Full Title of Article:

Formalization Of Theories of Design Using: First Order Logic Augmented With A Causal Calculus; And Entropy of a Design Defined In Terms Of The Function, Behavior and Structure Ontology.

Author: Karthik Gurumurthi

Affiliation: Light Pharma, Inc., Cambridge, MA, USA

Email Address: karthik.gurumurthi@gmail.com

Number of Pages in Manuscript: 92

Number of Tables in Manuscript: 6

Number of Figures in Manuscript: 8

Formalization Of Theories of Design Using: First Order Logic Augmented With A Causal Calculus; And Entropy of a Design Defined In Terms Of The Function, Behavior and Structure Ontology.

Abstract:

A symbolic logical framework (*L*) consisting of first order logic augmented with a causal calculus has been provided to formalize, axiomatize and integrate theories of design. *L* is used to represent designs in the Function-Behavior-Structure (FBS) ontology in a single, widely applicable language that enables the following: seamless integration of representations of function, behavior and structure; and generality in the formalization of theories of design. FRs, constraints, structure and behavior are represented as sentences in *L*. FRs are represented (as abstractions of behavior) in the form of existentially quantified sentences, the instantiation of whose individual variables yields the representation of behavior. This enables the logical implication of FRs by behavior, without recourse to apriori criteria for satisfaction of FRs by behavior. Functional decomposition is represented to enable lower level FRs to logically imply the satisfaction of higher level FRs. The theory of whether and how structure and behavior satisfy FRs and constraints is represented as a formal proof in *L.* Important general attributes of designs such as solution-neutrality of FRs, probability of satisfaction of requirements and constraints (calculated in a Bayesian framework using Monte Carlo simulation), extent and nature of coupling, etc. have been defined in terms of the representation of a design in *L*. The entropy of a design is defined in terms of the above attributes of a design, based on which a general theory of what constitutes a good design has been formalized to include the desirability of solution-neutrality of (especially higher level) FRs, high probability of satisfaction of requirements and constraints, wide specifications, low variability and bias, use of fewer attributes to specify the design, less coupling (especially circular coupling at higher levels of FRs), parametrization, standardization, etc..

Key words: Formalization of theories of design, first order logic, causal calculus, Function-Behavior-Structure, entropy of a design.

1. Introduction

The past few decades have seen the concurrent and integrated development of the science of design as well as overarching frameworks of practice of design (e.g. Simon(1996), Papalambros (2015), Pahl and Beitz (1988), Hubka and Eder (1988), Pugh (1991), Suh (1990), Ullrich and Eppinger (1995), Otto and Wood (2001), etc.). The relationship between the science of design and frameworks of the practice of design can be seen to be analogous to the relationship between the natural sciences and engineering. The natural sciences provide knowledge and theories about objects and phenomena of the natural world that form the basis for the design of systems to transform the natural world from one state to another. Similarly, the science of design should provide knowledge and theories about the general artifacts and processes of design, that should then form the basis for the development of the practice of design. Scientific method (e.g. Peirce (1878), Fann (1970), Popper (2005)) for the natural sciences is generally characterized by the construction of formal, axiomatic theories involving the following: logical induction and/or abduction to frame axioms or premises; deduction (from axioms or premises) to explain observations and make predictions; and induction (in the form of tests of hypotheses) to negate or corroborate predictions, thereby falsifying or corroborating a scientific theory. The success of the scientific method in the natural sciences is, at least partly, due to the construction of integrated, axiomatic theories that explain a number of phenomena in a variety of domains, based on a relatively small number of axioms. Similarly, the development of the science of design needs the development of formal, axiomatic, integrated theories that describe, explain, predict and allow the negation or corroboration of propositions about designs (e.g. in terms of their functions, structure and behavior) and design processes (that are used to develop design artifacts). Theories of design could pertain to the following: a) What constitutes a design; b) *How* a given design brings about a transformation in its external environment (i.e., theories about *how* a design satisfies its requirements and constraints); c) What is a good design; c) What constitutes a design process; d) *How* does a design process yield a design; e) What is a good design process; etc.. Since the above types of theories of design are interrelated (e.g. how a design satisfies its requirements is likely to influence the assessment of the quality of the design, and a good design process would be expected to yield good design artifacts) a unified and standard

formal language and logic is required to express, integrate and axiomatize the various types of theories of design mentioned above.

The question then arises as to what might be a suitable formal logic (or logics) for the design sciences. The answer to this depends on the general types of representation and reasoning needed in the above types of theories of design. Simon (1996) broadly defined the activity of designing as devising courses of action to change existing situations to preferred ones, and posited that the design artifact forms the interface between its internal and external environments. A theory of how a design brings about a desired change in its environment needs a definition of a design in terms of its constituent elements, and their relationships with one another and with the environment, i.e., an ontology of design artifacts and processes. A product design process involves various activities executed in an iterative and integrated manner (Pahl and Beitz (1988), Hubka and Eder (1988), Gero (1990), Suh (1990), Ullrich and Eppinger (1995), Otto and Wood (2001)), including the following: Defining the requirements and constraints that the product (e.g. automobiles, computers, etc.) should satisfy; Defining the structure of the product components; Specifying and/or anticipating the behavior of the product; and Evaluating the product for the satisfaction of its requirements and constraints. Pahl and Beitz (1988) describe a hierarchical approach of defining the functions of the product and choosing devices to execute the functions until a fully detailed product is realized. They describe the functions of the product as transformations of input materials, signals and energy into output materials, signals and energy. Individual functions are achieved using working principles (mechanisms), and the overall hierarchy of functions being achieved by the integration of working principles into a working structure. The behavior of the working principles and working structures is implicit in the representations or descriptions of the working principles and structures. Hubka and Eder (1988) proposed that the transformation of the environment from a given state to a desired state is achieved through a transformation system consisting of inputs, a transformation process using a transformation technology, and various operators (human, technical, information and management systems) that together enable the transformation process to achieve the desired state of the environment. The artifact that is designed itself is a technical system consisting of the following: inputs (materials, energy, information); functions (capabilities of

the product that produce the desired effects on the environment); organs (system of parts that achieves a given function); constructional parts; chains of actions caused by organs under the given inputs; outputs; and effects on the environment. The hierarchy of the transformation system consists in each operator itself being a transformation system. Suh (1990) proposed that the requirements of a product be analyzed into functional requirements (FRs) and constraints, and that design parameters be identified to satisfy FRs in such a manner that the resulting design is functionally uncoupled or decoupled design and has the least information content, i.e., the highest probability of satisfaction of FRs and constraints. Gero (1990) proposed the Function-Behavior-Structure (FBS) ontology wherein: functions define the purpose of the product; the structure defines the organization of the components of the product; the behavior of the product is derived from its structure, and verified against expected behavior derived from the functions of the product. Gero (1990) represents functions, in a language of commands, as combinations of verbs and nouns (e.g. “provide daylighting”), and behavior using the language of qualitative physics (Bobrow (1984)). Umeda, et al. (1990, 1996) proposed a Function-Behavior-State diagram, wherein the state of the product subsumes its structure. They represented functions as intentions to perform an action in a certain manner (e.g. “to move table precisely”), and described the causal decomposition of functions (analysis of a function into causally related sub-functions) to guide the selection of devices whose behavior satisfies the causal relationships among functions. They represented behavior using the qualitative physics of processes (Forbus (1984)). They assessed the satisfaction of functions using an apriori mapping between functions and expected behavior, simulating the behavior of the structure of the design, and evaluating whether the simulated behavior satisfies expected behavior. Vescovi, et al. (1993) represented functions in their Casual Functional Representation Language (CFRL) as logical combinations of causal process descriptions (CPDs), i.e., networks of temporally and causally related nodes, wherein each node is defined in terms of conditions to be satisfied by expected behavior, in a certain physical context and by a certain device. The above representation of functions constituted an abstraction of expected behavior. They represented behavior as temporally or causally ordered trajectories of states, simulated using qualitative physics (Forbus (1984, 1990)). The evaluation of satisfaction of functions by behavior was implemented by verifying whether the states constituting behavior satisfy the conditions for behavior specified by corresponding nodes constituting the functions. They also used

operators such as ALWAYS and SOMETIMES to characterize the extent of matching between CPDs and behavior. Goel, at al. (2009), in their Structure-Behavior-Function (SBF) framework, extended CFRL to include the concept of substances (attributes of objects such as angular momentum), bidirectional causality, etc.. Functions in Gero's (1990) ontology correspond to functions in the ontologies of Pahl and Beitz (1988), Otto and Wood (2001), Vescovi, et al. (1993) and Umeda, et al. (1990) and to organs in Hubka and Eder (1988). Structure in Gero's (1990) ontology corresponds to the concept of structure in the ontologies of Pahl and Beitz (1988), Otto and Wood (2001), Vescovi, et al. (1993), to the concept of state as a generalization of structure in Umeda, et al. (1990) and to the concept of component structure of Hubka and Eder (1988). Behavior in Gero's (1990) ontology corresponds to the concept of behavior in Vescovi, et al. (1993) and Umeda, et al. (1990), to the concept of action sequences in Hubka and Eder (1988), and is implicit in the concept of working principle of Pahl and Beitz (1988). Thus the FBS ontology is apparent in a variety of representations of design, as described above.

A central question pertaining to any design is: "*How* will the design satisfy its requirements and constraints?". An explanation of how a design will satisfy its requirements and constraints can be seen as a theory of satisfaction of requirements and constraints by the structure and behavior of the design. Formalization of theories of design also requires a formalization of the theory of how a design satisfies its requirements and constraints. Such a theory would enable the evaluation of whether a design makes logical sense and its ability to satisfy its requirements and constraints. A theory of satisfaction of requirements and constraints by a design needs a formal language and logic for its representation. In the FBS ontology, the use of different languages for representing functions and behavior, respectively, was partially motivated by the preference for retaining the flexibility of choosing a language to simulate behavior (e.g. qualitative physics) independently of the choice of language to represent functions (Vescovi, et al. (1993)). However, using different languages to represent functions and behavior, respectively, requires the apriori specification of criteria for the satisfaction of functions, either in the language of behavior as in Gero (1990) or in the language of functions as in Vescovi, et al. (1993). On the other hand, behavior, by its very definition, could be shown to satisfy functions without the need for pre-

specified criteria for satisfaction of functions (e.g. the behavior of an internal combustion engine generating 75 horse power of mechanical energy at 2000 cycles per minute, by definition, satisfies the function of generating mechanical energy). This kind of satisfaction of functions by behavior is a logical implication of functions by behavior. Behavior can be shown to logically imply functions, without recourse to apriori criteria for satisfaction of functions by behavior, only in a common language used to represent both functions and behavior, accompanied by a logic that enables formal proofs of satisfaction of functions by behavior. Further, the modeling of behavior in Gero (1990), Vescovi, et al. (1993) and Umeda, et al. (1990) is based on qualitative physics, which is suited to model the behavior of physical systems (Forbus (1984)), but not the behavior of any general system. Thus there is a need for a single, standard, formal language and logic to represent, generally and in an integrated manner, functions, behavior, structure and theories of how structure and behavior realize functions.

Product design theories often identify general attributes of a good design, e.g.: solution-neutrality of functions; independence of functional requirements; functional uncoupling or decoupling (modularity); standardization, symmetry; minimal information content; etc. ((Pahl and Beitz (1998), Suh (1990), Ullrich and Eppinger (1995)). Suh (1990) organized attributes of a good design into an axiomatic system, wherein a subset of these attributes are identified as axioms (e.g. independent functional requirements, minimal information content) and others are derived from these axioms (e.g. functional uncoupling or decoupling). In order to integrate various attributes of a good design proposed by various theories of design into a cogent, unified system, the general attributes of a good design have to be interpreted in terms of a common or standard representation of a design. The FBS ontology, given its use various theories of design as described above, is a good candidate for such a common or standard representation of design. Further, the general attributes of a good design, such as those mentioned above, have to be organized into a system of propositions in a manner that enables the following: a) assessment of internal consistency of the system of propositions; b) identification of axiomatic and derived propositions; and c) identification of additional propositions entailed by the existing axiomatic and derived propositions. Apart from formally unifying diverse theories of what constitutes a good design, such a formal theory might also yield additional desirable attributes of a design not contained in the original set of attributes represented in the theory.

Theories of design processes could be descriptive (e.g. they might describe how the activity of designing is carried out) or prescriptive (e.g. they might describe how the activity of designing should be carried out) (Finger and Dixon (1989), Chakrabarti and Blessing (1995)). Prescriptive design theories could be expressed as a mixture of descriptions (e.g. definition of a function in (Pahl and Beitz (1988)) as well as prescriptions (instructions or imperative sentences) (e.g. ensure the solution-neutrality of functions (Pahl and Beitz (1988)). For example, the Axiomatic Design Theory (ADT) of Suh (1990) often uses prescriptive language such as "*Maintain* the independence of Functional Requirements" to describe a good design process. Logics whose sentences are imperatives (commands) are modal logics (logics that qualify the truth value of a proposition, using qualifiers such as "necessarily true", "should be true" (language of commands), "intended to be true", "will be true in the future", etc.) (Cresswell, et al. (2012)). Also, the logic of the language of functions of Gero (1990) (language of commands), Umeda, et al. (1990) (language of intentions), Vescovi, et al. (1993) (use of operators such as ALWAYS and SOMETIMES) and Suh (1991) (language of commands), as described above, can all be considered as forms of modal logic. Simon (1996) pointed out paradoxes that arise in imperative logic (the logic of commands) when directly combined with the rules of declarative logic. For example, the imperative in ADT of "Maintain the independence of FRs" when represented in symbolic logical form Maintain(Independence(FRs)) could paradoxically imply Maintain (Independent(FRs) v ~Independent(FRs)), since in declarative logic any proposition A logically implies A or ~A. Such paradoxes can often be found in modal logic (e.g. Sometimes(A) implies Sometimes(A or Not A) if rules of declarative logic are directly applied, which is a contradiction, since "A or Not A" is always true in declarative logic). To avoid such paradoxes Simon (1996) proposed that predicate (first order) logic (Boolos and Jeffrey (1989) is adequate for the application of the scientific method in design. He proposed that imperative statements be reduced to declarative statements through the use of utility functions. For example, if a utility function for a design is proposed that carries a higher value for a design whose FRs are independent, then the imperative "Maintain the independence of FRs" can be reduced to stating that a design whose FRs are independent has a higher utility. Thus, since utility functions

themselves can be adequately represented and reasoned about within the framework of predicate (i.e. first order) logic, predicate logic is sufficient to formalize the science of design (Simon (1996)).

The activity of designing can be seen as a form of planning (Simon(1996)). The situation calculus (McCarthy and Hayes (1981)) was introduced to formally represent and reason about plans in an artificial intelligence context. A situation is defined as "the complete state of the universe at an instant of time". Each situation is associated with an instant of time. Changes in situations over time are represented using fluents, which are functions of situations ranging over situations (situational fluents) or truth values (propositional fluents). Fluents are also used to represent the results of actions. Causality (i.e. one situation causing another) is represented, using situational fluents, as a proposition A in a situation x causing a situation y to come about in which the proposition B is true. Attempts to formalize theories of action and change (including the situation calculus), engendered problems in maintaining the consistency the theory in the face of incomplete knowledge of situations. Reasoning under conditions of incomplete knowledge is characteristic of common sense reasoning (Mueller (2014)). Formalization of common sense reasoning about actions and change requires ensuring the consistency of the theory as new knowledge about the world becomes available. This sometimes necessitates revising the premises or conclusions of the theory as new knowledge about the world becomes available. In contrast to classical logic which is monotonic in nature (i.e., the truth values of sentences in a theory are not subject to change), formal reasoning under conditions of incomplete knowledge is non-monotonic in nature (i.e., the truth values of sentences in a theory could change as new knowledge is incorporated as premises into the theory) (McDermott and Doyle (1980), Reiter (1980)). Non-monotonic reasoning engenders problems in maintaining the consistency of theories of action and change as new knowledge becomes available. These include the following: the Frame Problem (i.e., the problem of having to specify a potentially infinite number of fluents whose values do not change over situations or time) (McCarthy and Hayes (1981)); Qualification Problem (i.e. the problem of having to specify a potentially infinite number of preconditions for a change) (McCarthy (1977)); and the Ramification Problem (i.e. the problem of having to specify a potentially infinite number of consequences of a change) (Finger (1987)). Methods such as default reasoning with the closed world

assumption (Reiter (1980)), circumscription (McCarthy (1980)) and explicit representation of cause-effect relationsips (e.g. McCain and Turner (1997)) are used to address the above problems. Default reasoning uses rules of inference that are known mostly to be true (e.g if x is a bird, then x can fly), unless there is specific information based on which to conclude otherwise (e.g. x is a penguin and penguins cannot fly). The closed world assumption (Reiter (1978)) is the assumption that if a certain sentence is not explicitly stated to be true of a given object, then it is assumed that the sentence is not true of the object (e.g. if it is not explicitly stated that x is a penguin, then it is assumed that x is not a penguin). In a similar vein, circumscription involves apriori identifying all the arguments of a predicate for which sentences containing the predicate are true in a given theory (e.g apriori identifying all the x's for which "x is a bird", and "if x is a bird, then x can fly" are true). The explicit representation of cause-effect relationships (e.g. as causal laws (McCain and Turner (1997)) mitigates the Ramification problem by distinguishing between the following: inferences of effects from causes; and inferences of causes given evidence of effects. The event calculus (Kowalski and Sergot (1989), Shanahan(1999)) represents and reasons about change in terms of events pertaining to specific objects, predicates and times of interest, in contrast to the situations of situation calculus that pertain to the state of the entire universe at a given point in time. The event calculus (Shanahan (1999)) represents temporal changes as events that occur at points in time or propositions that hold over points or durations of time. Causality is represented in the event calculus as a material implication of events or propositions that hold over time by events or actions causing them. Pearl (2000) organized causal knowledge within the framework of structural causal models, consisting of exogenous variables, endogenous variables and functions causally relating exogenous to endogenous variables, to make predictions, identify potential interventions (actions to achieve a desired effect) and hypothesize about counterfactuals. In structural causal models, the structure of causal relationships is represented as a directed acyclic graph (DAG) whose nodes represent parameters of interest (e.g. fluents) and whose arcs represent causal relationships among the parameters of interest. The value of a node in the DAG of causal relationships is determined as a function of the values of its immediate parents in the DAG. Pearl (1988) pointed out the importance of explicitly taking into account causality in default reasoning to reflect differences between causal and evidential support for inferences (a proposition inferred as the effect of a cause known to

have occurred weakens the basis for abducting the occurrence of other potential causes for the same effect). He categorized default rules (inferences) as causal (e.g. expectation of smoke from fire) or evidential (e.g. inference of fire from smoke). Geffner (1990) formalized causal theories in propositional modal logic. McCain and Turner (1997) formalized causal theories, in terms of propositions and actions, as sets of causal laws (of the form A => B to be read as "If A is true, then B has a causal explanation", which includes the case of A causing B, and where A can be a logical combination of propositions or actions, and B is a proposition). Bochman (2003, 2004) provided a causal calculus that integrated causal theories (involving propositions and not actions) with propositional logic. The causal calculus of Bochman (2003, 2004) was generalized to enable causal explanation of formulas expressible in first order logic (Lifschitz (1997), Lifschitz and Yang(2013)), which also enabled the logical representation of structural causal models of Pearl (2000) (Bochman and Lifschitz (2015)).

The logical representation of the FBS ontology of a design requires the representation of the following: objects (e.g. parts, sub-assemblies, etc.); attributes of objects (e.g. dimensions, materials, position, etc.); relationships among objects (e.g. a piston being a part of the piston sub-assembly); behavior of objects (e.g. rotation of crank from 0 degrees to 180 degrees causes the piston to move from top dead center position to bottom dead center position and inlet valve to open, etc.); FRs as abstractions of behavior (e.g. crank is rotated, piston is retracted, etc.); constraints on FRs (e.g. ranges of power generated within corresponding ranges of frequencies of crank rotation), etc.. Quantification (universal or existential) over objects such as parts or instants of time will be an important property to have in the representation of theories of design (e.g. "for all cylinders in the IC engine, the length of the cylinder is 90 mm" can be used to standardize the specifications of the cylinders in the IC engine; and "there exists time instants t1 and t2 such that in the duration [t1, t2], the power generated by the IC engine is greater than 0", is a statement that can be used to logically represent the FR of "power is generated by the IC engine"). Further, any scientific theory progresses from observing correlations and explaining them using causal theories (e.g. correlation between an attribute A (e.g. proportion of unburnt fuel in exhaust gas of an IC engine) and attribute B (e.g. air-fuel ratio at the inlet of the IC engine) of a design could be discovered to be due to a causal relationship between A and B (a lower air-fuel ratio causes a higher proportion of unburnt fuel to be

present in the exhaust gases of the IC engine due to reduction in the quantity of oxygen that is required to burn the fuel) or due to a common cause C affecting both A and B (e.g. the correlation between the speeds of the piston and valves being can explained by the causal relationship between the speed of the crank shaft and the speeds of the piston and valves, etc.). Thus a formal theory of why a design displays a certain behavior or how a design satisfies its requirements would also include the representation of causation of states or events by other states or events.

The situation calculus (McCarthy and Hayes (1981) treats both physical things (e.g. parts) as well as situations as objects (subject to universal and existential quantification), while defining a situation as "the complete state of the universe at an instant of time". The concept of situation in the situation calculus lacks intuitiveness in the context of designing, which is done in very specific contexts and not taking into account the state of the world as a whole. Further, intuitively, from the standpoint of classical first order logic, a situation could be seen as the conjunct of all the predicates that hold true for the objects of the universe at a given point in time. Therefore, quantification over situations could intuitively seen be as quantification over predicates. Thus, from the standpoint of classical first order logic, the situation calculus has the implicit structure of a higher (second) order logic. The event calculus (Shanahan (1999)) does not quantify over things (e.g parts) in its ontology, and therefore is also not an intuitive choice to represent the FBS ontology of a design. On the other hand, classical first order logic can intuitively represent the following: parts and sub-assemblies as objects; the attributes and relationships among parts as predicates; behavior as logic programs involving first order formulas, FRs as existentially quantified first order formulas (obtained by replacing individuals in first order formulas representing states constituting behavior with existentially quantified individual variables), constraints as formulas in classical first order logic, etc.. Higher order logics, such as second order logic (Boolos and Jeffrey (1989)), provide more expressive power by allowing quantification over predicates, functions and sentences, which is not allowed in classical first order logic. However, higher order logics also have important limitations (e.g. it is not always possible to assess whether a theory in second order logic is consistent (i.e., free of contradictions) (Boolos and Jeffrey (1989)), and perhaps consequently not as widely used as classical first order

logic. A limitation of classical first order logic, relevant to the representation of theories of design, is that it does not have the means to explicitly infer effect from cause in a manner that is distinct from other kinds of implication (e.g. inference based on correlation). The causal formalisms of Geffner (1990), McCain and Turner (1997) and Bochman (2003, 2004) do not use the language of classical first order logic to represent causal theories. The formalisms of Lifschitz (1997) and Lifschitz and Yang (2013) can represent causal theories in terms of first order formulas. However, the logical representation of causal rules (e.g. A => B) in the formalisms of Geffner (1990), McCain and Turner (1997), Lifschitz (1997), Bochman (2003, 2004) and Lifschitz and Yang (2013) pertain to inferring from A whether B has a causal explanation, rather than stating that A causes B (possibly originating in Pearl's (1988) idea of distinguishing statements with causal explanations from statements with non-causal (e.g. evidential) explanations in non-monotonic reasoning). However, the simplest representation of causality in behavior would be statements such as "A causes B" rather than statements such as "A implies that B has been caused".

Simon (1996) described a complex system as one having a large number of parts with many interactions, and also described the importance and pervasiveness of hierarchy in complex systems. The Information Axiom of ADT (Suh (1990)) expressed the probability of a design satisfying its FRs in terms of information entropy (Shannon (1948)). Braha and Maimon (1998) analyze the complexity of a design into structural (in terms of the number and variety of symbols used to represent the design) and functional (in terms of the probability of satisfaction of requirements and constraints) complexity. They quantify the level of abstraction of a design, in terms of its information content, relative to that at the highest level of abstraction of the design. The definition of information content by Braha and Maimon (1998) allows for the explicit characterization of the impact of standardization and symmetry of parts on the information content of a design. El-Haik and Yang (1999) expressed the complexity of a design in terms of Boltzmann entropy having the following components: variability (pertaining to the probability of satisfaction of requirements and constraints); vulnerability (reflective of the size of the design problem in terms of number of FRs, sensitivity of FRs to design parameters, and dependencies among design parameters); and correlation (among design parameters). Frey, et al. (2000) provided

approaches to compute the information content of decoupled designs, and showed that summing information content without regard to the dependencies among design parameters (DPs) could substantially overestimate the information content of a design. Guenov (2002) provided a measure of entropy, associated with coupling in the design matrix of ADT, for use in high level design. Khan, et al. (2007) used the concept of entropy to measure the diversity of components to satisfy functions, i.e. they measured, to an extent, the standardization of components used in a design. Modrak, et al. (2015) applied the measure of entropy of coupling developed by Guenov (2002) to characterize the complexity associated with mass customization of products. Krus (2013, 2015) characterized the evolution of a design through various phases (such as: the generation of design space, i.e. the set of all possible designs; generation of concepts; screening and optimization of concepts) in terms of reducing entropy (i.e. increasing information) corresponding to each stage of the design process. Krus (2013, 2015) also enabled the explicit characterization of the impact of standardization and parametric design of parts on the information content of the design space.

Taking into account the considerations discussed above pertaining to a suitable formal logic for the science of design, this work provides a symbolic logical framework (L), consisting of classical first order logic augmented with a calculus of causal processes (to represent statements such as "A causes B", where A and B are sentences in the language of classical first order logic) to formalize, axiomatize and integrate the following types of theories of design: a) What is a design (formally represented as a system of requirements (including FRs), constraints, structure and behavior); b) Whether and how a design satisfies its requirements and constraints (represented as a formal theory, based on monotonic reasoning, of satisfaction of requirements (including FRs) and constraints by the structure and behavior of the design); and c) What is a good design (represented as a formal, axiomatic theory of general attributes of a good design). To formalize and axiomatize the theory of what constitutes a good design (i.e., the general attributes of a good design), this work does the following: a) defines general attributes of a good design (e.g. those pertaining to the probability of satisfaction of requirements and constraints and change manageability of the design) in terms of the representation of a design in L; b) provides a measure of entropy(Shannon (1948), Lebowitz (1993)) of a design, which generalizes the concept of information

content of Suh (1990) to include other general attributes of a design (e.g. dimensionality, solution-neutrality of FRs, change manageability, etc.); and c) posits a utility function of a design that serves as an index of the quality of the design from a design-theoretic perspective (i.e., the extent to which the design exhibits general attributes of a good design such as solution-neutrality of requirements, high probability of satisfaction of requirements and constraints, change mangeability, etc.) that is defined to vary monotonically and inversely with the entropy of the design (from here onwards where quality of the design is mentioned, it means the quality of the design from a design-theoretic perspective, as described above). This utility function representing the quality of a design from a design theoretic perspective enables the re-framing of imperative statements (e.g. ensure the solution-neutrality of FRs) as declarative statements in predicate logic (e.g. improving the solution-neutrality of FRs reduces the entropy of the design, and therefore increases the quality of the design), as suggested by Simon (1998).

Since the emphasis of this work is on representing theories of design in a suitable logical framework rather than automating the generation or improvement of designs, the type of reasoning focused on in this work is monotonic reasoning. However, the use of a causal calculus in this work naturally mitigates the Qualification and Ramification problems (McCain and Turner (1997)). This could further be strengthened within the logical framework used in this work by methods of circumscription involving inclusion of statements such as the following: a) A v B v C <=> D if only A or B or C are explicitly stated in the theory as being the direct causes of D; A & B & C <=> D if only A, B and C in conjunction are explicitly stated in the theory as being the direct causes of D (analogous to the literal completion approach of McCain and Turner (1997), to limit the immediate causes of an effect to only those that are explicitly stated in the theory); b) If for all values of an individual variables x, y,.. within the corresponding ranges of values R1, R2, ... , a given sentence A is true, then include a sentence in the theory that states that the sentence A is true only for the values of x, y,... within their respective ranges R1, R2,…(an example of predicate completion (McCarthy (1980)). The above two approaches of circumscription have been illustrated in this work (see footnote for Table 3), and can be used to apply *L* in a non-monotonic setting.

Section 2 describes the symbolic logical framework (*L*) for the science of design. Section 3 discusses the representation of designs (as systems of requirements (including FRs), constraints, structure and behavior) in *L*. Section 4 presents the following types of formal theories of design: a) Whether and how the structure and behavior of a design satisfy its requirements and constraints (expressed as a formal proof of satisfaction of requirements and constraints by the structure and behavior of the design); and b) An axiomatic theory of what constitutes a good design (i.e. the general attributes that characterize a good system of requirements, constraints and structure and behavior). Section 5 illustrates the representation and analysis of designs in the symbolic logical framework *L* using the example of a schematic IC engine. Section 6 provides a summary and discussion of this work. Appendix 1 provides definitions of important concepts used through the remaining sections of this work. Appendix 2 provides the calculus of causal processes.

2. Symbolic Logical Framework *(L)*

The symbolic logical framework (*L*) described in this work consists of a language and logic to describe and reason about objects and processes. *L* builds on classical first order (i.e. predicate) logic (Boolos and Jeffrey (1988)) by incorporating a calculus of causal processes. The aspect of *L* that is predicate logic enables the description and reasoning about states of objects. The calculus of causal processes of *L* enables the description and reasoning about causal relationships among states of objects over time.

In this work: objects having certain attributes or being in certain relationships with other objects is called a state; a state that holds for a duration of time is called a state-duration (SD); a process is defined as a description of states of objects over time; and a causal process is a process that includes descriptions of causal relationships among states or SDs. An SD is a sentence in classical first order logic that describes a state holding over a specified duration of time. An SD is abbreviated in *L* as (S, D), wherein S denotes a state and D a duration of time. The representation of an SD in the form (S, D) in *L* is an abbreviation for a sentence in classical first order

logic. For example, in this work, an SD abbreviated as (S, [t_1, t_2]) can represented in classical first order logic as follows:

- $\forall t ((t \geq t_1 \,\&\, t \leq t_2 \,\&\, t_1 \leq t_2) \Rightarrow A(a, b, .., t))$, where A is a predicate with the individuals a, b,..,t as arguments.
- $\forall t((t \geq t_1) \,\&\, (t \leq t_2)) \Rightarrow \forall x \exists y... A(x, y, \ldots, t))$, where A is a predicate with the individual variables x, y,..,t as arguments.

Processes (as defined in Appendix 1.1) are represented in *L* in the following ways: a) SDs; b) sentences formed by conjunction or disjunction of states and at least one SD; and c) sentences containing one or more causal implication ($^{C}\Rightarrow$) signs. Sentences containing one or more causal implication ($^{C}\Rightarrow$) signs represent causal processes in *L*. Each causal implication ($^{C}\Rightarrow$) sign corresponds to a causal process. The sentence, formed from states or SDs only using the operations of negation, conjunction or disjunction, which is immediately to the left of the causal implication ($^{C}\Rightarrow$) sign is called the input sentence of the corresponding causal process, and that immediately to the right of the causal implication ($^{C}\Rightarrow$) sign is called the output sentence of the causal process. The use of *L* to represent recursive processes is described in Appendix 1.1. Definitions of objects, attributes, states, SDs, processes, etc. are given and their representation in *L* described in Appendix 1.1. The language of *L* is defined in Supplementary Tables 1 and 2. The calculus of causal processes is given in Appendix 2.

3. Representation of the design of an artifact in the symbolic logical framework of *L*

A design is represented in the language of *L* as a set of sentences stating the requirements (including FRs), constraints, structure and behavior of the design. Requirements (including FRs), constraints and structure are represented in *L* as sentences that are states, SDs, or logical combinations of states or SDs using negation, conjunction or disjunction. In addition, every sentence representing an FR in *L* contains one or more existentially quantified individual (object) variables, as described further in the following paragraph. Behavior is

represented in *L* as a temporal or causal process. Sentences in *L* that represent behavior as a causal process have one or more instances of the causal implication (^{C}=>) sign.

In this work functional requirements (FRs) are expressed as SDs to be achieved by the design (e.g. similar to the definition of function as an effect in Chandrasekaran and Josephson(1997)). FRs are represented in *L* as SDs with existential quantifiers. For example, the FR to generate power is representable in *L* as: $\exists t_1 \exists t_2 \exists v \forall t((t \geq t_1 \ \& \ t \leq t_2 \ \& \ t_1 \leq t_2) \Rightarrow (\text{Power(IC Engine)} = v \ \& \ v > 0))$ (i.e., there exists a time at which the power of the IC engine is greater than zero). Constraints that characterize FRs, e.g. specified range of power of an IC engine (also called performance requirements (PRs) in this work), can be obtained from FRs with values specified for one or more existentially quantified variables. For example, the performance requirement that an IC engine generate 75 hp of power (representable in *L* as: $\exists t_1 \exists t_2 \forall t((t \geq t_1 \ \& \ t \leq t_2 \ \& \ t_1 \leq t_2) \Rightarrow (\text{Power(IC Engine)} = 75 \text{ hp}))$ can be obtained from the representation of the corresponding FR to generate power by specifying the value of 75 hp for the existentially quantified variable *v* in the representation of the FR given above. Similarly, expected behavior of the design can be obtained from an FR or PR by specifying values for one or more existentially quantified variables in the FR or PR. For example the expected behavior of generating 75 hp of power in the power stroke of the IC engine (representable in *L* as: $\forall t((t \geq 30 \text{ ms} \ \& \ t \leq 40 \text{ ms}) \Rightarrow (\text{Power(IC Engine)} = 75 \text{ hp}))$ can be obtained from the representation of the corresponding FR above by specifying values for the existentially quantified variables v, t_1 and t_2, or can be obtained from the representation of the corresponding PR above by specifying values for the existentially quantified variables t_1 and t_2. The structure of a design is represented as a set of states in *L*. The behavior of a design is represented as a process (temporal or causal) in *L*, i.e. as a set of sentences involving SDs and, if behavior is represented as a causal process, one or more causal implication (^{C}=>) signs. *L* allows the representation of designs at various levels of abstraction (including at the conceptual design stage as well as stages involving progressively more detail)

Appendix 1.2 provides definitions pertaining to designs, their representation in *L*, and the explication of their attributes in terms of their representation in *L.* The representation of the requirements, including FRs, structure

and behavior of a design in the language of *L* is illustrated in Tables 1 to 3 for the design of a schematic IC engine shown in Figure 1 of Section 5.

3.1 Functional decomposition in the framework of *L*

An FR (FR_X) represented (as an existentially quantified sentence) in the language of *L* can be decomposed into (i.e., elaborated at the next level of detail as) a temporal or causal process involving child FRs (e.g. FR_{X1}, FR_{X2}, FR_{X2}, ...). However, for an FR (FR_X) decomposed as above, exactly one of the child FRs (FR_{X1} or FR_{X2} or FR_{X2} or ...) should logically imply the parent FR (FR_X). For the behavior of the design to satisfy the FRs (and PRs) of the design, each lowest level FR (or PR) of the design should be logically entailed by one or more sentences describing the behavior of the design. For example, in the functional decomposition of FR_1 (Air-fuel mix is sucked into the cylinder) in Table 1, exactly one child FR (FR_{13}) of FR_1 logically implies FR_1.

4. Formalization of Theories of Design in *L*

The following types of theories of design are formalized using *L*: a) Theories of a *whether and how* the structure and behavior of a design satisfy the requirements and constraints of the design; and b) Theory of what constitutes a good design (in the sense of a system of requirements, constraints, structure and behavior). The formalization of the above kinds of theories is described below.

4.1 Formalization of theories of whether and how the structure and behavior of a design satisfy the requirements and constraints of the design

A formal, axiomatic theory, describing how the structure and behavior of a design satisfy its requirements and constraints, can be represented in *L*. Such a theory (henceforth called the theory of satisfaction of the requirements and constraints of a design) is illustrated (see Section 5) in Table 5 for the schematic IC engine

shown in Figure 1, whose requirements, constraints, structure and behavior are shown in Tables 1-3. In a theory of satisfaction of the requirements and constraints of a design, the specified structure and behavior of the design constitute the axioms (or premises), and the sentences derived from these axioms constitute the theorems (or conclusions based on the premises). The structure and behavior of a design can be said to satisfy a given requirement of the design if and only if the sentence representing the requirement in *L* occurs as a theorem (i.e., the satisfaction of the requirement is entailed by the specified structure and behavior constituting the premises) in the above formal, axiomatic theory of satisfaction of requirements and constraints of the design. Similarly, a given constraint is said to be satisfied (i.e., not violated) by the structure and behavior of a design if and only if the negation of the constraint is not entailed by the specified structure and behavior of the design. Consequently, if the sentence in *L* representing a constraint is entailed by the specified structure and behavior of the design, then also the constraint is said to be satisfied by the specified structure and behavior of the design.

If the structure and behavior do satisfy the requirements and constraints of the design, then *how* the structure and behavior of a design satisfy the design's requirements and constraints is expressed as a formal proof in *L*. It can be seen from Section 3 above that FRs, PRs and behavior can be represented in *L* in a hierarchy of existential quantification. A PR that is obtained by specifying the values of one or more existentially quantified variables in an FR logically implies the FR. Similarly a description of behavior that is obtained by specifying the values of one or more existentially quantified variables in a PR or an FR logically implies the PR and FR, respectively. For example, the PR that an IC engine generate 75 hp of power (represented in *L* as $\exists t_1 \exists t_2 \forall t((t \geq t_1 \ \& \ t \leq t_2 \ \& \ t_1 \leq t_2) \Rightarrow (\text{Power(IC Engine)} = 75 \text{ hp}))$ logically implies the satisfaction of the FR to generate power represented in *L* as $\exists t_1 \exists t_2 \exists v \forall t((t \geq t_1 \ \& \ t \leq t_2 \ \& \ t_1 \leq t_2) \Rightarrow (\text{Power(IC Engine)} = v \ \& \ v > 0))$. Similarly, the behavior of generating 75 hp of power in the power stroke of the IC engine (represented in *L* as $\forall t((t \geq 20 \text{ ms} \ \& \ t \leq 30 \text{ ms}) \Rightarrow (\text{Power(IC Engine)} = 75 \text{ hp}))$ logically implies the satisfaction of the PR that the IC engine generate 75 hp of power (represented in *L* as $\exists t_1 \exists t_2 \forall t((t \geq t_1 \ \& \ t \leq t_2 \ \& \ t_1 \leq t_2) \Rightarrow (\text{Power(IC Engine)} = 75 \text{ hp}))$ and also logically implies the satisfaction of the FR to generate power represented in *L* as $\exists t_1 \exists t_2 \exists v \forall t((t \geq t_1 \ \& \ t \leq t_2 \ \& \ t_1 \leq t_2)$

=>(Power(IC Engine) = v & $v > 0$)). This possibility of logical implication of FRs and PRs by behavior, and the logic of *L* including its causal calculus, together enable the proof of satisfaction of FRs and constraints by the structure and behavior of the design. Table 4 illustrates the proof of satisfaction of FRs and constraints by the structure and behavior of an illustrative IC engine.

4.2 Formalization of theories of what constitutes a good design

Garvin (1987) proposed eight dimensions of quality of a product, viz.: performance, features, reliability, durability, conformance, serviceability (maintainability), aesthetics and perceived quality. Bralla (1996) proposed upgradability as an additional dimension of quality. The above nine dimensions of quality can be distilled into two generally desirable attributes of a design: a) satisfaction of requirements (including functionality, performance, features, conformance, aesthetics, etc.); and b) change manageability (including maintainabilty and upgradability). The information axiom of ADT (Suh(1990)) and the consequent desirability of wide specifications, low variability and low bias pertain to the attribute of requirements satisfaction. The independence axiom of ADT (Suh(1990)) and its implications pertaining to the desirability of uncoupled or decoupled designs over coupled designs, and the concept of modularity of a design (Ullrich and Eppinger (1995)) pertain to the attribute of change manageability of a design. Other generally desirable attributes of a design (Pahl and Beitz(1988), Ullrich and Eppinger(1995), Suh(1990)) include FRs stated in a solution-neutral manner so as to allow a wide number and variety of solutions to satisfy them, standardization of and symmetry in the components of a product, and a parametric design (Woodbury (2010), Oxman and Gu (2015)).

The formal representation of a design in terms of its requirements (including FRs), constraints, structure and behavior in the language of *L* enables the definition of the above desirable properties of a design in terms of the representation of the design in the language of *L.* Appendix 1 provides definitions of various general attributes of a design, in terms of the representation of the design in the framework of *L,* including the following: solution-neutrality of FRs of a design; coupling in designs; coupled, uncoupled and decoupled designs; probability of

success (and consequently, information content) of a design; standardization of and symmetry in the components of a design; and parametric designs.

The formalization of theories of what constitutes a good design (in terms of requirements, constraints, structure and behavior) can be achieved by representing the above generally desirable attributes of a design as sentences in a formal theory of design. Further, axiomatization of such a theory requires that one or more sentences in the theory be axioms from which the remaining of the above generally desirable attributes of a design be derived as theorems.

The approach taken in this work for the creation of such a formal, axiomatic theory of design, is now described. First, the concept of entropy (Shannon (1948), Lebowitz (1993)) underlying the information axiom of ADT (Suh (1991)) is generalized to include the following pertaining to a design: a) solution-neutrality of FRs (at various levels in the hierarchy of FRs); b) dimensionality (the number of mutually independent, constraned attributes) of the design; c) the probability of satisfaction of requirements and constraints by the structure and behavior of the design (calculated using a Bayesian framework (Pearl (2000)); and d) change manageability of the design. The entropy associated with the probability of satisfaction of requirements and constraints of the design is further analyzed into the components of width of specified ranges for the attributes of the design, variability of attributes of the design around their respective central tendencies, and bias in the attributes of the design (i.e., shift of the central tendencies of the attributes of the design from their respective specified target values). The entropy of change manageability of a design takes into account the nature (e.g. circular or non-circular), extent (number of) and relative importance of dependencies among sibling sub-designs at various levels in the hierarchy of the design. The entropy of a design as defined in this work, in addition to taking into account the factors mentioned above, also enables the explicit characterization of the impact of standardization, symmetry and the extent of parametrization of a design, on the overall entropy of the design. In this work we posit that lesser the entropy of a design the better is the design. The entropy of a design in terms of its components mentioned above is defined in Appendix 1.

Having defined the entropy of a design as above, a utility function "Q(d)" of a design (d) representing the quality of a design from a design-theoretic perspective is posited. The quality of a design from design-theoretic perspective (Q(d)) is the degree to which the design (d) exhibits the general attributes of a good design such as those discussed above (e.g. solution-neutrality, high probability of satisfaction of requirements and constraints, change-manageability, etc.). The definitions of generally desirable attributes of a design in terms of its representation in *L*, the definition of the entropy of a design, and the concept of the quality of a design from a design-theoretic perspective together enable the creation of a formal, axiomatic theory of what constitutes a good design (in the sense of a system of requirements, constraints, structure and behavior).

The first (and only) axiom of the theory states that lower the entropy of a design, the higher its quality (i.e., better the design) from a design-theoretic perspective. From this axiom and the definition of the entropy of a design are derived various theorems that formally state the desirability of the following: solution-neutrality of FRs, a small number of independent attributes of a design; wide specifications; low variability and bias in the attributes of the design; standardization of parts and components; symmetry of parts and components; uncoupled designs over coupled designs; of circular over non-circular dependencies among sibling sub-designs; avoiding dependencies among sibling sub-designs at higher levels in the hierarchy of sub-designs than at lower levels; and avoiding non-circular dependencies at higher levels in the hierarchy of sub-designs than at lower levels. As an example of how such a theory could help identify attributes of a good design over and above the set of attributes discussed above, the desirability of parametric designs (Woodbury (2010), Oxman and Gu (2015)) is also derived as part of the theory discussed above. For example, the practice of following rules of thumb to specify a relatively large number of attributes (e.g. various parameters of the inlet and exhaust valves of an IC engine, such as valve stem diameter, thickness of the valve head, etc.) in terms of the specifications of a small set of attributes of the design (e.g. the specified diameter of the valve head (Heywood (1988)), could be seen as a type of parametric design.

The formal, axiomatic theory of design is given below.

Definitions of terms

d ≡ A given design.

d' ≡ A design obtained by modifying a given design d (specific types, if any, of modifications by which d' is obtained from d are specified within the context of the corresponding theorems in the formal theory below).

Q(d) ≡ Quality of the design d from a design-theoretic perspective (see Appendix 1 for the definition of quality of a design from a design-theoretic perspective).

E(d) ≡ Entropy of design d (see Appendix 1 for the definition of entropy of a design).

O(d)≡ Degree of solution-neutrality of design d (see Appendix 1 for the definition of solution-neutrality of a design).

K(d) ≡ Dimensionality of design d (see Appendix 1 for the definition of dimensionality of a design).

M(d) ≡ Number of combinations of values of attributes of design d that lie within the specifications of design d (see Appendix 1 for how this quantity is calculated).

V(d) ≡ Number of combination of values of attributes of design d that lie within the natural bounds of variation of the attributes (see Appendix 1 for how this quantity is calculated).

B(d) ≡ Number of combination of values of attributes of design d that lie between the specified target values and central tendencies (e.g. expected values) of these attributes (see Appendix 1 for how this quantity is calculated).

S(d) ≡ Degree of standardization of design d (see Appendix 1 for the definition of degree of standardization of a design).

Sym(d) ≡ Degree of symmetry of design d (see Appendix 1 for the definition of degree of symmetry of a design).

T(d) ≡ Degree of parametrization of design d (see Appendix 1 for the definition of degree of parametrization of a design).

U(d) ≡ Number of non-circular dependencies among sibling sub-designs of d (see Appendix 1 for the definition of non-circular dependency among sibling sub-designs of a design).

C(d) ≡ Number of circular dependencies among sibling sub-designs of d (see Appendix 1 for the definition of circular dependency among sibling sub-designs of a design).

E(d, i) ≡ Entropy of design d at level i in the hierarchy of FRs of d.

Q(d, i) ≡ Quality of design d, from a design-theoretic perspective, at level i in the hierarchy of FRs of d.

C(d, i) ≡ Number of circular dependencies among sibling sub-designs of design d, at level i in the hierarchy of FRs of d.

U(d, i) ≡ Number of non-circular dependencies among sibling sub-designs of design d, at level i in the hierarchy of FRs of d.

$\Delta^X(d', d) \equiv X(d') - X(d)$, where X(d) is one of the following: E(d); Q(d); K(d); O(d); V(d); B(d); S(d); or Sym(d).

$\Delta^Y((d', j), (d, i)) \equiv Y(d', j) - Y(d, i)$, where Y(d, i) is one of the following: E(d, i); Q(d, i); C(d, i); or U(d, I) (and similarly for Y(d', j)).

Axiom 1: The quality of a design from a design-theoretic perspective varies inversely with the entropy of the design. That is, lower (higher) the entropy of a design the better (worse) is its quality from a design-theoretic perspective. (Here onwards, the phrase "better the design" is taken to mean "better the design from a design theoretic perspective").

That is:

$(\Delta^E(d', d) < 0 \Rightarrow \Delta^Q(d', d) > 0)$ & $(\Delta^E(d', d) > 0 \Rightarrow \Delta^Q(d', d) < 0$ & $((\Delta^E(d', d) = 0) \Rightarrow (\Delta^Q(d', d) = 0))$.

The following theorems follow from the definition of entropy (see Appendix 1) and Axiom 1 above (the statements of the theorems in words are by default qualified by trade-offs that might exist among various components of entropy of a design):

Theorem 1: The more solution-neutral the FRs of a design, the better the design.

That is: $\Delta^{O}(d', d) > 0 \Rightarrow \Delta^{E}(d', d) < 0 \Rightarrow \Delta^{Q}(d', d) > 0$

Theorem 2: The solution-neutrality of an FR higher in the hierarchy of FRs of a design is more important than the solution-neutrality of an FR lower in the hierarchy of FRs of the design.

Consider a design d wherein two or more FRs are not solution neutral. Consider design d' obtained from design d only by making a non-solution-neutral FR in design d solution-neutral. Similarly, consider design d'' obtained from design d only by making another non-solution-neutral FR in design d solution-neutral. Let x be the level of the non-solution-neutral FR, in the hierarchy of FRs of d, that was made solution-neutral in design d'. Similarly, let y be the level of the non-solution-neutral FR, in the hierarchy of FRs of d, that was made solution-neutral in design d''. Then:

$(x < y) \Rightarrow \Delta^{O}(d', d'') > 0 \Rightarrow \Delta^{E}(d', d'') < 0 \Rightarrow \Delta^{Q}(d', d'') > 0$

(by definitions of solution-neutrality of a design, entropy of a design and Axiom 1).

Theorem 3: Lower the dimensionality of a design (k), the better the design.

Consider a design d' obtained from design d only by reducing the dimensionality (number of constrained independent attributes) of d. Then:

$\Delta^{K}(d', d) < 0 \Rightarrow \Delta^{E}(d', d) < 0 \Rightarrow \Delta^{Q}(d', d) > 0$

(by the following: definition of design d' with respect to design d given above; definition of dimensionality of a design; definition of entropy of a design; and Axiom 1).

Theorem 4: Wider the specified ranges of attributes of a design, the better the design.

Consider a design d' obtained from design d only by changing the specified range of values, i.e. widening or narrowing the specifications for one or more attributes of design d. Then:

$\Delta^{M}(d', d) > 0 \Rightarrow \Delta^{E}(d', d) < 0 \Rightarrow \Delta^{Q}(d', d) > 0$

(by definition of entropy of a design and Axiom 1).

Theorem 5: Lower the variability in the attributes of the structure and behavior of a design about their mean, the better the design.

Consider a design d' that is obtained from design d only by reducing the variation of one or more attributes of the structure or behavior of d about their respective means. Then:

$\Delta^V(d', d) < 0 \Rightarrow \Delta^E(d', d) < 0 \Rightarrow \Delta^Q(d', d) > 0$

(by definition of entropy of a design and Axiom 1).

Theorem 6: Lower the bias in the values of attributes of the structure and behavior of a design from the specified target value of the attribute, the better the design.

Consider a design d' that is obtained from design d only by changing the bias in the values of the attributes of the structure or behavior of d from the specified target values for the attributes. Then:

$\Delta^B(d', d) < 0 \Rightarrow \Delta^E(d', d) < 0 \Rightarrow \Delta^Q(d', d) > 0$

(by definition of entropy of bias of a design and Axiom 1).

Theorem 7: More the standardization of parts or components of a design, the better the design.

Consider a design d' obtained from design d only by changing the extent of standardization of parts or components of d. Then:

$\Delta^S(d', d) > 0 \Rightarrow \Delta^K(d', d) < 0 \Rightarrow \Delta^E(d', d) < 0 \Rightarrow \Delta^Q(d', d) > 0$

(since standardization of parts or components reduces the number of constrained independent attributes of the design; and by definition of dimensionality of a design, definition of entropy of a design and Axiom 1).

Theorem 8: More the symmetry in the parts or components of a design, the better the design.

Consider a design d' obtained from design d only changing the extent of symmetry in the form of the parts or components of d. Then:

$\Delta^{Sym}(d', d) > 0 \Rightarrow \Delta^{K}(d', d) < 0 \Rightarrow \Delta^{E}(d', d) < 0 \Rightarrow \Delta^{Q}(d', d) > 0$

(since the value of one attribute in a pair of attributes that are mirror images of each other can be obtained from the value of the other attribute in the pair, thereby reducing the number of constrained independent attributes of the design; by definitions of dimensionality and entropy of a design; and Axiom 1).

Theorem 9: Fewer the dependencies among sibling sub-designs of a design, the better the design.

Consider a design d' obtained from design d only by changing the number of dependencies among sibling sub-designs of d. Then:

$(\Delta^{U}(d', d) < 0 \ \& \ \Delta^{C}(d', d) \leq 0) \text{ v } (\Delta^{U}(d', d) \leq 0 \ \& \ \Delta^{C}(d', d) < 0) \Rightarrow \Delta^{E}(d', d) < 0 \Rightarrow \Delta^{Q}(d', d) > 0$

(by definition of entropy of a design and Axiom 1).

Theorem 10: An uncoupled design is better than a coupled design.

Consider a design d that has one or more circular or non-circular dependencies among its sibling sub-designs. Consider a design d' obtained from design d only by changing the number dependencies (circular or non-circular) among sibling sub-designs of design d. Then:

$(\Delta^{U}(d', d) + \Delta^{C}(d', d) = -(U(d) + C(d)) \Rightarrow \Delta^{Q}(d', d) > 0$

Proof:

$U(d) > 0 \text{ v } C(d) > 0$ (by definition of d)……..…..(1)

$(U(d) + C(d)) > 0$ (by (1))……..…….(2)

$(\Delta^{U}(d', d) = -U(d)) \Leftrightarrow (U(d') = 0)$ (by definition of d' and d above)……..(3)

$(\Delta^C(d', d) = -C(d)) <=> (C(d') = 0)$ (by definition of d' and d above)……..(4)

Therefore:

$((\Delta^U(d', d) = -U(d) \ \& \ \Delta^C(d', d) = -C(d)) => (U(d') = 0) \ \& \ (C(d') = 0)$ (by (3) and (4))……...(5)

$(U(d') = 0) \ \& \ (C(d') = 0)$ <=> Uncoupled(d') (by definition of uncoupled design)……....(6)

$U(d) > 0 \ \& \ C(d) > 0$ (by definition of d)…….…..(7)

$(\Delta^U(d', d) + \Delta^C(d', d) = -(U(d) + C(d)) => (\Delta^U(d', d) + \Delta^C(d', d)) < 0$ (by 7)…….....(8)

$(\Delta^U(d', d) + \Delta^C(d', d)) < 0 => \Delta^E(d', d) < 0$ (by definition of entropy of a design)……..(9)

$(\Delta^U(d', d) + \Delta^C(d', d) = -(U(d) + C(d)) => \Delta^Q(d', d) > 0$ (by (8), (9) and Axiom 1).

Hence proved.

Theorem 11: Within a given level of FRs, eliminating a single circular dependency is more important than eliminating a non-circular dependency in a design.

Consider a design d that has one or more circular dependencies and one or more non-circular dependencies among its sibling sub-designs at level z in its hierarchy of FRs. Consider a design d' obtained from design d only by eliminating a circular dependency and a design d'' obtained from design d only by eliminating a non-circular dependency, each at level z in the hierarchy of FRs of d. Then:

$(\Delta^E(d'', d) < \Delta^E(d', d) < 0) \ \& \ (\Delta^Q(d'', d) > \Delta^Q(d', d) > 0)$

(by definition of entropy of a design and Axiom 1).

Theorem 12: Eliminating a circular dependency at a given level in the hierarchy of FRs of a design is more important than eliminating a circular dependency at a lower level in the hierarchy of FRs of a design.

Consider a design d that has one or more circular dependencies among its sibling sub-designs at level z and level $z + m$ ($m > 0$) in its hierarchy of FRs. Consider a design d' obtained from design d only by eliminating a circular

dependency at level z of FRs of d and a design d'' obtained from design d only by eliminating a circular dependency at level z+m of the FRs of d. Then:

$(\Delta^E(d', d) < \Delta^E(d'', d) < 0)$ & $(\Delta^Q(d', d) > \Delta^Q(d'', d) > 0)$

(by definition of entropy of a design and Axiom 1).

Theorem 13: Eliminating a non-circular dependency at a given level in the hierarchy of FRs of a design is more important than eliminating a non-circular dependency at a lower level in the hierarchy of FRs of a design.

Consider a design d that has one or more non-circular dependencies among its sibling sub-designs at level z and level z + m (m > 0) in its hierarchy of FRs. Consider a design d' obtained from design d only by eliminating a non-circular dependency at level z of FRs of d and a design d'' obtained from design d only by eliminating a non-circular dependency at level z+m of the FRs of d. Then:

$(\Delta^E(d', d) < \Delta^E(d'', d) < 0)$ & $(\Delta^Q(d', d) > \Delta^Q(d'', d) > 0)$

(by definition of entropy of a design and Axiom 1).

Theorem 14: Eliminating a circular dependency at a given level in the hierarchy of FRs of a design is more important than eliminating a non-circular dependency at a lower level in the hierarchy of FRs of a design.

Consider a design d that has one or more non-circular dependencies among its sibling sub-designs at level z and level z + m (m > 0) in its hierarchy of FRs. Consider a design d' obtained from design d only by eliminating a circular dependency at level z of FRs of d and a design d'' obtained from design d only by eliminating a non-circular dependency at level z+m of the FRs of d. Then:

$(\Delta^E(d', d) < \Delta^E(d'', d) < 0)$ & $(\Delta^Q(d', d) > \Delta^Q(d'', d) > 0)$

(by theorems 12 and 13).

Theorem 15: A decoupled design is better than a coupled design that is not decoupled (but within limits determined by the number of non-circular dependencies that are equivalent, in terms of their contribution to the entropy of a design, to a circular dependency, at various levels of FRs of a design).

Consider a design d that is coupled but not decoupled. Let design d' be obtained from design d only by eliminating all the circular dependencies in d and adding zero or more non-circular dependencies. Let d'_c be the design obtained from design d only by eliminating all the circular dependencies in d. d'_u be the design obtained from design d only by adding zero or more non-circular dependencies Then:

$(\Delta^E(d'_c, d) - \Delta^E(d'_u, d) < 0) \Rightarrow (\Delta^E(d', d) < 0) \Rightarrow (\Delta^Q(d', d) > 0)$

(By definition of entropy of a design and Axiom 1).

Theorem 16: More parametric the design (i.e., higher the proportion of dependent to independent attributes of a design), the better it is.

Consider a design d' obtained from design d only by changing the extent of parametrization of d. Then:

$\Delta^T(d', d) > 0 \Rightarrow \Delta^Q(d', d) > 0$

Proof:

$\Delta^T(d', d) > 0 \Rightarrow \Delta^K(d', d) < 0$ (since parametrization of parts or components reduces the number of constrained independent attributes of the design)……..(10)

Therefore:

$\Delta^T(d', d) > 0 \Rightarrow \Delta^E(d', d) < 0$ (by (10) and Theorem 3)…..(11)

Therefore:

$\Delta^T(d', d) > 0 \Rightarrow \Delta^Q(d', d) > 0$ (by (11) and Axiom 1).

Thus it can be seen from the above formalization that notions of what constitutes a good design, such as a high probability of satisfaction of requirements and constraints, change manageability (enabled by modularity), low dimensionality, high standardization, high symmetry, high parametrization, etc., are often not mutually independent, but have to be traded off against one another in accordance with the needs of stakeholders of the design, including the users of the design and the organization designing a product or solution. For example, a design that is highly optimized for performance might entail a reduced degree of standardization than otherwise might have been possible (e.g. an IC engine re-optimized for weight might use plastic components where previously aluminum components might have been used, thereby resulting in reduced use of standard parts). It can also be seen that formalization of the concept of coupling in design necessitates the recognition (especially in the light of the trade-off among attributes of a good design discussed above) of the following nuances of coupling in design (e.g. over and above whether a design is uncoupled, coupled or decoupled): a) the extent of coupling (e.g. number of non-circular and circular dependencies); b) the relative importance of circular compared to non-circular coupling; and c) the relative importance of coupling in higher levels of FRs compared to that in lower levels of FRs. Also, it can be seen that the above formalization enables the explicit identification of the desirability of parametric designs and provides a formal explanation of why parametrization in designs is desirable (i.e. parametrization tends to reduce the entropy of a design by reducing its dimensionality).

As can be seen from the description of the entropy of a design and the consequent theory of of what constitutes a good design given above, the entropy of a design could potentially be reduced, and therefore the design potentially improved, by one or more of the following ways: a) making FRs solution-neutral; b) reducing the dimensionality of the design (k); c) increasing the probability of success of the design; and d) reducing coupling in the design. The dimensionality of the design could be reduced by multiple ways including: eliminating one or more FRs; eliminating components or parts; increasing standardization of parts or components; increasing symmetry of parts or components; making the design more parametric (e.g. adopting empirically validated rules of thumb whereby certain attributes of a part or component are defined as functions of other attributes of the part or component). The probability of success of a design could be improved by widening the specifications and/or

reducing the variability and bias of attributes of the design. Coupling in the design could primarily be reduced by eliminating circular dependencies (i.e., trying to make the design decoupled), and secondarily by reducing or eliminating the number of non-circular dependencies (i.e., trying to make the design uncoupled, once it has been made decoupled) among sibling sub-designs of the design. However, reduction in one component of the entropy of the design could potentially be accompanied by an increase in another component of the entropy of the design. For example, reducing the dimensionality of a design by integrating parts could potentially be accompanied by increased coupling, and standardization of parts could potentially impact the probability of success of the design. Thus, it can be seen that the structure of entropy of a design could be used to guide the search for improvements in the design, while the values of the parameters determining the impact of dimensionality, solution-neutrality, circular or non-circular couplings, etc. on the entropy of the design (see Appendix 1 for definition of entropy of design in terms of the above) could be used to guide trade-offs among various considerations such as solution-neutrality, dimensionality, probability of success, change manageability, etc. of a design.

5. Illustrative example of a schematic IC engine

A schematic of a spark ignition IC engine shown in Figure 1 (a) and 1(b) is used to illustrate the representation and analysis of designs using the symbolic logical framework *L* discussed above. For the illustrative IC engine, the representation of the following in the language of *L* is shown in Tables 1, 2 and 3, respectively: a) requirements and constraints; b) structure; and c) behavior. Table 4 shows the proof (and expressions to calculate the probability) of satisfaction of the requirements and constraints of the illustrative IC engine by its structure and behavior. Supplementary Figures 1 (a) and 1 (b) illustrate the calculation of the truth values of requirements and constraints of the illustrative IC engine given its structure and the truth values of the input conditions of its behavior. Supplementary Figure 2 illustrates the calculation of the probability of satisfaction of the requirements and constraints of the illustrative IC engine, given the probabilities of its pre-conditions, using a Monte Carlo simulation. The Monte Carlo simulation generates multiple combinations of truth values for the pre-conditions,

assuming that each pre-condition follows a binomial distribution with probability of occurrence as specified in Supplementary Figure 2. For each combination of truth values of pre-conditions, the truth values of the intermediate and output SDs are calculated based on the causal relationships specified in Table 3. The probabilities of occurrence of the intermediate and output SDs are then estimated from the truth table simulated as above.

----------Figure 1 (a) goes here----------

----------Figure 1 (b) goes here----------

----------Table 1 goes here----------

----------Table 2 goes here----------

----------Table 3 goes here----------

----------Table 4 goes here----------

Figure 2 shows the coupling (dependency matrix) for the illustrative IC engine (see Appendix 1.2 for a definition of the coupling (dependency) matrix of a design). Figure 3 (a) shows the entropy and components of the entropy of the illustrative IC engine. The relative weight of each component of entropy in Figure 3 (a) is obtained from Figure 3 (b), which illustrates the assignment of relative importance (weights) to various components of entropy of the illustrative IC engine. Since all the FRs in the illustrative IC engine are represented in a solution-neutral manner (see Appendix 2.1 for a definition of solution-neutrality of an FR), its entropy of solution-neutrality is zero. Supplementary Figure 3 (a) shows the calculation of the entropies of dimensionality and conformance of the illustrative IC engine. Supplementary Figure 3 (b) shows the calculation of the entropy of the change manageability of the illustrative IC engine.

----------Figure 2 goes here----------

----------Figure 3 (a) goes here----------

----------Figure 3 (b) goes here----------

From the above, it can be seen that the design of the illustrative IC engine, as represented in *L*, has the following general attributes: a) the FRs of the illustrative IC engine have been stated in a solution-neutral manner (see Tables 1 and 2); b) there exists a proof of satisfaction, in the logic of *L*, of the requirements and constraints of the design by its structure and behavior (see Table 4); c) the design has a non-zero probability of satisfaction of requirements and constraints by its structure and behavior (see Supplementary Figure 2); and d) the design is decoupled (see Figure 2) (see Appendix 1.2 for a definition of a decoupled design).

The structure of entropy (design-theoretic (as defined in Appendix 1.2), and not the thermodynamic entropy of heat engines) of the illustrative IC engine enables the search for improvements in its design. For example, the entropy of the design of the illustrative IC engine could potentially be reduced by: a) reducing the dimensionality of the design; b) increasing the probability of satisfaction of the requirements and constraints of the design; and c) improving the change-manageability of the design. The dimensionality of the design could potentially be reduced, for example, by making the inlet valve unidirectional (thus eliminating the complex valve train used to control the timing of opening and closing the valve), standardizing the inlet and exhaust valves, eliminating the components for spark generation (e.g. by converting the IC engine into a compression ignition engine), etc.. The probability of satisfaction of requirements and constraints of the design could potentially be improved by appropriately modifying the specifications of and controlling the variation of statistically significant factors that determine the performance of the design. Since the illustrative IC engine is a decoupled design, improving its change manageability might not take priority over the other sources of entropy of the design.

The reduction in one component of entropy of the design obtained by each of the above potential improvements to the design of the illustrative IC engine might also have a tendency to increase the entropy of another component of entropy of the design, unless further adjustments are made to the design. For example, making the inlet valve unidirectional, while reducing the entropy of dimensionality, could increase the entropy of satisfaction of requirements and constraints by reducing the power output of the engine. Thus, since it is likely that any given design improvement could reduce one component of entropy while increasing another, the

selection of potential improvements in a design has to be guided by the overall impact of the improvements on the entropy of the design.

Figures 4 (a) and Figure 4 (b) illustrate the trade-offs among various components of entropy of a design associated with potential improvements in the design. Figure 4 (a) illustrates the overall impact of standardizing the inlet and exhaust values on the entropy of the design of the schematic IC engine, assuming that the above improvement reduces the power of the engine marginally. Figure 4 (b) illustrates the overall impact of changing the design of the inlet valve from a cam-operated poppet valve to a unidirectional valve, assuming that this change, while eliminating the complex mechanism (valve train) required to control operation of the valve, causes a substantial reduction in the power of the engine. Such analyses could help prioritize ideas for improving a design by analyzing the impact of the potential improvement on the various components of entropy, as well as the overall entropy, of the design.

Designers could assign different relative weights to the components of entropy depending on the purpose and context of the design, as shown in Figure 3(b). For example, the relative weight of entropy of satisfaction of requirements and constraints could be an order of magnitude higher than the other components of entropy if the product is used in a context where the cost of non-conformance is really high (e.g. surgical instruments), and the relative weight for change manageability could be relatively high in contexts where new versions of the product are released frequently (e.g. software products or automobiles). Figure 4(c) illustrates the sensitivity of the illustrative IC engine to the relative importance of various components of entropy shown in Figure 3 (b) (in this case, the effect of +/- 10% change in the relative importance of each component of entropy on that component of entropy of the illustrative IC engine). For example, it can be seen that the design of the illustrative IC engine, as is, is not impacted by the relative importance (weight) assigned to solution neutrality or circular coupling in the design. This is because the design, as is, is solution-neutral and does not contain any circular coupling. Also, as expected from the weights given in Figure 3 (b), the design of the illustrative IC engine is most sensitive to conformance to requirements and constraints, relatively less sensitive to dimensionality and marginally sensitive

to non-circular coupling. Thus the weights assigned to each component of entropy could potentially play a role in guiding the search for improvements of the design by emphasizing (or de-emphasizing) the components of entropy that are important in the context of the design.

6. Summary and Discussion

This work provides a symbolic logical framework (*L*), consisting of first order logic augmented with a causal calculus, to formalize, axiomatize and integrate the following types of theories of design: a) What is a design; b) How does a design satisfy its requirements and constraints (represented as a formal theory of satisfaction of requirements (including FRs) and constraints by the structure and behavior of the design); and c) What is a good design (represented as a formal, axiomatic theory of general attributes of a good design). A design is formally represented in *L*, in keeping with the FBS ontology (Gero (1990), Vescovi, et al. (1994), Umeda, et al. (1990)), as a system of requirements (including FRs), temporal and causal relationships among FRs, constraints, structure and behavior. The wide applicability of *L* enables the following: use of a single language to represent functions, behavior and structure, thereby enabling their seamless integration; and generality in the formalization of theories of design.

A central question pertaining to any design (in the sense of an artifact) is: "*How* will the design satisfy its requirements and constraints?". *L* enables the formal proof of satisfaction of FRs by the structure and behavior of the product through the use of logical, material and causal implication. FRs are represented as existentially quantified forms of behavior using *L*, which enables the logical implication of FRs (e.g. power is generated) by behavior (e.g. 75hp of power is generated). The formal representation of a design in terms of its requirements (including FRs), constraints, structure and behavior in the language of *L* enables the definition of generally desirable properties of designs (e.g. solution-neutrality of requirements, high probability of satisfaction of requirements and constraints, change manageability enabled by a modular architecture, standardization, symmetry, etc. (Pahl and Beitz (1988), Ullrich and Eppinger (1995), Suh (1990)) in terms of the representation

of the design in the language of *L*. Solution-neutrality of an FR is defined as the absence (non-usage), in the representation of the FR in the language of *L*, of individuals or predicates specific to a solution (structure and behavior) for the FR. Coupling in the design of an artifact represented in *L* is a dependency of one sub-design (structure and behavior intended to satisfy a specific FR) on a sibling sub-design in the hierarchy of the design and its sub-designs. An uncoupled design is one wherein there are no couplings. A coupled design is one where there are one or more couplings. A decoupled design is a coupled design where the couplings are not circular. The probability of success of a design is defined as the probability of the structure and behavior of the design satisfying all of its requirements and constraints. The probability of success of a design is computed in a Bayesian framework (Pearl (2000)). The entropy of a design is then defined in terms of general attributes of a design represented in the language of *L.* The definition of entropy of a design used in this work is a generalization of the concept of entropy (Shannon (1948)) underlying the information axiom of ADT (Suh (1990)), to encompass the following: solution-neutrality of FRs; dimensionality (i.e., the number of independent constrained attributes) of the design; satisfaction of requirements and constraints; and change-manageability. The entropy of satisfaction of requirements and constraints is further analyzed into the entropies of the following pertaining to the attributes of the design: width of specifications; variability about the central tendency of the attribute; and bias (shift of the central tendency of the attribute from the specified target value for the attribute). The entropy of change manageability of a design takes into account the number and types (circular and non-circular) of dependencies (couplings) among sibling sub-designs in the hierarchy of sub-designs of the design. The quality of a design from a design-theoretic perspective (i.e., the degree to which a design exhibits desirable attributes such as those discussed above) is then defined as a utility function of the design that increases (or decreases) with decreasing (or increasing) entropy of the design.

A formal, axiomatic theory of what constitutes a good design is then constructed based on the following: a) definitions of general attributes of a good design as represented in the language of *L*; b) definition of the entropy of a design in terms of or related to the definitions of general attributes of a good design as represented in the language of *L*; and c) the utility function Q representing the quality of a design from a design-theoretic

perspective. In the formal, axiomatic theory of design thus constructed, it is taken as an axiom that lower the entropy of a design, the higher its quality (i.e., the better it is) from a design-theoretic perspective. Based on the above axiom, and the definition of entropy of a design, the desirability of general attributes of a design such as solution-neutrality of FRs, wide specifications, low variability and bias, less coupling (i.e. a more modular architecture), higher standardization and symmetry, etc. follow as theorems.

The use of *L* is illustrated in the context of a schematic IC engine. The requirements (including FRs), constraints, structure and behavior of the IC engine are represented in the language of *L*. A formal proof of satisfaction of the requirements and constraints of the illustrative IC engine by its structure and behavior is provided using the logic of *L*. The probability of satisfaction of the requirements and constraints of the illustrative IC engine by its structure and behavior is demonstrated in a Bayesian framework using a Monte-Carlo simulation. It is shown that the FRs of the illustrative IC engine are represented in *L* in a solution-neutral manner. It is also shown that the design of the illustrative IC engine, as represented in the language of *L*, is decoupled. The computation of the entropy of the illustrative IC engine is demonstrated. The use of the entropy of the design to guide the search for and assess trade-offs among potential improvements in the design is illustrated. The specification of the relative importance of various components of entropy of a design in consonance with the purpose and context of the design activity, to guide the search for solutions, is described and illustrated.

Although monotonic reasoning has primarily been used in this work in keeping with the emphasis of this work on formalizing theories of design rather than automating the generation of designs, the causal calculus of *L* naturally mitigates the Qualification and Ramification problems encountered in non-monotonic reasoning. Further, approaches of circumscription (analogous to literal completion of McCain and Turner (1997)) and predicate subscription (McCarthy (1980) have been illustrated (see footnote of Table 3), and can be used to apply *L* in a non-monotonic setting.

L, being a first order logic, is undecidable (Boolos and Jeffrey (1989)), i.e., there is no method by which it can always be determined whether any given sentence in *L* is derivable (or not derivable) from any given set of premises in *L.* An implication of the undecidability of *L* in the context of the representation and reasoning about designs is that it cannot always be determined, using the logic of *L*, whether or not a given structure and behavior of a design, as represented in *L*, will satisfy its requirements and constraints. This is not to say that whether the structure and behavior of a design satisfy its requirements and constraints cannot be determined at all. For most practical cases, especially at the conceptual design stage, it should be possible to logically ascertain whether or not the structure and behavior imply the satisfaction of requirements and constraints. In fact, if the requirements and constraints of a design are indeed implied, in the logic of *L*, by the structure and behavior of a design, it can eventually be shown that this is so, through repeated application of the rules of logic of *L* (by virtue of the fact that first order logic is semi-decidable (Boolos and Jeffrey (1989)). However, the time required to demonstrate the soundness of a sound design could, in the worst case, grow exponentially with the size (e.g. number of symbols used) of description of the design in terms of its structure, behavior, requirements and constraints (by virtue of the fact that first order logic has exponential worst-case complexity (Papadimitriou (1994), Vardi (1982)).

An important aspect of design is the joint consideration of form and function (e.g. Palmer and Shapiro (1993)). This work focuses on the function and not the form or geometry of a design. This work does not, as yet, include in its scope approaches for the automatic synthesis of designs (e.g. using a search for design prototypes as in Gero(1990), Umeda, et al. (1990) or using grammar based approaches such as those described by Stiny and Gips (1971), Schmidt and Cagan (1995), Sridharan and Campbell (2005), etc.). The above limitations could be the subject matter for future work. In particular, the following could be explored in the future: automated search or synthesis of designs or design improvements guided by a formal, axiomatic, integrated theory of what constitutes a good design; and use or extension of the symbolic logical framework *L* to formalize, axiomatize and integrate theories of what constitutes a good design process.

7. Competing Interests: The **a**uthor declares none.

8. References

Bobrow, D. G. (1984). Qualitative Reasoning about Physical Systems: An Introduction. *Artificial Intelligence 24(1-3),* 1-5.

Bochman, A. (2003). A logic for causal reasoning. *In IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003,* 141–146. Acapulco: Morgan Kaufmann.

Bochman, A. (2004). A causal approach to nonmonotonic reasoning. *Artificial Intelligence 160*,105–143.

Bochman, A., & Lifschitz, V. (2015). Pearl's causality in a logical setting. *In Twenty-Ninth AAAI Conference on Artificial Intelligence.*

Boolos, G. & Jeffrey, R. (1989). *Computability and Logic*, 3 rd. Edition. Cambridge University Press.

Braha, D. and Maimon, O. (1998). The Measurement of a Design Structural and Functional Complexity. *IEEE Trans. Syst. Man Cybern., Part A. Syst. Humans, 28 (4),* 527–535.

Bralla J. G. (1996). *Design for excellence*. McGraw Hill, New York.

Chakrabarti, A. and Blessing, L. (2015). *A review of theories and models of design*. *Journal of the Indian Institute of Science 95.4*, 325-340.

Chandrasekaran, B. and Josephson, J. R. (1997). Representing function as effect. *In: Modarres, M. (ed.), Proc. Functional Modeling Workshop, Paris, France.*

Cresswell, M. J. and George, E. H. (2012). *A new introduction to modal logic*. Routledge.

El-Haik, B., and Yang, K., (1999). The Components of Complexity in Engineering Design. *IIE Trans., 31*, 925–934.

Erden, M. S. et al. (2008). A review of function modeling: Approaches and applications. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2008), 22,* 147–169..

Fann, K., (1970), *Peirce's Theory of Abduction*, Martinusnijhoff, The Hague.

Finger, S. and Dixon, J. R. (1989). A review of research in mechanical engineering design. Part I: Descriptive, prescriptive, and computer-based models of design processes. *Research in engineering design 1.1*, 51-67.

Forbus, K. D. (1984). Qualitative process theory. *Artificial intelligence 24.1-3*, 85-168.

Forbus, K. D. (1990). The qualitative process engine. *Readings in qualitative reasoning about physical systems. Morgan Kaufmann,* 220-235.

Frey, D. and Jahangir, E. (1999). Differential entropy as a measure of information content in axiomatic design. *Proceedings of the 1999 ASME Design engineering technical conference, Las Vegas, USA.*

Garvin, D. A. (1987). Competing on the Eight Dimensions of Quality. *Harvard Business Review, 65 (November-December)*, 101–9.

Gero, J. S. (1990). Design prototypes: a knowledge representation schema for design. *AI magazine*, *11*(4), 26.

Gero, J. S., & Kannengiesser, U. (2007). A function–behavior–structure ontology of processes. *AI EDAM: Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, *21*(04), 379-391.

Goel, A., Rugaber, S., & Vattam, S. (2009). Structure, behavior & function of complex systems: The SBF modeling language. *International Journal of AI in Engineering Design, Analysis and Manufacturing*, *23(1)*, 23-35.

Guenov, M. D. (2002). Complexity and Cost Effectiveness Measures for Systems Design. *Manufacturing Complexity Network Conf, Downing College, Cambridge, UK, April 9–10, 2002*, 455–466.

Geffner, H. (1990). Causal Theories for Nonmonotonic Reasoning. *In AAAI*, 524-530.

Heywood, J. (1988). *Internal combustion engine fundamentals*. McGraw-Hill Education.

Hubka, V., & Eder, W.E. (1988). *Theory of Technical Systems: A Total Concept Theory for Engineering Design*. New York: Springer-Verlag (completely revised translation of Hubka, V. (1984) Theorie Technischer Systeme 2 ed, Berlin: Springer-Verlag, 2nd edition of Hubka, V. (1974) Theorie der Maschinensysteme, Berlin:Springer-Verlag)

Khan, W. A. and Angeles, J., (2007). The role of entropy in design theory and methodology. *Proc. CDEN/C2E2 2007 Conference, Winnipeg, Alberta, Canada*.

Kowalski, R., and Marek S. A logic-based calculus of events. *Foundations of knowledge base management. Springer, Berlin, Heidelberg*. 23-55.

Krus, P., (2013). Information Entropy in the Design Process, *ICoRD'13: Global Product Development, Springer*, 101-112.

Krus, P. (2015). Design space configuration for minimizing design information entropy. *ICoRD'15–Research into Design Across Boundaries Volume 1. Springer, New Delhi,* 51-60.

Lebowitz, J. L. (1993). Boltzmann's entropy and time's arrow. *Physics today 46 (1993)*, 32-32.

Lifschitz, V. (1997). On the logic of causal explanation. *Artificial Intelligence 96*,451–465.

Lifschitz, V. and Yang, F. (2013). Functional completion. *Journal of Applied Non-Classical Logics 23(1-2),*121–130.

McCarthy, J. (1959). Programs with common sense. Mechanization of thought processes, *Vol. I. London: Her Majesty's Stationery Office*.

McCarthy, J. (1977). Epistemological problems of artificial intelligence. *In Proceedings of the Fifth International Joint Conference on Artificial Intelligence, pages 1038-1044, Cambridge, MA.*

McCarthy, J. (1980). Circumscription — a form of non-monotonic reasoning. *Artificial Intelligence, 13*, 27–39.

McCarthy, J., and Patrick J. H. (1981). Some philosophical problems from the standpoint of artificial intelligence. *Readings in artificial intelligence. Morgan Kaufmann,* 431-450.

McDermott, D. V. & Doyle, J. (1980). Non-monotonic logic I. *Artificial Intelligence, 13*: 41-72.

Mueller, E. T. (2014). *Commonsense reasoning: an event calculus based approach*. Morgan Kaufmann

Modrak, V, and Slavomir B. (2015). Using axiomatic design and entropy to measure complexity in mass customization. *Procedia CIRP 34,* 87-92.

Otto, K., & Wood, K. (2001). *Product design: techniques in reverse engineering and new product design*. Prentice-Hall.

Oxman, R. and Gu, N. (2015). Theories and models of parametric design thinking. *In:Proceedings of the 33rd ECAADE Conference, Vienna,* 2-6.

Pahl, G. and Beitz, W. (1988). *Engineering Design: A Systematic Approach*, Springer-Verlag, Berlin.

Palmer, R. S., & Shapiro, V. (1993). Chain models of physical behavior for engineering analysis and design. *Research in Engineering Design*, *5*(3-4), 161-184.

Panos Y. P. (2015). Design Science: Why, What and How. *Design Science, 1, e1 doi:10.1017/dsj.2015.1*

Papadimitriou, C., (1994). Computational Complexity. *New York: Addison-Wesley*.

Pearl, J. (1988). Embracing causality in default reasoning. *Artificial Intelligence, 35(2)*, 259-271.

Pearl, J. (2000). *Causality: Models, Reasoning, and Inference,* Cambridge: Cambridge University Press.

Peirce, C., (1878), Deduction, Induction, and Hypothesis. *Popular Science Monthly, 13*, 470-82.

Popper, K. (2005). The logic of scientific discovery. Routledge.

Pugh, S. (1991). *Total design: integrated methods for successful product engineering*. Addison-Wesley.

Reiter, R.(1978) On closed world data bases, in: *H. Gaillaire and J. Minker (Eds.), Logic and Data Bases (Plenum Press, New York),* 55-76.

Reiter, R. (1980). A Logic for Default Reasoning, *Artificial Intelligence, vol.13 (1980)*, 81–132.

Schmidt, L. C., & Cagan, J. (1995). Recursive annealing: a computational model for machine design. *Research in Engineering Design*, *7(2), 102-125.*

Shanahan, M. (1999). The event calculus explained. *Artificial intelligence today. Springer, Berlin, Heidelberg*, 409-430.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal 27.3*, 379-423.

Simon, H. A., (1996). *The Sciences of the Artificial*, MIT Press, Cambridge, MA.

Sridharan, P., & Campbell, M. I. (2005). A study on the grammatical construction of function structures. *AIE EDAM*, *19(03)*, 139-160.

Stiny, G., & Gips, J. (1971). Shape Grammars and the Generative Specification of Painting and Sculpture. In *IFIP Congress (2) (Vol. 2, No. 3).*

Suh, N. P. (1990). *The principles of design.* Oxford University Press.

Ulrich, K. T., & Eppinger, S.D. (1995). *Product Design and Development*, McGraw-Hill, New York.

Umeda, Y., Takeda H., Tomiyama T., Yoshikawa H. (1990). Function behavior and structure. *AIENG'90, Applications of AI in Eng., Computational Mechanics Publications and Springer Verlag,* 177–193.

Umeda, Y., Ishii, M., Yoshioka, M., Shimomura, Y., & Tomiyama, T. (1996). Supporting conceptual design based on the function-behavior-state modeler. *AI Edam, 10(4)*, 275-288.

Vardi, M., (1982), "The Complexity of Relational Query Languages," in Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing, pp. 137–146.

Vescovi, M., Iwasaki Y., Fikes R., and Chandrasekaran B. (1993). CFRL: A Language for Specifying the Causal Functionality of Engineered Devices. *In Eleventh National Conference on AI: AAAI Press/MIT Press*, 626-633.

Woodbury, R. (2010). *Elements of Parametric Design*, Routledge, N.Y.

I. Appendix 1: Definitions

Appendix 1.1 Definitions pertaining to the world and its representation in *L*

1. Object

An object is a thing such as an IC engine, sub-assembly of the IC engine (e.g. piton sub-assembly), a part of an IC engine (e.g. piston, crank shaft, etc.), a feature of a part (e.g. cylinder bore, cavity in the cylinder head, etc.). Objects are represented using individual letters (or names) (e.g. a, b, c,..) in *L* (see Appendix 2.1).

2. Domain

A domain is a set of objects (e.g. domain of IC engine consisting of the IC engine, its sub-assemblies and its parts). A domain is represented in *L* as a set of individual letters or names, e.g. (a, b, c, etc.).

3. Simple (or atomic) Object

A simple object is one that is not composed from other objects (e.g. a piston is a simple object in the domain consisting of the parts and sub-assemblies of an IC engine).

4. Compound Object

A compound object is one that is composed from other objects (e.g. a piston sub-assembly, consisting of piston, piston pin, etc. is a compound object in the domain consisting of the parts and sub-assemblies of an IC engine).

5. Attribute or Property of an Object

An attribute or property of an object is a descriptor of the object. For example, being of a cylindrical shape, having a diameter of 90 mm, having a length of 50 mm, being made of aluminum, etc. are attributes or properties of a piston. An attribute of an object is represented in *L* using a function symbol whose argument is the individual letter representing the object and whose range is the set of individual letters that represent the values that the attribute can take.

6. State of an object

6.1 State of a simple object

The state of a simple object is the set of properties of the object and the relationships in which the object exists with other objects. For example, the state of a piston can be described in terms of its position in the engine cylinder, its temperature, etc..

6.2 State of a compound object

The state of a compound object is the set of properties that the object, and its constituent objects, have and the relationships that the object, and its constituent objects, have with other objects. For example, the state of a piston assembly can be described in terms of the attributes (e.g. position, velocity) and relationships of the piston assembly (e.g. type of motion relative to cylinder bore) and its constituent parts (e.g. temperature of piston, type of fit between piston and piston ring, etc.). The state of a simple or compound object is represented as a sentence in *L* (excluding the use of the causal implication (^{C}=>) sign in the sentence representing the state of the object). For example, the state of the simple object piston being in the top dead center position in the cylinder of the IC engine and having an instantaneous linear velocity of 0 mm/sec can be represented in *L* as: (Postion (Piston) = Top Dead Center) & (LinearVelocity (Piston) = 0 m/s). The state of the compound object of the piston sub-assembly describing the position and velocity of the piston at a given point in time and the frictional force on the piston ring contained in the piston assembly can be represented in *L* as: (Postition (Piston) = Top Dead Center) & (LinearVelocity (Piston) = 0 m/s) & (FrictionalForce (Piston Ring) = 100 newtons).

7. Atomic State

An atomic state is a state that is not composed from other states using conjunction or disjunction. For e.g., the rotational speed of a crank shaft being 3000 rpm is an atomic state of the crank shaft. Atomic states are represented as atomic sentences (sentences that are not formed form other sentences using conjunction or disjunction) in *L*. For example, the state of a piston being in the top dead center position in the cylinder of the IC

engine is an atomic state, and is represented in *L* using the atomic sentence*:* (Position (Piston) = Top Dead Center).

8. Compound State

A compound state is a state that is composed from other states using conjunction or disjunction. Compound states are represented as compound sentences (sentences that are formed form other sentences using conjunction or disjunction) in *L*. For example, the state of the piston being in the top dead center position in the cylinder of the IC engine and having an instantaneous linear velocity of 0 mm/sec is a compound state and can be represented in *L* using the compound sentence: (Position (Piston) = Top Dead Center) & (LinearVelocity (Piston) = 0 m/s).

9. Duration

A duration is continuous period of time with a start time and end time, including the start and end times. When the end time is the same as the start time, the duration reduces to an instant of time.

10. State-Duration [SD]

An SD is a state that persists over a duration of time. In this work, an SD abbreviated as (S, $[t_1, t_2]$) can represented in classical first order logic as follows:

- $\forall t ((t \geq t_1) \& (t \leq t_2)) \Rightarrow A(a, b, ..))$, where A is a predicate with the individuals a, b,..(but not t) as arguments.
- $\forall t((t \geq t_1) \& (t \leq t_2)) \Rightarrow \forall x \exists y... A(x, y, ..))$, where A is a predicate with the individual variables x, y,.. (but not t) as arguments.

For example, the SD of the inlet valve of an IC engine being open from 0 ms to 10 ms can be abbreviated in *L* as: (Open (Inlet Valve), [0 ms, 10 ms]) and fully represented as a sentence in classical first order logic as:

$\forall t$ (($t \geq 0$ ms) & ($t \leq 10$ ms)) => Open(Inlet Valve)).

11. Atomic SD

An atomic SD is an SD whose state is an atomic state. For example, the SD of the inlet valve of an IC engine being open from 0 ms to 10 ms, and represented in *L* as $\forall t$ (($t \geq 0$ ms) & ($t \leq 10$ ms)) => Open(Inlet Valve)), is an atomic SD.

12. Compound SD

A compound SD is an SD whose state is a compound state. For example, the SD of the inlet valve of an IC engine being open from 0 ms to 10 ms, and the exhaust valve being closed in the same duration, and represented in *L* as $\forall t$ (($t \geq 0$ ms) & ($t \leq 10$ ms)) => Open(Inlet Valve) & Closed (Exhaust Valve)) is a compound SD.

13. Process

A process is the evolution of states of objects over time. A process can be represented in *L* as a sentence formed as follows:

- By the conjunction or disjunction of one or more states and at least one SD;
- Containing at least two distinct instants of time in the union of the durations in all the SDs constituting the process.

For example, the following represents in *L* the process of opening and closing of the inlet and exhaust valves of an IC engine in a single cycle of its operation: $\forall t$ (($t \geq 0$ ms & $t \leq 10$ ms) => Open(Inlet Valve)) & $\forall t$(($t > 10$ ms & $t \leq 40$ ms) => Closed (Inlet Valve)) & $\forall t$ (($t \geq 0$ ms & $t \leq 30$ ms) => Closed (Exhaust Valve)) & $\forall t$ (($t > 30$ ms & $t \leq 40$ ms) => Open (Exhaust Valve)) & RotationalSpeed (Crank Shaft) = 3000 rpm).

13.1 Causal Process

A causal process is a process that includes descriptions of causal relationships among states or SDs (i.e., which combinations of states or SDs cause which other combinations of states or SDs).

A causal process is represented in *L* as a sentence containing one or more causal implication (^{C}=>) signs. Each instance of a causal implication (^{C}=>) sign in a sentence corresponds to a causal process. Each causal implication (^{C}=>) sign in a sentence should be immediately preceded by a sentence formed from states or SDs, using only the operations of conjunction and disjunction, called the input sentence of the causal process. Each causal implication (^{C}=>) sign in a sentence should be immediately succeeded by a sentence formed from states or SDs, using only the operations of conjunction and disjunction, called the output sentence of the causal process. In a causal process containing multiple causal implication ($^{c}\Rightarrow$) signs, causal implication ($^{c}\Rightarrow$) sign in the process denotes a causal sub-process in the given causal process.

For example, the following sentence represents the causal process of opening an inlet valve :

- $(\forall t((t = 0 \text{ ms}) \Rightarrow \text{Closed(Inlet Valve)}) \ \& \ \text{RotationalSpeed(Crank Shaft)} = 3000 \text{ rpm})) \ ^{c}\Rightarrow \forall t((t > 0 \text{ ms} \ \& \ t \leq 10 \text{ ms}) \Rightarrow \text{Open(Inlet Valve)})$.

The following sentence represents the causal process of opening and then closing an inlet valve:

- $(\forall t((t = 0 \text{ ms}) \Rightarrow \text{Closed(Inlet Valve)}) \ \& \ \text{RotationalSpeed(Crank Shaft)} = 3000 \text{ rpm}) \ ^{c}\Rightarrow \forall t((t > 0 \text{ ms} \ \& \ t \leq 10 \text{ ms}) \Rightarrow \text{Open(Inlet Valve)})) \ \& \ (\forall t((t = 10 \text{ ms}) \Rightarrow \text{Open(Inlet Valve)}) \ \& \ \text{RotationalSpeed(Crank Shaft)} = 3000 \text{ rpm}) \ ^{c}\Rightarrow \forall t((t = 20 \text{ ms}) \Rightarrow \text{Closed(Inlet Valve)}))$.

The first example of a process above contains a single $^{c}\Rightarrow$ sign and the second contains two $^{c}\Rightarrow$ signs.

Each state or SD in the input sentence of the causal process corresponding to a given causal implication ($^{c}\Rightarrow$) sign is called an input state or SD of the causal process. Similarly, each state or SD in the output sentence of the causal process corresponding to a given causal implication ($^{c}\Rightarrow$) sign is called an input state or SD of the causal process. In a causal process (P) if the output sentence of a causal sub-process (P1) is also an input sentence or logically implies or contained in the input sentence of another causal sub-process (P2), then the output sentence of P1 is called an intermediate sentence of the given causal process P. The states or SDs of an intermediate sentence of causal process are called intermediate states or SDs. All the input and output sentences of the causal sub-processes of a given causal process that are not intermediate sentences of the given causal process are also input and output sentences, respectively, of the given causal process.

A causal process represented in *L* has to satisfy the following conditions:

- Not flowing backward in time: If the causal process consists of only one causal implication ($^{c}\Rightarrow$) sign, then no output SD of the causal process should have the start time of its duration earlier than the start time of the duration of any input SD of the causal process. If the causal process consists of multiple causal implication ($^{c}\Rightarrow$) signs, then for any causal sub-process corresponding to a causal implication ($^{c}\Rightarrow$) sign in the given causal process, no output SD of the causal sub-process should have the start time of its duration earlier than the start time of the duration of any input SD of the causal sub-process. This condition subsumes within it the condition of non-circularity of causal processes, i.e., the output SD of a causal process cannot have an effect on the input SD of the causal process.
- Non-circularity: The output sentence of a causal process cannot have an effect on the input sentence of the causal process. This implies the condition of non-reflexivity, i.e., a state or SD cannot cause itself.
- Non-redundancy: Neither the input nor the output sentence of a causal process should not be a tautology. The input sentence of a causal process should not logically imply its output sentence.

More rules pertaining to the usage of the causal implication ($^{c}\Rightarrow$) sign are given in Section 2.1.

13.2 Recursive Process

A recursive process is represented in *L* by specifying the following: a sentence that is a starting criterion for the process; a sentence that is a stopping criterion, if required, for the process; a causal process that executes when its starting condition is satisfied and so long as its stopping condition is not true; and with each iteration of the causal process causing the starting condition for the next iteration of the process to be true. For example, the recursive process of keeping the inlet valve open in an IC engine for the first 10 ms of each cycle of duration 40ms while the ignition is on can be represented as follows:

$\forall t(t = 0 \text{ ms} <=> T(t))$……(initialization of time as a starting criterion; T is a predicate used to enable the assignment of a specific instant of time to the individual variable t)……..……...(1)

$\forall t(t \geq 0 \text{ ms} \ \& \ t \leq 4000 \text{ ms} => On(ignition))$…….....(initiation and stopping criterion)..……...(2)

$\forall t(t = 0 \text{ ms} => Closed(Inlet Valve))$…….....(initiation and stopping criterion)..…….…..(3)

$\forall t($
$($
$T(t) \ \& \ On(ignition) \ \& \ Closed \ (Inlet \ Valve)$
$)$
$\overset{c}{\Rightarrow}$
$($
$\forall t_1($
$((t_1 > t \ \& \ t_1 \leq t + 10 \text{ ms}) => Open \ (Inlet \ Valve))$
$\&$
$((t_1 > t + 10 \text{ ms} \ \& \ t_1 \leq t + 40 \text{ ms}) => Closed \ (Inlet \ Valve))$
$)$
$\&$
$\sim T(t) \ \& \ T(t+40 \text{ ms})$……..(condition for next iteration of process)..……(4)
$)$
)…….…(recursive causal process)..…….….(5)

13.3 Deterministic and Non-deterministic Processes

Consider a process (causal or non-causal) executed n times. The process is deterministic if and only if, in each of the n executions of the process, the truth value of each state or SD constituting the process remains the same. A process that is not deterministic is non-deterministic (or stochastic).

14. Probability of success of a causal process

The probability of success of a causal process is defined as the conditional probability of the output sentence of the causal process being true given the truth of the input sentence of the causal process.

Let C be a causal process whose input sentence is A and output sentence is B.

Then the probability of success of a causal process C (P(C)) ≡ P(B/A).

Appendix 1.2 Definitions pertaining to designs, their representation in *L*, and the explication of their attributes in terms of their representation in *L*

1. Requirement

A sentence that has to be made true by the structure and behavior of a design. A requirement is represented in *L* as a sentence that does not include a causal implication sign.

2. Functional Requirement (FR)

A requirement that is an abstraction of the behavior expected from a design. An FR is represented in *L* as a sentence (not including a causal implication sign) containing existentially quantified individual variables whose instantiation as individual constants (representing specific objects) can be used to obtain a representation of behavior that satisfies the FR by logical implication. For example, $\exists t_1 \exists t_2 \exists v \forall t((t \geq t_1 \ \& \ t \leq t_2 \ \& \ t_1 \leq t_2) \Rightarrow (\text{Power(IC Engine)} = v \ \& \ v > 0))$ (i.e., the IC engine generates powe) is a representation of an FR, such that assigning specific values to the variables t_1, t_2 and v can be used to obtain a representation of behavior such as $\forall t((t \geq 30 \text{ ms} \ \& \ t \leq 40 \text{ ms}) \Rightarrow (\text{Power(IC Engine)} = 75 \text{ hp}))$ (i.e. the IC engine generates power of 75 hp during 30 ms to 40 ms of its cycle of operation) that satisfies the FR by logical implication.

3. Constraint

A sentence that should not be negated by the structure and behavior of a design. constraint is represented in *L* as a sentence that does not include a causal implication sign.

4. Structure of a design

The attributes of and relationships among the objects that constitute that constitute the design. A structure is represented in *L* as a set of sentences that do not include SDs, and do not include the causal implication sign.

5. Behavior of a design

A process (causal or non-causal) involving the objects that constitute the design. Behavior is represented in *L* as a set of sentences that could (and typically would) include the causal implication sign.

6. Solution Concept

The structure and behavior of a design that are intended to satisfy the requirements and constraints of the design.

7. Design

A system of requirements (including FRs), constraints, structure and behavior.

8. Sub-Design

A sub-system (d) of a design (D) such that: d is also a design; and the FRs of d are contained in the hierarchy of FRs of D.

9. Sibling Sub-Designs

Sub-designs (d1 and d2) of a design D are sibling sub-designs of design D if and only if the highest level FRs of d1 and d2, are siblings in the hierarchy of FRs of D.

10. Solution-neutrality of an FR

An FR in the hierarchy of FRs of a design is solution-neutral if and only if it is not defined in terms of any object that is not already contained in a solution concept selected for a higher level FR of which the given FR is a descendant. For example, let S_1 (valve train linking crank shaft motion to inlet valve opening and closing) be a solution concept for (the highest level FR in the hierarchy of FRs) FR_1 (the inlet valve is opened) and S_{11} (valve train with overhead cam and follower) be a solution concept for FR_{11} (motion is transferred from crank shaft to inlet valve), which is a child of FR_1. Since FR_{11} does not refer to objects (e.g. overhead cam) in S_{11} that are not already contained in the higher level solution concept (S_1), FR_{11} is stated in a solution-neutral manner. However, if FR_{11} were stated as "inlet valve is pushed by overhead cam" then FR_{11} would not be solution-neutral, since the object overhead cam is contained in the solution concept (S_{11}) for FR_{11}, but not contained in the solution concept (S_1) of FR_1. Similarly, if $FR_{11...1}$ is the child of a child of (and so on) of FR_1 then, for $FR_{11...1}$ to be solution-neutral, $FR_{11...1}$ can refer to any object (e.g. crank shaft) explicitly contained in the representation of the solution concepts of the parent of, the parent of parent of (and so on) of $FR_{11...1}$, but cannot refer to any object (e.g. overhead cam) that is contained in the solution concept for $FR_{11...1}$ but not in the solution concepts of the parent of, the parent of parent of (and so on) of $FR_{11...1}$.

11. Fragment of a design

A fragment of a design is a set of objects or sentences of a design. Examples of fragments of a design of an IC engine include: inlet valve; Open(inlet valve); $\forall t((t > 0 \text{ ms} \ \& \ t \leq 10 \text{ ms}) \Rightarrow \text{Open(Inlet Valve)})$; (Open(Inlet Valve) & Position(Piston) = BDC) $^{c}\Rightarrow$ InCylnder(Air-Fuel Mix); etc.

12. Dependency among fragments of designs

A fragment B of a design depends on a fragment A of a design if and only if:

- A is contained in B; or

- A causes B; or
- A implies B, where A is a sentence that describes structure or behavior and B is an FR or constraint. A special case of this condition is when A is a sentence that implies B by virtue of B being an an existentially quantified form of A (e.g. behavior A logically implies FR B, since the instantiation of one or more existentially quantified variables in B with individual constants yields A).

13. Dependency (coupling) among designs

A design D2 depends on design D1 if and only if there is a term in D2 that depends on a term in D1.

14. Dependency (coupling) matrix of a design

Consider the square matrix shown in Supplementary Figure 4. $\{D_1, D_2, ...D_i,...D_n\}$ in the matrix are all the sub-designs in the hierarchy of sub-designs in a given design D, ordered such that the lower triangular portion of the matrix in Supplementary Figure 1 has the minimum possible number of non-zero elements. If D_i and D_j are not sibling sub-designs in the hierarchy of sub-designs of D (which includes the case where i =j), then $K_{i,j} = 0$. If D_i and D_j are sibling sub-designs in the hierarchy of sub-designs of D, then:

- $K_{i,j} = N_{i,j}$; where $N_{i,j}$ is the number of dependencies from D_i to D_j.
- A dependency from Di to Dj is called a non-circular (or linear) dependency if and only if i< j, i.e., if and only if the cell defined by row i and column j in the matrix shown in Supplementary Figure 1 lies in the upper triangular region of the matrix.
- A dependency from Di to Dj is called a circular (or non-linear) dependency if and only if i> j, i.e., if and only if the cell defined by row i and column j in the matrix shown in Supplementary Figure 1 lies in the lower triangular region of the matrix.

A matrix, as shown in Supplementary Figure 1, satisfying the description given above is called the dependency matrix of design D.

15. Coupled Design

A design is coupled if and only if there is a dependency among any two sibling sub-designs of the design. That is, a design is coupled if and only if there is at least one non-zero element in its dependency matrix.

16. Uncoupled Design

A design that is not a coupled design is an uncoupled design. That is, a design is uncoupled if and only if all the elements in its dependency matrix are zero.

17. Decoupled design

A design is decoupled if and only if all the elements in the lower triangular portion of its dependency matrix are zero and there exists at least one non-zero element in the upper triangular portion of the dependency matrix.

18. Probability of Success of a Design

The probability that all the requirements and constraints of a design are satisfied.

19. Information Content of a Design

The logarithm to the base 2 of the reciprocal of the probability of success of a design (Suh(1990)).

That is, information content of design D

$\equiv \log_2(P(D))$; where P(D) is the probability of success of design D.

20. Valid Design

A design whose representation in *L* does not contain any inconsistencies within the logic of *L*. A design that is not valid will contain contradications in its representation (e.g. in structure, behavior, FRs, constraints, their relationships, etc.), which will make the design impossible to realize and/or render unreliable assessments of the satisfaction of FRs and constraints by the structure and behavior of the design.

21. Sound Design

A design whose structure and behavior satisfy its requirements and constraints.

22. Dimensionality of a design

The dimensionality of a design is the minimum number of independent attributes (parameters) required to completely specify the design. The dimensionality of a design is specific to the level of abstraction of a design. For example, the dimensionality of a design at the conceptual design stage is likely to be lower than its dimensionality at the detailed design stage.

23. Degree of parametrization of a design

The degree of parametrization of a design is defined as the ratio of the total number of attributes used to completely specify a design to the dimensionality of the design. That is:

Degree of parametrization of design D (g(D)) ≡ j/k; where:

- j is the number of parameters used to completely specify the design D
- k is the dimensionality of design D

For example, let attributes required to completely specify the geometry of an inlet valve be as follows: base diameter, base thickness, stem diameter and stem length. Thus the total number of attributes required to completely specify the geometry of the inlet valve (j) = 4. However, let the practice of designing of inlet valves follow the following rules of thumb:

- Base thickness = 1/10th of base diameter
- Stem diameter = 1/4th of base diameter

Then the minimum number of independent attributes required to completely specify the geometry of the inlet valve are base diameter and stem length. Thus the dimensionality of the geometry of the inlet valve (k) = 2. Thus the degree of parametrization of the inlet valve = j/k = 4/2 = 2.

24. The entropy of dimensionality of a design

The entropy of dimensionality of a design D ($E^{Dim}(D)$) is defined as the natural log of the dimensionality of D. That is:

$E^{dim}(D) \equiv \ln(K(D))$

where K(D) is the dimensionality of design D.

25. Degree of solution-neutrality of an FR

The degree of solution-neutrality of an FR in the hierarchy of FRs of a design is defined as the total number of potential solutions for the FR (aggregated over the potential solutions for all its descendant FRs, if any, in the hierarchy of FRs). For an FR that is at the lowest level in the hierarchy of FRs of a design, the degree of solution-neutrality is simply the number of solution concepts enumerated for that FR. For an FR that is not at the lowest level in the hierarchy of FRs, the degree of solution-neutrality is recursively enumerated over its descendant FRs in the hierarchy of FRs, as follows:

$N_i = \sum_{j=1}^{S(i)} N_{i,j}$

$N_{i,j} \equiv \prod_{k=1}^{m(i,j)} N_{i,j,k}$; where:

- N_i is the degree of solution-neutrality of FR_i;
- S(i) is the number of solution concepts, at the level of abstraction of FR_i, enumerated (e.g. from a catalog of solution concepts) to potentially satisfy FR_i;
- $N_{i,j}$ is the degree of solution-neutrality of FR_i considering only the solution concept j defined at the level of abstraction of FR_i;
- m (i, j) is the number of child FRs of FR_i entailed by the choice of solution concept j for FR_i;
- $N_{i,j,k}$ is the degree of solution-neutrality of $FR_{i,k}$ entailed by the choice of solution concept j for FR_i of design D.

For example, let FR_1 be "Allow air-fuel mix into the IC engine". Solution concepts at the level of abstraction of FR_1 could be enumerated as follows: $S_1 \equiv$ Poppet valve mechanism; $S_2 \equiv$ Butterfly valve mechanism; $S_3 \equiv$ Unidirectional valve mechanism. Consider S_1 (Poppet valve mechanism). This solution concept could entail the following next level FRs: $FR_{11} \equiv$ Open poppet valve; $FR_{12} \equiv$ Close poppet valve. The solution concepts for FR_{11}

at the level of abstraction of FR_{11} could be enumerated as follows: $S_{11} \equiv$ Overhead cam mechanism; $S_2 \equiv$ Cam, pushrod and tappet mechanism. Similarly, the solution concepts for FR_{12} at the level of abstraction of FR_{12} could be enumerated as follows: $S_{21} \equiv$ Cam slot mechanism; $S_{22} \equiv$ Spring. Similarly let 2 second level FRs be entailed for each of the solution concepts S2 and S3 for FR1, and let each of the second level FRs entailed by S2 and S3, respectively, have 2 solution concepts enumerated at the corresponding level of abstraction of FRs. Thus the degree of solution-neutrality of $FR_{11} = 2$ (number of elements in the set $\{S_{11}, S_{12}\}$, since FR_{11} is at lowest level of FRs considered in this example), and similarly, that of $FR_{12} = 2$. Therefore the solution-neutrality of FR_1 considering S_1 as the selected solution concept =2*2 =4. Similarly, the degree of solution-nuetrality of FR_1 considering S_1 and S_2, respectively as the solution-concept selected for FR_1, is 4 each. Therefore the solution-neutrality of FR_1 (obtained as the sum of its solution-neutrality considering each of S1, S2 and S3 individually as the selected solution concept) = 4+4+4 = 12.

26. Degree of solution-neutrality of the totality of FRs of a design

The degree of solution-neutrality of the totality of FRs of a design is the product of the degrees of solution-neutrality of the highest level FRs of the design. Thus:

$N_D \equiv \prod_{i=1}^{m} N_i$; where:

- N_D is the degree of solution-neutrality of design D;
- m is the number of highest level FRs of design D;
- N_i is the degree of solution-neutrality of highest level FR_i of design D.

26.1 Alternative definition of the degree of solution-neutrality of the FRs of a design

An alternative definition of the degree of solution-neutrality of the totality of FRs of a design is given below:

$N_D \equiv \prod_{i=1}^{m} \prod_{j=1}^{n(i)} (f_i Z)^{g(i,j)}$; where:

- N_D is the degree of solution-neutrality of the totality of FRs of D;
- i is a given level in the levels of hierarchy of FRs of design D;
- m is the number of levels in the hierarchy of FRs of design D;
- j is a given FR in level i of hierarchy of FRs of D;
- n(i) is the number of FRs in level i of FRs of D;
- $f_i = 1/10^{(i-1)}$
- g(i, j) = 1 if $FR_{i,j}$ is solution-neutral; and 0 otherwise
- Z is a number greater than 1 (e.g. 1000), chosen to reflect the average degree of solution-neutrality lost if a single potential solution concept is eliminated due to a lack of solution-neutrality of an FR at the highest level in the hierarchy of FRs of the design.

This alternative definition of the degree of solution-neutrality of a design considers solution-neutrality of an FR as a binary variable (using the term g(i, j) above), emphasizes the relative importance of solution-neutrality at the higher levels in the hierarchy of FRs (using the term fi above) and does not require an explicit enumeration of all possible solution concepts over all possible hierarchies of FRs of a design.

27. Entropy of solution-neutrality of the FRs of a design

The entropy of solution-neutrality associated with an FR_i is defined as the natural log of the reciprocal of the degree of solution-neutrality of the FR. The entropy of solution-neutrality of a design (d) ($E^{SN}(D)$) is the natural log of the reciprocal of the degree of solution-neutrality of the totality of FRs in the hierarchy of FRs of the design.

Thus $E^{SN}(D) = \ln(1/O)$, where O is the degree of solution-neutrality of the totality of FRs in the hierarchy of FRs of the design.

28. The degree of flexibility of specifications of a design (M)

The degree of flexibility of specifications of a design is the number of combinations of discrete values, of the independent attributes required to completely specify the design, which are consistent with (i.e. do not violate) the specifications of the design.

The number of discrete values that an attribute (X) can take is defined as follows:

- Case 1: X is a discrete variable, i.e., X can take only discrete values. Then the number of discrete values that X can take is the number of values of X that lie within the range of X that is specified in the requirements or constraints of D. For example, if X is the state of an on-off switch, then X can take only two discrete values (on and off).
- Case 2: X is a continuous variable. Let the specified range of X be $[x_1, x_2]$. The range of discrete values of x is defined as follows: $(x_1, x_1 + r, x_1 + 2r, x_1 + 3r, \ldots. x_1 + nr, x_2)$, where r is the precision of the measurement system specified to measure x, and n is the maximum whole number such that $x_1 + nr \leq x_2$. For example, if X is a continuous variable with specified range of [1, 3.5] units, and the precision of the measuring instrument specified to measure X is 0.5 units, then the range of discrete values of X is given by [1, 1.5, 2.0, 2.5, 3.0, 3.5].

Thus if a design D is completely specified by 2 attributes X1 (with m discrete values in its specified range) and X2 (with n discrete values in its specified range), then the degree of flexibility of specifications of D is given by m*n.

29. Entropy of specifications of a design ($E^S(D)$)

The entropy of specifications ($E^S(D)$) of a design D is defined as follows:

$E^S(D) = \ln(1/M)$, where M is the degree of flexibility of specifications of the design D.

30. Entropy of variability of a design ($E^V(D)$)

The entropy of variability of a design D, denoted by $E^V(D)$, is defined as follows:

$E^V(D) = \ln[V]$; where V is the number of combinations of discrete values in the natural range of variation (e.g. within 3 sigma confidence limits) of the attributes in the structure and behavior of D.

For example, if the natural range of variation of the diameter of the base of the inlet valve of an IC engine is [19.9, 20.1] mm and that of the diameter of the stem of the inlet valve is [4.95, 5.05] mm, both specified to be measured with a precision of 0.01 mm, then the entropy of variability of both of the above attributes taken together = ln((20.1-19.9)/0.01*(5.05-4.95)/0.01)) = ln(20*10) = ln(200).

The approach to defining the entropy of variability of a design described above is partly analogous to that defined in Krus (2015) to estimate design information entropy for a the range of a continuous variable divided into equal regions, each with an identical probability of occurrence.

31. Entropy of bias of a design ($E^B(D)$)

The entropy of bias of a design D, denoted by $E^B(D)$, is defined as follows:

$E^B(D) = \ln[B]$; where B is the number of combinations of discrete values in the difference between the mean and target values of the attributes in the structure and behavior of D.

For example, if the target value of the diameter of the inlet valve of an IC engine is 20 mm and mean of the diameter of the inlet valve is 19.95 mm, the process shift of the diameter of the base of the inlet valve is (19.95 - 20) mm = 0.05 mm. If the specified precision of measurement of the diameter of the inlet valve is 0.01 mm, then the number of discrete values in the process shift of the diameter of the inlet valve is [0.05/0.01] = 5. Similarly, if the number of discrete values in the process shift of the diameter of the stem of the inlet valve is 5, then the entropy of process shift of the diameters of the base and stem of the inlet valve, combined, is equal to ln(5*10) = ln(50).

32. Degree of coupling of a design

The degree of coupling of a design is defined as follows:

$$H \equiv \prod_{i=1}^{L} (J_1 g_i)^{c_i} (J_2 g_i u_i)$$

Where:

H ≡ Degree of coupling of the given design (D);

L ≡ Number of levels in the hierarchy of FRs of D;

i ≡ A given level of FRs in the hierarchy of FRs of D;

$g_i \equiv 1/10^{(i-1)}$;

J_1 ≡ A number greater than 1 and sufficiently large (e.g. $10^{(L-1)}$) to reflect the importance accorded to a circular coupling among sibling FRs of the design;

J_2 ≡ A number greater than 1 and sufficiently smaller than J_1 (e.g. $J_1/1000$) to reflect the much lower importance accorded to a non-circular coupling compared to a circular coupling in a design;

c_i ≡ Number of circular couplings among sibling sub-designs in level i of FRs;

u_i ≡ Number of non-circular couplings among sibling sub-designs in level i of FRs.

The above definition of degree of change manageability of a design gives a higher importance (weight, using the term g_i above) to couplings among higher level FRs than to couplings among lower level FRs.

33. Entropy of change manageability of a design ($E^{CM}(D)$)

The entropy of change manageability of a design D, denoted by $E^{CM}(D)$, is defined as follows:

$$E^{CM}(D) = \ln[H]$$

Where H is the degree of coupling of the design D.

34. Entropy of requirements and constraints (including specifications) of a design ($E^R(D)$)

The entropy of the requirements and constraints (including specifications) of a design D, denoted by $E^R(D)$, is defined as follows:

$E^R(D) = \ln[L^k/(O*M)]$; where:

- L is a sufficiently large constant real number to ensure that $E^R(D)$ is positive for the design D;
- k is the dimensionality of D;
- O is the degree of solution-neutrality of the totality of FRs of D;
- M is the degree of flexibility of specifications of D.

35. Entropy of Satisfaction of Requirements and Constraints of a Design($E^{RS}(D)$)

The entropy of satisfaction of requirements and constraints of a design is defined as the sum of the following types of entropies of the design: entropy of requirements and constraints; entropy of variability; and entropy of bias. That is:

$$E^{RS}(D) = E^R(D) + E^V(D) + E^B(D)$$
$$= \ln(L^k/MO) + \ln(V) + \ln(B)$$
$$= \ln(L^kVB/MO).$$

36. Entropy of conformance of a design to its specifications

The entropy of conformance to specifications of a design D is defined as:

$E^{Conf}(D) = \ln(\Omega)$; where:

$$\Omega = \prod_{i=1}^{n} \Psi_i$$

- $\Psi_i = 1$ if and only if the natural variation of parameter i lies within the specification limits of parameter i;
- $\Psi_i = N_i^{10}$ if and only if the natural variation of parameter lies outside the specification limits of parameter i;
- Ni = number of points within the natural variation of parameter I that lie outside the specification limits of parameter i.

The entropy of conformance to specifications of a design D subsumes within it considerations of variability and bias, and could be used instead of entropy of satisfaction of requirements and constraints in cases where the preference is for the following: a) Subsuming considerations of variability and bias into the overall consideration of conformance to specifications; and/or b) Estimation of a value of entropy of the design rather than bounds on entropy of a design. For example, when the satisfaction of an FR is determined as a logical implication of the FR by some behavior, only the lower bound of probability of satisfaction of an FR can be estimated based on the probability of occurrence of the behavior that satisfies the FR (since If A => B, then $P(A) \leq P(B)$).

33. Entropy of a Design

The entropy of a design (E(D)) is defined as the sum of the following: entropy of satisfaction of requirements and constraints; and entropy of change manageability. That is:

$$E(D) = E^{RS}(D) + E^{CM}(D)$$
$$= \ln(L^k VB/M) + \ln(H)$$
$$= \ln(L^k VBH/MO).$$

33.1 Alternative definition of entropy of a design

An alternative definition of the entropy of a design (E(D)) is as the sum of the following: entropy of solution neutrality, entropy of dimensionality, entropy of conformance and entropy of change manageability. That is:

$E(D) = E^{SN}(D) + E^{Dim}(D) + E^{Conf}(D) + E^{CM}(D)$.

Situations in which this definition of entropy might be preferred include the situations where the use of entropy of conformance to specifications (as opposed to entropy of satisfaction of requirements and constraints) might be preferred (as described above in definition of entropy of conformance to specifications).

34. Quality of a design from a design theoretic perspective (Q(D))

The quality of a design (D) from a design theoretic perspective, denoted by Q(D), is an index of the extent to which a design exhibits the attributes of a good design as discussed in Section 4.2. In this work Q(D) is a real number in the range $[0, \infty]$. Q(D) varies with the entropy of the design (E(D)) as described in Section 4.2.

35. Standardization of parts or components of a design

A part or component of a design is defined to be standardized if and only if there are at least two instances of the part or component in the design and all instances of the part or component in the design carry identical specifications.

The degree of standardization of parts or components of design D is defined as:

$S(D) \equiv N(D)/U(D)$; where:

- N(D) is the total number of parts or components of design D; and
- U(D) is the total number of unique parts or components of design D.

For the sake of simplicity, in this work the scope of consideration of standardization of parts or components is restricted to the product and does not span across products (i.e., the definition of standardization in this work does not include reuse of a part or component across products, or industry-wide standards (e.g. of nuts or bolts). However, the scope of consideration of standardization could be easily broadened as follows:

S(D) ≡ N(D)*R(D)/U(D); where S(D), N(D) and U(D) are as defined above, and R(D) ≡ the number of unique parts or components that are specified as per industry standards or are re-used across the products of the organization

36. Symmetry of parts or components of a design

The degree of symmetry of parts or components of design D is defined as:

Sym(D) ≡ $(1/U(D))\sum_{i \in Z} k_i$; where:

- Z is the set of unique parts or components of design D
- U(D) is the total number of unique parts or components of design D.
- k_i = the total number of planes of bilateral symmetry or axes of radial symmetry that are used in specifying the part or component i.

II. Appendix 2: Calculus of Causal Processes

Let SD represent a state-duration (as defined in Appendix 1.1). Then the following form a calculus of processes:

1. Non-reflexivity: $\sim(SD_1 \overset{c}{\Rightarrow} SD_1)$. That is an SD cannot cause itself.
2. Non-symmetry (unidirectionality of causation): $(SD_1 \overset{c}{\Rightarrow} SD_1) \Rightarrow \sim (SD_2 \overset{c}{\Rightarrow} SD_1)$
3. Associativity: $(SD_1 \overset{c}{\Rightarrow} (SD_2 \overset{c}{\Rightarrow} SD_3)) \Leftrightarrow (SD_1 \overset{c}{\Rightarrow} SD_2) \overset{c}{\Rightarrow} SD_3)) \Leftrightarrow (SD_1 \overset{c}{\Rightarrow} SD_2 \overset{c}{\Rightarrow} SD_3)$
4. Distributivity over output SDs: $SD_1 \overset{c}{\Rightarrow} (SD_2 \vee SD_3) \Leftrightarrow (SD_1 \overset{c}{\Rightarrow} SD_2) \vee (SD_1 \overset{c}{\Rightarrow} SD_3)$
5. Distributivity over input SDs: $(SD_1 \vee SD_2) \overset{c}{\Rightarrow} SD_3 \Leftrightarrow (SD_1 \overset{c}{\Rightarrow} SD_3) \& (SD_2 \overset{c}{\Rightarrow} SD_3)$
6. Composition (sequential conjunction): $(SD_1 \overset{c}{\Rightarrow} SD_2) \& (SD_2 \overset{c}{\Rightarrow} SD_3) \Leftrightarrow (SD_1 \overset{c}{\Rightarrow} SD_2 \overset{c}{\Rightarrow} SD_3)$
7. Composition (parallel conjunction): $((SD_1 \overset{c}{\Rightarrow} SD_2) \& (SD_3 \overset{c}{\Rightarrow} SD_4)) \Rightarrow ((SD_1 \& SD_2) \overset{c}{\Rightarrow} (SD_3 \& SD_4))$
8. Composition (parallel disjunction): $((SD_1 \overset{c}{\Rightarrow} SD_2) \vee (SD_3 \overset{c}{\Rightarrow} SD_4)) \Rightarrow ((SD_1 \vee SD_2) \overset{c}{\Rightarrow} (SD_3 \vee SD_4))$
9. Abstraction: $(SD_1 \overset{c}{\Rightarrow} SD_2) \Rightarrow (SD'_1 \overset{c}{\Rightarrow} SD'_2)$

   Where SD'_i is an existentially quantified sentence, from which the sentence SD_i is obtained by assigning values to one or more existentially quantified variables in SD'_i.

The rules described above are equally applicable when the SDs above are substituted by sentences that are logical combinations of SDs using only the & or v connectives.

III. Supplementary Material

III.1 Supplementary Figures

-------Supplementary Figure 1 (a) goes here----------

-------Supplementary Figure 1 (b) goes here----------

-------Supplementary Figure 2 goes here----------

-------Supplementary Figure 3 (a) goes here----------

-------Supplementary Figure 3 (b) goes here----------

-------Supplementary Figure 3 (b) goes here----------

-------Supplementary Figure 4 goes here----------

III.2 Supplementary Tables

-------Supplementary Table 1 goes here----------

-------Supplementary Table 2 goes here----------

1V. Figures

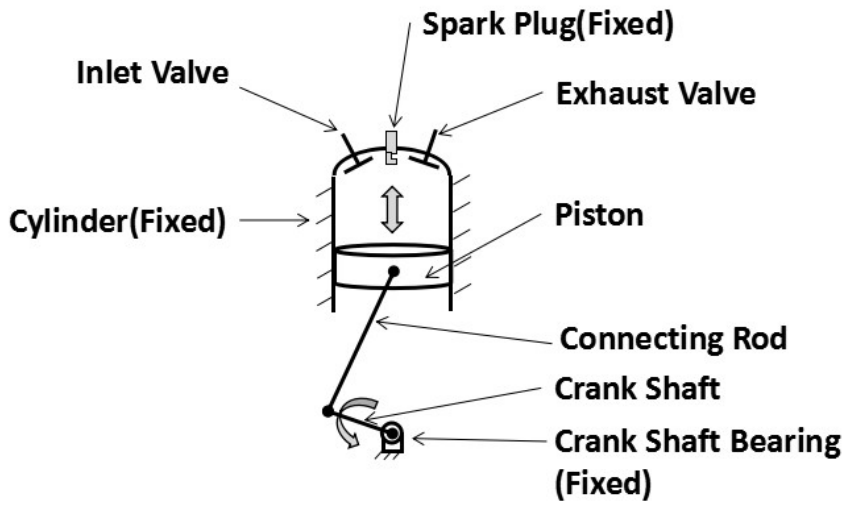

Figure 1(a): Schematic of a spark ignition IC engine.

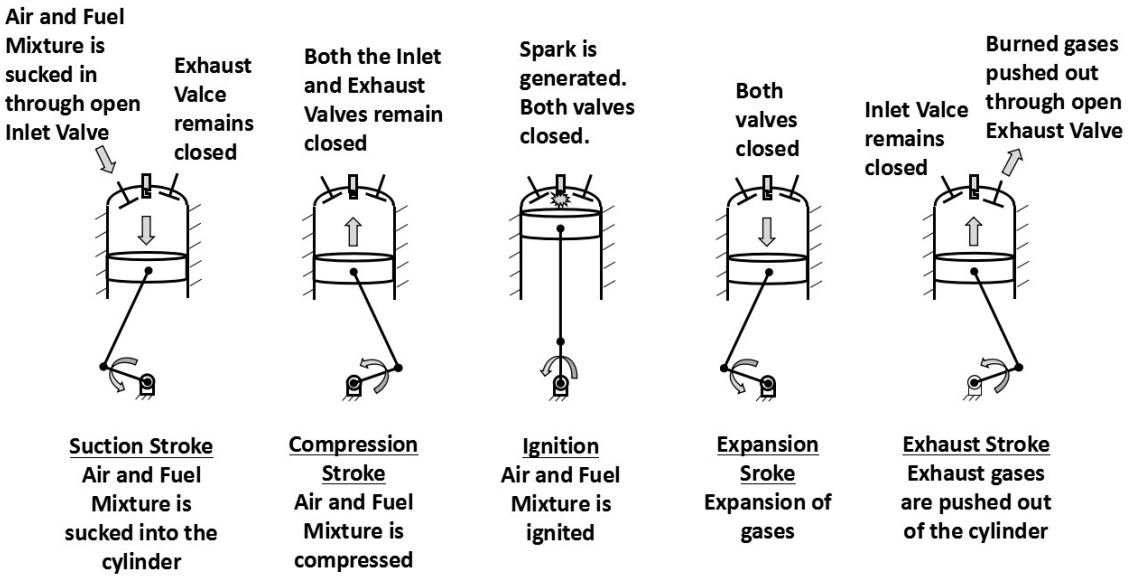

Figure 1(b): Schematic operation of a four stroke spark ignition IC engine.

	D	D_1	D_{11}	D_{12}	D_{13}	D_2	D_{21}	D_{22}	D_{23}	D_3	D_{31}	D_{32}	D_4	D_{41}	D_{42}	D_{43}	D_5	D_{51}	D_{52}	D_{53}
D																				
D_1						5							1				1			
D_{11}				1																
D_{12}																				
D_{13}																				
D_2										3			3							
D_{21}								1												
D_{22}																				
D_{23}																				
D_3													1							
D_{31}												1								
D_{32}																				
D_4																	4			
D_{41}															1					
D_{42}																1				
D_{43}																				
D_5																				
D_{51}																				1
D_{52}																				1
D_{53}																				

Figure 2: Coupling (dependency) matrix of the of the illustrative IC engine.

Entropy Component	Entropy Component	Overall Relative Weight	Entropy (Unweighted)	Entropy (Weighted)
Requirements Satisfaction	Solution-neutrality	7.00%	0.00	0.000
	Dimensionality	7.00%	207.23	14.506
	Conformance	56.00%	139.23	77.971
	Subtotal			***92.478***
Change Manageability	Circular Coupling	*29.97%*	*0*	*0.000*
	Non-circular Coupling	*0.03%*	*20.95*	*0.006*
	Subtotal			***0.006***
Total				**92.484**

Figure 3 (a): Entropy of the illustrative IC engine.

Relative Weights of Components of Entropy of Design				
Entropy Component	**Relative Weight**	**Entropy Sub-component**	**Relative Weight**	**Overall Relative Weight**
Requirements Satisfaction	70%	Solution Neutrality	10.00%	**7.00%**
		Dimensionality	10.00%	**7.00%**
		Conformance	80.00%	**56.00%**
Change Manageability	30%	Circular Coupling	99.90%	**29.97%**
		Non-circular Coupling	0.10%	**0.03%**
Total	**100%**			**100%**

Figure 3(b): Relative weights of various components of entropy of the design of the illustrative IC engine.

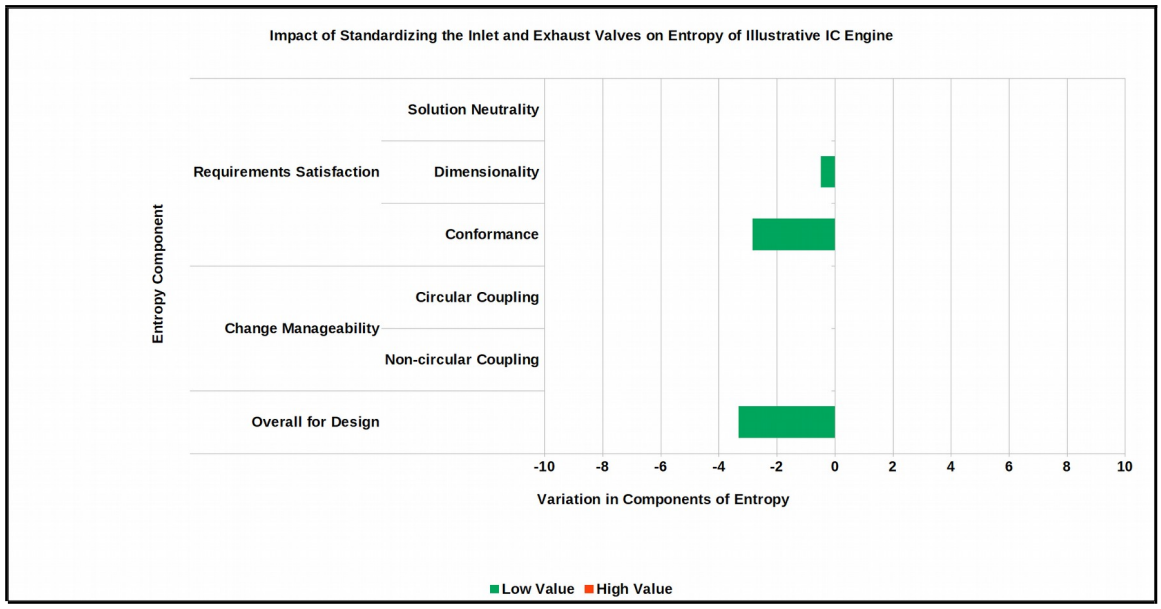

Figure 4 (a): Potential impact of improvements in the design of the illustrative IC engine on the entropy of the IC engine: Standardizing inlet and exhaust valves.

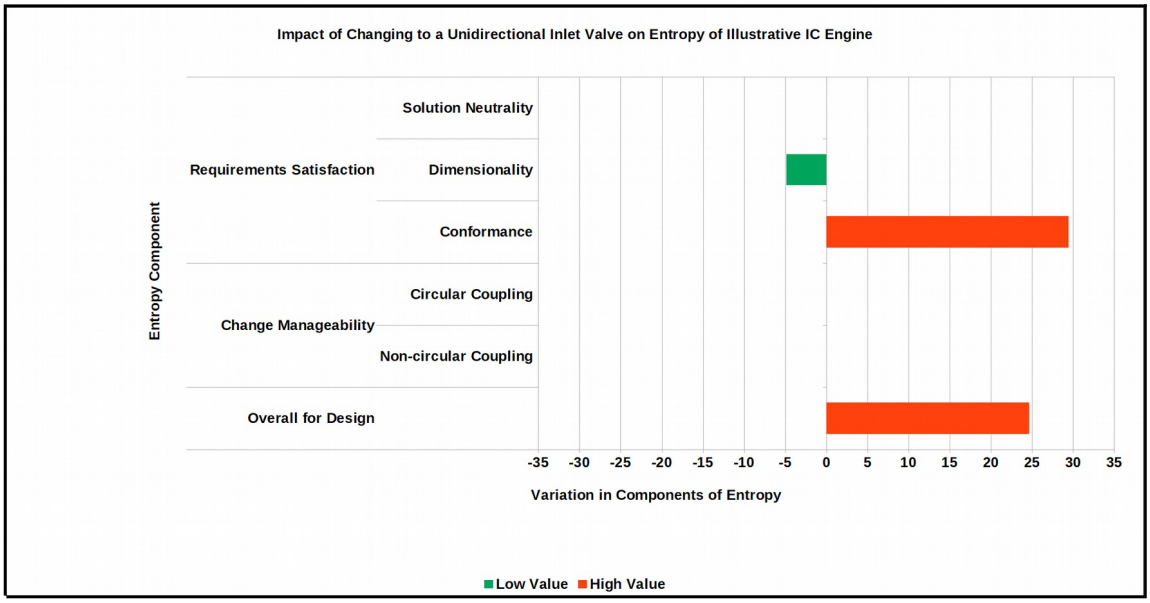

Figure 4 (b): Impact of potential improvements in the design of the illustrative IC engine on the entropy of the IC engine: Changing the inlet valve to a unidirectional valve.

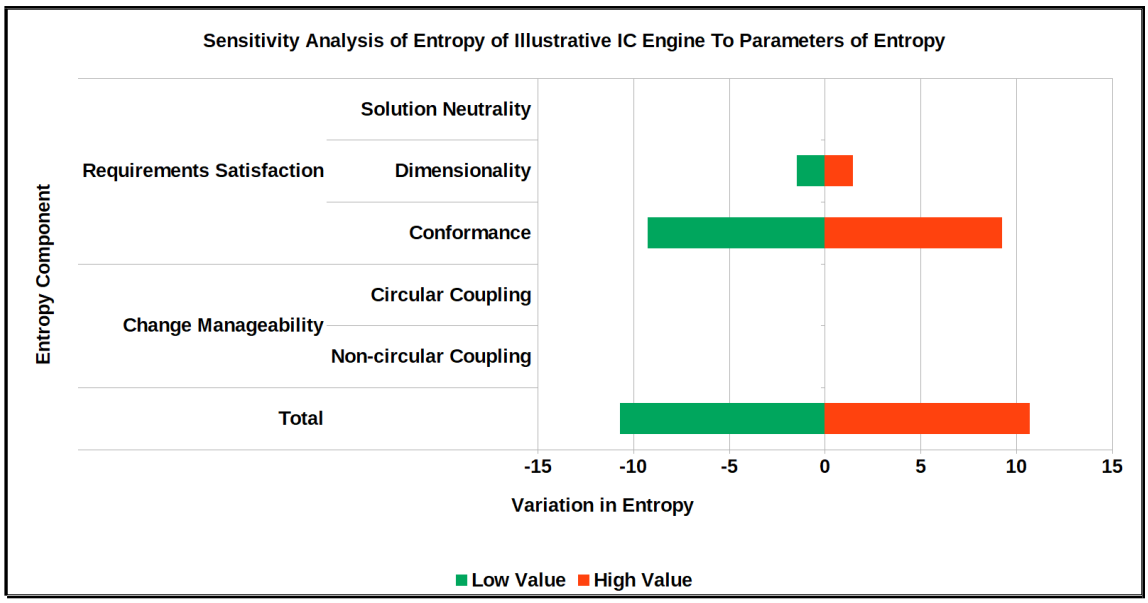

Figure 4 (c):Sensitivity analysis of the entropy of the design of illustrative IC engine to +/- 10% change in the relative weights specified for various components of entropy.

Structure				Behavior						Functional Requirements *(Satisfaction Of)*			Constraints (including Performance Requirements) *(Satisfaction Of)*		
				Input Conditions		Causal Process		Output SDs of Causal Processes							
Sentence	Truth Value (0-False; 1-True)	Sentence	Truth Value (0-False; 1-True)	Sentence	Truth Value (0-False; 1-True)	Identifier	Executed (0-False; 1- True)	Sentence	Truth Value (0-False; 1-True)	Sentence	Truth Value (0-False; 1-True)	Implied By SD	Sentence	Truth Value (0-False; 1-True)	Implied By SD
S1	1	S14 (≡ S8 & S9 & S10 & S11 & S12 & S13)	1	I10	1	P10	1	SD10	1	FR1	1	FR13	PR11	1	SD40
S11	1			I11	1	P11	1	SD11	1	FR11	1	SD11	PR12	1	SD20 & SD43
S12	1			I12	1	P12	1	SD12	1	FR12	1	SD12	PR1	1	PR11& PR12
S2	1			I13	1	P13	1	SD13	1	FR13	1	SD13	C1	1	S13
S3	1			I14	1	P20	1	SD20	1	FR2	1	FR23			
S31	1			I40	1	P21	1	SD21	1	FR21	1	SD21			
S32	1			I41	1	P22	1	SD22	1	FR22	1	SD22			
S4	1			I42	1	P23	1	SD23	1	FR23	1	SD23			
S5	1			I43	1	P30	1	SD30	1	FR3	1	FR32			
S6	1					P31	1	SD31	1	FR31	1	SD30			
S7	1					P40	1	SD40	1	FR32	1	SD31			
S8	1					P41	1	SD41	1	FR4	1	FR41			
S9	1					P42	1	SD42	1	FR41	1	SD40			
S10	1					P43	1	SD43	1	FR42	1	SD41			
S11	1					P50	1	SD50	1	FR43	1	SD42			
S12	1					P51	1	SD51	1	FR5	1	SD52			
S13	1					P52	1	SD52	1	FR51	1	SD51			
S15	1					P53	1	SD53	1	FR52	1	SD52			
S16	1					P54	1	SD54	1	FR53	1	SD53			

Supplementary Figure 1 (a): Calculating the truth values of satisfaction of Requirements and Constraints given the truth values of pre-conditions (including structure and input conditions of behavior): All pre-conditions satisfied.

Structure				Behavior						Functional Requirements *(Satisfaction Of)*			Constraints (including Performance Requirements) *(Satisfaction Of)*		
				Input Conditions		Causal Process		Output SDs of Causal Processes							
Sentence	Truth Value (0-False; 1-True)	Sentence	Truth Value (0-False; 1-True)	Sentence	Truth Value (0-False; 1-True)	Identifier	Executed (0-False; 1- True)	Sentence	Truth Value (0-False; 1-True)	Sentence	Truth Value (0-False; 1-True)	Implied By SD	Sentence	Truth Value (0-False; 1-True)	Implied By SD
S1	1	S14 (≡ S8 & S9 & S10 & S11 & S12 & S13)	1	I10	1	P10	1	SD10	1	FR1	0	FR13	PR11	0	SD40
S11	1			I11	1	P11	1	SD11	1	FR11	1	SD11	PR12	0	SD20 & SD43
S12	1			I12	1	P12	1	SD12	1	FR12	1	SD12	PR1	0	PR11 and PR12
S2	1			I13	1	P13	1	SD13	0	FR13	0	SD13	C1	1	S13
S3	1			I14	0	P20	1	SD20	1	FR2	0	FR23			
S31	1			I40	1	P21	1	SD21	1	FR21	1	SD21			
S32	1			I41	1	P22	1	SD22	1	FR22	1	SD22			
S4	1			I42	1	P23	1	SD23	0	FR23	0	SD23			
S5	1			I43	1	P30	1	SD30	1	FR3	0	FR32			
S6	1					P31	1	SD31	0	FR31	1	SD30			
S7	1					P40	1	SD40	0	FR32	0	SD31			
S8	1					P41	1	SD41	0	FR4	0	FR41			
S9	1					P42	1	SD42	0	FR41	0	SD40			
S10	1					P43	1	SD43	0	FR42	0	SD41			
S11	1					P50	1	SD50	0	FR43	0	SD42			
S12	1					P51	1	SD51	0	FR5	0	SD52			
S13	1					P52	1	SD52	0	FR51	0	SD51			
S15	1					P53	1	SD53	0	FR52	0	SD52			
S16	1					P54	1	SD54	0	FR53	0	SD53			

Supplementary Figure 1 (b): Calculating the truth values of satisfaction of Requirements and Constraints given the truth values of pre-conditions (including structure and input conditions of behavior): (a) Not all pre-conditions satisfied (Air-Fuel mixture not available at the inlet)

Structure				Behavior						Functional Requirements *(Satisfaction Of)*		Constraints (including Performance Requirements) *(Satisfaction Of)*		Overall Probability of Success (Satisfaction of all FRs and Constraints) of Design
				Input Conditions		Causal Process		Output SDs of Causal Processes						
Sentence	Probability	Sentence	Probability	Sentence	Probability	Identifier	Probability	Sentence	Probability	Sentence	Probability	Sentence	Probability	
S1	0.9973	S14 (≡ S8 & S9 & S10 & S11 & S12 & S13)	0.9900	I10	0.9973	P10	0.9973	SD10	0.9945	FR1	≥ 0.969	PR11	≥ 0.9395	≥ 0.919
S11	0.9973			I11	0.9973	P11	0.9973	SD11	0.9880	FR11	≥ 0.988	PR12	≥ 0.9315	
S12	0.9973			I12	0.9973	P12	0.9973	SD12	0.9805	FR12	≥ 0.9805	PR1	≥ 0.9315	
S2	0.9973			I13	0.9973	P13	0.9973	SD13	0.9690	FR13	≥ 0.969	C1	≥ 0.969	
S3	0.9973			I14	0.9973	P20	0.9973	SD20	0.9920	FR2	≥ 0.9595			
S31	0.9973			I40	0.9973	P21	0.9973	SD21	0.9845	FR21	≥ 0.9845			
S32	0.9973			I41	0.9973	P22	0.9973	SD22	0.9775	FR22	≥ 0.9775			
S4	0.9973			I42	0.9973	P23	0.9973	SD23	0.9595	FR23	≥ 0.9595			
S5	0.9973			I43	0.9973	P30	0.9973	SD30	0.9855	FR3	≥ 0.9525			
S6	0.9973					P31	0.9973	SD31	0.9525	FR31	≥ 0.9855			
S7	0.9973					P40	0.9973	SD40	0.9395	FR32	≥ 0.9525			
S8	0.9973					P41	0.9973	SD41	0.9355	FR4	≥ 0.9395			
S9	0.9973					P42	0.9973	SD42	0.9330	FR41	≥ 0.9395			
S10	0.9973					P43	0.9973	SD43	0.9315	FR42	≥ 0.9355			
S11	0.9973					P50	0.9973	SD50	0.9295	FR43	≥ 0.933			
S12	0.9973					P51	0.9973	SD51	0.9300	FR5	≥ 0.9265			
S13	0.9973					P52	0.9973	SD52	0.9265	FR51	≥ 0.93			
S15	0.9973					P53	0.9973	SD53	0.9230	FR52	≥ 0.9265			
S16	0.9973					P54	0.9973	SD54	0.9260	FR53	≥ 0.923			

Supplementary Figure 2: Computation of the probability of satisfaction of requirements and constraints of the illustrative schematic IC engine.

Component	Attribute	units	Specifications Of Attribute				Variability Of Attribute			Number of Discrete Values of Attributes Contained In –			Net Points Outside Specified Range	Entropy of Conformance
			Target Value	Lower Specification Limit (LSL)	Upper Specification Limit (LSL)	Specified Precision of Measurement of Attribute	Central Tendency Of Attribute (E.g. Mean)	Standard Deviation Of Attribute	Magnitude of Bias	Specified Range of Attribute	Six Standard Deviations from Central Tendency	Bias		
IC Engine	Power	hp	75	71.25	78.75	0.00075	74.25	1.071	0.750	10000	8571	1000	285.71	5.65
	Rotational Speed	rpm	2000	1900	2100	0.02	2020	28.571	20.000	10000	8571	1000	285.71	5.65
Cylinder	Inner Diameter of Bore	mm	90	89.91	90.09	1.8E-05	90.045	0.026	0.045	10000	8571	2500	1785.71	7.49
	Stroke Length	mm	50	49.95	50.05	1E-05	49.975	0.014	0.025	10000	8571	2500	1785.71	7.49
Inlet Valve	Diameter	mm	30	29.97	30.03	6E-06	30.015	0.009	0.015	10000	8571	2500	1785.71	7.49
Exhaust Valve	Diameter	mm	20	19.98	20.02	4E-06	20.01	0.006	0.010	10000	8571	2500	1785.71	7.49
Spark Plug	Voltage	volt	20000	19980	20020	0.004	20010	5.714	10.000	10000	8571	2500	1785.71	7.49
Piston	Diameter	mm	89.9	89.81	89.99	1.798E-05	89.86	0.026	0.045	10000	8571	2500	1785.71	7.49
Connecting Rod	Length	mm	100	99.9	100.1	2E-05	99.95	0.029	0.050	10000	8571	2500	1785.71	7.49
Crank	Radius	mm	50	49.95	50.05	1E-05	50.025	0.014	0.025	10000	8571	2500	1785.71	7.49
Timing components-Inlet valve	(Dimensionality assumed = 10; Number of values outside of specified range is assumed as shown in the corresponding column)												5.9049E+14	34.01
Timing components-Exhaust valve	(Dimensionality assumed = 10; Number of values outside of specified range is assumed as shown in the corresponding column)												5.9049E+14	34.01
Entropy of conformance for the overall design													2.942963E+60	139.23
Dimensionality of the design (K)														30.00
Entropy of Dimensionality of the design (= ln(1000^K))														207.23

Supplementary Figure 3 (a): Calculation of the entropies of dimensionality and conformance of the illustrative IC engine.

Level of FR	Number of Circular Couplings (c)	Number of Non-circular Couplings (u)	Weght for Level of FR (w)	Degree of Circular Coupling (= w^c)	Degree of Non-circular Coupling (= w*u if u >0; else 1)
1	0	15	10000	1	150000
2	0	8	1000	1	8000
3	0	0	100	1	1
4	0	0	10	1	1
5	0	0	1	1	1
Overall degree of coupling (product of degrees of coupling at each level)				1	1.20E+09
Entropy of Change Manageability for each type of coupling (=ln(overall degree of coupling of given type))				0	20.91
Entropy of Change Manageability (including circular and non-circular couplings)					20.91

Supplementary Figure 3 (b): Calculation of the entropy of change manageability of the illustrative IC engine.

	D_1	D_2	D_3	...	D_i	...	D_n
D_1	$K_{1,1}$	$K_{1,2}$	$K_{1,3}$	...	$K_{1,i}$	...	$K_{1,n}$
D_2	$K_{2,1}$	$K_{2,2}$	$K_{2,3}$	...	$K_{2,i}$	...	$K_{2,n}$
D_3	$K_{3,1}$	$K_{3,2}$	$K_{3,3}$	...	$K_{3,i}$	...	$K_{3,n}$
...	...	...	...	...	...	...	...
D_i	$K_{i,1}$	$K_{i,2}$	$K_{i,3}$	...	$K_{i,i}$	...	$K_{i,n}$
...	...	...	...	...	...	...	...
D_n	$K_{n,1}$	$K_{n,2}$	$K_{n,3}$	...	$K_{n,i}$	...	$K_{n,n}$

Supplementary Figure 4: Coupling (Dependency) matrix of a design.

V. Tables

Table 1: Requirements and constraints of the illustrative IC engine.

Alias	Description	Representation in *L*
Hierarchy of Functional Requirements (Functional Decomposition)		
FR_1	Air-fuel mix is sucked into the cylinder	$\exists t_{11}\exists t_{12}\ \forall t((t \geq t_{11}\ \&\ t \leq t_{12}\ \&\ t_{11} \leq t_{12}) =>$ InCylinder (Air-Fuel Mix))
FR_{11}	Inlet valve is opened	$\exists t_{111}\exists t_{112}\ \forall t((t \geq t_{111}\ \&\ t \leq t_{112}\ \&\ t_{111} \leq t_{112}) =>$ Open(Inlet Valve))
FR_{12}	Piston is moved to its lowest point in the cylinder(bottom dead center(BDC)).	$\exists t_{121}\forall t((t = t_{121}) =>$ Position(Piston) = BDC))
FR_{13}	Air-fuel mix is sucked into the cylinder	$\exists t_{131}\forall t((t = t_{131}) =>$ InCylinder (Air-Fuel Mix))
FR_2	Air-fuel mix is compressed	$\exists t_{21}\exists t_{22}\ \forall t((t \geq t_{21}\ \&\ t \leq t_{22}\ \&\ t_{21} \leq t_{22}) =>$ Compressed(Air-Fuel Mix))
FR_{21}	Inlet valve is closed	$\exists t_{211}\exists t_{212}\ \forall t((t \geq t_{211}\ \&\ t \leq t_{212}\ \&\ t_{2111} \leq t_{212}) =>$ Closed(Inlet Valve))
FR_{22}	Piston is moved to TDC to its highest point in the cylinder(top dead center(top dead center(TDC))	$\exists t_{221}\forall t((t = t_{221}) =>$ Position(Piston) = TDC))
FR_{23}	Air-fuel mix is compressed	$\exists t_{231}\forall t((t = t_{231}) =>$ Compressed(Air-Fuel Mix))
FR_3	Air-fuel mixture is ignited	$\exists t_{311}\exists t_{312}\ \forall t((t \geq t_{311}\ \&\ t \leq t_{312}\ \&\ t_{311} \leq t_{312}) =>$ Ignited(Air-Fuel Mix))
FR_{31}	Spark is generated	$\exists t_{311}\forall t((t = t_{311}) =>$ Generated (Spark))
FR_{32}	Air-fuel mixture is ignited	$\exists t_{321}\forall t((t = t_{321}) =>$ Ignited(Air-Fuel Mix))
FR_4	Power is generated	$\exists t_{41}\exists t_{42}\exists v\ \forall t((t \geq t_{41}\ \&\ t \leq t_{42}\ \&\ t_{41} \leq t_{42})$ =>(Power(IC Engine) = v & $v > 0$))
FR_{41}	Power is generated	$\exists t_{411}\exists v\forall t((t = t_{411}) =>$(Power(IC Engine) = v & $v > 0$))
FR_{42}	Combustion gases are expanded	$\exists t_{421}\exists t_{422}\ \forall t((t \geq t_{421}\ \&\ t \leq t_{422}\ \&\ t_{421} \leq t_{422}) =>$ Expanded(Combustion Gases))
FR_{43}	Piston is moved to BDC	$\exists t_{431}\forall t((t = t_{431}) =>$ Position(Piston) = BDC))
FR_5	Combustion gases are exhausted from the cylinder	$\exists t_{51}\exists t_{52}\ \forall t((t \geq t_{51}\ \&\ t \leq t_{52}\ \&\ t_{51} \leq t_{52}) =>$ Exhausted(Combustion Gases))
FR_{51}	Exhaust valve is opened	$\exists t_{511}\exists t_{512}\ \forall t((t \geq t_{511}\ \&\ t \leq t_{512}\ \&\ t_{511} \leq t_{512}) =>$ Open(Exhaust Valve))
FR_{52}	Piston moves to TDC	$\exists t_{521}\forall t((t = t_{521}) =>$ Position(Piston) = TDC))
FR_{53}	Combustion gases are exhausted from the cylinder	$\exists t_{531}\forall t((t = t_{531}) =>$ Exhausted(Combustion Gases))
Causal relationships among FRs		

P'_1	Suction stroke	$(FR_{11}$ & $FR_{12})$ ^{c}=> FR_{13}
P'_2	Compression stroke	$(FR_{13}$ & FR_{21} & $FR_{22})$ ^{c}=> FR_{23}
P'_3	Ignition	$(FR_{23}$ & $FR_{31})$ ^{c}=> FR_{32}
P'_4	Power stroke	FR_{32} ^{c}=> FR_{41} ^{c}=> FR_{42} ^{c}=> FR_{43}
P'_5	Exhaust stroke	$(FR_{43}$ & $FR_{51})$ ^{c}=> FR_{52} ^{c}=> FR_{53}
Performance Requirements		
PR_1	The IC engine generates power of 75 hp (+/- 4 hp) at 3000 rpm (+/- 150 rpm)	$\exists t_{31}\exists t_{32}\forall t((t \geq t_{31}$ & $t \leq t_{32}$ & $t_{31} \leq t_{32})$ =>(Power(IC Engine) ≥ 71 hp & Power(IC Engine) ≤ 75 hp & RotationalSpeed(Crank Shaft) ≥ 2850 rpm & RotationalSpeed(Crank Shaft) ≤ 3150 rpm))
PR_{11}	The IC engine generates power of 75 hp (+/- 4 hp)	$\exists t_{31}\exists t_{32}\forall t((t \geq t_{31}$ & $t \leq t_{32}$ & $t_{31} \leq t_{32})$ =>(Power(IC Engine) ≥ 71 hp & Power(IC Engine) ≤ 75 hp))
PR_{12}	The crank shaft rotates at 3000 rpm (+/- 150 rpm)	$\exists t_{31}\exists t_{32}\forall t((t \geq t_{31}$ & $t \leq t_{32}$ & $t_{31} \leq t_{32})$ =>(RotationalSpeed(Crank Shaft) ≥ 2850 & RotationalSpeed(Crank Shaft) ≤ 3150 rpm))
Constraints		
C_1	Displacement volume of Piston is not more than 2 liters	(DisplacementVolume(Piston) **≤** 2 liters

Footnotes:

FR-Functional Requirement**;** PR - Performance Requirement

Existentially quantified instants of time in the definition of FRs above should satisfy constraints imposed by the definition of causal processes (see Appendix 1.1) and the causal calculus (see Appendix 2) (e.g. $t_{111} \leq t_{11}$ & $t_{121} \leq t_{11}$; $t_{11} \leq t_{21}$ & $t_{211} \leq t_{21}$ & $t_{221} \leq t_{21}$).

Table 2: Structure of the illustrative IC engine.

Alias	Description	Representation in *L*
S_1	The cylinder head assembly is a component of the IC engine	Parent(Cylinder Head Assembly) = IC Engine
S_{11}	The cylinder head is a component of the Cylinder Head Assembly	Parent(Cylinder Head) = Cylinder Head Assembly
S_{12}	The spark plug is a component of the Cylinder Head Assembly	Parent(Spark Plug) = Cylinder Head Assembly
S_2	The cylinder block is a component of the IC engine	Parent(Cylinder Block) = IC Engine
S_3	Piston and Connecting Rod Assembly is a component of the IC engine	Parent(Piston and Connecting Rod Assembly) = IC Engine
S_{31}	Piston is a component of the Piston and Connecting Rod Assembly	Parent(Piston) = Piston and Connecting Rod Assembly
S_{32}	Connecting Rod is a component of the Piston and Connecting Rod Assembly	Parent(Connecting Rod) = Piston and Connecting Rod Assembly
S_4	Crank Shaft is a component of the IC engine	Parent(Crank Shaft) = IC Engine
S_5	Inlet Valve is a component of the IC engine	Parent(Inlet Valve) = IC Engine
S_6	Exhaust Valve is a component of the IC engine	Parent(Exhaust Valve) = IC Engine
S_7	Relative motion at joint between crank shaft and crank shaft bearing is of type rotation	RelativeMotionType(Crank Shaft, Crank Shaft Bearing) = Rotation
S_8	Relative motion at joint between crank shaft and connecting rod is of type rotation	RelativeMotionType(Crank Shaft, Connecting Rod) = Rotation
S_9	Relative motion at joint between connecting rod and piston is of type rotation	RelativeMotionType(Connecting Rod, Piston) = Rotation
S_{10}	Relative motion between piston and cylinder block is of type sliding	RelativeMotionType(Piston, Cylinder Block) = Sliding
S_{11}	Relative motion between inlet valve and cylinder head is of type sliding	RelativeMotionType(Inlet Valve, Cylinder Head) = Sliding
S_{12}	Relative motion between exhaust valve and cylinder head is of type sliding	RelativeMotionType(Exhaust Valve, Cylinder Head) = Sliding
S_{13}	Displacement volume of piston is 1.31 liters	DisplacementVolume(Piston) = 1.31 liters
S_{14}	Four bar mechanism described by the degrees of freedom and types of joints among the parts of the IC engine as described in sentences S_8 to S_{13}.	$S_{14} \equiv S_8 \,\&S_9 \,\&\ldots S_{19}$

S_{15}	Structure (mechanism involving parts such as cam shaft, cam, etc.) converting the motion of the crank shaft into the motion of the inlet and exhaust valves	Not represented here in terms of its component parts for the sake of simplicity, but used in Table 6 as an input to the causal process transforming the movement of the crank shaft to the movements of the inlet and exhaust valves.
S_{16}	Structure (mechanism involving parts such as crankshaft position sensor, high tension coil, spark plug, etc.) generating a spark as the piston reaches TDC in the power stroke.	Not represented here in terms of its component parts for the sake of simplicity, but used in Table 6 as an input to the causal process transforming the movement of the crank shaft to the movements of the inlet and exhaust valves.

Table 3: Behavior of the illustrative IC engine.

Alias	Description	Representation in *L*
Input conditions (sentences)		
I_{10}	Crank shaft is at 180 degrees (pointing towards the piston)	∀t((t = 0 ms) => Orientation(Crank Shaft) = 180°)
I_{11}	Inlet valve is closed at the start of the suction process	∀t((t = 0 ms) => (Closed(Inlet Valve))
I_{12}	Exhaust valve is closed until the start of the exhaust process	∀t((t ≥ 0 ms & t ≤ 30 ms) => Closed(Exhaust Valve))
I_{13}	Piston is at its highest point in the cylinder(Top Dead Center (TDC)	∀t((t = 0 ms) => Position(Piston) = TDC)
I_{14}	Air-fuel mixture is available at the inlet valve at the start of the suction process	∀t((t >= 0 ms) => AvailableAtInlet(Air-Fuel Mix))
I_{40}	Fuel to Air ratio = 0.067	∀t((t >= 0 ms) => FuelToAirRatio(Air-Fuel Mix) = 0.067)
I_{41}	Fuel conversion efficiency of engine = 0.31	∀t((t >= 0 ms) => FuelConvEff(IC Engine) = 0.31)
I_{42}	Density of air at inlet to engine = 1.125 kg/m^3	∀t((t >= 0 ms) => AirDensity(Inlet) =1.125 kg/m^3)
I_{43}	Volumetric Efficiency(VE) = 0.85	∀t((t >= 0 ms) => VolEff(IC Engine) = 0..85)
Causal Processes		
P_{10}	Move crank shaft (anti-clock wise) from 0 degrees to 180 degrees	I_0 ^{c}=> SD_{10}
P_{11}	Open inlet valve at the start of the suction process	S_{15} & I_{11} & SD_{10} ^{c}=> SD_{11}
P_{12}	Move piston from TDC to its lowest point in the cylinder (bottom dead center (BDC)	S_{14} & I_{13} & SD_{10} ^{c}=> SD_{12}
P_{13}	Air-fuel mix is sucked into the cylinder	I_{14} & SD_{11} & SD_{12} ^{c}=> SD_{13}
P_{20}	Move crank shaft from 180 degrees to 360 degrees	SD_{10} ^{c}=> SD_{20}
P_{21}	Close inlet valve at the start of the compression process	S_{15} & SD_{20} ^{c}=> SD_{21}
P_{22}	Move piston from BDC to TDC during the compression process	S_{14} & SD_{20} ^{c}=> SD_{22}
P_{23}	Compress air-fuel mixture to the required pressure	SD_{13} & SD_{22} ^{c}=> SD_{23}

P_{30}	Spark plug generates a spark	S_{16} & SD_{20} ^{c}=> SD_{30}
P_{31}	P_{31}: The air-fuel mixture is ignited	SD_{30} & SD_{23} ^{c}=> SD_{31}
P_{40}	Power is generated	SD_{31} & I_{40} & I_{41} & I_{42} & I_{43} & S_{13} ^{c}=> SD_{40}
P_{41}	Combustion gases are expanded	SD_{40} ^{c}=> SD_{41}
P_{42}	Piston is moved to BDC	SD_{41} ^{c}=> SD_{42}
P_{43}	The crank shaft is rotated to 270 degrees	S_{14} & SD_{42} ^{c}=> SD_{43}
P_{50}	The crank shaft moves from 270 degrees to 720 degrees under its own momentum	SD_{43} ^{c}=> SD_{50}
P_{51}	The exhaust valve remains open during the exhaust process	S_{43} & S_{15} ^{c}=> SD_{51}
P_{52}	The piston moves from BDC to TDC	S_{14} & SD_{50} ^{c}=> SD_{52}
P_{53}	Combustion gases are exhausted from cylinder	SD_{51} & SD_{52} ^{c}=> SD_{53}
P_{54}	The exhaust valve closes at the end of the exhaust process	S_{15} & SD_{50} ^{c}=> SD_{54}
Intermediate SDs		
SD_{10}	The crank shaft is oriented at 180° at the end of the suction stroke.	$\forall t((t = 10 \text{ ms}) \Rightarrow \text{Orientation}(\text{Crank Shaft}) = 180°)$
SD_{11}	The inlet valve is open during the suctions stroke.	$\forall t((t > 0 \text{ ms} \& t \leq 30 \text{ ms}) \Rightarrow \text{Open}(\text{Inlet Valve}))$
SD_{12}	The piston at BDC at the end of the suction stroke.	$\forall t((t = 10 \text{ ms}) \Rightarrow \text{Position}(\text{Piston}) = \text{BDC})$
SD_{13}	The air-fuel mixture is in the cylinder during the suction stroke.	$\forall t((t = 10 \text{ ms}) \Rightarrow \text{InCylinder}(\text{Air-Fuel Mix}))$
SD_{20}	The orientation of the crank shaft is at 360° at the end of the compression stroke.	$\forall t((t = 10 \text{ ms}) \Rightarrow \text{Orientation}(\text{Crank Shaft}) = 360°)$
SD_{21}	The inlet valve is closed immediately after the suction stroke and remains so for the rest of the strokes.	$\forall t((t > 10 \text{ ms} \& t \leq 40 \text{ ms}) \Rightarrow \text{Closed}(\text{Inlet Valve}))$
SD_{22}	The piston at TDC at the end of the compression stroke.	$\forall t((t = 20 \text{ ms}) \Rightarrow \text{Position}(\text{Piston}) = \text{TDC})$
SD_{23}	The air-fuel mix is compressed during the compression stroke.	$\forall t((t = 20 \text{ ms}) \Rightarrow \text{Compressed}(\text{Air-Fuel Mix}))$
SD_{30}	The spark is generated at the end of the compression stroke.	$\forall t((t = 20 \text{ ms}) \Rightarrow \text{Generated}(\text{Spark}))$
SD_{31}	The air-fuel mixture is ignited at the beginning of the power stroke.	$\forall t((t = 20.1 \text{ ms}) \Rightarrow \text{Ignited}(\text{Air-Fuel Mix}))$
SD_{40}	75 hp of power is generated in the power stroke.	$\forall t((t > 20 \text{ ms} \& t \leq 30 \text{ ms}) \Rightarrow (\text{Power}(\text{IC Engine}) = 75 \text{ hp})$
SD_{41}	Combustion gases are expanded in the power stroke.	$\forall t((t > 20 \text{ ms} \& t \leq 30 \text{ ms}) \Rightarrow (\text{Expanded}(\text{Combustion Gases}))$

SD_{42}	Piston is at BDC at the end of the power stroke.	∀t((t = 30 ms) =>Position (Piston) = BDC)
SD_{43}	Orientation of the crank shaft is 540° at the end of the power stroke.	∀t((t = 30 ms) =>Orientation (Crank Shaft) = 540°)
SD_{50}	Orientation of the crank shaft is 720° at the end of the exhaust stroke.	∀t((t = 40 ms) =>(Orientation (Crank Shaft))
SD_{51}	Exhaust valve is open during the exhaust stroke.	∀t((t > 30 ms & t ≤ 40 ms) => Open(Exhaust Valve))
SD_{52}	The piston is at TDC at the end of the exhaust stroke.	∀t((t = 40 ms) => Position(Piston) = TDC)
Output SDs		
SD_{53}	Combustion gases are exhausted during the exhaust stroke.	∀t((t = 40 ms) => Exhausted (Combustion Gases))
SD_{54}	The exhaust valve is closed immediately after the exhaust stroke.	∀t((t > 40 ms => Closed(Exhaust Valve))

Footnotes:

- *L* provides flexibility to specify attributes or parameters to be equal to some value (e.g. density of air at inlet to engine = 1.125 kg/m^3) or to be within a range of values (E.g. power of IC engine = 75 hp +/- 5 hp), as appropriate to the context (e.g. conceptual stage of design or later stages of design involving more detail, whether the attribute is a categorical (e.g. material type) or numerical attribute (e.g. power), etc.).
- Examples of circumscription that could be applied to the above definition of behavior are as follows:
    - S_{15} & I_{11} & SD_{10} <=> SD_{11} (corresponding to the causal process P_{11} defined above as: S_{15} & I_{11} & SD_{10} ^{c}=> SD_{11}) to limit the immediate cause of SD_{11} to only that explicitly stated in the theory, i.e. S_{15} & I_{11} & SD_{10}. (this is analogous to the literal completion approach of McCain and Turner (1997)).
    - (Position(Piston) = TDC) => ∀t((t = 0 ms V t = 20 ms V t = 40 ms) (this is an example of predicate circumscription (McCarthy (1980)).

Table 4: Formal proof (and expressions to calculate probability) of satisfaction of requirements and constraints by the structure and behavior of the schematic IC engine.

Type of Sentence (Premise/ Inference)	Sentence (Alias)	Implied By	Expression of Probability of Occurrence
Premise	S_1	Not Applicable (since the sentence is a premise)	$P(S_1)$
Premise	S_{11}	--”--	$P(S_{11})$
Premise	S_{12}	--”--	$P(S_{12})$
Premise	S_2	--”--	$P(S_2)$
Premise	S_3	--”--	$P(S_3)$
Premise	S_{31}	--”--	$P(S_{31})$
Premise	S_{32}	--”--	$P(S_{32})$
Premise	S_4	--”--	$P(S_4)$
Premise	S_5	--”--	$P(S_5)$
Premise	S_6	--”--	$P(S_6)$
Premise	S_7	--”--	$P(S_7)$
Premise	S_8	--”--	$P(S_8)$
Premise	S_9	--”--	$P(S_9)$
Premise	S_{10}	--”--	$P(S_{10})$
Premise	S_{11}	--”--	$P(S_{11})$
Premise	S_{12}	--”--	$P(S_{12})$
Premise	S_{13}	--”--	$P(S_{13})$
Inference	S_{14}	S_8, S_9, S_{10}, S_{11}, S_{12} and S_{13}	$P(S_{14}) = P(S_8 \ \& \ S_9 \ \& \ S_{10} \ \& \ S_{11} \ \& \ S_{12})$
Premise	S_{15}	Not Applicable (since the sentence is an premise)	$P(S_{15})$
Premise	S_{16}	--”--	$P(S_{16})$
Premise	I_{10}	--”--	$P(I_{10})$
Premise	I_{11}	--”--	$P(I_{11}) = P(I_{10})*P(I_{11}/ I_{10})$
Premise	I_{12}	--”--	$P(I_{12}) = P(I_{10})*P(I_{12}/ I_{10})$
Premise	I_{13}	--”--	$P(I_{13}) = P(I_{10})*P(I_{13}/ I_{10})$
Premise	I_{14}	--”--	$P(I_{14})$
Premise	I_{40}	--”--	$P(I_{40})$
Premise	I_{41}	--”--	$P(I_{41})$
Premise	I_{42}	--”--	$P(I_{42})$
Premise	I_{43}	--”--	$P(I_{43})$

Premise	P_{10}	--”--	$P(P_{10})$
Premise	P_{11}	--”--	$P(P_{11})$
Premise	P_{12}	--”--	$P(P_{12})$
Premise	P_{13}	--”--	$P(P_{13})$
Premise	P_{20}	--”--	$P(P_{20})$
Premise	P_{21}	--”--	$P(P_{21})$
Premise	P_{22}	--”--	$P(P_{22})$
Premise	P_{23}	--”--	$P(P_{23})$
Premise	P_{30}	--”--	$P(P_{30})$
Premise	P_{31}	--”--	$P(P_{31})$
Premise	P_{40}	--”--	$P(P_{40})$
Premise	P_{41}	--”--	$P(P_{41})$
Premise	P_{42}	--”--	$P(P_{42})$
Premise	P_{43}	--”--	$P(P_{43})$
Premise	P_{50}	--”--	$P(P_{50})$
Premise	P_{51}	--”--	$P(P_{51})$
Premise	P_{52}	--”--	$P(P_{52})$
Premise	P_{53}	--”--	$P(P_{53})$
Premise	P_{54}	--”--	$P(P_{54})$
Inference	SD_{10}	I_{10} and P_{10} (by causal implication)	$P(SD_{10}) = P(I_{10})*P(P_{10})$
Inference	SD_{11}	S_{15}, I_{11}, SD_{10} and P_{11} (by causal implication)	$P(SD_{11}) = P(S_{15} \& I_{11} \& SD_{10})*P(P_{11})$
Inference	SD_{12}	S_{14}, I_{13}, SD_{10} and P_{12} (by causal implication)	$P(SD_{12}) = P(S_{14} \& I_{13} \& SD_{10})*P(P_{11})$
Inference	SD_{13}	I_{14}, SD_{11}, SD_{12} and P_{13} (by causal implication)	$P(SD_{13}) = P(I_{14} \& SD_{11} \& SD_{12})*P(P_{13})$
Inference	SD_{20}	SD_{10} and P_{20} (by causal implication)	$P(SD_{20}) = P(SD_{10})*P(P_{20})$
Inference	SD_{21}	S_{15}, SD_{11}, SD_{20} and P_{21} (by causal implication)	$P(SD_{21}) = P(S_{15} \& SD_{11} \& SD_{20})*P(P_{21})$
Inference	SD_{22}	S_{14}, SD_{12}, SD_{20} and P_{22} (by causal implication)	$P(SD_{22}) = P(S_{14} \& SD_{12} \& SD_{20})*P(P_{22})$
Inference	SD_{23}	I_{12},SD_{21}, SD_{22}, SD_{13} and P_{23} (by causal implication)	$P(SD_{23}) = P(I_{12} \& SD_{13} \& SD_{21} \& SD_{22})*P(P_{23})$
Inference	SD_{30}	S_{16}, $SD_{20\backslash}$ and P_{30} (by causal implication)	$P(SD_{30}) = P(SD_{20})*P(P_{30})$
Inference	SD_{31}	S_{16}, $SD_{23\backslash}$ and P_{31} (by causal implication)	$P(SD_{31}) = P(S_{16} \& SD_{23})*P(P_{31})$
Inference	SD_{40}	I_{40}, I_{41}, I_{42}, I_{43}, SD_{31} and P_{40} (by causal implication)	$P(SD_{40}) = P(I_{40} \& I_{41} \& I_{42} \& I_{43} \& SD_{31})*P(P_{40})$
Inference	SD_{41}	SD_{40} and P_{41} (by causal implication)	$P(SD_{41}) = P(SD_{40})*P(P_{41})$
Inference	SD_{42}	I_{12}, SD_{22}, SD_{41} and P_{42} (by causal implication)	$P(SD_{42}) = P(I_{12} \& SD_{22} \& SD_{41})*P(P_{42})$
Inference	SD_{43}	S_{14}, SD_{42} and P_{43} (by causal implication)	$P(SD_{43}) = P(S_{14} \& SD_{42})*P(P_{43})$
Inference	SD_{50}	SD_{43} and P_{50} (by causal implication)	$P(SD_{50}) = P(SD_{43})*P(P_{50})$

Inference	SD_{51}	S_{15}, I_{12}, SD_{43} and P_{51} (by causal implication)	$P(SD_{51}) = P(S_{15} \& I_{12} \& SD_{43})*P(P_{51})$
Inference	SD_{52}	S_{14}, SD_{42}, SD_{50} and P_{52} (by causal implication)	$P(SD_{52}) = P(S_{14} \& SD_{42} \& SD_{50})*P(P_{52})$
Inference	SD_{53}	SD_{21}, SD_{51}, SD_{52} and P_{53} (by causal implication)	$P(SD_{53}) = P(SD_{21} \& SD_{51} \& SD_{52})*P(P_{53})$
Inference	SD_{54}	S_{15}, SD_{50}, SD_{51} and P_{54} (by causal implication)	$P(SD_{54}) = P(S_{15} \& SD_{50} \& SD_{51})*P(P_{54})$
Inference	C_1	S_{13} (by logical implication)	$P(C_1) = P(S_{13})$
Inference	PR_{11}	SD_{40} (by logical implication)	$P(PR_{11}) \geq P(SD_{40})$
Inference	PR_{12}	SD_{20} and SD_{43} (by logical implication)	$P(PR_{12}) \geq P(SD_{20} \& SD_{43})$
Inference	PR_1	PR_{11} and PR_{12} (by logical implication)	$P(PR_1) = P(PR_{11} \& PR_{12})$
Inference	FR_{11}	SD_{11}(by logical implication; SD_{11} can be obtained from FR_{11} by assigning values to existentially quantified variables in FR_{11})	$P(FR_{11}) \geq P(SD_{11})$
Inference	FR_{12}	SD_{12}(by logical implication)	$P(FR_{12}) \geq P(SD_{12})$
Inference	FR_{13}	SD_{13}(by logical implication)	$P(FR_{13}) \geq P(SD_{13})$
Inference	FR_1	FR_{13}(by logical implication)	$P(FR_1) \geq P(FR_{13}) \geq P(SD_{13})$
Inference	FR_{21}	SD_{21}(by logical implication)	$P(FR_{21}) \geq P(SD_{21})$
Inference	FR_{22}	SD_{22}(by logical implication)	$P(FR_{22}) \geq P(SD_{22})$
Inference	FR_{23}	SD_{23}(by logical impFlication)	$P(FR_{23}) \geq P(SD_{23})$
Inference	FR_2	FR_{23}(by logical implication)	$P(FR_2) \geq P(FR_{23}) \geq P(SD_{23})$
Inference	FR_{31}	SD_{30}(by logical implication)	$P(FR_{31}) \geq P(SD_{30})$
Inference	FR_{32}	SD_{31}(by logical implication)	$P(FR_{32}) \geq P(SD_{31})$
Inference	FR_3	FR_{32}(by logical implication)	$P(FR_3) \geq P(FR_{32}) \geq P(SD_{31})$
Inference	PR_{11}	SD_{40}(by logical implication)	$P(PR_{11}) \geq P(SD_{40})$
Inference	PR_{12}	SD_{22} & SD_{42}(by material implication)	$P(PR_{12}) \geq P(SD_{22} \& SD_{42})$
Inference	PR_1	PR_{11} & PR_{22}(by logical implication)	$P(PR_1) \geq P(PR_{11} \& PR_{12})$
Inference	FR_{41}	PR_{11}(by logical implication)	$P(FR_{41}) \geq P(PR_{11}) \geq P(SD_{40})$
Inference	FR_{42}	SD_{41}(by logical implication)	$P(FR_{42}) \geq P(SD_{41})$
Inference	FR_{43}	SD_{42}(by logical implication)	$P(FR_{43}) \geq P(SD_{42})$
Inference	FR_4	FR_{41}(by logical implication)	$P(FR_4) \geq P(FR_{41}) \geq P(PR_{11}) \geq P(SD_{40})$
Inference	FR_{51}	SD_{51}(by logical implication)	$P(FR_{51}) \geq P(SD_{51})$
Inference	FR_{52}	SD_{52}(by logical implication)	$P(FR_{52}) \geq P(SD_{52})$
Inference	FR_{53}	SD_{53}(by logical implication)	$P(FR_{53}) \geq P(SD_{53})$
Inference	FR_5	FR_{53}(by logical implication)	$P(FR_5) \geq P(FR_{53}) \geq P(SD_{53})$

Supplementary Table 1: Logical symbols of the language *L*

Symbol	Meaning
~	Negation
&	Conjunction
V	Disjunction
=>	Implication
^{c}=>	Causal implication
<=>	Bi-directional implication
∃	Existential quantifier
∀	Universal quantifier
=	Equal to
<	Lesser than
>	Greater than
≤	Lesser than or equal to
≥	Greater than or equal to
(	Delimiter
)	Delimiter
,	Delimiter
≡	Defined as

Supplementary Table 2: Non-logical symbols of *L*

Symbol	Description
a, b, c,,…	individuals (or names)
x, y, z,…	individual variables whose values are specific individuals
f(x, y, z…)	Function symbols, whose arguments are individuals, individual variables or other functions, and that range over individuals
R(a, b,…)	Predicates that take as arguments one or more individuals or individual variables.

Author Biography

Karthik Gurumurthi is a mechanical engineer with a background in product development methodologies and advanced symbolic logic. His work experience includes product development, manufacturing, quality, risk and benefit-risk management, and advanced decision support systems.