A Formal Approach to Connectibility Affordances

Andrew J. Abbate, *Member, IEEE*, and Ellen J. Bass, *Senior Member, IEEE*

***Abstract*—Connectibility affordances, or opportunities for a user to establish input/output cable connections, can be critical to the safety and usability of complex systems. To support model-based analyses, this research introduces a formal approach: Connectibility Affordance VErification, Modeling, and ENumeration (CAVEMEN). CAVEMEN is applicable to a human-environment system encompassing physical entities with specified properties, a user with specified motor abilities, and connectibility affordance instances involving different combinations of source-target connections. The modeling technique leverages object-oriented principles to define one instance of a connectibility affordance with respect to one unique combination of connection sources and targets. An inspection technique supports the enumeration of connectibility affordance instances that are desired (supporting a correct connection) and undesired (supporting an incorrect connection). A model checking technique aids in verifying accuracy, meaning the user can actualize desired affordance instances, and robustness, meaning undesired affordance instances never emerge. An XML-based grammar, a model checking syntax translation tool, and a linear temporal logic specification of accuracy and robustness support the analyses. We demonstrate CAVEMEN with a pacemaker system case study. The inspection aids in identifying nine desired and 18 undesired instances of chamber-port connectibility. The trace evaluation shows that accuracy and robustness depends on whether an entity property of interest can change in the environmental context. These results indicate that CAVEMEN shows promise for analyzing connectibility affordances of a safety-critical system.**

***Index Terms*—Affordance, formal methods, human performance modeling, interface evaluation, model-based design, usability.**

I. Introduction

A COMPLEX system can incorporate connectible hardware that a user (e.g., installer and operator) needs to physically manipulate, such as cables, output connectors, and input sockets. To inform the design of connectible hardware, human factors engineering (HFE) researchers and standards organizations have developed measures that should be tested early in the design process. For example, connectible hardware can be considered *accurate* if the user can manipulate it in ways that establish operational configurations, and *robust* if the system helps to prevent the user from establishing incorrect configurations [1], [2].

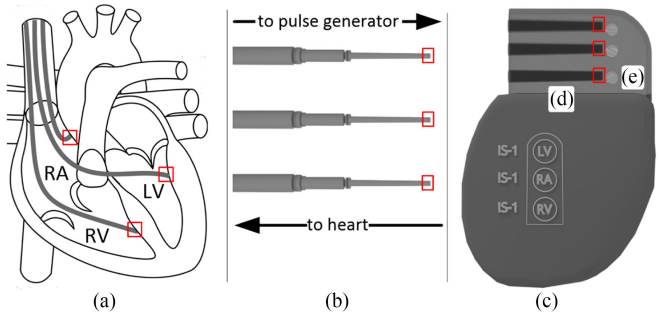

Fig. 1. Graphical rendering of a pacemaker system. Red boxes are added for reference in text. (a) Patient's heart showing segments of each lead. Lead distal tips are implanted in three chambers: LV stands for "left ventricle." RA stands for "right atrium." RV stands for "right ventricle." (b) Lead proximal tips. Arrows are added to identify connection targets of both lead tips (middle segments not shown). (c) Pulse generator. (d) Input ports. (e) Loosened set screw.

Consider the example of a surgically implanted pacemaker (see Fig. 1). Such a system provides heart failure patients with life-sustaining therapy via a programmed pulse generator. The pulse generator detects cardiac anomalies and delivers corrective electrical pulses to three heart chambers: the left ventricle (LV), which pumps oxygenated blood through the body; the right atrium (RA), which receives deoxygenated blood from the body; and the right ventricle (RV), which pumps deoxygenated blood to the lungs. The pulses are delivered via three identical leads having a distal tip that is implanted in a heart chamber and a proximal tip that is connected to a corresponding pulse-generator port [labeled on the pulse generator from top to bottom in Fig. 1(c)]. We characterize an operational configuration as follows: each lead distal tip is contained within the interior of exactly one heart chamber, the interior of a heart chamber is covering the front surface of exactly one lead distal tip, and the front surface of a proximal tip [see red boxes in Fig. 1(b)] and the back surface of a pulse-generator port [see red boxes in Fig. 1(d)] are covering each other. Each pulse-generator port has a set screw that must be loosened for a lead proximal tip to be fully inserted [see Fig. 1(e)].

Suppose the configuration in Fig. 1 emerges during an implantation surgery: each lead distal tip is implanted in a target heart chamber, each lead proximal tip is disconnected from a target pulse-generator port [see Fig. 1(b) and (d)], and each port's set screw is loosened [see Fig. 1(e)]. To establish a chamber-port connection, the surgeon needs to align a proximal tip with a target pulse-generator port and move the pulse generator toward the proximal tip with sufficient force to establish the connection. Three instances of such an action support an accurate operational configuration via three correct connections: LV chamber to LV

Manuscript received April 10, 2018; revised August 30, 2018; accepted November 17, 2018. Date of publication January 31, 2019; date of current version November 21, 2019. The work of A. J. Abbate was supported by the 2015–2016 U.S. Department of Education GAANN Interdisciplinary Collaboration and Research Enterprise for Healthcare fellowship (Grant P200A150023). (*Corresponding author: Andrew J. Abbate.*)

A. J. Abbate was with Drexel University, Philadelphia, PA 19104 USA. He is now with the Pacific Science & Engineering Group, San Diego, CA 92121 USA (e-mail: andrewabbate@pacific-science.com).

E. J. Bass is with the College of Nursing and Health Professions and the College of Computing and Informatics, Drexel University, Philadelphia, PA 19104 USA (e-mail: ellen.j.bass@drexel.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/THMS.2018.2886265

port, RA chamber to RA port, and RV chamber to RV port. However, unsafe configurations for the patient, such as one in which the LV chamber is connected to the RV port, are also possible (in part because there are three of the same lead, rather than a physically different lead for each chamber-port connection). Thus, while utilizing three of the same lead could have logistical benefits (e.g., ease of manufacturing), such a design also enables incorrect chamber-port connections. This reflects a robustness problem—one that emerges in existing pacemaker systems.[1]

A. Connectibility Affordances

The pacemaker system example of chamber-port connectibility could be characterized as an *affordance* [3]—an opportunity for a user to execute an action in a human-environment system (HES) [4].

In this research, an HES encompasses a three-dimensional (3-D) spatial area, a set of physical objects (referred to as *entities*), a user, and contextual factors that shape human-device interaction (e.g., automation). The user has motor abilities to physically manipulate the entities, whereas the entities have properties that physically constrain or support physical manipulations. An affordance emerges when specified properties of the entities and motor abilities of the user co-occur. If an affordance emerges, then the user can actualize it by executing one corresponding motor action.

Our interpretation of affordance is one of several that have proven useful in model-based analyses of human-interactive systems. Extant models commonly specify an affordance with respect to one human operator, one environment, and one particular set of physical entities therein. However, as demonstrated with the pacemaker system example, an affordance can involve many different combinations of physically equivalent entities. Such an affordance can be defined in an object-oriented way, where each unique combination of physically equivalent entities supports one unique instance of an object-oriented affordance. This research focuses on object-oriented connectibility affordances.

To characterize object-oriented connectibility affordances, consider a system in which there are many duplicates of the same entities and thus many ways of connecting them. Every unique connection configuration can reflect a different instance of the same connectibility affordance. The number of connection configurations—and thus the number of affordance instances—is defined formally in (1), where A is the number of instances, E is the number of duplicate entities that can establish the connection, $sources_1, \ldots, sources_n$ is the number of connection sources on the entity, and $targets_1, \ldots, targets_n$ is the number of connection targets for each source

$$A = E(sources_1 \times targets_1 \times \cdots \times sources_n \times targets_n). \quad (1)$$

Currently, it could be difficult for analysts to enumerate and evaluate object-oriented connectibility affordances with respect to many duplicate entities in combination. The problem space is further complicated by emergent behaviors in the environmental context, including user behaviors, such as physically manipulating multiple entities in parallel, and system behaviors, such as automated sensing and actuation. Thus, analysts could benefit from an improved approach. Such an approach should facilitate the enumeration of desired/undesired affordance instances with respect to duplicate entities in combination. It should also incorporate unambiguous methods and measures for verifying accuracy and robustness. The measures should be applicable to desired/undesired instances of the same connectibility affordance, whereas the methods should be applicable to emergent behaviors in the environmental context.

B. Research Contributions

To support improved analyses of object-oriented connectibility affordances, this research introduces a formal, model-based approach: Connectibility Affordance VErification, Modeling, and ENumeration (CAVEMEN). CAVEMEN extends our earlier research [5], which employs model checking—a highly automated technique for "proving" system properties [6]—to support the identification of potential affordance problems. Extensions enable the enumeration of desired/undesired connectibility affordance instances with respect to many duplicate entities in combination, verification of accuracy and robustness, and modeling of emergent behaviors in different environmental contexts.

Two tools and two analysis techniques support these extensions. Tools include CAVEMEN-XML, a custom encoding language for specifying an HES and emergent affordances therein, and an automated translator, which parses an instantiated CAVEMEN-XML representation and generates model checking syntax.[2] Analysis techniques include inspection, a technique for the analyst to enumerate and characterize desired/undesired instances of the same connectibility affordance, and trace evaluation, a technique for verifying a specification of accuracy and robustness via model checking. We introduce an implementation of the trace evaluation technique for analyzing emergent behaviors in two environmental contexts: one in which a condition for the affordance to emerge cannot change, and one in which the same condition can change. We demonstrate an application of CAVEMEN with a pacemaker system case study.

II. Background

A. Affordance Modeling

Researchers at the intersection of ecological psychology and computer science have developed a variety of model-based approaches to affordance. As mentioned, CAVEMEN differs from extant approaches because it defines affordances in an object-oriented way, where one instance of an affordance involves a unique combination of duplicate environment entities. In contrast, extant approaches commonly model one affordance at a time with respect one fixed set of entities. These approaches commonly enable a normative, task-analytic approach to affordances, where human perception and cognition partially control what affordances emerge and what actions are taken. CAVEMEN offers a different perspective—one that exclusively addresses physical artifacts, but inclusively addresses both desired

[1] See for example https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfmaude/detail.cfm?mdrfoi__id=6920003&pc=DTB

[2] The XML schema, translation tool, and case study models described in this article are available for download at https://github.com/andrew-j-abbate/CAVEMEN

and undesired affordances. We next review a subset of the affordance modeling literature that reflects these differences.

Wells [7] integrates the concept of affordance with Turing's theory of computation [8]. An affordance model captures one normative, temporal ordering of human actions within a Turing machine representation. Symbolic variables represent a human operator, actions that can be executed, and properties of the environment. The model enables simulation of a human operator progressing toward one goal state of the environment by interacting with a specified set of entities.

Turvey [9] models affordances in the context of a particular task or objective. Environment properties and human properties operate as inputs to a "juxtaposition function" that defines the intersection of affordances and the human operator's goal-driven intentions. The model assumes that an affordance only emerges if it supports an intended behavior; and if an affordance emerges, then it is always actualized.

According to Stoffregen [4], one potential issue with Turvey's model is that it does not account for a human operator choosing against actualizing an affordance. He addresses this issue by removing the juxtaposition function and replacing it with a psychological choice function. Together, the affordance model and psychological choice function specify that an affordance can emerge even if it does not support an intended behavior, and the human operator will choose to actualize the affordance only if it supports an intended behavior.

In [10], Rothrock *et al.* integrate the models described in [9] and [7] with infrastructure that abstracts human perception. Three types of sensory functions represent what audio, visual, and haptic properties of the environment the human operator can perceive. An affordance emerges if the human operator can perceive relevant environment properties, whereas the human operator actualizes the affordance if it supports progress toward a goal [11].

In [12], Lenarčič and Winter leverage situation theory [13] to define a hierarchical model of affordance. Here, an affordance is composed of two hierarchical elements: one situation and one human operator. The situation is defined by a set of environmental conditions, and the human operator is defined by a set of cognitively/physically feasible abilities [12]. The model captures HES dynamics as the human operator executes goal-oriented actions.

B. Model Checking

Model checking commonly involves three sequential steps for the analyst: encoding a *formal model* of the target system, which can abstract a broad range of temporally evolving behaviors in terms of valued-variable states and next-state transitions [14]; encoding a specification that unambiguously characterizes a behavior of interest with respect to the formal model, typically using the semantics of a temporal logic, such as linear temporal logic (LTL) [15]; and invoking a model checker that searches the formal model for specification violations. If the model checker finds a specification violation, then it returns one trace of sequentially ordered states and transitions through the formal model leading up to the violation. If the specification characterizes a desired behavior, then the trace is called a *counterexample*. If the specification characterizes an undesired behavior, then the trace is called a *witness*.

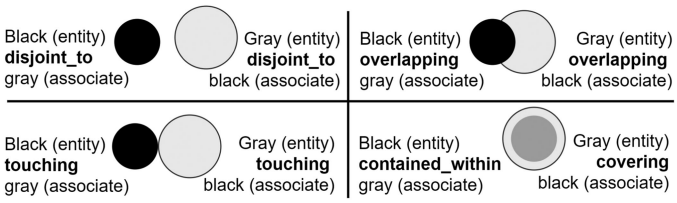

Fig. 2. Topology-keyword semantics in two dimensions. Topology keywords are in boldface text.

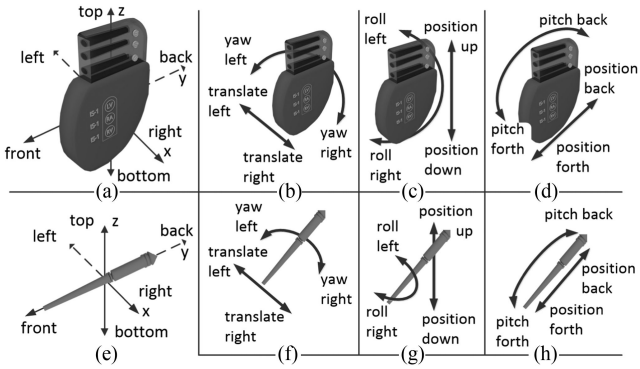

Fig. 3. (a) Surfaces of a pacemaker pulse generator and its ports. (b)–(d) Pulse generator movements. (e) Surfaces of lead proximal tips. (f)–(h) Lead proximal tip movements.

C. Analyzing Affordances via Model Checking

Our earlier approach [5] enables model checking analyses of affordances with respect to an HES encompassing a 3-D spatial environment, physical entities, and a human operator. We accomplish this using Stoffregen's affordance formalism [4], standard engineering terminology [16]–[18], and the Symbolic Analysis Laboratory (SAL) model checking system [19]. We describe these techniques ahead to aid in understanding CAVEMEN extensions.

1) Environment Model: The environment is modeled using one hierarchical variable *X*. Lower level variables represent the entities therein and their relevant physical properties. For reduced ambiguity, the analyst specifies each entity's part-whole composition, where parts are permanently attached fixtures, each of which has its own part-whole composition and properties.

Properties can be binary (true/false), numerical, or spatial relationship. Binary and numerical properties are modeled using variable names that aid in identifying what property is being represented. Spatial-relationship properties incorporate one topology keyword derived from [16], one direction keyword derived from [17], and one other entity (referred to as an "associate") with respect to which the spatial relationship is defined.

Five topology keywords (disjoint to, touching, overlapping, contained within, and covering) enable the analyst to specify 2-D connectedness of one entity with respect to one associate (see Fig. 2). Six direction keywords (top, bottom, left, right, back, and front) define unchanging directional surfaces of the associate [see for example the surfaces in Fig. 3(a) and (e)], which the analyst can either interpret as interior surfaces or exterior surfaces, depending on the application. Each topology-direction pair is mutually exclusive; for example, an entity can-

not be specified as simultaneously touching and overlapping the front surface of an associate. These semantics enable specification of 3-D connectedness between *any* surface(s) of an entity with respect to one *particular* surface of an associate. For reduced ambiguity, the analyst should encode an accompanying spatial-relationship property that specifies 3-D connectedness between the associate and one surface of the entity.

When instantiating a formal model, the analyst assigns each property one of the following behaviors: an unchanging initial-state value, which is used if the property never changes; a particular next-state value, which is used if the property only changes due to the user actualizing an affordance; or one of many possible next-state values, which is used if the property can change due to other events in the environmental context. An unchanging property can only operate as an affordance input, whereas a transitioning property can operate as an affordance input, output, or both (affordance inputs and outputs are explained in Section II-C3).

2) User Model: The user is modeled using one variable *Z*, which leverages the same hierarchical representation of the entities as the environment variable *X*. However, instead of representing physical properties of the entities, lower level variables of *Z* represent motor abilities to physically manipulate the entities. The model assumes that the user maintains one stationary position in the spatial environment and that all entities can be moved in parallel.

Each motor ability is uniquely defined with respect to each entity. Six degrees of freedom (6DoF) [18] keywords describe how the user can position, translate, and rotate an entity about its origin. The direction of each movement is specifiable along the *x*, *y*, and *z* axes with respect to the surfaces that direction keywords represent for spatial-relationship properties [see Fig. 3(b)–(d), where directions correspond to surfaces in Fig. 3(a)]. The analyst assigns each keyword a numeric value that represents the maximum force magnitude with which a movement can be executed. Specifying force magnitude is useful if actualizing an affordance requires the user to overcome some opposing force imposed by the environment, such as air pressure or friction. If these force magnitudes are unknown, then the analyst should assign each 6DoF keyword a value of 1 or 0, meaning the movement is possible or impossible for the user respectively.

3) Affordance Model: Affordances are modeled as input/output functions. Inputs are conditions that the entity properties and motor abilities must concurrently satisfy for the affordance to emerge. Outputs are conditions that entity properties must satisfy as an immediate consequence of the user actualizing the affordance. A model checking instantiation of an extant affordance formalism [4] supports these semantics

$$possesses(affordance)(Z, X). \tag{2}$$

Equation (2) is a nested Boolean function of affordances (*affordance*), the user (*Z*), and the environment (*X*). *affordance* is an enumerated list of one or more named affordance variables, where each variable name aids in identifying the affordance being modeled. The outer function takes one affordance variable as an input, whereas the inner function takes the user *Z* and the environment *X* as inputs. Using all three inputs, (2) returns *true* for all affordances whose input conditions are satisfied by lower level variables of *Z* and *X*. This is accomplished for each modeled affordance by encoding a conjunction of Boolean expressions, one for each entity-property input and one for each motor-ability input. The conjunctions for entity-property inputs should be encoded first, followed by the conjunctions for motor-ability inputs. This is because motor-ability inputs specify what movements are possibly needed in any HES configuration satisfying the entity-property inputs, such as one in which the entities are facing different directions and need to be aligned in order to establish a connection.

After instantiating all conjunctions, the analyst can utilize (2) to encode one next-state transition for each modeled affordance, where next-state values are affordance-output conditions. Multiple next-state transitions can be enabled in any state of the formal model (i.e., multiple affordances can emerge in parallel), but exactly one transition can execute (i.e., one affordance can be actualized). CAVEMEN inherits this feature of our prior approach because it improves physical validity. For example, a connection source could be simultaneously connectible to multiple connection targets, each of which occupies a different spatial location. Since one connection source cannot simultaneously occupy multiple locations, exactly one instance of the affordance can realistically be actualized at a time.

III. CAVEMEN Extensions

Here, we identify how CAVEMEN extends our earlier approach with respect to object-oriented connectibility affordances. Where applicable, we identify extant tools and techniques that inspire the extensions. A case study demonstration appears in Section V.

1) Characterization of Desired/Undesired Connectibility Affordance Instances: As mentioned, one knowledge gap is the difficulty for an analyst to characterize all instances of a connectibility affordance with respect to many duplicate entities in combination. The CAVEMEN-XML language and translator address this gap.

Using CAVEMEN-XML, the analyst can specify each entity once, including what quantity of the entity (one or more) is applicable to the affordance(s) of interest. Natural language keywords specify each entity's numerically quantifiable, binary, and spatial-relationship properties that are relevant, including if/how these properties can transition in a formal model. Our custom translator instantiates model checking syntax representing each duplicate entity, including all lower level entities, properties, and next-state transitions. This extension enables the translator to generate all instances of the same affordance with respect to duplicate entities. Two additional extensions facilitate characterization and enumeration of the affordance instances: specification of the inputs and outputs in a general way with respect to duplicate entities, and generation of model checking syntax for each instance with respect to correct combinations of duplicate entities.

To support the first extension, we leverage a technique that has proven useful in the enhanced operator function model (EOFM) task-analytic framework [20], [21]. In EOFM, a custom, XML-based grammar (EOFM-XML) incorporates keywords for specifying cardinal and temporal orderings of actions/activities that can be executed in a goal-driven task, where cardinal orderings include all activities/actions, at least one activity/action, or exactly one activity/action. An automated translation tool generates model checking syntax representing all of the ways to

execute the same goal-driven task(s) with respect to an instantiated EOFM-XML representation. For connectibility affordance applications, natural language keywords of CAVEMEN-XML enable the analyst to specify what combination of duplicate entities must satisfy the input or output condition: all, at least some number, or exactly some number. Each condition is otherwise specified as in our earlier approach: inputs are conditions that must be satisfied for the affordance to emerge and outputs are conditions that become satisfied as a consequence of actualizing the affordance.

To support the second extension mentioned above, the translator uses custom algorithms that generate model checking syntax for all instances of the same connectibility affordance. The duplicate entities and affordance input/output combinations specified in a CAVEMEN-XML representation have corresponding model checking syntax that uniquely identifies each instance of the same entity, property, motor ability, and affordance.

2) Enumeration of Desired/Undesired Connectibility Affordance Instances: One purpose of the extensions mentioned earlier is to support the enumeration of desired/undesired connectibility affordance instances. This is accomplished in CAVEMEN via an inspection technique: the analyst enumerates desired and undesired instances of the same connectibility affordance with respect to translated model checking syntax. A desired instance has inputs supporting a correct connection, whereas an undesired instance has inputs supporting an incorrect connection. An ideal result is no undesired instances, whereas a minimally acceptable result is at least one desired instance. The inspection result informs additional formal model syntax that is needed to support verification of accuracy and robustness.

3) Specification and Verification of Accuracy and Robustness: Currently, model checking of connectibility affordances is constrained to the model checking syntax that the analyst can manually encode and the specifications that the analyst can identify. To support the application of other model checking approaches, researchers have developed generalizable specifications and tools that facilitate the encoding processes [22]. In this vein, CAVEMEN incorporates support for connectibility affordance applications.

To inform generalizable specifications, we introduce an LTL specification of accuracy and robustness (3), where the first line is accuracy and the second line is robustness. Accuracy means that a specified set of desired instances will eventually be actualized, and robustness means that no undesired instances will ever emerge. "F" is a temporal operator meaning "eventually," "G" is a temporal operator meaning "always," and "∀" is a variable quantifier meaning "for all"

$$\left(\begin{array}{l} \mathsf{F}(\forall d : desired \,|\, actualized[d]) \wedge \\ \quad \mathsf{G}(\forall u : undesired \,|\, \neg(possesses(u)(Z, X)) \end{array}\right). \tag{3}$$

The variables *d* and *u* come from the enumerated lists *desired* and *undesired*, respectively, both of which the analyst instantiates based on the inspection result: *desired* includes desired affordance instances of interest, and *undesired* includes all undesired instances. The translator automatically instantiates *actualized*, which is an indexed array of Boolean-valued affordance instances. The translator assigns each index (representing one affordance instance) an initial value of *false* and an irreversible next state of *true* once the affordance is actualized. These semantics enable the formal model to keep track of which affordance instances have been actualized.

The translator instantiates two forms of the specification that support different kinds of trace evaluations: one positive form, which is encoded as shown in (3), and one negative form, which has "¬" (meaning "not") outside the large, left-hand parenthesis shown in (3). The positive form is verified to search for counterexamples. If a counterexample is returned, then it will show one trace through the formal model in which one or more undesired affordance instances emerge, a desired instance is not actualized, or both (an undesired result). No counterexamples means that there are no violations of accuracy and robustness (a desired result). The negative form is verified to search for witnesses. If a witness is returned, then it will show a trace through the model in which all specified desired instances are actualized without an undesired instance ever emerging (a desired result). No witnesses means that the system is not both accurate and robust (an undesired result). Specifications are verified using the SAL infinite bounded model checker (SAL-INF-BMC), as its algorithms are optimized for generating traces [19] and avoiding the problem of state-space explosion [23].

4) Modeling of Emergent Behaviors in the Environmental Context: Two CAVEMEN extensions support improved analyses of emergent behaviors in the environmental context: formal modeling of emergent user motor abilities to move subsets of the entities in parallel, and evaluating accuracy and robustness with respect to different environmental contexts.

As mentioned, our earlier approach assumes that the user can always move all entities in parallel. For object-oriented connectibility affordances, one limitation of such an assumption is that the combination of entities that can be moved in parallel is critical to what instance of the affordance emerges. Consider the pacemaker system example of chamber-port connectibility. If one lead distal tip is implanted within each chamber and all lead proximal tips are disconnected from all pulse-generator ports (see Fig. 1), then which instance of chamber-port connectibility emerges depends in part on which lead proximal tip the surgeon can move in parallel with the pulse generator (such as by gripping either entity with each hand).

Robotics researchers have developed ways of formally modeling these kinds of parallel movement abilities by specifying that a robot has two arms and two hands, both of which can be utilized at the same time to move entities [24]. While such a technique has proven useful in robotics applications, modeling humans in the same way could be inappropriate per international accessibility standards [25]. Thus, to support the first extension mentioned before, CAVEMEN incorporates an alternative technique. Using natural language keywords of CAVEMEN-XML, the analyst can specify which entities (and how many duplicates) the user can move in parallel. The translator generates model checking syntax enabling exactly one such set of movements in every state of the formal model.

To support the second extension mentioned above, we leverage a trace evaluation technique that has proven useful in task-analytic applications [26]: comparing the model checking results of two formal models with respect to a one-line change of intermediate-language syntax. To employ this technique in CAVEMEN, the analyst identifies an entity property of interest

that operates as an affordance input. In one CAVEMEN-XML representation, the analyst specifies that this property has an unchanging initial state. In a second CAVEMEN-XML representation, the analyst specifies that this same property can transition nondeterministically, which abstracts a different environmental context. The analyst then verifies both translated formal models with respect to the same instantiation of accuracy and robustness. This technique aids in identifying whether emergent behaviors can affect accuracy and robustness.

IV. CAVEMEN-XML

CAVEMEN-XML enables the analyst to specify a static representation of an HES and connectibility affordances. It employs eXtensible Markup Language (XML) [27], an international standard for representing structured data. An XML document is defined within a single *root* element. Lower level elements are called *children*. All children of the root element can have valued attributes, text content, and children.

In CAVEMEN, an XML Schema Document (XSD) [28] grammar enforces CAVEMEN-XML syntax that is needed for the translator to generate model checking syntax. The translator, which can be called from a desktop web browser, validates CAVEMEN-XML syntax against the grammar. We next describe the syntax, structure, and semantics of CAVEMEN-XML. Elements are in boldface text, attributes are in italic text, and attribute values are in quotes. Subsection titles are children of the root element, **hes**.

A. Environment

A CAVEMEN-XML document has one **environment** element, which represents a 3-D spatial area. One or more child **entity** elements represent the physical objects therein.

Each **entity** element has two attributes: *name* and *quantity*. The *name* attribute value should be an alphabetic string that aids in identifying what entity is being specified. To support formal model translation, heterarchical entities must have unique *name* attribute values. The *quantity* attribute value is a positive integer that specifies how many of the entities are applicable to the affordance(s) of interest. One or more **property** child elements specify the entity's numerical, binary, and spatial-relationship properties.

Each **property** element has three attributes: *evolution*, *type*, and *name*. The *evolution* attribute can have one of three values: "human" if a property only transitions due to the user actualizing an affordance, "environment" if the property can transition due to other events in the environmental context (or also due to the user actualizing affordances), or "none" if the property never transitions. The *type* attribute can have one of three values: "Boolean" if the property is binary, "numerical" if the property is numerically quantifiable, or "spatial" if the property is a spatial relationship. Text content, which is only applicable for spatial-relationship properties, must reference one or more dot-separated *name* attribute values of one or more **entity** elements, where each dot specifies one step down in the **entity** element hierarchy. This convention is needed to specify reciprocal spatial relationships between the entity, surface(s) of the associate, and vice versa.

TABLE I
Quantity-Operator VALUE SEMANTICS

"all"	"at_least-*x*"	"exactly-*x*"
All entity instances must satisfy the condition	*x* or more entity instances must satisfy the condition	*x* entity instances must satisfy the condition, while all others must not

B. Human

A CAVEMEN-XML document has one **human** element, which represents the user. Child elements are **ability** and **ability-set**.

Each **ability** element has two attributes: *name* and *entity*. The *name* attribute value is a unique alphabetic string that aids in identifying what ability is being modeled. The **entity** attribute identifies what entity can be moved. Its value references one or more dot-separated *name* attribute values of **entity** elements.

For each child element of **ability**, attributes identify what 6DoF movements the user can execute. Each value is a positive number that specifies the maximum force magnitude with which the user can execute the movement. If force magnitudes are unknown, then each value should be "1" to specify that the movement is possible. Omitting an attribute is equivalent to specifying that the movement is impossible for the user.

Each **ability-set** element specifies parallel movements that can be executed in a mutually exclusive way with respect to all other movements. The *name* attribute value is a unique alphabetic string that aids in identifying what set of movements is being modeled. One or more **ability-ref** child elements specify what movements can be executed. Each **ability-ref** element has two attributes: *name* and *quantity*. The *name* attribute references an **ability** element by its *name* attribute. The *quantity* attribute value is a positive integer that specifies how many of the referent entity (i.e., the *entity* attribute value of the referenced **ability** element) can be moved in parallel.

C. Affordance

One or more **affordance** elements represent input/output affordances. Each **affordance** element has one valued attribute, *name*, which is a unique alphabetic string that aids in identifying what affordance is being modeled. Child elements are **environment-input**, **human-input**, and **environment-output**. We next describe the attributes and text content of these child elements.

Each **environment-input** element specifies an entity-property condition that must be satisfied for the affordance to emerge. It has three attributes: *name*, *quantity-operator*, and *equality-operator*. The *name* attribute value references the *name* attribute value of a **property** element. Together, the text content and *equality-operator* attribute specifies the condition with respect to the referent property (see Table II). If the referent property's *type* attribute value is "numerical" or "Boolean," then text content can be a logical expression (i.e., one that is either true or false). If the referent property's *type* is "spatial," then text content can be one topology keyword and one optional direction keyword. Omitting a direction keyword is equivalent to specifying that all six direction keywords must be assigned the same topology-keyword value. For any referent property, the

TABLE II
Equality-Operator VALUE SEMANTICS

Value	Applicable element(s)	Condition
"equal-to"	**environment-input**, **environment-output**, **human-input** children	The property or ability must be equal to the specified text content
"not-equal-to"	**environment-input**, **environment-output**, **human-input** children	The property or ability must not be equal to the specified text content
"less-than-or-equal-to"	Numerical-type **environment-input/ environment-output**, **human-input** children	The property or ability must be less than or equal to the specified text content
"greater-than-or-equal-to"	Numerical-type **environment-input/ environment-output**, **human-input** children	The property or ability must be greater than or equal to the specified text content
"less-than"	Numerical-type **environment-input/ environment-output**, **human-input** children	The property or ability must be less than the specified text content
"greater-than"	Numerical-type **environment-input/ environment-output**, **human-input** children	The property or ability must be greater than the specified text content

quantity-operator attribute specifies how many duplicate entities, each of which has an instance of the same referent property, must satisfy the condition (see Table I).

Each **human-input** element specifies a motor-ability condition that must be satisfied for the affordance to emerge. It has two attributes: *name* and *quantity-operator*. The *name* attribute value references the *name* attribute value of an **ability** element. Optional child elements specify the movements that are possibly needed in order to actualize the affordance. Together, the text content and *equality-operator* attribute of each child element specifies one condition, as described in Table II. Omitting child elements is equivalent to specifying that all movements of the entity identified in the referent **ability** element are needed to actualize the affordance, which is either all maximum force magnitudes or "1" (meaning the movement must be possible). The *quantity-operator* attribute specifies how many duplicate entities, each of which has an instance of the same ability, must satisfy the condition (see Table I).

Each **environment-output** child element specifies what entity-property changes occur as a consequence of actualizing the affordance. Its attributes and text context are encoded in the same way as described for **environment-input** elements mentioned earlier.

V. CASE STUDY

Here, we apply the CAVEMEN approach in a pacemaker system case study of chamber-port connectibility. We encode a CAVEMEN-XML representation representing the pacemaker system described in Section I. We model one heart having three applicable chambers, one pulse generator having three applicable ports, one set screw for each port, three leads, one proximal tip and one distal tip for each lead, and one surgeon (i.e., the user) who can move the pulse generator and any one lead proximal tip in parallel. We then translate the CAVEMEN-XML representation to SAL in order to conduct inspection and trace evaluation analyses.

Based on (1), we expect the translator to instantiate 27 instances of the chamber-port connectibility as there are three leads, one distal connection source having three targets, and one proximal connection source having three targets ($27 = 3(1 \times 3 \times 1 \times 3)$). To support the inspection, we define a desired instance as one supporting a correct chamber-port connection: LV chamber to LV port, RA chamber to RA port, and RV chamber to RV port. The translator appends integers to CAVEMEN-XML syntax in order to uniquely define duplicate entities; thus, for the purpose of this case study, we refer to the chambers and pulse-generator ports as follows: "1" corresponds to "LV," "2" corresponds to "RA," and "3" corresponds to "RV."

To support the trace evaluation, we verify the negative form of (3) to search for witnesses—traces through the formal model showing that it is possible for the system to satisfy the accuracy and robustness specification.

As mentioned in Section I, the case study pacemaker system can be considered accurate with respect to chamber-port connectibility because the surgeon can actualize three desired instances—one for each lead. In the parlance of CAVEMEN, we identify exactly one set of initial environment-input conditions in which the system can be considered accurate: all three set screws are loosened, all three lead distal tips are implanted correctly, and all three lead proximal tips are disconnected from pulse-generator ports (see the configuration shown in Fig. 1).

We hypothesize that these environment-input conditions cannot support both accuracy and robustness with respect the chamber-port connectibility, unless different conditions emerge. One such set of conditions includes which pulse-generator port set screws are loosened, which lead proximal tip the surgeon can move, and in which heart chamber the same lead's distal tip is implanted. To support accuracy, the set screw of the correct target port must be loosened (enabling the correct connection). To support robustness, set screws of incorrect target ports must be tightened (disabling incorrect connections). If the states of all three set screws can change, then three desired instances of chamber-port connectibility can be actualized without an undesired instance ever emerging. If states of the set screws are unchanging, then it should be impossible for the system to be both accurate and robust.

To test this hypothesis, we encode two CAVEMEN-XML representations: one in which the sets screws have unchanging initial-state values (the first model) and one in which their values can transition nondeterministically (the second model). We then translate, verify, and compare model checking results of these two models with respect to the negative form of the accuracy and robustness specification. We expect the first model to return no witnesses (an undesired result) and the second model to return a witness (a desired result).

A. CAVEMEN-XML Representations

Both CAVEMEN-XML representations are 63 lines. Three top-level **entity** elements (direct children of **environment**) include "Heart" (see Fig. 4c), which represents the patient's heart; "PulseGenerator" (see Fig. 4j), which represents the pulse generator; and "Lead" (see Fig. 4m), which represents three leads. "Heart" has one child entity named "Chamber" (see Fig. 4b),

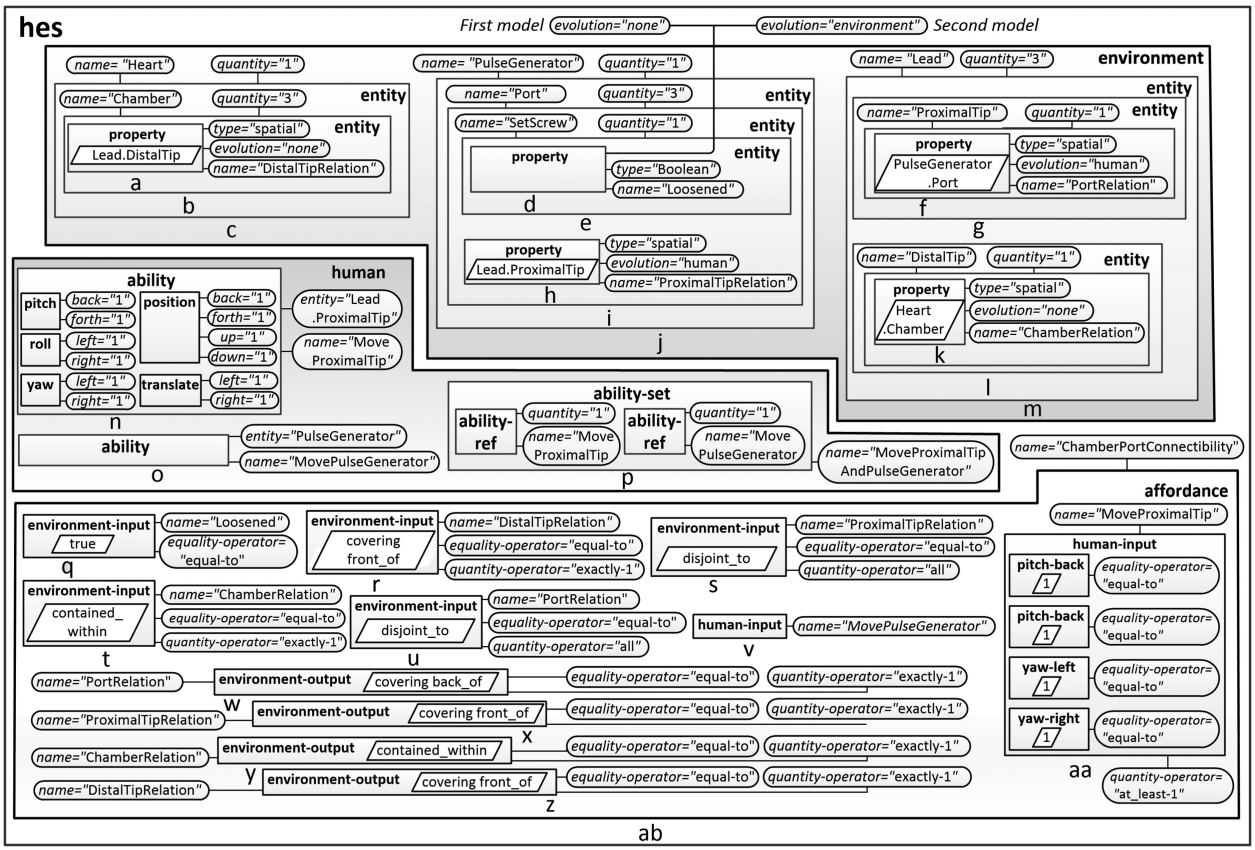

Fig. 4. CAVEMEN-XML representation of the pacemaker system HES. Elements, attributes, attribute values, and text content are represented using the conventions of Section IV. Letters are added for reference in text of Section V-A. Shading aids in differentiating child elements of **hes**: **environment** (white to gray), **human** (gray to white), and **affordance** (none).

which represents the three applicable chambers (LV, RV, and RA). "PulseGenerator" has one child entity named "Port" (see Fig. 4i), which represents the pulse generator's three ports. "Port" has one child entity named "SetScrew" (see Fig. 4e), which represents the set screw of each port. "Lead" has two child entities: "ProximalTip" (see Fig. 4g), which represents each lead's proximal tip; and "DistalTip," which represents each lead's distal tip (see Fig. 4l).

There are five applicable properties for the modeled entities. In the first model, three properties are specified as unchanging (*evolution* = "none"): one spatial property of "Chamber" named "DistalTipRelation," which represents the spatial relationship between a heart chamber and a lead distal tip (see Fig. 4a); one Boolean property of "SetScrew" named "Loosened," which represents whether a set screw is loosened (see Fig. 4d); and one spatial property of "DistalTip" named "ChamberRelation," which represents the spatial relationship between a lead distal tip and a heart chamber (see Fig. 4k). In the second model, "Loosened" is specified as changing independently of chamber-port connectibility (*evolution* = "environment").

In both models, two properties are specified as changing due to the surgeon actualizing an affordance (*evolution* = "human"): one spatial property of "Port" named "ProximalTipRelation," which represents the spatial relationship between a pulse generator port and a lead distal tip (see Fig. 4h); and one spatial property of "ProximalTip" named "PortRelation," which represents the spatial relationship between a lead distal tip and a pulse generator port (see Fig. 4f).

Both models include a **human** element having three child elements that represent the surgeon's motor abilities. One **ability** element named "MoveProximalTip" specifies that the surgeon can move a lead proximal tip along all axes and directions (see Fig. 4n). Another ability element named "MovePulseGenerator" specifies the same movements for the pulse generator (see Fig. 4o, child elements not depicted are identical to those of Fig. 4n). One **ability-set** element named "MoveProximalTipAndPulseGenerator" specifies that the surgeon can move the pulse generator and one lead proximal tip in parallel (see Fig. 4p). We do not model the surgeon's ability to move two lead proximal tips in parallel because such an ability is unneeded to actualize chamber-port connectibility.

One **affordance** element named "ChamberPortConnectibility" represents the affordance of interest (see Fig. 4ab). Five **environment-input** elements specify that "ChamberPortConnectibility" emerges if all of the following conditions hold: at least one pulse-generator port's set screw is loosened (see Fig. 4q), exactly one lead distal tip is contained within a heart chamber (see Fig. 4t), exactly one heart chamber is covering the front surface of a lead distal tip (see Fig. 4r), all pulse generator ports are disjoint to a lead proximal tip (see Fig. 4s), and all lead proximal tips are disjoint to a pulse-generator port (see Fig. 4u). These conditions capture many possible HES configurations, including the one shown in Fig. 1.

Two **human-input** elements specify that the following conditions must hold for "ChamberPortConnectibility" to emerge: the surgeon can position, translate, pitch, yaw, and roll the

pulse generator in all directions (see Fig. 4v); and the surgeon can pitch and yaw at least one lead proximal tip in all directions (see Fig. 4aa). These conditions capture the movements that are potentially needed for the surgeon to actualize chamber-port connectibility in any HES configuration satisfying the **environment-input** conditions.

Four **environment-output** elements specify that the following conditions must hold as a consequence of actualizing "ChamberPortConnectibility": exactly one lead proximal tip is covering the back surface of a pulse generator port (see Fig. 4w), exactly one pulse-generator port is covering the front surface of a lead proximal tip (see Fig. 4x), exactly one lead distal tip is contained within a target heart chamber (see Fig. 4y), and exactly one heart chamber is covering the front surface of a lead distal tip (see Fig. 4z).

B. Inspection

We invoked the translation tool to generate SAL models for the two CAVEMEN-XML representations. A total of 2662 lines of SAL code were generated for the first model. A total of 2668 lines of SAL code were generated for the second model. Both translated models represented 27 instances of the affordance, where nine are desired and 18 are undesired. The nine desired instances support a correct chamber-port connection, whereas the 18 undesired instances support an incorrect chamber-port connection.

To aid in understanding the logic of the translation tool, all input conditions for one of nine desired instances are shown in (4). References to Fig. 4 specify what line(s) of (4) correspond to **environment-input** elements of the CAVEMEN-XML representation. Horizontal lines demarcate sets of input conditions that correspond to the same Fig. 4 letter

$$
\begin{array}{l}
X.PulseGenerator.Port_1.SetScrew_{Loosened} = true \ \wedge \text{(Fig. 4q)} \\
\hline
\forall d : \{front\_of,\ back\_of,\ top\_of,\ bottom\_of,\ left\_of,\ right\_of\} \mid \\
X.Lead_1.DistalTip[Heart.Chamber_1][d] = contained\_within \ \wedge \\
\neg(X.Lead_2.DistalTip[Heart.Chamber_1][d] = contained\_within) \wedge \\
\neg(X.Lead_3.DistalTip[Heart.Chamber_1][d] = contained\_within) \wedge \text{(Fig. 4t)} \\
\hline
X.Heart.Chamber_1[Lead_1.DistalTip][front\_of] = covering \ \wedge \\
\neg(X.Heart.Chamber_2[Lead_1.DistalTip][front\_of] = covering) \ \wedge \\
\neg(X.Heart.Chamber_3[Lead_1.DistalTip][front\_of] = covering) \ \wedge \text{(Fig. 4r)} \\
\hline
X.PulseGenerator.Port_1[Lead_1.ProximalTip][d] = disjoint\_to \wedge \\
X.PulseGenerator.Port_2[Lead_1.ProximalTip][d] = disjoint\_to \wedge \\
X.PulseGenerator.Port_3[Lead_1.ProximalTip][d] = disjoint\_to \wedge \text{(Fig. 4s)} \\
\hline
X.Lead_1.ProximalTip[PulseGenerator.Port_1][d] = disjoint\_to \wedge \\
X.Lead_2.ProximalTip[PulseGenerator.Port_1][d] = disjoint\_to \wedge \\
X.Lead_3.ProximalTip[PulseGenerator.Port_1][d] = disjoint\_to \wedge \text{(Fig. 4u)} \\
\hline
(\forall m : \{pitch\_back, pitch\_forth, yaw\_left, yaw\_right, translate\_left, \\
translate\_right, position\_up, position\_down, position\_back, position\_forth\} \mid \\
Z.PulseGenerator[m] = 1) \ \wedge \text{(Fig. 4v)} \\
\hline
(\forall m : \{pitch\_back,\ pitch\_forth,\ yaw\_left,\ yaw\_right\} \mid \\
Z.Lead_1.ProximalTip[m] = 1) \text{ (Fig. 4aa)}
\end{array} \tag{4}
$$

The depicted input conditions support a desired instance of chamber-port connectibility because they specify a connection between port-1 and chamber-1. In (4), such a connection is established via lead-1. Input conditions for the eight remaining desired instances have similar syntax, but they reflect other correct connections via leads 1, 2, and 3, such as port-2 and chamber-2 via lead-3.

TABLE III
MODEL CHECKING RESULTS

Model	Result	Verification time (s)
First	*no witnesses*	10.29
Second	*witness*	20.16

C. Trace Evaluation

To support the trace evaluation, we instantiated two enumerated-list variables: *desired*, which incorporates three desired instances of the affordance (one for each lead), and *undesired*, which incorporates the 18 undesired instances. The selected desired instances include $ChamberPortConnectibility_1$, which supports connecting chamber-1 to port-1 via lead-1; $ChamberPortConnectibility_{14}$, which supports connecting chamber-2 to port-2 via lead-2; and $ChamberPortConnectibility_{27}$, which supports connecting chamber-3 to port-3 via lead-3. Using these variables, we verified the negative form of (3) in order to search for witnesses in both translated formal models.

Verification was conducted using SAL-INF-BMC [19] on a 3.5-GHz workstation with 64-GB RAM running the Ubuntu 16.04 desktop. Results are shown in Table III.

No witnesses were returned in the first model, indicating that if the pulse-generator ports' set screws have unchanging initial states, then the system cannot be considered both accurate and robust with respect to chamber-port connectibility.

The witness returned for the second model has five steps in which none of the 18 undesired instances emerged, but all three identified desired instances were actualized. In every step, all three lead distal tips are implanted within the three respective heart chambers as specified in (4). In the first step, the surgeon can move the pulse generator and the proximal tip of lead-2 as specified in (4). The set screw of port-2 is loosened, whereas the set screws of ports 1 and 3 are not. Thus, $ChamberPortConnectibility_{14}$ is actualized, resulting in a connection between chamber-2 and port-2 via lead-2. In the second step, the surgeon can move the pulse generator and the proximal tip of lead-3 as specified in (4). The set screw of port-3 is loosened, whereas the set screws of ports-1 and 2 are not. Thus, $ChamberPortConnectibility_{27}$ is actualized, resulting in a connection between chamber-3 and port-3 via lead-3. In the third step, the surgeon can move the pulse generator and the proximal tip of lead-1 as specified in (4). The set screw of port-1 is loosened, whereas the set screws of ports 2 and 3 are not. Thus, $ChamberPortConnectibility_1$ is actualized, resulting in a connection between chamber-1 and port-1 via lead-1. No transitions execute in the fourth or fifth step; thus, the final state is one in which each lead establishes a correct chamber-port connection. This result indicates that if the states of all three set screws can change, then it is possible for the pacemaker system to be both accurate and robust with respect to chamber-port connectibility.

VI. Discussion and Conclusions

This paper has introduced CAVEMEN, a formal approach to object-oriented connectibility affordances. CAVEMEN-XML enables the analyst to specify a formal representation of the HES, including parallel entity movements that are possible for a user and connectibility affordances with respect to duplicate entities in combination. The translator reduces the need for manually encoding model checking syntax by automatically generating all instances of the same connectibility affordance in the native syntax of SAL [19], including positive and negative instantiations of the accuracy and robustness specification. The inspection technique supports enumerating desired/undesired affordance instances with respect to translated model checking syntax. The trace evaluation technique supports analyses of accuracy and robustness in different environmental contexts.

We demonstrated CAVEMEN with a pacemaker system case study. The inspection showed that the HES supports 27 instances of chamber-port connectibility: nine desired and 18 undesired. The trace evaluation showed that different environmental contexts can affect accuracy and robustness: if an affordance input of interest never changes (whether a pulse generator port's set screw is loosened), then the system cannot be both accurate and robust; if the same affordance input can change nondeterministically, then the system can be both accurate and robust. These results indicate that CAVEMEN shows promise for analyzing object-oriented connectibility affordances of a safety-critical system.

A. Methodological Considerations

The case study illustrated methodological benefits of CAVEMEN with respect to the identified research contributions, including the enumeration of desired/undesired connectibility affordance instances, verification of accuracy and robustness, and modeling of emergent behaviors in different environmental contexts.

Regarding the enumeration of desired/undesired connectibility affordance instances, we showed that CAVEMEN can address the number of unique connection configurations defined in (1). In the case study, the translator successfully generated an expected result of 27 instances, whereas the inspection aided in identifying which instances are desired and undesired respectively. Thus, the inspection technique could be useful on its own—independently of the trace evaluation—as a way of determining what kinds of connections are possible due to duplicate entities and connection sources/targets.

Regarding the verification of accuracy and robustness, we showed that it is possible for trace evaluation results to correctly reflect an actual accuracy and robustness problem; i.e., the result of *no witnesses* for the first model indicates that it is possible for a surgeon to actualize an undesired instance of chamber-port connectibility—an expected result in light of the source material from a U.S. national database. This result indicates that the trace evaluation technique could be useful in a real-world application.

Regarding the modeling of emergent behaviors in different environmental contexts, we showed that nondeterministic state-transition behavior of one entity property can produce accuracy and robustness differences. Design insights gleaned from this technique come by imagining what could enable the behaviors observed in a returned witness. For example, one interpretation of the second model is that a hypothetical, automated control system adjusts set screws to ensure that only desired chamber-port connectibility instances emerge, such as by loosening the port-2 set screw and tightening the others if the surgeon is touching the lead-2 proximal tip (as in step-1 of the returned witness). Such an interpretation could inform the operational concept of an improved pacemaker system, although a different approach would be needed to model the envisioned automation.

An overarching benefit is reducing the amount of manually encoded syntax that is needed to support the analyses. For example, instantiating and translating the case study CAVEMEN-XML representation reflected a 2611-line reduction of manually encoded syntax. There is also a reduced need for knowledge of LTL semantics, as the translator generates both positive and negative instantiations of (3).

B. Limitations and Future Work

As shown with the case study analysis, our translation tool could generate thousands of lines of SAL code from a CAVEMEN-XML instantiation. This observation, which is mainly an artifact of duplicate-entity specifications, raises at least two scalability concerns: first, the translation algorithms could overwhelm a web browser, and second, infinite-bounded model checking could take an impracticably long time to execute. Future work should explore ways of reducing these concerns, such as by conducting benchmarking studies to inform a more efficient XML-to-SAL translation mechanism (see for example [29]).

Our interpretation of affordance is based on an extant formalism [4]; however, it could be limited with respect to other human-integrated system considerations. For example, Baber [30] indicates that we do not define the "rationale for performing the action [associated with an affordance] in the first place." He thus proposes an extended framework (called *Forms of Engagement*) that considers the user's perception, culture, and goals. Future work should explore ways of utilizing CAVEMEN in such a holistic context. One step toward achieving this could involve combining a CAVEMEN formal model with formal models of task behavior and perceptual artifacts. For example, a formal task model, such as EOFM [31], could aid in identifying whether connectibility affordances support a normative sequence of actions (or whether a normative procedure prevents undesired affordances from emerging), whereas a formal signifier model, such as BIGSIS [32], could aid in identifying whether perceivable properties of environment entities support correct connections (e.g., color-coded connection sources and targets).

Our interpretations of accuracy and robustness are inspired by extant HFE standards [1], [2]; however, there could be other interpretations that are not constrained to one type of connectibility affordance. For example, we did not demonstrate a way of verifying accuracy and robustness with respect to other affordances (e.g., "set screw tightenable/loosenable") or multiple connectibility affordances in combination. Future work should thus explore an extended set of accuracy and robustness specifications.

REFERENCES

[1] *Human Factors Engineering–Design of Medical Devices*," ANSI/AAMIHE75:2009/(R)2013, AAMI, Arlington, VA, USA, 2013.

[2] *Ergonomics of Human-System Interaction – Part 210: Human-Centered Design for Interactive Systems*, ISO 9241-210:2010, Geneva, Switzerland, 2017.

[3] J. J. Gibson, "The theory of affordances," in *Perceiving, Acting, and Knowing: Toward an Ecological Psychology*, J. B. Robert E Shaw, Ed. Hillsdale. Oxford, U.K.: N. J. Lawrence Erlbaum Assoc., 1977, pp. 67–82.

[4] T. A. Stoffregen, "Affordances as properties of the animal-environment system," *Ecol. Psychol.*, vol. 15, no. 2, pp. 115–134, 2003.

[5] A. J. Abbate and E. J. Bass, "Modeling affordance using formal methods," in *Proc. Annu. Meeting Human Factors Ergonom. Soc.*, 2017, pp. 723–727.

[6] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model Checking.* Cambridge, MA, USA: MIT Press, 1999.

[7] A. J. Wells, "Gibson's affordances and Turing's theory of computation," *Ecol. Psychol.*, vol. 14, no. 3, pp. 140–180, 2002.

[8] A. M. Turing, "On computable numbers, with an application to the entscheidungs problem," *J. Math*, vol. 58, no. 345–363, pp. 230–265, 1936.

[9] M. T. Turvey, "Affordances and prospective control: An outline of the ontology," *Ecol. Psychol.*, vol. 4, no. 3, pp. 173–187, 1992.

[10] L. Rothrock, R. Wysk, N. Kim, D. Shin, Y.-J. Son, and J. Joo, "A modelling formalism for human-machine cooperative systems," *Int. J. Prod. Res.*, vol. 49, no. 14, pp. 4263–4273, 2011.

[11] J. Joo *et al.*, "Agent-based simulation of affordance-based human behaviors in emergency evacuation," *Simul. Model. Pract. Theory*, vol. 32, pp. 99–115, 2013.

[12] A. Lenarčič and M. Winter, "Affordances in situation theory," *Ecol. Psychol.*, vol. 25, no. 2, pp. 155–181, 2013.

[13] J. Barwise and J. Perry, "Situations and attitudes," *J. Philos.*, vol. 78, pp. 668–691, 1981.

[14] D. L. Parnas, "On the use of transition diagrams in the design of a user interface for an interactive computer system," in *Proc. 24th Nat. ACM Conf.*, 1969, pp. 379–385.

[15] E. M. Clarke, E. A. Emerson, and A. P. Sistla, "Automatic verification of finite-state concurrent systems using temporal logic specifications," *ACM Trans. Program. Lang. Syst.*, vol. 8, no. 2, pp. 244–263, 1986.

[16] M. J. Egenhofer and J. Herring, "A mathematical framework for the definition of topological relationships," in *Proc. 4th Int. Symp. Spatial Data Handling.*, 1990, pp. 803–813.

[17] A. U. Frank, "Qualitative spatial reasoning about distances and directions in geographic space," *J. Vis. Lang. Comput.*, vol. 3, no. 4, pp. 343–371, 1992.

[18] J. Bird and C. Ross, *Mechanical Engineering Principles*. New York, NY, USA: Routledge, 2012.

[19] L. De Moura, S. Owre, and N. Shankar, "The SAL language manual," Comput. Sci. Lab., SRI Int., Menlo Park, CA, USA, Tech. Rep. CSL-01-01, 2003.

[20] M. L. Bolton, R. I. Siminiceanu, and E. J. Bass, "A systematic approach to model checking human-automation interaction using task analytic models," *IEEE Trans. Syst., Man Cybern., Part A, Syst. Humans*, vol. 41, no. 5, pp. 961–976, Sep. 2011.

[21] M. L. Bolton and E. J. Bass, "Enhanced operator function model (EOFM): A task analytic modeling formalism for including human behavior in the verification of complex systems," in *The Handbook of Formal Methods in Human-Computer Interaction*, B. Weyers, J. Bowen, A. Dix, and P. Palanque, Eds. New York, NY: Springer, 2017, pp. 343–377.

[22] M. L. Bolton, N. Jimenez, M. M. van Paassen, and M. Trujillo, "Automatically generating specification properties from task models for the formal verification of human–automation interaction," *IEEE Trans. Human-Mach. Syst.*, vol. 44, no. 5, pp. 561–575, Oct. 2014.

[23] A. Biere, A. Cimatti, E. M. Clarke, O. Strichman, and Y. Zhu, "Bounded model checking," *Adv. Comput.*, vol. 58, pp. 117–148, 2003.

[24] E. Şahin, M. Cakmak, M. R. Doğar, E. Uğur, and G. Üçoluk, "To afford or not to afford: A new formalization of affordances toward affordance-based robot control," *Adapt. Behav.*, vol. 15, no. 4, pp. 447–472, 2007.

[25] *Guide for Addressing Accessibility in Standards*, ISO, Geneva, Switzerland, ISO/IEC Guide 71:2014(E), 2014.

[26] A. J. Abbate, E. J. Bass, and A. L. Throckmorton, "A formal task analytic approach to medical device alarm troubleshooting instructions," *IEEE Trans. Human-Mach. Syst.*, vol. 41, no. 1, pp. 53–65, Feb. 2016.

[27] L. Quin, "Extensible markup language 1.0," 1997. [Online]. Available: http://www.w3.org/XML/Core/

[28] C. Sperberg-McQueen and H. Thompson, "The W3C XML schema 1.0," 2001. [Online]. Available: http://www.w3.org/XML/Schema

[29] M. L. Bolton, X. Zheng, K. Molinaro, A. Houser, and M. Li, "Improving the scalability of formal human–automation interaction verification analyses that use task-analytic models," *Innov. Syst. Softw. Eng.*, vol. 13, pp. 1–17, 2017.

[30] C. Baber, "Designing smart objects to support affording situations: Exploiting affordance through an understanding of forms of engagement," *Frontiers Psychol.*, vol. 9, 2018, Art. no. 292.

[31] M. L. Bolton, E. J. Bass, and R. I. Siminiceanu, "Generating phenotypical erroneous human behavior to evaluate human–automation interaction using model checking," *Int. J. Human-Comput. Stud.*, vol. 70, no. 11, pp. 888–906, 2012.

[32] A. J. Abbate and E. J. Bass, "A formal methods approach to semiotic engineering," *Int. J. Human-Comput. Stud.*, vol. 115, pp. 20–39, 2018.

Andrew J. Abbate (S'15–M'17) received the Ph.D. degree in biomedical science from the School of Biomedical Engineering, Science, and Health Systems, Drexel University, Philadelphia, PA, USA, in 2017.

He is currently a Senior Human Factors Engineer with the Pacific Science & Engineering Group, San Diego, CA, USA, where he specializes in visual analytics and human-autonomy integration. His research focuses on developing model-based tools and techniques that can be used to inform human-integrated systems design early in the engineering process.

Ellen J. Bass (M'98–SM'03) received the Ph.D. degree in industrial and systems engineering from the Georgia Institute of Technology, Atlanta, GA, USA.

She is currently a Professor and the Chair of the Department of Health Systems and Sciences Research, College of Nursing and Health Professions, Drexel University, Philadelphia, PA, USA, a Professor with the Department of Information Science, Drexel University's College of Computing and Informatics, and an Affiliate Professor with the Drexel University's School of Biomedical Engineering, Science and Health Systems. She has more than 30 years of human-centered systems engineering research and design experience in air transportation, healthcare, and other domains and the focus of her research is to develop theories of human performance, quantitative modeling methodologies, and associated analysis methods that can be used to evaluate human-system interaction in the context of total system performance.