A CATEGORY OF DESIGN STEPS

Breiner, Spencer (1); Subrahmanian, Eswaran (1,2)
1: National Institute of Standards and Technology (NIST), United States of America; 2: Carnegie Mellon University, United States of America

Abstract

In this paper we critique the interpretation of Concept-Knowledge theory in terms of logical decidability. We argue instead that concepts and knowledge should be regarded as logical language and axioms, the two main components of a logical theory. Based on this proposal and using tools from the mathematics of category theory, we propose a category of logical designs to act as a formal interpretation for the dynamic operators which define the design processes of C-K theory.

Keywords: C-K theory, Category theory, Product modelling / models, Knowledge management, Innovation

Contact:
Dr. Spencer Breiner
National Institute of Standards and Technology (NIST)
ITL
United States of America
spencer.breiner@nist.gov

Please cite this paper as:
Surnames, Initials: *Title of paper*. In: Proceedings of the 21st International Conference on Engineering Design (ICED17), Vol. 7: Design Theory and Research Methodology, Vancouver, Canada, 21.-25.08.2017.

1 INTRODUCTION

Concept-Knowledge (C-K) theory is a semi-formal model of design which aims to explicitly account for the process of innovation, characterized by the creation of new, unexpected products and methods. The formal aspect of the theory arises from a connection with symbolic logic, in which both concepts and knowledge are regarded as logical propositions, differentiated by a logical property called decidability. C-K theory then analyses the processes of design in terms of four operators which embody navigation through the space of possible (and impossible) designs (Hatchuel & Weil, 2003).

Our goal here is to argue *against* this interpretation, and to provide a better one. The existing interpretation suffers from several problems, leading to a mismatch between the usage of C-K theory and the mathematical structure of logical systems. Our proposal is simpler and more direct, associating concepts with symbolic language and knowledge with axioms, the two key components of a first-order logical theory.

In order to formalize the dynamic aspects of C-K theory, we introduce some ideas from the theory of categories, a mathematical discipline which studies abstract composable processes. A well-known connection between categories and logic (Makkai & Reyes, 1977) allows us recast our C-K designs as categorical structures. We can then use mappings between categories, called functors, in order to interpret the design operators of C-K theory.

2 C-K THEORY AND ITS LOGICAL INTERPRETATION

Concept-Knowledge (C-K) theory is a theory of design which aims to explicitly account for innovation. It arises from the observation that traditional problem-solving theories of design focus on decision-theoretic models which are insufficient to explain how new objects, outside a given array of choices, can be generated.

C-K theory postulates two spaces—the “Concept Space” and the “Knowledge Space”—which we explore in the process of design (Figure 1). In particular, four "operators" allow us to modify these two spaces. The internal operators are *concept expansion* (C→C) and *knowledge expansion* (K→K). The first allows us to brainstorm and create new concepts without regard to feasibility, while the second allows us to analyze and recombine existing knowledge to achieve new insights. The essence of design, though, lies in the external operators, *conjunction* (C→K) and *disjunction* (K→C). These allow for a back and forth interaction between the two spaces, testing concepts to generate new knowledge and studying knowledge to suggest new concepts.

Hatchuel & Weil (2003) present an example in the development of a magnesium-carbon dioxide (Mg-CO2) engine for use in a robotic Mars mission. The initial design question is whether an Mg-CO2 could fuel the engine on a Mars mission; they might be lighter than traditional engines because the CO2 reactant can be harvested from Mars' atmosphere (K→C). Researchers conducted landed mass calculations (a proxy for launch costs) for both types of engine, finding that the Mg-CO2 engine was not viable (C→K). Rather than ending their analysis here, the researchers introduced several characteristics (e.g., planned/unplanned) which a mission might satisfy (C→C). By analyzing the specific requirements in these various situations, the researchers were able to identify a range of optimal performance criteria for various niche functions (C→K). Comparing these to the known characteristics of Mg-CO2 engines, they were able to identify useful applications in certain unplanned and emergency situations (K→K).

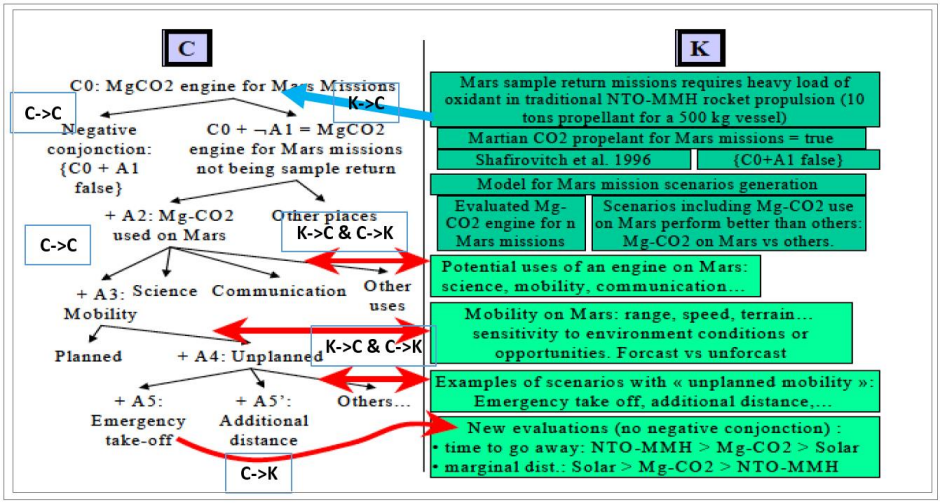

Figure 1. C-K operators in Mg-CO2 engine

C-K theory aims to not only structure innovation in design, but also to formalize it, through a connection to symbolic logic (an alternative formalization in terms of action logic can be found in Salustri (2005)). Hatchuel & Weil propose such an interpretation in (2003), which we will refer to as the HW interpretation. Under this approach we can think of both concepts and knowledge as propositions (true/false statements). A proposition counts as knowledge if it is *decidable*, which is to say that it can be proved definitively true or false. By contrast, concepts in the HW interpretation are propositions whose truth cannot be ascertained from the existing base of knowledge.

Furthermore, Hatchuel, et al. (2007) argue that design seeks to take an undecidable concept such as "Mg-CO2 engine for Mars mission" and move it into the realm of knowledge by demonstrating (or refuting) its feasibility. They claim that, in this respect, design can be viewed as a generalization of *forcing*, a mathematical technique from abstract set theory which can be used to produce new universes of sets satisfying certain desirable characteristics, and that this provides "a new scientific identity" for design theory.

3 THE NEED FOR A NEW INTERPRETATION

While we agree that C-K theory is useful in practice, as a framework for organizing innovation in design, we believe that its formal foundations are a little shaky. In this section, we present three arguments questioning the HW interpretation of C-K theory suggested by Hatchuel, et al. (2003). We find that their approach using decidability fails to line up with the language it uses, the mathematics which grounds it, and the practice which it aims to describe.

The first hint that the received view might not capture the appropriate relationship between C and K is linguistic. If a concept is a proposition whose truth value is unknown, then as I gain information concepts disappear to be replaced by knowledge.

But this is wrong. My concept of "a unicorn" does not stop being a concept simply because I know that such beasts are mythical; "a tree" is still a concept though we know that they exist. Indeed, the idea that concepts are propositions stems from an implicit conflation between a concept C and a certain proposition related to it: "There is an x which is a C".

The problem is that, if C should be associated with any proposition, it should be the following simpler statement: "x is a C". In terms of symbols, it is the difference between these two formulas:

$$\big(\exists x.\, C(x)\big) \nLeftrightarrow C(x)$$

The formula on the left must be either true or false (though we might not know which one). The formula on the right simply does not have a truth value; the statement "x is a C" is neither true nor false because its value depends on another quantity x. Therefore, the formula on the left defines a *truth function* mapping each x to a truth value. We say that x is a free variable in C.

The second clue which argues against the HW interpretation is mathematical. Although most descriptions of C-K theory are couched in terms of propositions, it is clear that the logician's "propositional calculus" is not up to the demands. This is because the propositional calculus is the study of truth values rather than truth functions, and therefore incapable of handling the sort of distinctions

raised above. Instead we will need something more like the "predicate calculus" which studies both individuals and the properties they exhibit. We do not mean to suggest that C-K theorists are unaware of this necessity; rather, our concern is that the use of propositional language oversimplifies our informal intuitions. For simplicity, we will focus classical first-order logic, but the discussion applies much more broadly, both to simpler logics (e.g., algebraic or intuitionistic) and to stronger ones (e.g., multi-sorted programming languages or higher-order logic).

This shift, from the logic of truth to the logic of individuals, is important for the following reason: such logical theories require a specified symbolic language before one can even talk about propositions. This raises concerns when we are told that concepts and knowledge are propositions, but the language in which they are written is left unmentioned. More tellingly, even precise presentations like Kazakçi (2009) take the logical language as a given, with no indication of how it relates to (or could be derived from) a given design problem.

The third and most nettlesome problem with the HW interpretation is methodological: it simply cannot be applied in practice. This is not to say that C-K theory is not useful. Far from it: the references in the section 2 make it clear that C-K theory can be quite valuable. However, that value seems to be largely independent of the theory's logical interpretation.

This disconnect takes several forms. Most apparent, perhaps, is the fact that C-K based investigations typically proceed in natural language and are rarely, if ever, given a formal translation. From this it is clear that the formal interpretation is not necessary for (and perhaps orthogonal to) the typical applications of C-K theory in design.

The connection with forcing is even more tenuous. We are unaware of any attempt to analyze a real-life design case study through the lens of forcing. Forcing constructions have several well-defined components. In a design context, what is the relevant "ground model" of set theory to base our analysis? How would one identify or construct a generic filter? Where and how can we recover a new design from the extension model which is produced from mathematical forcing?

And what are the implications of this new identity for design practice? "[W]e can better understand why Design practice can be *disconcerting*, *controversial* and *stressful*" (emphasis in original). But if this is all that the formalism grants us, then it may not be worth the bother.

4 AN ALTERNATE INTERPRETATION

In this section, we will present and analyze an alternative logical interpretation for C-K theory: we associate the concepts of a design with a symbolic language, and its knowledge with the logical axioms. In this way, each design stage is formally represented by a logical theory, which we might think of as the theory of the artefact.

This new proposal has several nice features. First, it is grounded in criticisms in the previous section and so meets several of those objections. Second, it is much simpler than the HW interpretation, and ties design to more fundamental logical objects. It also generalizes the HW interpretation, so that any existing examples can be adapted to fit our proposal.

More specifically, we start with some basic concepts in the language $\mathbb{L}$. We can use logical connectives to combine these basic concepts into more complicated formulas. The collection of all these formulas (or all those we have considered thus far) determines our set of concepts C. Similarly, we can think of the axioms as basic knowledge, but these can be combined through proof to yield new theorems. The knowledge base K should be thought of as the set of such proven or provable theorems.

First of all, notice that we avoid the mathematical and linguistic concerns of the previous section. Mathematically, the logical language which was given short shrift in the HW interpretation is now absolutely central, forming one of the twin pillars of C-K theory. Linguistically, unicorns and trees remain concepts for us, regardless of what we might know about them.

Next consider the extent of the HW interpretation. Their concepts--the undecidable propositions--are, in particular, written as formulas. This means that all of concepts of the decidability interpretation are also ours, though the reverse it not true. As for knowledge, this is associated with the propositions known to be true or false, while we associate them with the propositions known to be true. However, this is only a technical distinction; a proposition is provably false if and only if its negation is probably true. This is obvious in classical (Boolean) logic by the rule of double negation, but also holds in more general intuitionistic contexts (e.g., open-world semantics). Thus the C and K in our new interpretation are strictly more inclusive than the alternative.

The most important advantage of our new interpretation is its simplicity. Language and axiom sit at the very core of logic; they are the first concepts introduced in a formal presentation. Thus, our interpretation embeds design at the heart of logical analysis. In addition, logical languages and axioms are finite (or finitely generated), so that we can calculate and compute with them.
Contrast this with undecidability. Though of undeniable importance in modern logic, undecidability is a much more sophisticated idea, requiring explication of formal proof before it can even be defined. Moreover, constructions like forcing (used to verify undecidability) are inherently infinitary, limiting their application in real-world domains. The simpler interpretation we propose should make it easier to formalize practical applications of C-K theory in a logical context. Building up a base of such examples and case studies is a critical area for future research.

5 CATEGORY THEORY

In this section, we introduce some ideas from a branch of mathematics called category theory with an eye towards formalizing the dynamical component of C-K theory: the four design steps C→C, C→K, K→C, K→K.
More generally, we believe that CT provides the language and theory to support new tools for building and managing the sorts of logical models that design requires. It's application here is quite tempting: CT is a theory of arrows. A proper introduction to CT is beyond the scope of this paper and we omit many details (in particular, discussion of identity arrows). See Spivak (2014) for a good introduction.
A category $\mathcal{C}$ contains two types of entities: objects $(X, Y, Z, ...)$ and arrows $(f: X \to Y, Y \overset{g}{\to} Z, ...)$. Intuitively, objects can be thought of as states or types while arrows are processes or functions which operate on them. Every arrow has an input and an output (called the *domain* and *codomain*), indicated by writing $f: X \to Y$. So far this is just a directed graph. The critical feature of a category is a composition operation that allows us to chain the output of one arrow to the input of another: $f.g$ means ``do f, then g."
The most familiar example of a category is **Sets**, the category of sets and functions; given two functions $f: X \to Y$ and $g: Y \to Z$, the composite function $f.g$ is defined by the usual formula $g(f(x))$. Many other ``concrete" categories are constructed by enhancing sets and functions with additional structure. **Vect**, the category of vector spaces and linear maps, is a typical example. In other categories, the objects and arrows may be completely abstract, lacking any semblance of internal structure. For example, any directed graph defines a category where objects are nodes and arrows are paths of edges; to compose we just concatenate paths at their endpoints.
Although the notion of a category is exceedingly abstract, this is necessary for its application across such a wide range of disciplines. Table 1 provides a suggestion of the breadth of categorical analysis. Composable processes occur throughout mathematics and the sciences, and this means that all can be analyzed through the lens of CT.
Our analysis here will rely on the last two entries in the chart. One strand comes from the categorical approach to formal logic, allowing us to represent logical theories as categories. The other is more radical: CT is self-referential. This means that we can use the machinery of category theory to structure and study categories themselves, and hence any of the other entities from physics and logic and the rest which can be rendered in these terms.

Table 1. Categories in familiar contexts

In …	**…the objects are…**	**… and the arrows are…**
Physics	Systems	Processes
Computer Science	Datatypes	Computable functions
Data Science	Tables	Foreign keys
Statistics	Probability spaces	Stochastic kernels
Logic	Formulas	Definable functions
Category theory	Categories	Functors

6 CATEGORIES IN LOGIC

So far we have argued for a new interpretation of the concepts and knowledge in C-K theory, as the as the language and axioms of a logical theory. This interpretation is, in a sense, static: at any particular stage of design, we can formally represent the state of the design as a logical theory. Now we would like to use category theory to capture the dynamical aspects of design steps.

Our strategy is to use first represent logical theories as categories, and then use the mappings of categories, called *functors*, to represent mappings between theories. In fact, functors are already important from a static perspective, to capture the relationship between language and theory. We can only sketch the details her, but see Makkai and Reyes (1977) for a detailed presentation.

The first step is to turn a logical theory $\mathbb{T}$ into a category $\mathcal{T}$. Speaking (very) roughly, an object of $\mathcal{T}$ is a formula $\varphi(x)$ (open, with free variable(s) x) and an arrow $\sigma: \varphi(x) \to \psi(y)$ is a definable function (total, single-valued relation) $\sigma(x, y)$. Functional relations are closed under the composition of relations, $\exists y.\, \sigma(x, y) \wedge \tau(y, z)$, so we can take this as a definition of composition in $\mathcal{T}$.

An important special case is $\mathcal{L}$, the category generated by the empty theory (no axioms) and associated with the language $\mathbb{L}$ itself. This category has the special property of being *freely generated* which makes them easier to analyze. With the same language, $\mathcal{T}$ and $\mathcal{L}$ have the same formulas and so the same objects. However, $\mathcal{T}$ may have arrows that do not occur in $\mathcal{L}$, and arrows which are distinct in $\mathcal{L}$ may be provably equal in $\mathcal{T}$.

As indicated in the last section, categories themselves are the objects of a (meta-)category **Cat**; the arrows between them are called *functors*. Furthermore, the logical connectives and quantifiers ($\wedge$,$\vee$, $\neg$, $\exists$, etc.) correspond to structural features of a category such as Cartesian products ($\wedge$), image factorization ($\exists$), etc. The particular case of classical, first-order logic corresponds to Boolean, coherent structure, and we restrict our attention to *structure-preserving* functors.

Definition. A functor $F: \mathcal{C} \to \mathfrak{D}$ sends objects to objects and arrows to arrows in such a way that

- $h: C \to C'\$$ in $\mathcal{C} \implies F(h): F(C) \to F(C')$ (domain/codomain coherence)
- $F(h.k) = F(h).F(k)$ (compositionality)
- $F(\varphi \wedge \psi) = F(\varphi) \wedge F(\psi), F(\exists x.\, \varphi) = \exists x.\, F(\varphi), \ldots$ (structure preservation)

One good class of examples define the set-theoretic semantics of first-order logic. For simplicity, consider a language $\mathbb{L}$ with one sort X and one relation $R(x, x')$. An $\mathbb{L}$-structure M consists of an *underlying set* $|M|$ and an *extension* $R^M \subseteq |M|^n$. Given this information, we can define the extension of any formula recursively in terms of intersection, union, direct image, etc. This yields a functor $\mathcal{L} \to$**Sets** sending each formula to its extension. The recursive specification merely recasts the structure preservation requirement, and composition is preserved because it is defined from the logical structure. The relationship between language and theory can also be expressed as a functor $q: \mathcal{L} \to \mathcal{T}$.

Furthermore, the satisfaction relation between the structure M and the theory $\mathbb{T}$ can be expressed in terms of functors. M satisfies $\mathbb{T}$ ($M \vDash \mathbb{T}$) if and only if the functor M factors through $\mathcal{T}$, i.e. there is another structure-preserving functor $\bar{M}: \mathcal{T} \to$**Sets** such that $M = q.\bar{M}$.

One major advantage of the categorical approach is that it provides a precise, detailed language for analyzing functorial relations (see Makkai and Reyes (1977) for definitions). For example, axiomatic extensions $\mathbb{T} \subseteq \mathbb{T}'$ (in the same language) determine a class of functor $\mathcal{T} \twoheadrightarrow \mathcal{T}'$ which are *(essentially) surjective on objects* and *full on subobjects*. These maps, indicated by double-headed arrows, are called *quotients*; they represent a strong categorical version of surjectivity which captures the notion of "same language" (up to definitional equivalence).

In contrast, linguistic extensions $\mathbb{L} \subseteq \mathbb{L}'$ induce functors $\mathcal{L} \rightarrowtail \mathcal{L}'$ which are often *conservative* (equivalently in the context, *faithful*). More generally, conservative funtors $\mathcal{T} \rightarrowtail \mathcal{T}'$ provide a categorical analog for injectivity, and indicates that the new theory does not collapse or identify any distinctions which could be made in the original. And just as any function can be factored as a surjection followed by an injection, any structure-preserving functor can be factored as a quotient followed by a conservative map. Thus, CT provides a language for lifting the structural methods in set theory to more sophisticated contexts.

7 THE CATEGORY OF DESIGN STEPS

With the categorical machinery of the last section in hand, we are ready to provide a formal definition of a category of C-K designs. We then consider the four design steps identified by C-K theorists, and show how these fit into our proposed definition. In some sense, we can regard these as the atomic operations from which more complicated design steps are constructed. To justify this claim, we prove an easy factorization theorem which asserts that any design step can be factorized into a purely linguistic extension followed by a purely axiomatic extension.

Definition. The category $\mathfrak{D}$ of C-K designs (in a classical, first-order ambient logic) is defined as follows:

- An object $D \in \mathfrak{D}$ consists of:

  a functor $q: \mathcal{L} \twoheadrightarrow \mathcal{T}$ where,
  - $\mathcal{L}$ and $\mathcal{T}$ are structured (Boolean, coherent) categories,
  - $\mathcal{L}$ is freely generated by a first-order logical language,
  - q is structure-preserving and a quotient.
- An arrow $F: D \to D'$ (figure 2) consists of
  - a pair of functors $F_L: \mathcal{L} \to \mathcal{L}'$ and $F_T: \mathcal{T} \to \mathcal{T}'$ where
  - both F_L and F_T are structure preserving, and
  - the resulting square commutes: $F_L.q' = q.F_T$.
- Composition in $\mathfrak{D}$ is defined by composition of functors:
  - $(F.G)_L = F_L.G_L$ and $(F.G)_T = F_T.G_T$.

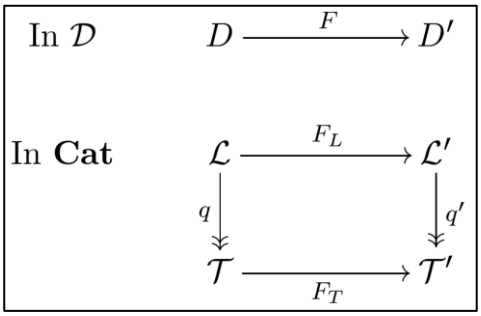

Figure 2. An arrow in $\mathfrak{D}$

In a map of designs, the language functor $\mathcal{L} \to \mathcal{L}'$ tells us how to interpret the basic language of the initial design in terms of (possibly complex) formulas in the new language. The map of theories $\mathcal{T} \to \mathcal{T}'$ ensures that the axioms of the new design entail the axioms of the original design, relative to the interpretation provided by F_L. In fact, the theory functor is completely determined by the language functor; for a given F_L there may not be an associated F_T (in which case there is no map of designs), but if there is such a map it is unique. This follows from the fact that q is a quotient functor.

Now we would like to examine the relationship between our category of designs and the C-K theorist's four design steps: C→C, C→K, K→C, K→K.

First consider the C→K design step. When we first introduce a new concept into our C-K model, the next step will be to pin down the new concept by specifying what we already (implicitly) know about it. For example, we might have a concept of parenthood as a binary relation $P(p, c)$ where $p, c: Person$. As it stands, this is a purely linguistic/conceptual structure, but we also know things about parenthood, such as the fact that every child has a parent: $\forall c \exists p. P(p, c)$.

In terms of our formal representation, C becomes $\mathcal{L}$, K becomes $\mathcal{T}$, and the C→K step becomes the quotient functor $q: \mathcal{L} \to \mathcal{T}$. Thus, under our proposed interpretation, a C→K step is already present in *every* design stage. Once we understand concepts as purely linguistic, without connection to (un)decidability, we recognize that concepts must *always* be supplemented with knowledge. Formally speaking, this means that the C→K design step is essentially trivial, acting by the identity at both the linguistic and theoretical levels (see Figure 3).

Next consider a knowledge extension K→K. According to our rubric, this should correspond to a functor $\mathcal{T} \to \mathcal{T}'$. Such a functor fits neatly into a design arrow in which the linguistic functor is the identity. Most often, this results from an extension of theories $\mathbb{T} \subseteq \mathbb{T}'$ through the addition of one or more axioms (written in the same language). More generally, we do not require axiomatic extension, so long as the original axioms are entailed by the new. For example, we might specify in $\mathbb{T}'$ that a child has exactly two (biological) parents, from which our first axiom would follow.

The analysis of concept expansion appears similar, but hides an important difference. We expect that a C→C design step should correspond to a functor $\mathcal{L} \to \mathcal{L}'$. Again, we can define such a functor from either a simple linguistic extension $\mathbb{L} \subseteq \mathbb{L}'$, or by sending each piece of the initial language to some formula of the latter. If we are working in a purely conceptual setting, with no axioms, then we can regard this as a design step in which the vertical quotients are trivial. However, if our initial state is a full design with axioms, then this conceptual expansion will leave us in an inconsistent state. After the

conceptual expansion we are left with a diagram $\mathcal{T} \twoheadleftarrow \mathcal{L} \to \mathcal{L}'$, without enough information to attach a theory to the extended language.

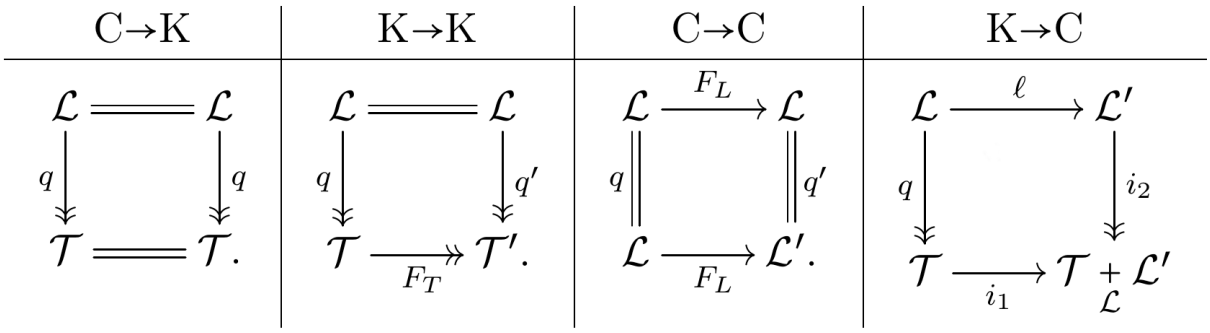

Figure 3. C-K design operators as arrows in $\mathfrak{D}$

The path to a solution passes through the fourth operator K→C. According to our proposal, we should think of a K→C design step as a functor $\mathcal{T} \to \mathcal{L}'$. Unfortunately, such a functor typically cannot exist, as it would require proving the axioms of $\mathcal{T}$ without the use of any axioms in $\mathcal{L}$'. At best we could expose the original language $\mathcal{L}$ and define an interpretation $\mathcal{L} \to \mathcal{L}'$, at which point we are again in the problematic state $\mathcal{T} \twoheadleftarrow \mathcal{L} \to \mathcal{L}'$.

From this data we can construct a *pushout*; this is a categorical operation analogous to a union in the category of categories. This allows us to construct one new object and two new arrows (one of which is a quotient) as shown in figure 3. Omitting a precise definition of the pushout, we can say that it is, roughly, the smallest theory which (i) interprets the language $\mathcal{L}'$, (ii) satisfies the axioms of $\mathcal{T}$, and (iii) which aligns the two interpretations of $\mathcal{L}$ inside $\mathcal{T}$ and $\mathcal{L}$', respectively.

Though the four C-K design steps can, to some degree, be embedded in our category of designs, some of them fit a bit awkwardly. The C→K step, though important to the notion of a design *stage*, is essentially trivial as a design *step*. The C→C expansion makes sense as an associate between languages, but does not yield a full design step at the level of theories. Thus, from the functorial perspective, it appears the source of a design step is less important than its target. In other words, we should pay careful attention to what is changed in a design step (language, theory or both), but less to what triggered the step.

To this end, we can define two important classes of *purely linguistic* and *purely theoretical* design steps. The former are constructed from a linguistic functor and a pushout operation, while the latter consist of theoretical functors without modification to the language. Furthermore, we have the following easy theorem, which follows more-or-less directly from the definition of the pushout:

Theorem. Any design step $F: D \to D'$ can be factored into a purely linguistic step followed by a purely theoretical step. Moreover, the factorization is unique up to categorical equivalence of the theoretical factor.

Proof. Given the design step F, we first apply the pushout operation to the linguistic functor F_L and the initial quotient q. This allows us to build a purely linguistic extension, the left-hand square (i) in figure 4. The pushout is unique up to equivalence of categories. Now we can apply the universal mapping property of the pushout (Spivak, 2014) to the theoretical map F_T and the latter quotient q'. This allows us to construct (uniquely up to natural isomorphism) the dashed arrow drawn in Figure 4. The resulting square (ii) is a purely theoretical extension, completing the proof.

8 CONCLUSION

In this paper, we argued against the HW interpretation of C-K theory based on several criteria: a linguistic mismatch, a technical omission and a (lack of) pragmatic application. Instead, we associate logical designs with the language and axioms of a logical theory, which we may think of as a ``theory of the object''. Based on this association, and borrowing tools from categorical logic, we defined a category of design processes, and identified interpretations for the C-K design operators inside this category.

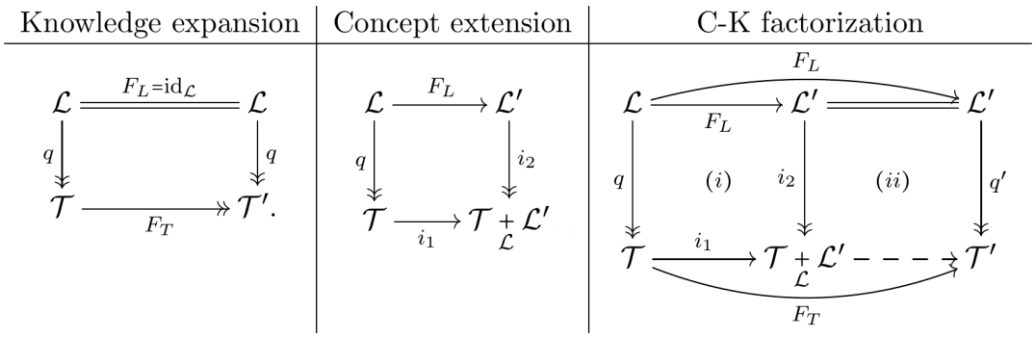

Figure 4. The factorization of a design step

This is only a first step towards a true formalization of C-K theory, and this opens up many avenues for future research. Perhaps most pressing among these is the formal modeling of actual case studies, so that we can see if our interpretation truly matches up with the actual design processes of C-K theory. After all, our main complaint for the HW interpretation was that it has not been very useful in practice. Thus, the development of case studies would both validate our proposed interpretation and identify areas where it needs further elaboration.

There is also more theory to be developed. The categorical mathematics introduced in this paper has direct bearing on the phenomena of expansion, and the creation of new products, which C-K theory was designed to explain. In particular, CT has been used to analyze the duality between logical theories and their instances. Under this duality, quotient functors (axiomatic expansions) correspond to the narrowing of semantic possibilities, as in classical decision theory, but the faithful functors generated by linguistic extension lead to precisely the sort of expansion recognized by C-K theorists.

Another area for future consideration concerns the genesis of design processes, which fit into our category of designs less easily than the effects of those processes. We believe that C-K theory may conflate two different types of knowledge, axiomatic and semantic/observational, and that distinguishing between these may lead to a better understanding of design inspiration.

REFERENCES

Hatchuel, A. and Weil, B. (2003), "A new approach of innovative design: an introduction to C-K theory", *ICED 2003*, Stockholm, Aug. 2003, Design Society, Paris, pp. 109-110 (executive summary).

Hatchuel, A. and Weil, B. (2007), "Design as forcing: deepening the foundations of C-K theory", *ICED 2007*, Paris, July 2007, Design Society, Paris, pp. 447-459.

Kazakçi, A. (2009), "A formalization of C-K design theory based on intuitionist logic," *ICORD 09*, Bangalore, Jan. 2009, Design Society, Paris, pp. 499-507.

Makkai, M. and Reyes, G. E. (1977), *First order categorical logic*, Springer, Berlin

Salustri, F. A. (2005), "Representing C-K theory with an action logic", *ICED 2005*, Melbourne, Aug. 2005, Engineers Australia, p. 1873.

Spivak, D. I. (2014), *Category theory for the sciences*, MIT Press, Cambridge.