}, x_P : P \to \mathbb{R}$.
Definition 5. A control parameter $c \in \mathbb{R}$ is a scalar.

https://doi.org/10.1017/pds.2026.10437 Published online by Cambridge University Press

Definition 6. A Roth Function $f \in R$ is a tuple:
$f = (n_f, i_f, o_f, c_f)$, where n_f is a unique name for the function, i_f, o_f with $i_f, o_f \in V$ are inputs and outputs of the function and c_f is a control parameter specifying the quantitative transition from i_f to o_f.
Definition 7. The Roth Function Set R is the set of all Roth Functions:
$R = \{f | f \text{ represents a Roth Function}\}$
Definition 8. The Quantitative Roth Catalogue is a subset of the Roth Functions (usable subset):
$QRC \subseteq R$.
Definition 9. A Function Structure FS is an admissible directed graph $FS = (N, K, \Lambda)$ with:
N a finite set of nodes
$K \subseteq N \times N$ a set of directed edges
$\Lambda : N \to \mathrm{R}$ a labeling, which maps each node to its Roth Function
For any edge $(u, w) \in \mathrm{K}$, let $\Lambda(u) = (n_u, i_u, o_u, c_u)$ and $\Lambda(w) = (n_w, i_w, o_w, c_w)$. Then the edge is admissible iff o_u is compatible with i_w under the combination rules associated with the Roth Functions.
Definition 10. A Quantitative Functional Decomposition Problem $QFDP$ is a tuple:
$QFDP = (S_x, i_x, o_x, \mathrm{QRC})$, where $S_x \subseteq \cup_{m \in \mathbb{N}_{>0}} V^m$ is a finite set of entity vector states. For every $s_x \in S_x$ there exists $m \in \mathbb{N}_{>0}$ with $s_x = (x_1, \ldots, x_m)$, $x_j \in V$, for all $j = 1, \ldots, m$. The input to the $QFDP$ is i_x and o_x is the output, with $i_x, o_x \in S_x$. QRC is the Quantitative Roth Catalogue.
Definition 11. The solution f_d to a $QFDP$ is a finite ordered sequence of Roth Functions $f_d = (f_1, \ldots, f_k), k \in \mathbb{N}_{>0}, f_j \in QRC$, for all j, such that there exists a state $s' \in S_x$ with $i_x \overset{f_d}{\Rightarrow} s'$ and $s' \models o_x$, which induces the Function Structure $FS(f_d)$.
This section is the direct answer to **RQ 1**. By using the formalizations, quantitative consistency can be incorporated into the Decomposition Problem and make it accessible to AI algorithms.

3.2. Roth catalogue adaptation

The Quantitative Roth Catalogue (QRC) from Definition 8 provides six function families, i.e. split, add, shape, transform, guide and store. These differ from the classical Roth Functions (Roth, 2000) only in quantitative refinement. Each QRC function declares a parent Roth Function $r \in R$, so that $QRC \subseteq R$ holds true by construction. More precisely, guide, add, split and store all refine their counterparts. Transform_energy/information refines transform. For shape_material, the parent is shape when only kind changes and transform_material when the phase label changes. The associated energetics are explicitly modelled in the energy domain rather than being implicit in the material operator as in the classical Roth Functions.
In this implementation, store and guide act on availability states. Store marks an entity as not available for routing, whereas guide marks it as available for routing. The add family co-locates specified quantities from two sources into a destination without creating or destroying any quantity. On the other hand, the split family moves a specified quantity from a source to a destination, likewise without creation or destruction. Cross-entity interactions for add and split are realized with interface adapters, which enforce unit compatibility, conservation of mass and energy and, where declared, coupling constraints such as stoichiometric ratios.
The shape_material function reclassifies material already present within an entity across kinds and/or phases while conserving total mass. If a phase change requires energy, e.g. melting, that energy is supplied by energy functions and routed via subsequent add or split operations. The transform family converts quantities between forms within an entity. Transform_energy and transform_information change energy or information with declared efficiencies, while material phase or form relabeling is represented with shape_material plus explicit energy transfers.
Table 1 summarizes the mapping from the initial Roth Functions to the QRC and clarifies how classical operators are realized in the quantitative setting used by the following algorithm.
RQ 2 is directly answered by describing the quantitative representation for functions and criteria, which enforce conservation and compatibility throughout the Function Structure. In that way qualitative links are replaced with parameterized and checkable interfaces.

https://doi.org/10.1017/pds.2026.10437 Published online by Cambridge University Press

Table 1. Quantitative adaptation

Classical Roth Functions	QRC Functions	Description
store_material; store_energy; store_information	store	same semantics
guide_material; guide_energy; guide_information	guide	availability for routing
add_with (all 9 variants)	add + adapter	co-locate quantitites; adapter enforces compatibility and conservation across entities
split_with (all 9 variants)	split - adapter	move quantity between entities; adapter enforces compatibility and conservation across entities
shape_material	shape_material	reclassify across kinds and/or phases; mass conserved; no implicit energetics
transform_material	shape_material + energy function	relabelling via shape; requires additional energy function
shape_energy; shape_information	transform_energy; transform_information	re-form within entity; modelled as transform with declared efficiency
transform_energy; transform_information	transform_energy; transform_information	convert with declared efficiency, then route

3.3. Algorithm

The functional decomposition is addressed by expressing it as a planning problem. Working in a numerical-planning setting where control parameters modulate the numerical effects of actions, the planning-as-satisfiability with Satisfiability Modulo Theories (SMT) is adopted as a suitable base paradigm (Heesch, Cimatti, et al., 2024). The planning problem is formulated using the feature-vector–based state-space representation introduced by Heesch, Ehrhardt, and Niggemann (2024), which encodes high-dimensional states as vectors and provides a uniform interface for translating the planning problem into an SMT formula. The resulting Algorithm 1 is shown below.

Algorithm 1 Quantitative Function Structurer

Input: Index Sets, *QFDP*: Quantitative Functional Decomposition Problem
Output: f_d: Sequence of parameterized Roth Functions, or "No solution found"

1: $(phases, energies, infos, streams, kinds) \leftarrow build_enums_from(Index\ Sets, QFDP)$
2: $Problem \leftarrow new_newProblem()$
3: $install_fluents(Problem, \{m_amt, e_amt, i_amt, s_amt, p_val, guided\})$
4: $add_objects(Problem, phases \cup energies \cup infos \cup streams \cup kinds \cup entities)$
5: $Actions \leftarrow build_actions_from_catalog(QRC, phases, energies, infos, kinds)$
6: **for** each $a \in Actions$ **do**
7: $\quad Problem.add_action(a)$
8: **end for**
9: $Problem.set_initial_state(encode_state_newProblem(i_x))$
10: $Problem.set_goal(encode_goal_newProblem(o_x))$
11: $Planner \leftarrow select_planner(backend, options)$
12: $res \leftarrow Planner.solve(Problem)$
13: **if** $res.plan = undefined$ **then**
14: $\quad$ **return** No solution found
15: **end if**
16: $f_d \leftarrow extract_action_sequence(res.plan)$
17: **return** f_d

The input needed for the Quantitative Function Structurer are the Index Sets from Definition 1 and the actual *QFDP* from Definition 10. The output is either a solution to the *QFDP* as in Definition 11 or an indication that no solution can be found. The algorithm starts with building the concrete domain for the instance in line 1, where the state space depends on the initial and the goal state of the *QFPD*. In line 2, a

https://doi.org/10.1017/pds.2026.10437 Published online by Cambridge University Press

Problem is created as an empty new problem, building on the Definitions 2-4. The next line registers the fluent families on the *Problem*, realizing the components of the entity space V as typed numeric state variables. Line 4 populates the object domains of the *Problem* with the union of all instance sets $(phases \cup energies \cup infos \cup streams \cup kinds \cup entities)$.
The following line 5 compiles the QRC into *Actions*. Each Roth tuple $f = (n_f, i_f, o_f, c_f)$ becomes a parameterized action with preconditions and effects. The for-loop from line 6 to line 8 adds all previously built *Actions* to the action set of the *Problem*. This makes the admissible Roth Function transitions available for planning as defined in Definition 9.
Line 9 then encodes the input state $i_x \in S_x$ from Definition 10 as a concrete valuation of the fluent families and installs it as a planning initial state. In the same way line 10 encodes the desired output state $o_x \in S_x$ from Definition 10 as goal constraints over the fluent families. Then a one-shot planning backend with given options is selected in line 11. This backend yields the solver for handling the planning problem. The choice is implementation specific and independent of the formal definitions given above. In line 12 the selected solver is invoked on the constructed planning problem. The result contains a candidate plan, meaning an ordered sequence of actions, i.e. Roth Functions or will indicate a failure or timeout.
The if-condition from line 13 to line 15 checks for an undefined plan or unsuccessful status. If no plan is available, the algorithm returns "No solution found" immediately. On a side note, this test does not reject empty plans of length 0, since these plans are valid when i_x already satisfies o_x. However, if the solver was successful, the ordered action sequence is extracted and assigned to f_d. The final line returns the solution which can directly be used to build the Function Structure FS.
The shown and explained Algorithm 1 answers **RQ 3**.

4. Experimental results

The algorithm is assessed using a coffee maker and a Fused Deposition Modeling (FDM) printer as case studies. The experiments were conducted on a system with an AMD Ryzen 9 9950X 16-core processor and 96 GB RAM. The planning problems derived from the functional decomposition problems were encoded in the Unified Planning Framework (UPF) (Micheli et al., 2025) and solved using the Tempest planning engine, which is based on Panjkovic and Micheli (2023, 2024).

4.1. Coffee maker

The first case study is a coffee maker. It is a suitable benchmark because it couples material and energy entities in a simple checkable way. Yet, it still exercises unit checks, conservation and interface compatibility. Its overall function is to produce hot coffee given it has access to water, coffee powder and electricity. The quantified inputs and outputs are displayed in Table 2.

Table 2. Coffee maker inputs and outputs

Inputs	Outputs
water (1.0kg; E_thermal=20600J)	brewed coffee (water=0.25kg; coffee_powder=0.01kg; E_thermal ∈ [22206J;24818J])
coffee_powder (0.25kg; E_thermal=20900J)	
grid (E_electrical=1000000J)	
cp_water=4180J/(kg*K)	

Presenting the problem to the Quantitative Function Structurer using the UPF and the Tempest planning engine, it outputs the following solution:
$\{add_material\ (src1 = water,\ src2 = coffee_powder,\ dst = brewed_coffee,\ amount1 = 1/4,$
$amount2 = 5764607523034235/576460752303423488 \approx 0.01)\};$
$\{guide_entity(x = grid)\};$
$\{transform_energy_electrical_to_thermal(x = grid,\ E_in = 1000000)\};$
$\{guide_entity(x = water)\};$
$\{split_energy_thermal(src = grid,\ dst = brewed_coffee,\ E_in = 24818)\}$

https://doi.org/10.1017/pds.2026.10437 Published online by Cambridge University Press

The resulting Function Structure is depicted in Figure 1 shown below.

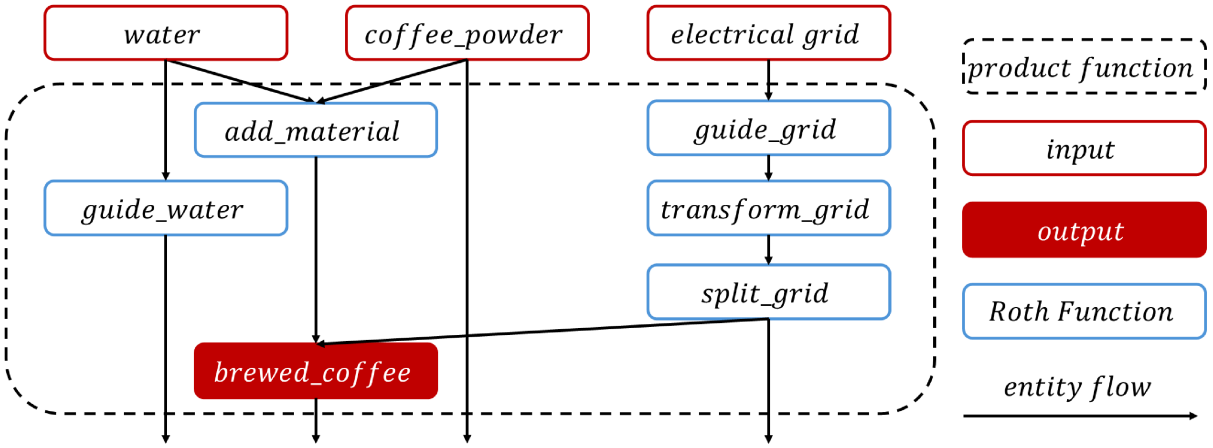


Figure 1. Derived function structure for the coffee maker

First, the desired masses of water and coffee powder are added together. Then the electricity is guided, before the entire electrical energy is transformed into thermal energy. The next sub function is the guidance of what is left of the water, followed by loading the brewed coffee with the appropriate amount of thermal energy. The final state entails the desired output of the brewed coffee along with the rest of the water, coffee powder and thermal energy.

4.2. FDM printer

The second case study is a minimal FDM printer. It complements the coffee maker as a suitable benchmark because it exercises also information alongside material and energy in one coherent instance. In contrast to the coffee maker, it also requires a material phase change. It thereby further stresses unit checks, conservation and interface compatibility across the entities. The overall function is to produce a two phased filament loaded with information. The quantified inputs and outputs are listed in Table 3.

Table 3. FDM printer inputs and outputs

Inputs	Outputs
filament (1.0kg; E_thermal=36000J)	filament ((liquid filament $\geq$ 0.05kg; E_thermal $\in$ [65000J;70000J]; I_analog=10), (liquid filament $\geq$ 0.8kg; E_thermal $\in$ [34000J;38000J]))
grid (E_electrical=1000000J)	
information (I_digital=10)	
cp_filament=1800J/(kg*K)	

Inputting the QFDP containing the above inputs and outputs in the Quantitative Function Structurer returns the following solution:
$\{Transform_material_filament_solid_to_liquid(x = filament,\ amount = 1)\};$
$\{Transform_information_digital_to_analog(x = information,\ I_in = 5761004643332338483/576460752303423488 \approx 10.00)\};$
$\{Guide_entity(x = grid)\};$
$\{Split_energy_electrical(src = grid,\ dst = filament1,\ E_in = 292733975779082240000/4278419646001971 \approx 68421.05)\};$
$\{Transform_information_digital_to_analog(x = information,\ I_in = 3602879701896397/576460752303423488 \approx 0.01)\};$
$\{Shape_material_filament_to_filament1_liquid(x = filament,\ amount = 900719925474099/4503599627370496 \approx 0.20)\};$
$\{Guide_entity(x = information)\};$
$\{Split_information_analog(src = information,\ dst = filament,\ I_in = 10)\};$
$\{Transform_energy_electrical_to_thermal(x = filament1,\ E_in = 292733975779082240000/4278419646001971 \approx 68421.05)\};$

https://doi.org/10.1017/pds.2026.10437 Published online by Cambridge University Press

$\{Add_material_filament_liquid_filament1_liquid(src1 = filament,\ src2 = filament,\ dst = filament1,\ amount1 = 3602879701896397/4503599627370496 \approx 0.80,\ amount2 = 900719925474099/4503599627370496 \approx 0.20)\}$
The resulting Function Structure is shown in Figure 2.

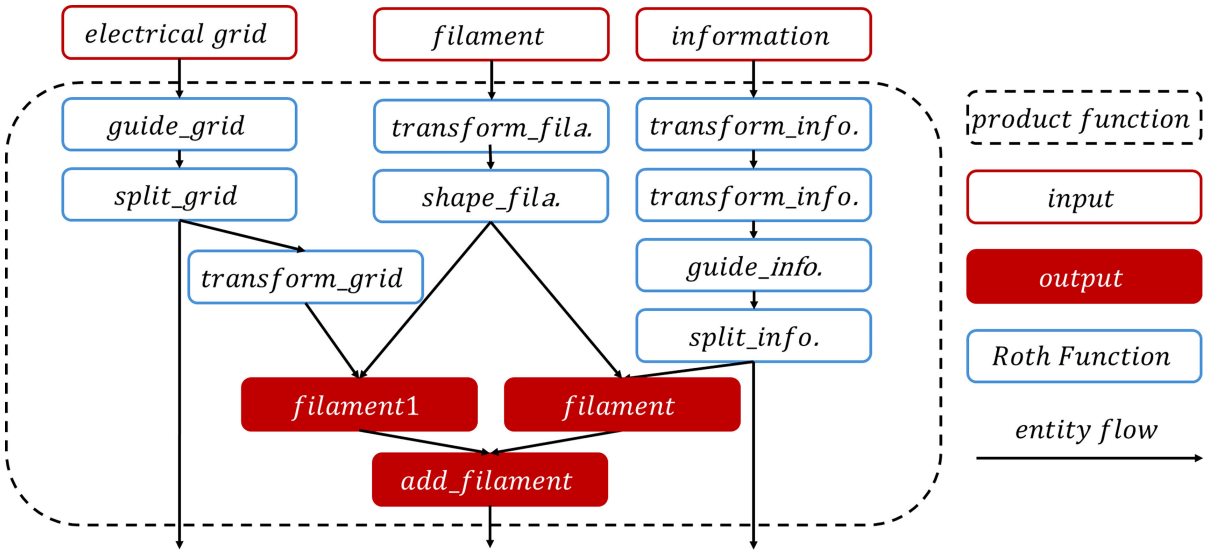


Figure 2. Derived function ftructure for the FDM printer

The filament is first transformed into a liquid state. Afterwards, the digital information is getting transformed into analog information. In a third step the electricity is being guided right before a part of it is being split off for usage in a, at this point empty, filament entity. The next function is to again transform an amount of information from digital to analog. Then, the appropriate amount of filament is shaped into filament1 leaving the rest in the filament entity. The information is being guided before it is split off with a part of it going into the filament. The energy for the filament1 is now being transformed into thermal energy. Finally, both filaments are being added back together to form the desired output.

4.3. Discussion

Both case studies satisfy the quantitative checks at the final state. Mass and energy targets lie within the specified bands and all unit and interface checks pass. The used solver works within exact rationales to avoid floating point errors. This leads to quantities appearing as long fractions in the solution. For better readability, the quantities were omitted in the resulting Function Structures. The assignments from Section 3.2 are reflected in the solutions. Shaping material serves as a bookkeeping label and whenever a phase label changes the necessary energy is supplied separately by an energy entity. This preserves Definition 8.

The solutions provided by the algorithm are sensitive not only to the problem specification, but also to the applied search heuristic. For the used solver the solutions are not guaranteed to be shortest. One topological order is reported and independent steps may run in parallel. The visualizations show one possible layout of the same Function Structure. For both case studies the Function Structures contain unnecessary functions, e.g. the guide functions for the coffee maker and FDM printer or the second transformation of information found in the FDM printer solution. To the best of the authors knowledge, there is currently no planner available which guarantees shortest-path-optimality for numerical planning problems with real-valued control parameters. Further, solutions depend heavily on the problem statement. If stated so, the solver might unintentionally opt for cooling off material to heat up other material, leading to unwanted results. The generated solutions obey physical laws by construction, which can rule out other intuitive but inconsistent steps. Once a particular state is known to be required at a specific location in the Function Structure, the overall product needs to be divided into appropriate subassemblies for which these intermediate steps then act as inputs or outputs. The runtime grows with the number of entities and allowed phases and theoretically with the size of the set QRC. Tightening the numeric bands for goals will reduce the admissible branches. The problems are formalized to be linear by the QRC, which aligns with the solver, i.e. an SMT backend.

https://doi.org/10.1017/pds.2026.10437 Published online by Cambridge University Press

The two case studies exercise a broad part of the framework. Together they activate five of the six function families, only omitting the explicit store function. They include cross-entity adapters and instantiate all three entity types, i.e. material, energy and information. Multiple constraint types are being covered, like mass and energy conservation and unit compatibility. They also show different routing patterns, e.g. heat provision or material co-location. Because the catalogue is the same across both instances and only the QFDP is different, the consistent success of checks for both case studies indicates that the method is not tied to a specific topology or parameter setting.

5. Conclusion

This paper makes three principal contributions, each corresponding to one of the posed RQs. It addresses **RQ1** in section 3.1 by formalizing the QFDP so that quantitative consistency can be asserted in the conceptual design phase and so that the problem is processable by known AI methods. Section 3.2 introduces a quantitative representation and composition scheme for mapping Roth Functions to a QRC in order to adapt for quantitative measures, which answers **RQ2**. The QFS algorithm introduced in section 3.3 finds correct solutions to the QFDP from formal problem statements. From these solutions, the Function Structures can directly be derived from. This directly satisfies **RQ3**.

The empirical results uphold the quantitative criteria introduced in the formalism. Mass and energy targets fall within the specified bands and all interfaces respect unit and type compatibility. Regarding the generalizability, both case studies demonstrate that the catalogue and validators operate consistently across different entity couplings. However, structural evaluation of concept design problems is limited. Publicly available and genuinely diverse decomposition problems are scarce and many variants share near identical Function Structures. So large scale evaluation would mostly retest the same topology. Consequently, carefully chosen cases were tested rather than a bulk run over near duplicates.

With quantified Function Structures, interfaces are specified with magnitudes and units, so mass and energy balances can be verified alongside entity type compatibility. This enables earlier plausibility assessments for design concepts, e.g. heating budget or material allocations and makes mismatches visible where they occur, not months later in the embodiment phase.

A limitation lies within the linear formalization, here phenomena with inherently nonlinear couplings fall out of scope. Examples include radiative heat transfer and fluid and flow losses for pumps and fans. The modelling choice, that material functions do not deal with thermal energy, shifts all thermodynamics into the energy entities, which is consistent for bookkeeping but different from some intuitive procedures. The proposed individual index sets are not intended to be exhaustive at this stage. While the authors assume that the provided sets will suffice for most decomposition tasks, certain cases may necessitate their individual extension.

Future work stems from the observed non minimal plans. They suggest a shift from feasibility focused solving to backends which are more optimized for the problem. Benchmarking different solver backends for set QFDPs will be a logic next step outside of the scope of this paper.

Quantified Function Structures change how problem statements need to be formulated for decomposition. Instead of vague targets, statements can specify known or wanted magnitudes, units, bands and interface types, reducing later iteration loops. Subsequent to functional decomposition, the search for solution principles can be driven by quantified requirements rather than informal fitting, e.g. selecting a heating solution principle that can deliver a stated energy band, narrowing down the search space. The effort of smoothing out the connection between these tasks in the conceptual design phase remains.

References

Bender, B., & Gericke, K. (Eds.). (2021). *Springer eBook Collection. Pahl/Beitz Konstruktionslehre: Methoden und Anwendung erfolgreicher Produktentwicklung* (9. Auflage). Springer Vieweg. https://doi.org/10.1007/978-3-662-57303-7

Cimatti, A., Griggio, A., Mover, S., Roveri, M., & Tonetta, S. (2022). Verification modulo theories. *Formal Methods in System Design*, 60(3), 452–481. https://doi.org/10.1007/s10703-023-00434-x

Ehrlenspiel, K., & Meerkamm, H. (2017). *Integrierte Produktentwicklung: Denkabläufe, Methodeneinsatz, Zusammenarbeit (6., überarbeitete und erweiterte Auflage). Hanser eLibrary*. Carl Hanser Verlag. https://doi.org/10.3139/9783446449084

Gero, J. S. (1990). Design Prototypes: A Knowledge Representation Schema for Design. *AI Magazine*, 11(4), 26. https://doi.org/10.1609/aimag.v11i4.854

https://doi.org/10.1017/pds.2026.10437 Published online by Cambridge University Press

Haddad, M.-S., & Seibel, A. (2025). Functional decomposition of technical products based on large language models and Monte Carlo tree search. *Proceedings of the Design Society*, 5, 1913–1922. https://doi.org/10.1017/pds.2025.10205

Heesch, R., Cimatti, A., Ehrhardt, J., Diedrich, A., & Niggemann, O. (2024). A Lazy Approach to Neural Numerical Planning with Control Parameters. In U. Endriss (Ed.), *Frontiers in Artificial Intelligence and Applications Series: v.392. Ecai 2024: 27th European Conference on Artificial Intelligence, 19-24 October 2024, Santiago de Compostela, Spain - Including 13th Conference on Prestigious Applications of Intelligent Systems (PAIS 2024)* (1st ed.). IOS Press Incorporated. https://doi.org/10.3233/FAIA241000

Heesch, R., Ehrhardt, J., & Niggemann, O. (2024). Integrating Machine Learning into an SMT-Based Planning Approach for Production Planning in Cyber-Physical Production Systems. In *Communications in Computer and Information Science: Vol. 1948. Artificial Intelligence. Ecai 2023 International Workshops* (1st ed. 2024, Vol. 1948, pp. 318–331). Springer Nature Switzerland; Imprint Springer. https://doi.org/10.1007/978-3-031-50485-3_33

Kitamura, Y., & Mizoguchi, R. (2003). Ontology-based description of functional design knowledge and its use in a functional way server. *Expert Systems with Applications*, 24(2), 153–166. https://doi.org/10.1016/S0957-4174(02)00138-0

Koller, R., & Kastrup, N. (1998). *Prinziplösungen zur Konstruktion technischer Produkte* (2., neubearbeitete Auflage). *Springer eBook Collection Computer Science and Engineering*. Springer Berlin Heidelberg.

Micheli, A., Bit-Monnot, A., Röger, G., Scala, E., Valentini, A., Framba, L., Rovetta, A., Trapasso, A., Bonassi, L., Gerevini, A. E., Iocchi, L., Ingrand, F., Köckemann, U., Patrizi, F., Saetti, A., Serina, I., & Stock, S. (2025). Unified Planning: Modeling, manipulating and solving AI planning problems in Python. *SoftwareX*, 29, 102012. https://doi.org/10.1016/j.softx.2024.102012

Mikes, A., Edmonds, K., Stone, R [Robert], & DuPont, B. (2021). AutoFunc: A Python package for automating and verifying functional modeling. *Journal of Open Source Software*, 6(58), 2362. https://doi.org/10.21105/joss.02362

Mohammed, O. M., & Shammari, A. Z. M. (2021). A Procedural Algorithmic Approach for Functional Structure Construction. *Engineering, Technology & Applied Science Research*, 11(1), 6819–6832. https://doi.org/10.48084/etasr.4012

Panjkovic, S., & Micheli, A. (2023). Expressive Optimal Temporal Planning via Optimization Modulo Theory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(10), 12095–12102. https://doi.org/10.1609/aaai.v37i10.26426

Panjkovic, S., & Micheli, A. (2024). Abstract Action Scheduling for Optimal Temporal Planning via OMT. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(18), 20222–20229. https://doi.org/10.1609/aaai.v38i18.30002

Rosenthal, P., Demke, N., Mantwill, F., & Niggemann, O. (2024). Plan-Based Derivation of General Functional Structures in Product Design. In *2024 IEEE 7th International Conference on Industrial Cyber-Physical Systems (ICPS)* (pp. 1–8). IEEE. https://doi.org/10.1109/ICPS59941.2024.10640039

Roth, K. (2000). *Konstruieren mit Konstruktionskatalogen: Band 1: Konstruktionslehre (3. Auflage, erweitert und neu gestaltet)*. Springer. https://doi.org/10.1007/978-3-642-17466-7

Roth, K. (2001). *Konstruieren mit Konstruktionskatalogen: Band 2: Kataloge* (3. Auflage, mit wesentlichen Ergänzungen). *Springer eBook Collection Computer Science and Engineering*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-17467-4

Tensa, M., Edmonds, K., Ferrero, V., Mikes, A., Soria Zurita, N., Stone, R [Rob], & DuPont, B. (2019). Toward Automated Functional Modeling: An Association Rules Approach for Mining the Relationship between Product Components and Function. *Proceedings of the Design Society: International Conference on Engineering Design*, 1(1), 1713–1722. https://doi.org/10.1017/dsi.2019.177

Umeda, Y., Ishii, M., Yoshioka, M., Shimomura, Y., & Tomiyama, T. (1996). Supporting conceptual design based on the function-behavior-state modeler. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 10(4), 275–288. https://doi.org/10.1017/S0890060400001621

Verein Deutscher Ingenieure (1997). *VDI-Richtlinie 2222 Blatt 1:* Konstruktionsmethodik Methodisches Entwickeln von Lösungsprinzipien. VDI-Verlag.

Verein Deutscher Ingenieure (2019). *VDI-Richtlinie 2221 Blatt 1:* Entwicklung technischer Produkte und Systeme Modell der Produktentwicklung. Beuth Verlag.

Verein Deutscher Ingenieure Verband der Elektrotechnik Elektronik Infomartionstechnik (2005). *VDI/VDE-Richtilinie 3682:* Formalisierte Prozessbeschreibungen. VDI-Verlag.

https://doi.org/10.1017/pds.2026.10437 Published online by Cambridge University Press