INTERNATIONAL DESIGN CONFERENCE - DESIGN 2010
Dubrovnik - Croatia, May 17 - 20, 2010.

A FORMAL ACCOUNT OF THE DUAL EXTENSION OF KNOWLEDGE AND CONCEPT IN C-K DESIGN THEORY

L. Hendriks and A.O. Kazakci

Keywords: C-K design theory, formalization, design logic, design reasoning, dual expansion

1. Introduction

The aim of the present work is to contribute to the formalization of C-K design theory. C-K design theory (CKT) is an abstract theory of design providing a high level, general description of design reasoning [Hatchuel and Weil 2003, 2007, 2009]. The theory claims that any creative design reasoning is based on the distinction and interaction of two kinds of entities: *concepts* and *knowledge*. In CKT, knowledge is constituted by any piece of information (or proposition) that can be given a logical status by the designer (i.e. the designer can affirm that the proposition is true or false). Concepts, on the other hand, are novel and creative propositions that, at the moment of their creation, cannot be declared as true or false. Design progresses by building and expanding concepts in C space (that contains concepts) using knowledge from K space (that contains known propositions). Since concepts represent new ideas, it is often the case that new knowledge is required to continue further elaboration: K space is expanded by acquiring new knowledge. Under this perspective, a *design process* is the result of the *interaction and dual expansion of concepts and knowledge* [Hatchuel and Weil 2003, 2007, 2009].

1.1 Formalization of CKT: an open issue

CKT and derived methods have seen many successful industrial applications [Hatchuel et al. 2004]. The theory has also contributed to different literatures ranging from management of innovation to economy of innovation [Le Masson et al. 2006]. Contrastingly, although CKT is a theory of reasoning, *its formal and logical aspects have yet to be explored in depth*. Despite the fact that, in its various presentations, CKT refers to many formal theories and approaches (like Set Theory [Hatchuel and Weil 2003], Category Theory [Hatchuel and Weil 2002], Forcing [Hatchuel and Weil 2007], *a complete formalization of the theory is still an open issue.*
Nevertheless, some steps for formalizing CKT have been taken in some recent work. [Hatchuel and Weil 2007] argues that there are significant similarities between the type of reasoning described by CKT and Forcing, a technique used in Set Theory for constructing alternative set theoretic models with desired properties. It is claimed that the parallel between Forcing and CKT is an important step for design theory in general but this issue needs more formal investigation. In a complementary approach, [Kazakci et al. 2008] shows that CK type reasoning can be implemented with much more simple formalisms. They use a propositional term logic to model the basic ideas of CKT. They suggest a notion of "models of K space" to emphasize that different knowledge structures (or formalism) used to model knowledge will yield different conceptive power and degrees of flexibility in reasoning.
[Kazakci 2009] presents a first-order logical formal account of CKT's core notions. To emphasize the constructive aspects of a design process, he uses Intuitionist logic. That work considers the *interaction*

of concepts and knowledge providing the necessary definitions that allows the interaction and expansion of the corresponding spaces. The current paper builds on this work and complements it in that we consider the core proposition of the theory - the *dual expansion of concepts and knowledge* and investigate the logical implications of such a principle.
The contribution provided is both a step towards better understanding the theoretical roots of C-K theory and design reasoning in general and towards the possibility of better founded comparison of C-K with other formal theories of design such as [Yoshikawa 1981] or [Zeng and Cheng 1991].

1.2 Approach

[Kazakci 2009] argues that, since knowledge in K space can be expanded (new objects and their properties can be learnt about the world), the knowledge space is a *partial theory* (in logical sense). The minimal structure for modeling the evolution of such a theory is that of a monotonic learning. In logic, the standard way to describe such models is by means of Kripke semantics. Kripke semantics (for non-monotonic logics such as the Intuitionist logic) imposes a *partial order on the stages of knowledge* of an individual. [Kazakci 2009] describes the evolution of a knowledge space based on the standard Kripke semantics for Intuitionist logic and he suggests that a concept space may be built based on each stage of knowledge. However, the evolution of concepts, hence, *the dual expansion of C and K spaces needs more elaboration*. Here, we suggest using an *extended Kripke structure* by considering a partial order on what we call *design stages*. A design stage is a tuple <K, C> where K is a partial theory representing knowledge and C is a (single) concept. A design stage might be extended by a *design move* either by expanding the concept, or by expanding the knowledge. As we discuss, this captures formally the principle of conjoint expansions of concepts and knowledge.

1.3 Plan of the paper

The plan of the paper is as follows. Section 2 gives a detailed discussion and analysis of some of the ideas of CKT from a logical modelling point of view. The insight provided by this discussion is used in Section 3 to introduce *design stages* and *design moves*. Details of the basic formal framework describing the dual expansion is presented and discussed during this section. Section 4 discusses possible extensions and some technical details. Section 5 concludes.

2. An analysis of dual expansion in CKT

In this section we provide an in-depth discussion of some of the ideas of CKT in order to set the ground motivation for the formal framework of the next section. After a brief presentation of CKT, an in-depth discussion of the notions of conceptual expansion, concept generation and knowledge expansion with respect to the formalism of first-order logic follows.

2.1 An overview of CKT

CKT claims that the fundamental properties of design reasoning are based on the distinction and interaction between two spaces;

- **Knowledge space** A knowledge space represents all the knowledge available to a designer (or to a group of designers) at a given time. These are propositions that the designer is capable of declaring as true or false; i.e., propositions whose logical status are known to the designer (e.g., some tires are made of rubber).
- **Concept space** A concept space represents propositions whose logical status are unknown and cannot be determined with respect to a given knowledge space. These are propositions that can be stated as neither true, nor false by the designer at the moment of their creation (e.g., some tires are made of dust).

Concepts are descriptions of an object of the form "C: there exist an object x with the properties p_1, p_2,..., p_n" such that C is *undecidable* with respect to current K.
According to CKT, creative design begins by a conceptual expansion that forms a concept. A novel and unusual property is added to a concept to form a new concept (e.g. tires for life). The elaboration of concept can then be continued either by further expansions (tires for life are made of silicon) or by

restrictions (that is by adding usual properties of the initial concept, e.g. tires for life are round). Conceptual expansions or restrictions are called **partitioning** in CKT.
When elaborating a concept space, a designer might use his or her K space, either to partition further the concepts, or to attempt a validation of a given concept. This last type of operation is called **K-validation** and it corresponds to the evaluation of a design description using knowledge. The result of a K-validation is positive, if the designer acknowledges that the proposition "there exist an object x with properties $p_1,p_2,...,p_n$" is true. The result is negative, if the knowledge available to the designer allows him to state that the proposition is false.
Often the validation of a concept is not readily possible. In order to validate concepts, new knowledge warranting the existence conditions of such an object should be acquired. In terms of CKT, knowledge should be expanded. The expansion of knowledge space is called **K-expansion**. The central proposition of CK theory is thus "design is the interaction and dual expansions concepts and knowledge" [Hatchuel and Weil 2003, 2007, 2009].

2.2 Design as expanding partitioning

One of the most intriguing parts of CKT, from a logical point of view, is its account of expanding partitions. "Partitioning" is a term coming from a general method of reasoning known as Branch and Bound Method (most of the features of CKT have been introduced by comparison to BBM, [Hatchuel and Weil 2002]). Let us try to reconstruct the idea, starting with a short (and simplified) account of the Branch and Bound Method.
Assume we are looking for the minimum of a function f(x), where $x \in \mathbf{R}$. One could divide the task into subtasks by partitioning **R** into, say, 5 parts R_1, R_2, R_3, R_4, R_5. (This is a partition if these parts have no element in common and their union is **R**.) If we can find the minimum of f(x) in each of these parts then the least one will be the minimum we are looking for. This is an example of Branching. Before searching these five minima, we can sometimes much easily find some upper bound and some lower bound for such a minimum on R_i. So let m_i be the minimum we are looking for, and we know $l_i \leq m_i \leq u_i$. If it turns out that $u_4 \leq l_2$ then we can already conclude that the minimum of R_2 will be larger than that of R_4 and hence we can skip looking for the m_2. This is the Bound part of the search method. The above description of BBM contains the principle idea of partitioning the 'search space' (branching) and choosing the best next subtask (identifying subtasks that can be disregarded).
A similar kind of reasoning can be applied to logical descriptions of objects. To try to reinterpret BBM in a logical context, let us recall the Venn diagrams that can be used either for (classical not intuitionistic!) propositional logic or (Aristotelian) syllogistic logic. A proposition P can be modeled as a subset [[P]] of some 'universe of discourse' U. In fact P causes a partitioning of U by $[[P]] \cup [[\neg P]]$. Adding another proposition (or concept) Q could further partition [[P]] into $[[P \wedge Q]] \cup [[P \wedge \neg Q]]$. If we see [[P]] as a set of possible 'outcomes' further branching into $[[P \wedge Q]]$ will only further *restrict* the possible outcomes; Figure 1a.
But how is an *expanding* partitioning possible? When a new concept $P \wedge R$ is formed for the fist time, the designer does not know whether an object with these properties (may) exist or not. That means there is no intersection between [[P]] and [[R]] when design starts. The aim of the process is to expand known Ps and Rs in the knowledge space so that such an object become known; Figure 1b.
This suggests that expansive partitioning is K-relative like the rest of the reasoning process. Hence, the notion of expansion in C space cannot be understood independently of the knowledge. Here, we suggest describing it by introducing a *supporting body of knowledge* K_P for every property P; Figure 1c. A proposition P is supported by some knowledge K_P if that knowledge allows to acknowledge (verify, warrants, etc) the existence of objects having the property P (although, as we shall see later, we will allow this support not being sufficient to prove P). In figure 1c, we see that although there are some knowledge K_P and K_R supporting the existence of objects having, respectively, properties P and R there is no knowledge confirming the existence of $P \wedge R$. The parts of the universe that are known do not include such intersection. In order to validate the expansion "attempt" $P \wedge R$, new knowledge that warrants the existence of $P \wedge R$ should be acquired. Hence, the idea of supporting body of knowledge allows us to capture the idea of dual expansion: The expansion $P \wedge R$ is possible if knowledge can be

extended to some $K_{P\wedge R}$. This logical interpretation of the notion of dual expansion will provide the primary insight of our logical framework. We shall nevertheless discuss some points that will prove helpful before proceeding to our formal framework.

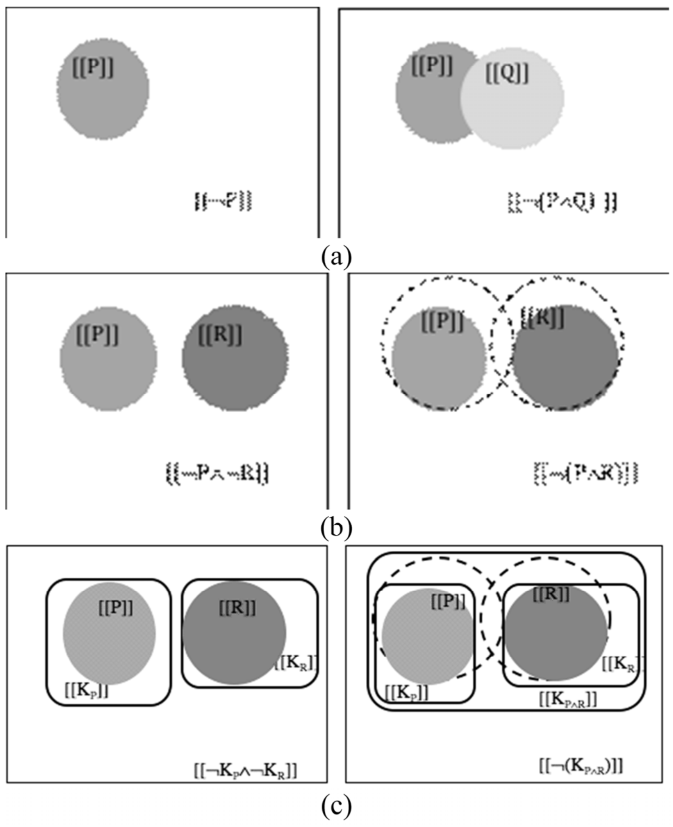

Figure 1. A description of expansive partitioning

2.3 Producing concepts from knowledge

Another intriguing idea from CKT is the emerging of new 'concepts' from (new) knowledge. Can we generate concepts using logic? Recall that in CKT for C to be a concept it has to be 'unknown', in as far as our body of knowledge allows us, whether there are instances of the concept or not. From the examples in CKT literature we can reconstruct a simple mechanisms at work here, using the language $\mathcal{L}_K$ in which our body of knowledge K is represented.

If we assume $\mathcal{L}_K$ is a first order logic language enriched with a set of constants for specific individuals and predicates (the *signature* of $\mathcal{L}_K$), it is only natural that extending K may also extend the signature. The new part of the body of knowledge may introduce new constants (Planck's h, the star Vega-β, President Obama) and predicates (being married to, prime, being the president of).

Concepts can now be generated from knowledge by recombination of expressions used in the body of knowledge. If 'Beatrix is the queen of The Netherlands' then 'x is the queen of y' is a phrase in $\mathcal{L}_K$. Which allows us to form an expression like: 'Planck is the queen of Vega-β'.

If we model phrases as formulas with one free variable x (a restriction we may lift later on) this amounts to forming conjunction of existing phrases, like in $\text{Boat}(x) \wedge \text{Flies}(x)$. Such a new phrase (if K does not imply that $\exists x.\ \text{Boat}(x) \wedge \text{Flies}(x)$ and neither that $\neg\exists x.\ \text{Boat}(x) \wedge \text{Flies}(x)$) could be used as a concept C and starting point for design.

Combined with knowledge extension this simple mechanism will supply the design process with a wealth of new concepts without any appeal to imagination or hidden creative powers. The mechanisms above can easily be extended further, e.g. by lifting the restriction on the type of phrases used. Like in the example 'Planck is the queen of Vega-β', where we could start the design turning this into the question ‘Wouldn’t it be nice if Planck is the queen of Vega-β?’. This could ‘branch’ (in a series of steps) into ‘absorbing very bright light to produce both energy and comfortable back ground illumination’.

2.4 Generating knowledge from concepts

Knowledge may guide the designer in avoiding branches that are known to be undesirable (like in case K contains some constraints on the amount of money to spend on producing a Car(x), combined with the knowledge that Gold(x) will raise the cost of production considerably). This is similar to the bounding part in Branch and Bound.

One simple mechanism of knowledge expansion occurs when the designer (or the design team) is aware of the body of knowledge K, say there knowledge is a part of K called K_0. Extending this K_0 could be done be Googling the web, asking experts etc.

A second mechanism could involve further research in the field K. The experts in the field might be unable to answer the questions of the design team. Further research and experimentation could be necessary. For example we might want to use carbon-epi-hexa-fluor-plexitude for the heat shield of our Vega-β-surveyor, but it is unknown what will happen if carbon-epi-hexa-fluor-plexitude is heated above 5 million degrees Celsius.

A third mechanism may occur when we try to combine parts of theories, say the nanotechnology with the neurobiology of humain brain, say, in order to use nanotechnological devices to record firing patterns of neurons. Whether such thing is possible may be a complete new subject for scientific research.

3. A logical framework

3.1 Preliminaries

In this section, we will describe the design process as generating design stages <K, C>, where K is some body of knowledge and C is a concept. In CKT one is especially interested in design stages where C is totally new for K. Here we will allow *degenerated stages that can be discarded once they are seen as inconsistent or in fact not new at al*l (hence C turns out to be feasible already based on K).

The design process may extend a design stage in principle in infinitely many ways. One could try to imagine all ‘existing’ possible ‘bodies of knowledge’ or all ‘possible concepts’ and try to describe these as (a special sort of) sets, such that the operations in the design process which transform a stage <K, C> into a new <K’, C’> can be defined as a special kind of (extended) ‘search operations’, not unlike known search algorithms (e.g. on databases or in linear programming).

Not only Ockham’s Razor makes the ‘existence’ of such ‘Knowledge Spaces’ or ‘Concept Spaces’ suspect, simply from a pragmatic point of view, some sort of constructive reasoning in the design process seems to be attractive. Operations in the design process defined using ‘mental images’ of infinite collections containing ‘lawless’ sequences are certainly not constructive [Kazakci 2009). Note that according to CKT there is no ‘algorithm’ or construction that will determine the next step in a design path. Such a path can be seen to behave ‘lawless’ in Intuitionist sense [Kazakci 2009).

Therefore, at least from a pragmatic point of view, it seems reasonable to assume that at each point in the design process the body of knowledge has a finite representation.

We introduce a basic logical framework that will allow us to represent in logical terms some of the ideas in CKT. We will be a little vague on the precise logical language $\mathcal{L}$ and the rules of the logic used. This is because we want to extend the framework later on. This could be done in different directions and we don’t want to assume too much about either the expressivity of the language or the strength of the logical rules.

Without any problem the reader may assume $\mathcal{L}$ is the language of predicate logic and the logic is the usual classical logic (although all our results will be valid in intuitionist logic as well: we will simply not use the axiom $A \vee \neg A$ or the rule $\neg\neg A \vdash A$ of classical logic). Note that, in a constructivist type of logic, like intuitionist logic, a proof of C from K is a construction, one that under a certain conditions

can be used as a *recipe* to construct an instance of Cx based on the constructions that exist according to K – which is a suitable characteristic for modelling design endeavour.
In our notation T |- A means a formula A is provable from the set of formulas T. T* will be the set of all formulas (in $\mathcal{L}$) derivable from T (T* = {A ∈ $\mathcal{L}$ | T |- A}). So T* ⊆ S* and T |- A implies S |- A.

3.2 Design Stages and design space

Let us model the body of knowledge (as a first approximation) as a finite set of formula, in the language of first order logic (predicate logic). Such a finite theory K will always be 'partial knowledge' and hence extendable. Again in a first approximation, we could model the 'concept' C as a formula (in predicate logic). To turn the design process in some sort of reasoning we introduce a logical relationship that may or may not exist between K and C: derivability. K |- C means: C is derivable from K.

Definition 1 A *design stage* in a design process is a tuple s = <K, C>, where K∪{C} is a set of formulas. A *design space* is a partially ordered set of stages, where $\langle K_0, C_0\rangle \leq \langle K_1, C_1\rangle \Leftrightarrow K_0^* \subseteq K_1^*$ and K_1, C_1 |- C_0
s = <K, C> is called

- Consistent ⇔ K |/- ¬C (and inconsistent o.w.)
- *Closed* ⇔ K |- C
- *Feasible* ⇔ s is consistent and closed
- *Open* ⇔ s is consistent and not closed

s_0 and s_1 are *equivalent*, $s_0 \equiv s_1$, if $s_0 \leq s_1$ and $s_1 \leq s_0$
$s_0 < s_1$ if $s_0 \leq s_1$ and $\neg(s_0 \equiv s_1)$
$\langle K_0, C_0\rangle$ and $\langle K_1, C_1\rangle$ *share the same Body of Knowledge* if $K_0^* = K_1^*$
The condition K_1, C_1 |- C_0 can be seen as a kind of 'bounding' to guarantee that if s ≤ s' then s' is indeed a 'solution' for the issue in s. The following facts are easy to prove.

Facts 2

1. <K, C> is open ⇔ K |/- C and K |/- ¬C
2. ≤ is reflexive: $s_0 \leq s_0$
3. ≤ is transitive: $s_0 \leq s_1$ and $s_1 \leq s_2$ then $s_0 \leq s_2$
4. if s consistent then s either feasible or open
5. if $s_0 \leq s_1$ and s_0 and s_1 share the same body of knowledge then s_1 open if s_0 open
6. If $\langle K_0, C_0\rangle \leq \langle K_1, C_1\rangle$ and $\langle K_1, C_1\rangle$ feasible then $\langle K_1, C_0\rangle$ feasible
7. $\langle K_0, C_0\rangle \equiv \langle K_1, C_1\rangle \Leftrightarrow K_0^* = K_1^*$ and K_0 |- $C_0 \leftrightarrow C_1$

Fact 1 is simply restating the definition of an open design stage (in line with [Kazakci 2009]). Fact 2 and fact 3 state that ≤ is in fact a partial ordering and so the design space is a partially ordered set. Fact 5 implies that only refining the concept without extending the knowledge base will never provide a feasible solution in the design process. Fact 6 proves that a design path ending in a feasible design stage also provides a feasible solution for the original concept C_0. Fact 7 shows that the equivalence relationship generated by ≤ is, from a logical point of view, a natural one as it is the same as telling that the bodies of knowledge are equivalent and for this body of knowledge the concepts are equivalent.

3.3 Extending design stages: design moves

The formal framework above is a quite minimal body of knowledge we will need in the following to make sense of the ideas of *conceptual and knowledge expansions* in CKT. Let us first introduce a simple definition for the signature of (sets of) formulas and a simple operation on design stages:

Definition 3 Let A be a formula, the *signature* of A, σ(A), is defined as the set of constants and predicates used in the atomic subformulas of A.

Definition 4 Let s = <K, C> and s' = <L, D> be design stages. The *product* of s and s', s⊗s' is defined as: s⊗s' = <K∪L, C∧D>

To define a design move that allows an expansion in a given design stage, we first will extend our definition of a design stage slightly. We will assume that the goal of the design is an individual object described by Cx, where Cx is a formula with x free.

Definition 5 Let s = <K, Cx> then s is *consistent* (*open*, *closed*, *feasible*) for a constant c if <K, Cc> is *consistent* (*open*, *closed*, *feasible*). A design stage s will be called *consistent* (*open*, *closed*, *feasible*) if s is *consistent* (*open*, *closed*, *feasible*) for some constant c $\in \sigma(K)$.

Definition 5 will allow us to use the same terminology as in the basic framework, but with a small abuse of terminology.

Definition 6 Let s = <K, Cx> and s' = <T, Dx> be design stages. The *product* of s and s', $s \otimes s'$ is a *design move* from s if s < $s \otimes s'$.

If Dx above is a literal (an atomic formula or the negation of an atomic formula) $s \otimes s'$ is called an (atomic) partition. If a partition results in a feasible design stage, it corresponds to a conjunction in CKT. ($s <_i s'$) will be used as a notation for s' being a design move from s to s'.

Definition 6 describes what seem to be the basic features of the expansions in CKT. In [Kazakci 2009], in C space, no other partitions than the atomic ones are considered. In the next section, we discuss how this can be generalized within the framework we provide here. Note that $s <_i s'$ is neither reflexive nor transitive. On the other hand if $s <_i s'$ then s < s', which is a transitive relationship.

The main difference with the existing literature about CKT, is that *partition is now defined as a kind of product of design stages in stead of a combination of separate steps extending knowledge and concepts*. The notions of K-expansion and C-expansion can be obtained as *special cases of design moves* as in definition 7.

Definition 7 Let s = <K, Cx> and s' = <T, Dx> such that s"=$s \otimes s'$ is a design move from s.

s" is called:

- pure K-expansion if Dx $\equiv$ **T**,
- pure C-expansion if $T^* \subseteq K^*$
- proper if neither pure K- or C-extending

In the case of pure C-extension, if we have $\sigma(T) \cap \sigma(K) \neq \emptyset$, then the partition is a restrictive. Otherwise, it corresponds to an expansive partition.

Here **T** is a propositional constant equivalent with all tautologies, making the concept part of <T, Dx> void. This operation gives rise to the design space depicted in figure 2.

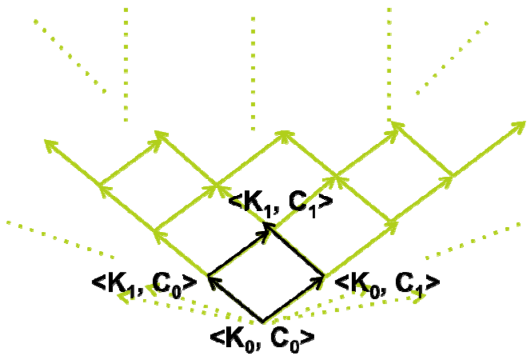

Figure 2. A design space and its possible evolutions by design moves

Example. Lest us consider again the concept of Fly(x) $\wedge$ Boat(x) . If we have a knowledge of the sort Fly(a) $\wedge$ Wing(a) for some object a, we can extend Fly(x) $\wedge$ Boat(x) to Fly(x) $\wedge$ Boat(x) $\wedge$ Wing(x) by a restrictive partition. On the other hand, using some knowledge about some object b such that Transparent(b) (and assuming that no other knowledge exist implying Transparent(x) is a property of some of the objects that fly), we can extend Fly(x) $\wedge$ Boat(x) to Fly(x) $\wedge$ Boat(x) $\wedge$ Transparent(x). Here this is modeled as <{Fly(a), Wing(a)}, Wing(x)> is feasible (there obviously is a constant, a, such that Fly(a), Wing(a) |- Wing(a)). We can extend < K, Fly(x) $\wedge$ Boat(x)> to <K $\cup$ {Fly(a), Wing(a)}, Fly(x) $\wedge$ Boat(x) $\wedge$ Wing(x)> for which we have {Fly, Wing, a} $\cap$ {Boat, Fly} $\neq \emptyset$ (restrictive partition). <{Transparent(b)}, Transparent(x)> is also feasible and it can be used to extend

$< K, Fly(x) \wedge Boat(x)>$ to $<K \cup \{Transparent(b)\}, Fly(x) \wedge Boat(x) \wedge Transparent(x)>$ which is an expansive partition.
In our framework, where the body of knowledge and the concept are always treated together, the additional knowledge Fly(a), Wing(a) is explicitly added to the previous knowledge base (if it was not yet there).
From a CKT point of view one might wonder about the 'design' status of a feasible <T, Dx>, as such a design stage already will belong to the knowledge domain. As mentioned before we allow also the borderline cases where s is inconsistent or where s is feasible. Instead of using the condition that <T, Dx> should be feasible for some constant $c \in \sigma(T)$ we could obviously have stated that T needs to be consistent and for some c $T \vdash Dc$.
Our framework suggests that <T, Dx> may well be the outcome of another design path, starting for example with $<\{\exists x(Fly(x) \wedge Wing(x)\}, Wing(x)>$: "there is something that can fly with wings". Hence, *our framework allows for subtasks in design and the use of their output*. This is a new dimension in CKT as it can allow modeling and analyzing formally the integration efforts of different design teams working on different parts of the design problem.
Finally, we introduced an extra condition on feasibility (or closing) of <K, Cx>, compared to CKT, by asking not only for $K \vdash \exists xCx$ to become true but requiring a constant c in $\sigma(K)$ such that $K \vdash Cc$. This is obviously a kind of constructivist twist: for $\exists xCx$ to become derivable one needs a *witness* c such that Cc becomes derivable. In the context of design this seems to make good sense (at least as a condition for feasibility), as we would not be satisfied with the outcome of the design process in the form "$\exists xCx$ is true" without being able to tell how to construct such an x.
For design moves, we can find some simple facts:
Facts 8
Let $s <_i s'$:
If s' is pure K-extending then:

- s' is open (consistent) $\Rightarrow$ s is open (consistent)
- s feasible and s' consistent $\Rightarrow$ s' is feasible

If s' is pure C-extending then:

- s and s' share the same body of knowledge
- s' closed $\Rightarrow$ s closed
- s open and s' consistent $\Rightarrow$ s' open
- s' feasible $\Rightarrow$ s feasible

Note that with an alternative definition of K-extending, using $K \vdash Dc$ for some $c \in \sigma(K)$ instead of $Dx \equiv \mathbf{T}$, we would lose fact 8.1. Pure K-extending partitions opening a closed stage contradict our intuition about pure K-extensions.
Up to now we have mainly introduced a kind of 'language' to describe a logical refinement of CKT. The integration of this framework with that of [Kazakci 2009] is the next step of our work. In the following, we will discuss some other possible extensions.

4. Possible further extensions

The framework above could be extended in many ways; a few of them are discussed below.

4.1 K-validation and validation of K: hypothetical reasoning

One of the issues that are not addressed in detail by CKT is the status of 'the knowledge'. In our framework we model the body of knowledge (BoK) as a set of formula K. It is possible to presuppose that the 'knowledge' in de BoK is always 'true', moreover that it is useable in the sense that if $\exists xCx$ is provable from K, somehow K will help us to find an c such that Cc is 'true'.
These 'background' assumptions play a role in the definition of K-validation as a kind of query on K whether $\exists xCx$. If the outcome of such a query is positive than <K, Cx> is considered feasible (and is no longer a design stage, but a part of knowledge).
Obviously in our framework such a K-validation does not play an important role, as implicitly we postulate that at the introduction of every new design stage the question whether $K \vdash C$ will be

examined. In CKT the K-validation seems to provide the ‘answer’ on the question posed by the original design stage, whereas in the current extension such an ‘answer’ is necessarily a design stage with a proof showing it to be feasible.
We can ask the question the other way around: what about the validation of knowledge? If we can assume that, our BoK consists of clear true statements like: “Obama is the president of the US”, this is not a problem. However, in design, it can be the case that the reasoning needs to continue based on assumptions and not only factual knowledge.
Such ‘hypothetical knowledge’ can be taken into account in a slightly more general extension of our framework. For example allowing extensions (and partitions) using <T, D> such that <T, D> is feasible and we, for the moment, assume that T could be true, without having the evidence yet.
A trivial example would be closing <K, C> by <K$\cup${C}, C>, which may lead to some further restrictions on some potential update operation (e.g. ‘update s_0 with s_1’, say, $s_0 \oplus s_1$), adding the design stage <T, D> to K. Such an operation would allow using T in the rest of the design path extending it. K-validation might then mean replacing all design stages from the BoK.

4.2 Stages as knowledge or concepts

Stages themselves can be used as ‘concepts’. A *primitive concept* is a formula, like a *primitive body of knowledge* is a set of formulas. A *stage* is a pair <K, C> where K is a BoK and C a concept. Obviously a primitive concept (BoK) is a concept (BoK). But also: If s is a stage then s is concept and {s} is a BoK. This would allow for example <K_2, <K_1, <K_0, C_0>>> a typical stage obtained by knowledge extension from <K_0, C_0>. Naturally, we want <K_2, <K_1, <K_0, C_0>>> to be equivalent to <$K_0 \cup K_1 \cup K_2$, C_0>.
For extension of a BoK K_0 by a stage <K_1, C_1>, one could always require feasibility of < K_0, <K_1, C_1>>. Observe that if <T, A> is a feasible stage and T is validated knowledge, we can replace <T, A> by T. On the other hand if T would be ‘hypothetical’ (finite) set of formulas, we could replace <K, <<T, A>, C>> by <K, $\wedge T \rightarrow C$>. In such a way the idea of using stages as knowledge or concepts could in some way already be seen as contained in our basic logical framework, allowing for C formulas in general, so also implications.

4.3 Extending or restricting the language

Our basic framework does also allow concepts that define relationships, like <K, C> with C = $\forall z\,(Rxz \rightarrow Szy)$, where R, S $\in \sigma(K)$. Such ‘concepts’ are about ‘finding’ a relationship Qxy such that $\forall x, y\ (Qxy \leftrightarrow \forall z\,(Rxz \rightarrow Szy))$ and thus extend the language of ‘the concept space’ (using the CKT terminology).
It is assumed in CKT that the BoK can contain all kind of knowledge, like aesthetic judgments, preferences, believes etc. Different types of knowledge can be considered by explicitly introducing some modalities in the language. One might consider introducing the language of epistemological logic (introducing some operator on formulas like [i]A for ‘agent i knows A’ or the more generic []A for ‘A is known’). Likewise operators for preference might be introduced for a more precise description of how design decisions are taken.
It is also possible to restrict the formal language in our framework to one known to be more ‘manageable’ than predicate logic (e.g. computationally). An example would be Description Logic, which is both an extension and a restriction of predicate logic.

5. Conclusion

The paper presents a contribution to the formalization efforts of C-K design theory. More specifically, we presented a first-order logical framework modelling the idea of dual expansion of concepts and knowledge in C-K theory. Ultimately, the present work and its possible extensions we discussed aims at a contribution to a clear and rigorous formal description of C-K theory using logic. The integration of our framework with [Kazakci 2009] and the comparison with other work in duality and co-evolution in design is the next step of our study.

References

Hatchuel, A., P. Le Masson and B. Weil (2004). C-K theory in practice: Lessons from industrial applications. 8th International Design Conference, Design 2004, Dubrovnik, Croatia.

Hatchuel, A. and B. Weil (2002). La théorie C-K: Fondaments et usages d'une théorie unifiée de la conception. Colloque Sciences de la Conception, Lyon.

Hatchuel, A. and B. Weil (2003). A new approach of innovative design : an introduction to C-K design theory. ICED'03, Stockholm, Sweden.

Hatchuel, A. and B. Weil (2007). Design as Forcing:Deepening the foundations of Ck theory. 16th International Conference on Engineering Design - ICED 2007, Knowledge, Innovation and Sustainability, Paris, France.

Hatchuel, A. and B. Weil (2009). "C-K design theory: an advanced formulation." Research in Engineering Design 19(4): 181-192.

Kazakci, A. (2009). A formalisation of CK design theory based on Intuitionist Logic. International Conference on Research into Design. ICORD09. A. Chakrabarti. Banglore, India, Research Publising Services: 499-507.

Kazakci, A., A. Hatchuel and B. Weil (2008). A model of C-K design theory based on a term logic: a formal background for a class of design assistants. International Design Conference 2008, Dubrovnik, Croatia.

Le Masson, P., B. Weil and A. Hatchuel (2006). Les processus d'innovation- Conception innovante et croissance des entreprises Paris, Hermès.

Yoshikawa, H., Ed. (1981). General design theory and a CAD system. Man-Machine Communication in CAD/CAM, North Holland Publishing.

Zeng, Y. and G. Cheng (1991). "On the logic of design." Design Studies 12(3): 137-141.

Dr. Akin Osman Kazakci
CGS - Centre de Gestion Scientifique -Mines ParisTech
60 boulevard Saint-Michel, 75272 PARIS Cedex 06
Telephone: 0(0 33) 1 40 51 92 08
Email:akin.kazakci@ensmp.fr