Ryerson University
Digital Commons @ Ryerson

Mechanical Engineering Publications and Research | Mechanical Engineering Publications

1-1-2012

Representing CK theory with an action logic

Filippo A. Salustri
Ryerson University, Canada, salustri@ryerson.ca

Follow this and additional works at: http://digitalcommons.ryerson.ca/me

Part of the Computer-Aided Engineering and Design Commons

Recommended Citation
F.A. Salustri. 2012. Representing CK theory with an action logic. Design Computing & Cognition. College Station, Texas.

This Article is brought to you for free and open access by the Mechanical Engineering Publications at Digital Commons @ Ryerson. It has been accepted for inclusion in Mechanical Engineering Publications and Research by an authorized administrator of Digital Commons @ Ryerson. For more information, please contact bcameron@ryerson.ca.

Reformulating CK theory with an action logic

Filippo A. Salustri
Ryerson University, Canada

CK theory is an interesting and unique theory of engineering design. This paper introduces *ALX3d*, a formal descriptive version of CK based on the action logic ALX3, which is able to represent aspects of the actions, preferences, beliefs, and knowledge of collaborating, imperfect agents (such as human designers). It is shown that all the basic notions of CK can be rendered in the logic of ALX3d with only one relatively minor change in how the CK terms *concept* and *knowledge* are defined and related. A case study of CK is used to show how ALX3d can also be used to describe some "real-world" situations. The advantages of ALX3d are that they recast CK in a form more readily understood by those accustomed to expert, knowledge-based, and formal systems; provide a "scientific" vehicle for reasoning about the design activities it can describe; and define a possible basis for the development of new, computer-based designers' aids.

Introduction

CK theory [1,2] presents an interesting and unique theory of engineering design, but the available literature does not cast CK in a sound logic. Without soundness, there are severe limits to the reliability of results that any "formal" theory can obtain. In this paper, the author will demonstrate that a *sound action logic* can be just as expressive as CK using only conventional notions of logic. That is, the spirit and benefits of CK can be preserved using conventional logic. The author has presented previous related work elsewhere [3]. Having a formal descriptive (not prescriptive) reformulation of CK yields new benefits that will be described.

CK is a high level description of designing based on the intuitive distinction between *concepts* and *knowledge*. Knowledge, in CK, consists of

Design Computing and Cognition DCC'12. J.S. Gero (ed),
pp. xx-yy. © Springer 2012

statements that are true. Concepts, on the other hand, are statements the logical statuses of which are *unknown*. They are groups of propositions that together describe a possible design solution. Designing, in CK, proceeds from a base of knowledge, and from which concepts are developed. These concepts are manipulated and eventually verified to become knowledge. Thus, a proposed design is a CK concept, and once the design has been appropriately verified, it becomes *known* as a suitable design solution. CK partitions the domain of all propositions into two spaces; *C-space* contains concepts and *K-space* contains knowledge. *Operators* are used to represent the kinds of activities undertaken by designers that transform propositions. These operators are broadly categorized by whether their inputs and outputs are concepts or knowledge. This leads to four categories of operators.

C→C operators transform one concept into another.

C→K operators transform a concept into knowledge through some validation action that establishes a logical status for the concept.

K→C operators generate concepts by transforming knowledge propositions. For example, given knowledge items *x* and *y*, but no further knowledge about *x* and *y*, one might create a concept consisting of $x \wedge y$, or $x \Rightarrow y$. These operators can be thought of as design concept generators.

K→K operators deduce new knowledge from existent knowledge; these are the conventional acts of inference common in analysis.

There have been many papers written about CK, and case studies indicate that CK can describe a variety of design activities. However, as they are all based on a few fundamental ideas, this author uses only those key papers that define CK. Further details about CK will be provided below.

Action logics are formal systems that address the activities undertaken by reasoning agents. ALX3 [4] is especially well suited because it is the only sound and complete action logic of which the author is aware that assumes the agents (i.e. human designers) exhibit *bounded rationality* – they are imperfect reasoners having imperfect/incomplete knowledge (per Simon [5]). CK also assumes bounded rationality.

The rest of this paper is organized as follows. Section 2 introduces ALX3. Section 3 then describes how ALX3 covers the scope of CK theory. The author does this by examining each key feature and notion of CK from [1] informally, and then translating it into a formal representation in ALX3, resolving any identified logical problems along the way. We refer to the formal design model thus developed as *ALX3d* (ALX3 for *d*esign). Section 4 then uses an example of the use of CK from the existent literature to demonstrate the use of ALX3d. Finally, Section 5 discusses some of the benefits that a formalization like ALX3d can afford.

Overview of ALX3

ALX3 is a sound and complete first-order action logic that incorporates knowledge, belief, preference, and action operators to represent the activities of multiple agents working with bounded rationality (per Simon [5]). ALX3 is completely documented in [4]. It assumes the usual apparatus of first order logic: constants, variables, functions, and relations, conjunction ($\wedge$), disjunction ($\vee$), negation ($\neg$), material implication ($\Rightarrow$), and universal ($\forall$) and existential ($\exists$) quantification over variables. We also use the notation *x: y* to indicate that *x* is defined as *y*.

In action logics, an *agent* is an entity that takes actions to achieve certain states. An agent *a* can know ($K_a\psi$) or not know ($\neg K_a\psi$) a proposition ψ; the agent may also believe ($B_a\psi$) or not believe ($\neg B_a\psi$) the proposition. ALX3 defines knowledge typically for formal systems as *true, justified belief.* Within the system, knowledge and belief are treated as two separate operators related by a definition of the former with respect to the latter. Other definitions of knowledge are possible without necessarily affecting the soundness of the logic. In the language of ALX3, $K_a\psi$: $B_a\psi \wedge \psi$. That is, for a proposition ψ, knowledge is equivalent to true, justified belief. Consider the statement *"It is raining in Tokyo."* The statement is either true or false, whether or not an agent knows it. The agent may *believe* the statement is true, but will not *know* until the agent takes appropriate actions to verify the belief.

One might question our definition of knowledge with regards to determining truth. Philosophically, a belief is true if it is true in an absolute sense, which means that all we have are beliefs. We would have to have access to some omniscient agent to guarantee that true things are in fact true. However, as engineers and scientists, we are accustomed to assuming truth by invoking other mechanisms. We accept the yield stress of cold-rolled steel as given in standard handbooks, for example, even though the actual yield stress of any specific steel part will not be exactly as the handbook defines. Similarly, we carry out physical testing of products to determine their performance knowing that some products will likely fail in the future – yet we accept the product's performance as true based on those tests. This notion of truth, though philosophically unsatisfying, is sufficient for us to have developed all the engineered products that have ever existed, and with a very high degree of overall success.

Generally, we assume truth within and relative to some specific *context*; so long as the context holds, we can expect a certain measure of truth to be acceptable. This is a typical scientific approach: given evidence that

some statement is true, and until incontrovertible evidence is found to the contrary, then accepting the truth of the statement is reasonable.

There is another important difference between knowledge and belief within the convention of formal systems. Knowledge is not "lost" – once you have it, you cannot lose it. Losing knowledge is not the same as just "forgetting" it. Forgetfulness is a function of an imperfect reasoning agent (i.e. an agent with bounded rationality) and not a function of the knowledge that is forgotten. Beliefs, on the other hand, can be *retracted* from a system of beliefs if some action proves that the belief cannot be true. This kind of action is the same as that which, when successful, identifies truth and thus knowledge. In both the author's work and in CK, these kinds of actions are *validation* actions that engineers use to "prove" their concepts – within a reasonable context. This makes beliefs similar to assumptions – statements that we may take to be true for the sake of achieving some goal unless and until they are demonstrated to be false.

ALX3 also uses a *many-worlds interpretation* based on action logic semantics. This means that the current state (or "world") is a collection of propositions, and that alternative (or future, or past, or just *other*) states are accessible from the current state through *actions* that cause propositions to be added or withdrawn. Thus, actions allow one to represent how states of knowledge and belief, and the propositions they include, change. Actions are written in ALX3 as $\langle a \rangle \ \psi$, where a is the action and ψ is a proposition that is true in any state that can occur because of executing the action. In other words, $\langle a \rangle \ \psi$ means that executing action a will make ψ true.

It is important to note that the propositions that define a state need not be written in ALX3. State-defining propositions must obey the rules of first order logic in order to maintain the soundness of ALX3, but otherwise, the language of the state propositions may differ. This is especially important with regards to the primitives that are used in those propositions. Any logic built on first order systems is acceptable. Since these propositions capture, among other things, the product being designed – a *product model* – this means that any first order product modeling logic can be used with ALX3 without affecting the soundness of ALX3 itself. This gives us the flexibility to treat the product modeling logic and the design process logic as essentially disjoint entities.

Associated with ALX3 actions is the notion of *accessibility* of states. A state t is accessible from another state s if there exists an action or a sequence of actions that can be described by propositions that are true at t, s, and all intermediate states. There are two accessibility relations in ALX3. Direct accessibility (DA) is defined as $DA_i\psi$: $\exists a \ \langle a_i \rangle \ \psi$; that is, agent i can reach a state where ψ holds directly from the current state by executing

action *a*. General accessibility (A) is defined as $A_i\psi$: $DA_i\psi \vee (DA_i\varphi \wedge (\varphi \therefore \psi))$; that is, agent *i* can reach a state where ψ holds either directly through one action or indirectly through a sequence of actions (the causation operator $\therefore$ is explained next).

A key feature of ALX3 is a special implication operator for causal relations. Here we write it $\varphi \therefore \psi$; that is, in all states (a) that are *closest* to current state and (b) where φ holds, ψ also holds. A *closest state* is one that (a) is accessible from current state (i.e. has an action allowing an agent to move to it from the current state) and (b) has the smallest possible changes from the current state. The semantics of ALX3 [4] provides a complete formal description of this operator, but that is beyond the scope of this paper. We note that the ALX3 causal operator is broader than the usual "scientific" sense of "cause and effect." In ALX3, cause can arise from simple preference (see below – i.e. something is caused because an agent prefers it to an alternative). It can also capture the relation between dependent and independent variables – i.e. the values of a set of independent variables *cause* the values of a set of dependent variables. As such, a causal relation in ALX3 can also be regarded as a kind of *explanation*.

Finally, agents may sometimes choose to *prefer* one state to another without necessary explanation. ALX3 supports this with a binary *preference operator*. We write this as $\psi P_a \varphi$; that is, agent *a* prefers closest states where ψ is true to closest states where φ is true. This is useful because it decouples what agents prefer from the rationale for that preference.

Applying ALX3 to design: ALX3d

In this section, we will explain how ALX3 can be used to represent CK theory. We will consider key concepts in CK one at a time, and show the corresponding element in ALX3d. No new operators are needed for this beyond those already in ALX3. This means that ALX3d is as sound a logic as ALX3.

Knowing versus believing

The fundamental departure in ALX3d from CK is the notion and application of *logical status*. In CK, only knowledge has logical status; concepts are not knowledge and have no logical status.

In ALX3d, on the other hand, we distinguish between the logical status of a proposition and whether an agent knows that logical status. That is, we use *belief* (a proposition that has a logical status not *known* to an agent as described in the previous section) as the comparable notion. That is to

say, an agent can believe a proposition and work with it even if it is false. While this definition of belief is not as subtle as its formal definition ALX3, it is sufficient for our purposes here.

We therefore make the following equivalences. CK's knowledge space (K-space) contains all true propositions [1]; we interpret this as saying that K-space contains all *known* propositions ($K_a\psi$). Similarly, CK's concept space (C-space) contains all propositions "that have no logical status" [1]; we interpret this as meaning that C-space contains propositions the logical statuses of which are not known to an agent, but to which an agent ascribes a logical status temporarily (until validation is possible) to attain some goal. That is, we are saying that C-space contains only *believed* propositions ($B_a\varphi$), and that ALX3 beliefs subsume CK concepts.

Initial and final conditions

In CK as in ALX3d, a design is complete when its description consists entirely of knowledge (and no beliefs), K_aD (where *D* is the design being developed). This is the final condition. We know the design is sufficient because we know it satisfies a conjunction of requirements *R*; that is, the agent knows that the requirements R imply at least one design D,

$$K_a(R \Rightarrow D). \tag{1}$$

Initially, in ALX3d, the designers know some of the requirements, K_aR_i, and may have some beliefs about the design, B_aD_i. Since beliefs may be falsified, there is no reason to undertake a design. Thus at least one requirement must be known (i.e. must be true). They would also believe that there are actions they can take that will lead to a state in which some superset of the initial requirements R_i (i.e. R) some design D in that state. We summarize this by saying that the goal of designing is to move from a state of *belief* to one of *knowledge*. This is consistent with CK, where design involves the transformation of concepts into knowledge. In ALX3d, we write this as:

$$K_aR_i \wedge B_aD_i \wedge B_aA_a(\exists R\ \exists D\ ((R \Rightarrow Ri) \wedge (R \Rightarrow D))). \tag{2}$$

Coarse description of design activity

In CK, K-space and C-space include propositions that capture design information. In ALX3d we impose a little more detail. The requirements, *R*, of a design problem is a conjunction of individual requirement propositions, R: $\wedge_i r_i$. *R* can be a conjunction of different *r*'s at each different state. Similarly, a design *D* is a conjunction of individual design propositions, D: $\wedge_i d_i$. *D* can be a conjunction of different *d*'s at each different state.

We identify an appropriate design by going from $B_a(R \Rightarrow D)$ to $K_a(R \Rightarrow D)$. This is done with a validation action by which *D* is evaluated with respect to *R*. (Validation actions are described below.) These actions are identical to C→K operators in CK. Once the designers know that $R \Rightarrow D$, then $K_a(R \Rightarrow D) \Rightarrow K_a D$ (because the knowledge operator in ALX3 distributes over implication) and they know the design too. That is, we know the design *D* is "right" because we have validated it against *R*.

We use causal implication in ALX3 to define the following:

$$K_a R \wedge B_a D \therefore K_a(R \Rightarrow D) \vee K_a(\neg(R \Rightarrow D)) \vee \neg K_a(R \Rightarrow D). \quad (3)$$

This says that if the designers were to *know* the requirements and *believe* to have a valid design, then one of three conditions holds.

$K_a(R \Rightarrow D)$ – The validation was successful and we now know that *D* satisfies *R*. We have found a solution.

$K_a(\neg(R \Rightarrow D))$ – The validation failed; *D* does not satisfy *R*. *D* and/or *R* will have to be changed to proceed.

$\neg K_a(R \Rightarrow D)$ – The validation could not be performed or completed. This would be the case if either *R* or *D* were lacking in sufficient detail.

We will examine these three conditions in further detail below.

ALX3d beliefs versus CK concepts

CK's first definition of "design" [1] can now be written as: design is the process of expanding *R* and *D* to go from Formula 2 to Formula 1.

The CK notion of *K-relativity* is consistent with ALX3. That is, in CK, determining the logical status of a proposition is done with respect to the available knowledge (K-space). If there is no K-space or if it is empty, nothing can be done. If K-space is not sufficiently rich, C→K, K→K, and K→C operators may be unable to give useful results.

In ALX3d, R⇒D is evaluated with respect to what is known to the design agents (our equivalent of K-space) in the current "world" – which is similar to K-relativity.

Where ALX3d and CK differ here is that CK does not seem to account for the differences in what each agent knows, whereas ALX3d can (at least to a degree). The ability to distinguish between what two agents know is inherent in ALX3. It is perfectly reasonable to write that $K_a\psi \wedge \neg K_b\psi$; that is, agent *a* knows ψ but agent *b* does not.

One can then use this to describe some aspects of the collaborations between multiple agents. For example, one may say that in the final state all agents must *know* the design, $\Lambda_a K_a D$. This might be reasonable in small projects, but in large and complex projects such as the design of a commercial airliner, this is not feasible or even necessary. In the latter

case, one might associate an agent *i* with an element of *D*. The element is the conjunction of only *some* of the propositions that constitute *D* – call it D_i; a necessary condition here would be $D \Rightarrow D_i$ in any desirable state. Thus, each agent knows a part of the design, K_iD_i. One may then imagine a supervisory agent *s* (a project manager or team leader), who would not need to know the design, but only that the agents in his team know their respective design elements. We can write this as $K_s(\forall i\ K_iD_i)$.

This is only a superficial description of the organizational/social relationships that would occur. The author's goal here is not to exhaustively describe these relationships, but only to suggest that ALX3 has the capacity to represent them. Detailing the organizational/social aspects of engineering teamwork will be the subject of a future paper. In the meantime, one can refer the interested reader to [4], in which some further details of how ALX3 can be used to describe organizations is discussed.

In CK, "the formulation of the requirements is a first concept formulation which is expanded by the designer in a second concept that is called the proposal" [1]. There is a problem, however, if we accept requirements as C-space elements, as implied in [1]. We must *know* at least some of the requirements, i.e. know their logical status, because in CK, one can only reason about things with defined logical status. Without this, the initial requirements, the initial design (the "first concept formulation"), and the design proposal are all extra-logical, which defeats the purpose of formalizations like CK. Other requirements may begin as beliefs (in C-space) and migrate through design actions to K-space, but we cannot create the design without actually *knowing* (some of) the product's requirements.

In ALX3d, on the other hand, we can reason about beliefs because they are assumed to have a "temporary" logical status as described above. This means that ALX3d is a richer representation than CK, without violating the intention of CK.

CK concepts have "no logical status." Without this, one cannot identify individual concepts from within sets of concepts because this is done in first order logic by finding a (possibly complex) predicate that is true for only one element of a set, thus identifying it. The treatment of sets is conventionally done in logical systems with *set theory*, and typically with a particular development of set theory called Zermelo-Fraenkel (ZF) set theory [11]. In ZF, there is a specific axiom, the *Axiom of Choice*, which defines how one may select individuals from sets. The axiom is necessary in any development requiring the individuation of specific set elements. In CK, then, the Axiom of Choice is excluded because concepts are without logical status. The set of concepts is defined implicitly by means of propositions that enumerate characteristics of any suitable concept.

This is unwarranted in ALX3d. Beliefs are assumptions regarding the logical status of propositions. The collection of beliefs (the belief structure) allows provisional and conditional reasoning. A belief can be retracted, which would then require validating all inferences that include the retracted belief. Nonetheless, reasoning is possible. Therefore, we do not need to exclude the Axiom of Choice in ALX3d; indeed, at this time, the author finds no conclusive argument either for or against the inclusion of the Axiom of Choice. This gives us a certain greater flexibility of design representation.

The capacity to reason about beliefs is an important element of ALX3d that CK does not provide. What designer would pursue a concept he did not *believe* would be fruitful? What company would pursue a product development project that its members did not believe would be successful? Designers cannot wait for the certainty of knowledge all the time, so they must make assumptions to proceed, and backtrack if those assumptions turn out to be wrong. The belief system support in ALX3d lets us do this.

Translating CK operators to ALX3d

In CK, one changes the logical status of concepts by adding or subtracting properties. In logic, we represent properties with propositions. For example, *weight(motor, 5kg)* asserts a motor has the property of weight with value 5kg. Description logics [7] can be used to develop ontologies (formal descriptions of bodies of knowledge) based on those properties, but this is beyond the scope of this paper.

In ALX3d, we add and subtract propositions that ascribe properties to *R* and *D*, such that subsequent application of validation actions hopefully turns beliefs into knowledge. From this, all four kinds of CK operators (C→C, C→K, K→C, and K→K) translate easily to ALX3d.

Consider an example in [1] about bicycles with pedals and "effective wings." "Bicycles with Pedals" (denoted by the predicate *bp*) leads to a ALX3d belief $B_a(\exists x\ bp(x))$, while "Bicycles with Effective Wings" (denoted by the predicate *bew*) leads to $B_a(\exists x\ bew(x))$. "Bicycles with pedals and effective wings" is written $B_a(\exists x[bp(x) \wedge bew(x)])$. Note that these are beliefs held by an agent – that is, they are design concepts. The real question is not whether such a design is possible but rather whether $R \Rightarrow x$; that is, does there exist a situation wherein a bicycle with pedals and effective wings is appropriate. If there is no such situation, then even considering the concept is pointless.

The answer depends on what is known (the content of CK's K-space). For example, in a dome with an atmosphere on the Moon or some other very low gravity setting, *bew(x)* might be perfectly reasonable. The reason

why *bew(x)* seems silly is because of the situation (context) we *assume* in the absence of specific knowledge of the implication. Context logics [6] and work on situated design [8] might also help here, but again this is beyond the scope of the current paper.

Let us now consider the CK operators in more detail.

In CK, K→C operators add or subtract properties, written as propositions in K-space, to or from concepts thus creating disjunctions in C-space [1]. This corresponds to the design activity of *generating alternatives*. These operators expand C-space with elements from K-space. The disjunction arises from considering that adding a property partitions the set of all extant concepts into a set of those that satisfy the property and another of those that do not. Furthermore, C→C operators expand or "flesh out" a concept by adding yet other propositions.

In ALX3d, these two kinds of operators are treated uniformly. We can add a new design proposition *d'* to *D*. To write this in ALX3d, we let R_1 and R_2 be any pair conjunctions of subsets of the requirements, such that $R_1 \wedge R_2 \Leftrightarrow R$. Then we can write:

$$B_a(R_1 \Rightarrow d') \therefore B_a(D': D \wedge d'), \text{ or} \tag{4a}$$

$$B_a(R_1 \Rightarrow \neg d') \therefore B_a(D': D \wedge \neg d'). \tag{4b}$$

Formula 4a reads that if a designer were to believe that some requirements that were theretofore unsatisfied, are satisfied by a new design proposition *d'*, then the agent would believe that the design can be improved by adding the new proposition to the current design. Formula 4b is similar, but for propositions believed to not fulfill any requirements.

In a sense, there is no real distinction in ALX3d between what CK calls calls K→C and C→C operators, because in ALX3d they are simply means to develop new concepts. If it is necessary, for some reason, to distinguish between them, then one only need recognize that K→C operators will involve knowledge operators in ALX3d while C→C operators will involve belief operators in ALX3d. The actions associated with these state transitions (implied by the ∴ operator) are actions by which a designer proposes new aspects of a design. This partitions states into those where *d'* holds and those where ¬*d'* holds.

We can see how this reasoning can be represented with ALX3d in the following hypothetical sequence of activities.

1. An initial state is assumed of $K_aR \wedge B_aD$. The design agent knows the (initial) requirements *R* and believes the (initial) design *D*.

2. The agent attempts to perform a validation action, but finds that validation cannot be done: $K_aR \wedge B_aD \therefore \neg K_a(R \Rightarrow D)$.
3. Since the validation cannot be done, either *D* or *R* must be explicated further. The agent chooses to expand *D* with a new proposition *d'*: $B_a(R_1 \Rightarrow d')\ D \therefore K_aR \wedge B_a(D \wedge d')$.
4. The agent attempts to validate the design again. This time, validation fails: $K_aR \wedge B_a(D \wedge d') \therefore K_a(\neg(R \Rightarrow (D \wedge d')))$.
   Since the first validation (step 2) could not be done, but including *d'* causes validation to fail, the only alternative is $\neg d'$.
5. The error is corrected: $K_a(\neg(R \Rightarrow (D \wedge d'))) \therefore K_aR \wedge B_a(D \wedge \neg d')$.

If validation of the design in step 5, cannot be performed, we know that neither $D \wedge d'$ nor $D \wedge \neg d'$ is a sufficient solution and that more expansion must be done to the design (and possibly the requirements).

If the validation in step 4 yielded $\neg K_a(R \Rightarrow (D \wedge d'))$, we would not be able to choose between *d'* and $\neg d'$ because neither led to a suitable design. Some alternative courses of action here include the following.

- The agent could try a different validation action since it might be the validation action itself that cannot operate on the available information in *R* and $D \wedge \neg d'$.
- The agent could pursue both $D \wedge d'$ and $D \wedge \neg d'$ as design alternatives until validation does give a distinct answer.
- The agent could change *R* and try to validate again.
- The agent could choose *d'* or $\neg d'$ based on the agent's own preferences (e.g. $d' P_a \neg d'$).
  Changing *R* is done just as changing *D*, by adding *r'* or $\neg r'$ to *R*.

If, on the other hand, the validation in step 5 *fails*, then neither *d'* nor $\neg d'$ is a suitable solution. This means that there is an error in *R*, since one of *d'* or $\neg d'$ must be true (whether the agent knows which one is true is not the point). In this case, one must use some sort of strategy to backtrack to earlier states until one finds a state in the history of changes to *R* where either *d'* or $\neg d'$ does hold.

The preceding discussion applies equally to CK's *expanding partitions* (operations that increase the number of possible design concepts) and *restricting partitions* (operations that decrease the number of possible design concepts). Expanding partitions are those for which the designer only believes *d'*, $B_a d'$; restricting partitions are those for which the designer knows *d'*, $K_a d'$.

In CK, the current state can be "backtracked" by returning to a previous state, but the theory itself does not formally describe this (for example, by some appropriate operator). In ALX3d, however, we can use *belief retraction* to formalize backtracking directly. We can write this as $D': D \backslash d'$

– that is, *D'* is like *D* but without *d'*. Backtracking in this manner applies only to beliefs and not to knowledge, because it is a principle of action logics that knowledge cannot become unknown once it is known. (Note: this is *not* the same as backtracking in logic programming languages like Prolog.)

Details of this strategy constitute future work; here it is sufficient to recognize that such representations are possible in ALX3d.

Let us now consider C→K operators, which turn a concept into knowledge in CK. In ALX3d, these are *validation actions*. Once a (design) concept becomes knowledge in CK, it is a sufficient design solution. In ALX3d, validation actions validate a belief, turning it to knowledge. As in CK, such actions include conducting mathematical analyses, experimental tests, etc. The only substantive difference is that in ALX3d the key belief that must be validated is the implication $R \Rightarrow D$, rather than the design *D* itself. In ALX3d, knowing *D* follows from knowing (validating) this implication.

Finally, CK's K→K operators expand knowledge space through logical and scientific reasoning. Any such operator is available within the first-order logic underlying ALX3d.

An example

In [10], Hatchuel *et al* present examples of the application of CK theory. In this section, the author will discuss how ALX3d can achieve at least the same level of description as CK. We will use one of the examples in [10]: the design of a new chemical (Mg-CO_2) rocket motor for use in Mars exploration missions. In [10], the case study is divided into four phases; we will follow the same layout here.

The initial state (Phase 0) is the proposal that a Mg-CO_2 engine would be "better" than the conventional solution. Per [10], we label this proposal C_0. In CK, the proposal is a *concept* because it has no logical status. In ALX3d, the proposal is a *belief*, a statement that we assume to be true and then reason with it until we can either prove or disprove it. We write it in ALX3d as B_aC_0.

In Phase 1 of the case study, an attempt was made to use the Mg-CO_2 concept for a *sample return mission* to Mars (labeled A_1 in [10]). We can write this in ALX3d as $B_a(C_0 \wedge A_1)$. An "evaluation" was then carried out by comparing the new motor to existent ones with respect to the key criterion of minimum landed mass on Mars for a sample return mission. This constitutes a validation action in ALX3d. It was found that the new motor

failed the validation; that is, the new motor is not as good as a conventional motor. In ALX3d as in CK, this only means that $K_a\neg(C_0 \wedge A_1)$. As in CK, and by the fundamental properties of first order logic, this does not necessarily imply that $\neg C_0$. So we can preserve our core belief, B_aC_0, by contending that $B_a\neg A_1$, which would account for the validation result. The designer's new belief is then $B_a(C_0 \wedge \neg A_1)$. We can write all this as:

$$(C_0P_a\neg C_0 \ \wedge \ (B_a(C_0 \wedge A_1) \ \therefore \ K_a\neg(C_0 \wedge A_1)) \ \Rightarrow \ K_a(\neg C_0 \vee \neg A_1)) \ \therefore \ B_a(C_0 \wedge \neg A_1). \quad (11)$$

Note that in CK, it is assumed at this point that A_1 is false; i.e. that the new motor will not work for a sample return mission. This is in fact incorrect. All that the agents can infer logically is that they *believe* the Mg-CO_2 motor will not work in this kind of mission; they only *know* that the combination $C_0 \wedge A_1$ will not work. At this point, CK would have us accept the validity of our main proposal C_0, but the whole point of the exercise is to determine if the concept has any merit at all. We see then that ALX3d is more expressive of the actual state of affairs in this case.

In Phase 2 of the case study, it is reported that a study conducted of mission profiles *excluding* sample return missions (i.e. $B_a(C_0 \wedge \neg A_1)$) yielded no positive results, but that this was due to an excessive number of attributes placed on the problem during evaluation. It is also suggested that CK provides a key insight here – that those excess attributes must be removed in order to discover other possible solutions. However, the current author has been unable to find a clear indication of how CK itself accommodates this. Indeed, the current author contends that this is a feature of an ontological representation of design problems as a composition of facts. This is how logic works in general, and is not a feature particular to CK. There is an old adage: *always question premises*. In this case, the premises are the attributes. Questioning them involves determining whether they are necessary or simply accepted by fiat, convention, or error.

In the case of the Mg-CO_2 motor, it is evident that all scenarios had at least one attribute in common: that the motor would be used in transit to Mars. This is the premise that is questioned in [10]. In fact, then, the belief (the CK concept) $B_a(C_0 \wedge \neg A_1)$ was interpreted *incorrectly* because the premise of using the motor in transit is not part of the concept; that is, A_1 (use for sample return missions) does not necessarily imply use in transit. It would appear then that human error is the root cause of this situation.

Let us assume, lacking other information from [10], that we should have distinguished transit as a key element of the mission profile. Let us further assume for the sake of simplicity that the mission profile can be exhaustively divided into two main segments: the transit to Mars (both go-

ing and coming) and the mission on Mars. We can rewrite the original belief $B_a(C_0 \wedge A_1)$ now as $B_a(C_0 \wedge A_{1t} \wedge A_{1m})$ where A_{1t} stands for "using the motor for transit" and A_{1m} stands for "using the motor on Mars."

Given this, the failure of the validation action noted above then tells us that if we wish to maintain C_0, then either A_{1t} or A_{1m} must be wrong. We would have then had the belief: $B_a(C_0 \wedge [A_{1t} \vee A_{1m}])$. That is, the Mg-CO2 motor might be suitable for either transit to Mars or operation on Mars, but not both.

Since all the investigated scenarios involved A_{1t}, the logical alternative here, regardless of the use of CK or ALX3d, is to use the Mg-CO2 motor for purposes other than the transit to Mars. Practically, this is equivalent to using the motor only on Mars, labeled A_2 in [10], which we can represent in ALX3d as $B_a(C_0 \wedge A_2)$, so long as we also accept that $A_2 \Rightarrow \neg A_1$.

Continuing through the case study, [10] then identifies four other attributes that constitute possible uses of the Mg-CO_2 motor on Mars: A_3 – "used for mobility," A_4 – "unplanned mobility," A_5 – "emergency lift-off," and A_5' – "additional distance." The systematic appearance of these alternatives follows from the use of CK only insofar as CK implies the use of breadth-first searches, which is our only logical course of action. A new concept is then specified in [10], which can be written in ALX3d as $B_a(C_0 \wedge A_2 \wedge A_3 \wedge A_4 \wedge A_5)$.

There are, however, two problems here. First, both this belief and its CK variant mean that the agent believes an appropriate design is a Mg-CO_2 motor used for *unplanned emergency lift-off mobility on Mars*; that is, the mission involves the *simultaneous* occurrence of A_2 through A_5, because of the logical conjunctions.

The current author believes this was not the actual intention. Rather, it makes more sense that the intention was for the new motor to be used in *any combination* of the situations denoted by $A_2 - A_5$. That is, a disjunction should have been used, i.e. $B_a(C_0 \wedge (A_2 \vee A_3 \vee A_4 \vee A_5))$, to correctly represent that any of $A_2 - A_5$ could constitute an appropriate use of the Mg-CO_2 motor.

The second problem arises from considering the nature of propositions $A_2 - A_5$. Specifically, three important facts are missing. First, A_2 is a generalization of $A_3 - A_5$; that is, "use on Mars" includes "use for mobility," "use for unplanned mobility," and "use for emergency lift-off." Second, A_3 is a generalization of A_4 and A_5; that is "use for mobility" covers both emergency and unplanned mobility. Third, some design activities must have occurred to get from A_2 to A_3 and then to A_4 and A_5; that is, for example, in moving from "use on Mars" to "use for unplanned mobility" implies some design action that identifies the required mobility as *unplanned*.

The current author therefore suggests the following ALX3d representation for situation reported in [10]:

$$((B_a(C_0 \wedge A_2) \therefore B_a A_3) \therefore B_a A_4) \therefore B_a(A_5 \vee A_5'). \quad (12)$$

Formula 12 captures a great deal about the situation:

- Initially, the agent believes the Mg-CO_2 motor is a viable alternative for use on Mars ($C_0 \wedge A_2$).
- There is a causal relation between *use on Mars* and *mobility on Mars* (A_3), so some design action must occur for the relation to hold.
- To achieve A_3, there must exist some design action (a "conceptual expansion" in CK) that moves the agent there. This is a human cognitive act connecting a means (the motor) to a desirable capability (mobility).
- Similarly, once the agent believes A_3, there is an action that will lead the agent to a new state where the mobility is *unplanned* (A_4).
- Finally, once the agent is in a state of believing A_4, there is an action that will lead the agent to believe either *emergency lift-off* or *additional distance* (A_5 or A_5') as alternate suitable situations.

We note that once we reach a state where A_5 or A_5' is true (and only in such states), then we can also say that $B_a(A_5 \vee A_5') \Rightarrow A_4 \Rightarrow A_3 \Rightarrow A_2$, which gives a causal chain back to the original propositions. Again, this demonstrates that ALX3d provides a richer representation of designing than CK, while remaining consistent with the intent and general principles of CK.

Finally, in Phase 3 of the case study, a comparison of the Mg-CO_2 concept and an alternative design, the ExoMars Rover, is reported. The concept used is that of Mg-CO_2 combustion for *unplanned mobility on Mars*; that is, A_5 and A_5' are not used. The ExoMars performance constraints are given as (a) motor weighing less than 60kg, (b) mission life of no more than 180 days, (c) maximum power consumption of 200W, and (d) minimum 10km range. These constraints are used to limit a performance domain that the Mg-CO_2 concept must satisfy. Based on existent knowledge (e.g. principles of rocket propulsion), two key design parameters for the Mg-CO_2 concept are discovered – motor mass (m_m), and mass of the CO_2 acquisition plant (m_p) – that can be used to calculate values for performance characteristic of lifetime (t), power (p), and range (r).

The values of the parameters exist within a bounded domain; any value set within the domain constitutes a possible solution, i.e. where the Mg-CO_2 motor concept can compete against the ExoMars alternative. The authors argue [10] that this opens up new possibilities for mission concepts and design alternatives that would not have been noticed otherwise.

Phase 3 is described in [10] using text and diagrams, and it is not necessarily clear which activities are derived from CK and which arise simply from the use of rational, logical reasoning in general, or innovative thinking about the problem. No matter which is actually the case, the current author will show that stages of development that occurred in Phase 3 can be represented directly in the language of ALX3d and consistently with CK. We recall that the goal is not to have ALX3d lead designers through the process, but rather to capture descriptively the reported design activities.

First, let ΔD be true only for any design concept D; that is, ΔD is a predicate that identifies design concepts. We would therefore assert ΔC_0 to mean that C_0 is a design concept. The set of all known design alternatives satisfying some propositions ψ is given in ALX3d by

$$D_a(\psi)\text{: } \{x\text{: } K_a\Delta x \Rightarrow B_a\psi\}. \tag{13}$$

The agent knows that Δx because the agent asserted it. Note that $K_a\Delta x$ (knowing that x is a design concept) does not imply $K_a(R\Rightarrow x)$ (knowing that x is an acceptable design concept for a given problem). For example, consider the previous example of bicycles with pedals and effective wings. Let B_0 be "bicycle," P_1 be "with pedals," and P_2 be "with effective wings." Furthermore, assume we were interested in finding alternatives that have wings (P_2) to "bicycles with pedals" ($B_0 \wedge P_1$). The set of alternatives is given by $D_a(P_2)$: $\{x\text{: } K_a\Delta x \Rightarrow B_aP_2\}$, which would include bicycles with pedals and any other design concept satisfying "with effective wings." Similarly, $D_a(P_1)$: $\{y\text{: } K_a\Delta y \Rightarrow B_aP_1\}$ would contain the alternatives to bicycles with effective wings that also satisfy "with pedals."

Now, returning to the Mars case study, let C_1: $C_0 \wedge A_4$; i.e. C_1 is the concept of using Mg-CO_2 motors for unplanned mobility on Mars. The designer can assert ΔC_1 as a possible design. The designers' state thus includes B_aC_1. To look for alternative concepts, we need to identify concepts that involve A_4 – all cases of unplanned mobility on Mars – but without C_0. We can write this as $C_1 \backslash C_0$. Now the set of all design alternatives is just:

$$D_a(C_1\backslash C_0)\text{: } \{x\text{: } K_a\Delta x \Rightarrow B_aA_4\}. \tag{14}$$

$D_a(C_1\backslash C_0)$ includes all the design concept alternatives to C_1. To gather these alternatives, the designers began with a belief C_1, and did the appropriate research (a C→K operator in CK) to find the design alternatives $D_a(C_1\backslash C_0)$. We can represent this as a causal relation in ALX3d:

$$B_aC_1 \therefore D_a(C_1\backslash C_0) \vee \neg D_a(C_1\backslash C_0). \tag{15}$$

That is, every subsequent state following the search for design alternatives is one that either definitely does or does not have such alternatives. Obviously, to continue the case study, we must assume that $D_a(C_1 \backslash C_0)$ is in fact the case.

One might ask: is there some feature of a state where $B_a C_1$ that draws the agent to look for alternatives? The original case study [10] only states "…the prototype should overcome the rover solution for the next known missions…." At this time, the current author can only propose that setting a goal of comparing concepts to alternatives is an extra-logical design principle. This activity may be a part of a validation action; that is, it could be one way to determine if a design concept has merit. This might suggest an axiom (a statement accepted as true but not provable within a logic) for ALX3d, but setting out exactly what this axiom might be remains an item for future study.

We can now describe this phase of the case study in ALX3d. Let the design parameters P for the Mg-CO_2 motor be m_m and m_p. Let the values of the design parameters be written as functions mapping a design parameter to a value: $m_m(d)$, $m_p(d)$. Let the performance metrics of any designs be $M: \{p, t, r\}$ (power, lifetime, range). The values of metrics can be written as functions $p(d)$, $t(d)$, and $r(d)$ for a design d.

The metric values lead to (or "cause" in ALX3d) the parameter values. That is, the case study indicates that p, t, and r were dependent values, and m_m and m_p were the independent values. In ALX3d, this is written:

$$\forall d[[m_m(d) \wedge m_p(d)] \therefore [p(d) \wedge t(d) \wedge r(d)]]. \quad (16)$$

Furthermore, the values can be partially ordered, e.g. $p(x) O_p p(y)$, for different designs, where O_p is a generalized ordering operator on p.

Constraints were defined in the case study based on knowledge of rocketry and physics. Let the constraints be written as: maximum power consumption $p = 200\ W$, expected operating life $t = 180\ days$, and minimum range of operation $r = 10\ km$. A condition for a satisficing design [5] is given by: $p(d)<p \wedge t(d)<t \wedge r(d)>r$. We can now write a satisficing goal for the Mg-CO_2 concept C_1 as a belief in a causal relation. Since the design is only a concept, we cannot *know* this satisficing relation, but only believe it. In ALX3d, we can write a satisficing goal for this case as:

$$G^s[C_1]: B_a[[m_m(C_1) \wedge m_p(C_1)] \therefore [p(C_1)<p \wedge t(C_1)<t \wedge r(C_1)>r]]. \quad (17)$$

This statement essentially captures the domain of possible values for the identified design parameters such that any design that satisfies this statement is a possible solution. This kind of *formal* representation of the goal of design activity is not available in CK theory.

We can also go beyond the case study somewhat by considering a way to find the best design within the domain of satisficing solutions given by $G^s[C_1]$. Let there be two designs based on C_1, defined by C_2: $C_1 \wedge x$ and C_3: $C_1 \wedge y$ and that both satisfy $G^s[C_1]$. We can use the formalism of trade-off goals [3] to capture the agent's preference for one satisficing design over another. Given C_2 and C_3 as defined above, and letting u, v, and z be other satisficing designs in $G^s[C_1]$, and letting φ and ψ stand for any two of the metrics, we can write the following.

$$G^t[\varphi(C_2)]: [\varphi(C_2)P_a\varphi(C_3)] \wedge [\varphi(C_2) \therefore \psi(u)] \wedge$$
$$B_a[\ \neg\exists z\ (\varphi(z)P_a\varphi(C_2) \wedge \varphi(z) \therefore \psi(v) \wedge \psi(u)P_a\psi(v)\ ], \tag{18a}$$

$$C_2P_aC_3 \Rightarrow B_a[\varphi(C_2)\ O_\varphi\ \varphi(C_3)]. \tag{18b}$$

This says that C_2 is preferred to C_3 if C_2 attains a "better" value of one of the metrics (φ) than does C_3, and doing so will not limit finding a more preferred value for one of the other metrics (ψ).

We have now developed a new model of the case study in [10] that is grounded far better in a formalism of design activities than CK can provide.

Potential Benefits of ALX3d

The author has introduced ALX3d, a formal theory of design activities built upon the action logic ALX3, and designed to account for the key features and intent of CK theory. A case study from the existent CK literature was reworked in ALX3d to demonstrate its representational richness.

The purpose of any formal model, including ALX3d, is only to provide a reasoning tool, a mechanism to allow one to reason in as rational and structured a manner about a domain. All models are by definition incomplete; else they would be indistinguishable from the thing being modeled. Models like ALX3d provide one perspective on a thing. There are other equally meaningful ways to think about design processes. ALX3d in no way discounts them; it only provides an alternative.

Furthermore, this work in no way invalidates CK. Rather, it demonstrates that the fundamental premises of CK are reasonable premises regarding the act of designing; namely, that there is an important difference between knowledge and concepts, and that a rational (logical) process can describe (but not necessarily explain) at least some parts of the act of designing. ALX3d also demonstrates the power of logical systems to capture essential aspects of design processes, especially the decisions that design-

ers must make based not only on knowledge but also on their beliefs and preferences.

ALX3d is a research tool, not something to be used by practicing designers. However, continued development of theories in mathematics and the sciences have often let eventually to practical benefits for designers. It is reasonable to assume the same could happen with logical theories like ALX3d and CK. As ALX3d matures, it will be possible to use it for several purposes in this regard, some of which include the following.

Appeal to formal systems researchers. CK theory, which has distinct benefits as a design research tool, is somewhat hindered because it does not conform to conventions of formal systems. ALX3d maintains the intent and basic principles of CK while casting it in a form more readily understood by those with grounding in formal systems. As such, ALX3d makes CK theory more appealing to the community of design researchers who understand and use formal systems, including researchers in artificial intelligence, computer science, and cognitive science.

Reasoning about documented design processes. Assuming a complete description of a design process as documented either in industry or the literature can be constructed (and the author currently believes this is entirely possible), then the description can be reasoned about using the inference rules that are built into ALX3 to study the process, and find and address its problematic aspects. This would significantly advance our understanding of the nature of engineering design.

Construction of new design processes. It may well be that in the natural course of analyzing design processes, new process descriptions may arise that could significantly improve the design capability of a group of designers. The description of rules for establishing trade-off goals, and for switching between different types of design tasks – as outlined in the preceding sections – are examples of this. It may be that new, "industry-strength" methods can be developed by considering different ways that such activities can be described in ALX3d.

Construction of new computer-based design aids. Logical systems are well suited to implementation in computer tools. It should be possible to use ALX3d to develop new design applications of artificial intelligence and knowledge-based systems. Such systems may also yield significant advantages for practicing designers. For example, it may be possible to develop tools that will suggest sequences of design activities that are more likely to lead quickly to better (or at least satisficing) designs. One may also envision case-based reasoning engines that use sequences of actions as cases.

Conclusion

ALX3d, a reformulation of CK theory based on the action logic ALX3, has been introduced. Beyond what is currently possible with CK, ALX3d leverages ALX3 to provide a richer framework for describing design activities in formal terms. While adding support to the CK approach, ALX3d also demonstrates the potential benefits of using formal systems in design research. Although ALX3d is still being developed, there are strong indications, as demonstrated in this paper that it may be a useful tool for design research.

References

1. A. Hatchuel and B. Weil (2009) CK design theory: an advanced formulation. Research in Engineering Design, 19(4):181–192.
2. A.O. Kazakci and A. Tsoukias (2004) Extending the CK design theory to provide theoretical background for personal design assistants. in D. Marjanovic (ed), DESIGN 2004, pages 45-52.
3. F.A. Salustri (2003) Towards an action logic for design processes. Proc. Intl. Conf. on Engineering Design, paper #1051.
4. Z. Huang (1994) Logics for agents with bounded rationality. PhD Thesis, University of Amsterdam.
5. H.A. Simon (1981) The Sciences of the Artificial. The MIT Press, Cambridge, Mass.
6. V. Akman and M. Surav (1996) Steps toward Formalizing Context. AI Magazine, 17(3):55-72.
7. R.J. Brachman, D.L. McGuinness, P.F. Patel-Schneider and L.A. Resnick (1991) LIVING WITH CLASSIC: When and How to Use a KL-ONE-Like Language. In Principles of Semantic Networks: Explorations in the Representation of Knowledge (ed. John F. Sowa); Morgan Kaufmann Publishers, Inc., San Mateo; pages 401-456.
8. J.S. Gero (2004) Situated design computing: Introduction and implications, in D. Marjanovic (ed), DESIGN 2004, pages 27-36.
9. M.J. French (1992) The Opportunistic Route and the Role of Design Principles. Research in Engineering Design, 4(3):185-190.
10. A. Hatchuel, P. Le Masson, and B. Weil (2004) CK theory in practice: lessons from industrial applications. In D. Marjanovic (ed), DESIGN 2004, pages 245-257.
11. I.M. Copi (1979) Symbolic Logic, 5/e. Prentice-Hall.