Contract-Based Hierarchical Modeling and Traceability of Heterogeneous Requirements

Nikhil Vijay Naik, Alessandro Pinto, and Pierluigi Nuzzo

***Abstract*—The design of complex mission-critical systems often follows a layered approach, which may lead to complicated, multi-level, multi-viewpoint requirement hierarchies. This heterogeneity makes it challenging to guarantee the traceability of the requirements across levels of abstraction and, consequently, the satisfaction of the requirements by a system implementation, especially when requirements at different abstraction levels are expressed using different mathematical formalisms and modeling languages. In this paper, we address this challenge by introducing *heterogeneous hierarchical contract networks (HHCNs)*, a formal model based on a graph of assume-guarantee contracts, for capturing and analyzing heterogeneous requirement hierarchies. We formulate the requirement *traceability validation* problem in terms of contract refinement relations between nodes in an HHCN. We then define *contract embeddings* to enable reasoning about refinements across levels of abstraction in the HHCN that are expressed using heterogeneous formalisms. Contract embeddings leverage the notion of conservative approximation to rigorously map contracts across levels of abstraction while ensuring that refinement is preserved independently of the formalism to which the contracts are mapped. We illustrate their effectiveness on a case study motivated by a multi-agent autonomous lunar rover mission.**

***Index Terms*—Cyber-physical systems, model-based design, requirement formalization, contract-based design, traceability validation**

I. Introduction

The growing complexity of mission-critical cyber-physical systems necessitates robust *requirement engineering* to enable seamless coordination among mission stakeholders and facilitate efficient execution of mission activities [1]. Beginning with a mission-level characterization of system objectives, such as the operational constraints on a spacecraft to fulfill science goals, these requirements percolate to sub-system requirements, e.g., pertaining to science instruments, thermal management, energy management, propulsion, navigation, and communication. In turn, these requirements shape the component-level requirements, e.g., sub-system requirements may be decomposed into requirements on sensors, sensor processing, state estimation, maneuver calculation, and actuation, which may be associated with architectural elements like hardware or software modules [2], [3]. As a result, these requirement patterns constitute multi-level, multi-viewpoint *requirement hierarchies*. Owing to this layered design approach, requirement formulation tends to be strongly influenced by the corresponding *level of abstraction*, producing *heterogeneous, hierarchical* sets of requirements.

Evaluating the satisfaction of mission-level requirements requires modeling frameworks to provide *traceability*, i.e., the ability to reason about the satisfaction of higher-level requirements from conclusions derived on a set of lower-level requirements. Additionally, these frameworks need to be amenable to verification, i.e., verifying that the hardware and software implementations satisfy the requirement hierarchy [1], [4]. Manual approaches to requirement formulation and traceability validation with natural language-based tools impose restrictions on the requirement syntax, e.g., limitations on the nesting of objectives within a single requirement, to streamline the formulation and stipulate patterns derived from current best practices for organizing requirements [4]. However, being natural language-driven, these approaches may introduce subjectivity, leading to requirements that might be ambiguous, conflicting, or incomplete [5]. Similarly, requirement hierarchies may be improperly structured, potentially resulting in incorrect claims of correctness that lead to unexpected design problems and errors discovered late in the design cycle [4]. This may render the system vulnerable to cost-intensive repairs or undesirable issues after deployment [6], [7]. The increasing adoption of architectures featuring autonomous decision-making artifacts, such as coordinated multi-agent autonomous systems, exacerbates these challenges, because requirement hierarchies governing such missions must ensure agent safety, correct coordination and adaptable operation in diverse scenarios while ensuring mission satisfaction [8].

An appealing solution to manage this complexity is to encapsulate requirements into a *formal* model of representation within a rigorous and structured framework. These frameworks can then be used to algorithmically provide strong guarantees on the soundness of the requirement hierarchy to assist faster vetting of the design. *Contract-based design* has been proposed as a general framework to manage system complexity by relying on formal, compositional representations of design elements and a rigorous algebra for their correct integration [9]–[11]. In a contract-based methodology, traceability between

Manuscript received XXXXXX; revised XXXXXX; accepted XXXXXX. Date of publication XX **MONTH** 2024; date of current version XX **MONTH** 2024. This work was supported in part by NSF under awards 1846524 and 2139982, and by a Jet Propulsion Laboratory (JPL) Graduate Fellowship. Part of this research was carried out at the JPL, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (NASA) (80NM0018D0004). Recommended for acceptance by XXXXXXXX. *(Corresponding author: Nikhil Vijay Naik.)*

Nikhil Vijay Naik (nikhilvn@usc.edu) is with the Department of Electrical and Computer Engineering, University of Southern California (USC), Los Angeles. Alessandro Pinto (alessandro.pinto@jpl.nasa.gov) is with the NASA JPL, California Institute of Technology, Pasadena. Pierluigi Nuzzo (nuzzo@eecs.berkeley.edu) is with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley and the Department of Electrical and Computer Engineering, USC.

requirements across hierarchical levels can be established via contract *refinement* relationships, where an aggregation of contracts at a lower level of the hierarchy is proven to refine a contract at a higher level. Contract refinement checking can also be used to solve certain verification problems. Algorithmic refinement analysis of *homogeneous* contract hierarchies, where all contracts are defined using the same formalism, has been explored in the literature [11]–[13]. However, for greater fidelity, it is often desirable to capture requirement hierarchies with heterogeneous models, by employing appropriate formalisms at different levels of abstraction [14]. While efforts in the literature [10], [14] have attempted to map contracts across heterogeneous formalisms for checking refinement in specific instances, a more general and systematic approach to refinement checking of heterogeneous contract hierarchies for traceability validation is still elusive.

This paper addresses these challenges by proposing a novel graph-based model, termed *heterogeneous hierarchical contract network (HHCN)*, to organize and formalize the heterogeneous set of requirements associated with a multi-level, multi-viewpoint system design process. The traceability validation problem can then be defined in terms of checking refinement between contracts belonging to adjacent layers in an HHCN, which may be represented using different formalisms. Secondly, we propose a method to rigorously map contracts from an abstraction level to another in the HHCN to perform refinement checking. These mappings, termed *contract embeddings*, are orthogonal to the particular method used to check refinement. Moreover, they ensure that the outcome of refinement checking is preserved independently of the level of abstraction at which the refinement is checked. We show the effectiveness of our HHCN model and the associated embeddings on the validation of the traceability of a requirement hierarchy motivated by a lunar rover mission.

II. Related Work

Architecture frameworks elucidating guidelines for system design have been proposed in the literature [2], [15]. Derived from current best practices, these frameworks outline general design principles, such as explication and vetting of all design assumptions and ensuring traceability of the design flow. However, they do not explicitly address requirement formalization or algorithmic methods for traceability validation.

Modeling and verification frameworks based on formal languages, e.g., TLA+ or the B-method, have been proposed in the literature, which also leverage *refinement* relations [16]–[18] to enable automated deductive reasoning and facilitate the application of analysis techniques such as theorem proving and model checking [19], [20]. Contract-based design aims to encompass and generalize existing frameworks toward a paradigm for compositional system modeling and design that is orthogonal to the specific formalisms, used for capturing the elements or aspects of a design, and the associated analysis techniques [9]. Hierarchical contract networks (HCNs) were first proposed as a foundation for contract-based hierarchical system synthesis in terms of the stepwise refinement of a system-level specification into an aggregation of components from a pre-characterized library of implementations [12], [21]. However, in previous work, all the refinements in an HCN were established between homogeneous contracts. In this paper, we extend the HCN model to define heterogeneous hierarchical contract networks (HHCNs) and provide a foundation for rigorous requirement engineering with heterogeneous contracts.

Analysis methods for heterogeneous system architectures have been proposed in the literature by coordinating multiple models of computation [22]–[24]. Other approaches represent components as sets of *behaviors* and map these sets across different formalisms through *behavioral relations* [25], later augmented by *vertical contracts* [14]. Contracts extend behavioral models by specifying the properties of a component in terms of a *pair* of sets, namely, the *assumptions* and the *guarantees* [9], [10]. Consequently, we build on behavioral relations and vertical contracts by further extending them to support the analysis of generic HHCNs.

Our approach is founded on *Galois connections*, which have been used to define mappings between behaviors at different abstraction levels [26]. Using a Galois connection, a set of behaviors in a *concrete* formalism can be mapped to an *abstract* formalism for efficient analysis. The result of the analysis can then be used to derive constraints on the set of behaviors in the concrete formalism [26], [27]. *Conservative approximations* extend Galois connections to enable abstraction-based methods for efficiently checking refinement preorders between sets of behaviors defined in the concrete formalism [27]–[29]. However, attempts at directly extending the notion of Galois connection from sets of behaviors to contracts has led to obstacles. While a Galois connection on sets of behaviors can be used to define an abstraction to map contract assumptions and guarantees from a concrete formalism to an abstract formalism, the same Galois connection may not be used to define a concretization map for contracts [30]. This paper overcomes this obstacle by rather leveraging conservative approximations to abstract and concretize contracts. Differently from previous approaches, aiming to transform both the assumptions and the guarantees of a contract, via the same abstraction function, from a concrete domain to an abstract domain where contract-based verification becomes tractable, our objective is, instead, to extend the notion of refinement checking and enable its effective evaluation between contracts defined in different formalisms.

III. Preliminaries

We introduce background notions on contract-based design, Galois connections, and conservative approximations, which are used throughout the paper.

A. Contract-Based Design

Assume-guarantee (A/G) contracts provide a rigorous algebra for the construction of systems from a set of components [9]. We define a *contract* $\mathcal{C}$ on a component M as a triple $\mathcal{C} = (V, A, G)$. V is a set of variables. A, termed the *assumptions*, and G, termed the *guarantees*, are sets of behaviors over V, which we assume expressed in a formalism

$\mathcal{B}$, as further detailed below. *M satisfies $\mathcal{C}$*, written $M \models \mathcal{C}$, if all its behaviors lie within the set G, given a set of assumptions A, i.e., if $M \subseteq G \cup \bar{A}$, where $\bar{A}$ denotes the complement of A. A contract is said to be *saturated* when G is replaced by $G \cup \bar{A}$. Any contract is equivalent to its saturated form. Consequently, in this paper, we assume that all contracts are saturated. A contract $\mathcal{C}$ is *consistent* when the set of guarantees $G \cup \bar{A}$ is nonempty, i.e., there exists at least one implementation of $\mathcal{C}$. It is said to be *compatible* when the set of assumptions A is nonempty, i.e., there is at least one valid environment for $\mathcal{C}$.

We can compare two contracts $\mathcal{C}_1 = (V, A_1, G_1)$ and $\mathcal{C}_2 = (V, A_2, G_2)$ through the *refinement* relationship, which is a preorder on contracts. $\mathcal{C}_1$ refines $\mathcal{C}_2$, written $\mathcal{C}_1 \preceq \mathcal{C}_2$, whenever $\mathcal{C}_1$ has weaker assumptions and stronger guarantees, namely, $A_1 \supseteq A_2$ and $G_1 \subseteq G_2$ [9]. To combine individual contracts, we define the *conjunction* of two contracts $\mathcal{C}_1$ and $\mathcal{C}_2$ as the greatest lower bound for the refinement relation, written as $\mathcal{C}_\wedge = \mathcal{C}_1 \wedge \mathcal{C}_2$ and given by the triple $(V, A_\wedge, G_\wedge)$, where $A_\wedge = A_1 \cup A_2$ and $G_\wedge = G_1 \cap G_2$. This operation can be used to represent a combination of requirements that must be satisfied simultaneously by a component. On the other hand, the *composition* operation on contracts is used to combine requirements associated with different components in an architecture. Formally, the composition operation between two contracts $\mathcal{C}_1$ and $\mathcal{C}_2$, written $\mathcal{C}_1 \otimes \mathcal{C}_2$, is equal to the contract $(V, A_\otimes, G_\otimes)$, where $A_\otimes = (A_1 \cap A_2) \cup (\overline{G_1 \cap G_2})$ and $G_\otimes = G_1 \cap G_2$. The composite contract delivers all the guarantees promised by each individual contract, hence the formula for the guarantees. The assumptions are instead computed such that any valid environment, in the context of each component, provides a suitable environment for the other, i.e., supports its assumptions. Both conjunction and composition are commutative and associative. We refer to the literature [9] for further treatment of the notions mentioned above. In this paper, we use contracts to capture requirements precisely and rigorously check their consistency, compatibility, and traceability by formulating and solving contract refinement checking problems.

B. Galois Connections and Conservative Approximations

We recall the notion of *Galois connection* as a way to transfer behaviors across heterogeneous formalisms. We consider behaviors in two formalisms, denoted by $\mathcal{B}$ and $\mathcal{B}'$, characterized by the set of all possible behaviors $\mathbf{B}$ and $\mathbf{B}'$, respectively [31]. For example, if $\mathcal{B}$ is a continuous-time formalism, $\mathbf{B}$ may include the set of trajectories of a continuous-time system; if $\mathcal{B}'$ is a discrete-time formalism, $\mathbf{B}'$ may include the traces of a discrete-time system. Intuitively, a Galois connection can be used to define mappings between behaviors in $\mathbf{B}$ and behaviors in $\mathbf{B}'$.

Definition 1 (Galois Connection)**.** *Let $(\mathbf{B}, \preccurlyeq_{\mathbf{B}})$, $(\mathbf{B}', \preccurlyeq_{\mathbf{B}'})$ be two partially ordered sets, characterizing formalisms $\mathcal{B}$ and $\mathcal{B}'$, respectively. A pair of functions $\alpha : \mathbf{B} \to \mathbf{B}'$ and $\gamma : \mathbf{B}' \to \mathbf{B}$ forms a* Galois connection *$\langle \alpha, \gamma \rangle$ if and only if $\forall A \in \mathbf{B}, A' \in \mathbf{B}'$,*

$$\alpha(A) \preccurlyeq_{\mathbf{B}'} A' \longleftrightarrow A \preccurlyeq_{\mathbf{B}} \gamma(A'). \tag{1}$$

We also say that $\langle \alpha, \gamma \rangle$ is a Galois connection from $\mathcal{B}$ to $\mathcal{B}'$.

If there exist distinct behaviors A_1, A_2 in $\mathbf{B}$ and A' in $\mathbf{B}'$, such that $A' = \alpha(A_1) = \alpha(A_2)$, then we say that $\mathcal{B}'$ is the *abstract* formalism and $\mathcal{B}$ is the *concrete* formalism. Let A be defined in $\mathcal{B}$. We can leverage Definition 1 to efficiently derive bounds for A as follows. We map A into an abstract domain in $\mathcal{B}'$ by defining a Galois connection $\psi = \langle \alpha, \gamma \rangle$ from $\mathcal{B}$ to $\mathcal{B}'$ such that computing $\alpha(A)$ is more tractable. We can then search for A' such that $\alpha(A) \preccurlyeq_{\mathbf{B}'} A'$ in the abstract formalism and transfer A' back to the concrete formalism by computing $\gamma(A')$. By (1), $\gamma(A')$ acts as a bound for A in the concrete formalism [26], [31].

As an example, let $(\mathbf{B}, \preccurlyeq_{\mathbf{B}})$ be the set of real numbers $\mathbb{R}$, equipped with the standard ordering $\leq$, and $(\mathbf{B}', \preccurlyeq_{\mathbf{B}'})$ be the set of integers $\mathbb{Z}$, equipped with the standard ordering $\leq$. Consider the functions $\alpha_u : \mathbb{R} \to \mathbb{Z},\ \alpha_u(r) := \lceil r \rceil$, $\gamma_u : \mathbb{Z} \to \mathbb{R},\ \gamma_u(s) := s$, where $\lceil . \rceil$ denotes the ceiling function. Then, $\psi_1 = \langle \alpha_u, \gamma_u \rangle$ is a Galois connection, satisfying the conditions given in Definition 1. We observe that multiple elements in $\mathbb{R}$ map to a single element in $\mathbb{Z}$ via α_u. Therefore, we say that $\alpha_u(r)$ is the *abstraction* of $r \in \mathbb{R}$, whereas $\gamma_u(s)$ is a *concretization* of $s \in \mathbb{Z}$. Consider the interval $\mathcal{T} = (2, 3]$ in $\mathbb{R}$. We use ψ_1 to find an upper bound of $\mathcal{T}$. Because $\forall r \in \mathcal{T}$, $\alpha_u(r) = \lceil r \rceil = 3$ holds, we observe that $\forall r \in \mathcal{T}, \alpha_u(r) \leq s_\top = 3$ and $s_\top = 3$ is an upper bound in $\mathbb{Z}$. We can then apply the concretization function γ_u to obtain $\gamma_u(s_\top) = 3$ and use (1) to conclude $\forall r \in \mathcal{T}, r \leq \gamma_u(s_\top) = 3$ and $\mathcal{T}$ is bounded by 3. To summarize, Galois connections can be leveraged to design abstraction-based methods to efficiently derive bounds on sets of behaviors. We recall below an equivalent definition of Galois connections [31].

Theorem 1 (Equivalent Definition of Galois Connection)**.** *Let $(\mathbf{B}, \preccurlyeq_{\mathbf{B}})$ and $(\mathbf{B}', \preccurlyeq_{\mathbf{B}'})$ be as defined above. A pair of functions $\alpha : \mathbf{B} \to \mathbf{B}'$ and $\gamma : \mathbf{B}' \to \mathbf{B}$ form a Galois connection $\psi = \langle \alpha, \gamma \rangle$ if and only if*

1) *$\forall A \in \mathbf{B}, A' \in \mathbf{B}' : A \preccurlyeq_{\mathbf{B}} \gamma \circ \alpha(A)$ and $\alpha \circ \gamma(A') \preccurlyeq_{\mathbf{B}'} A'$.*
2) *α and γ are* monotone *functions, i.e., $\forall A_1, A_2 \in \mathbf{B}$, if $A_1 \preccurlyeq_{\mathbf{B}} A_2$ then $\alpha(A_1) \preccurlyeq_{\mathbf{B}'} \alpha(A_2)$. Similarly, $\forall A'_1, A'_2 \in \mathbf{B}'$, if $A'_1 \preccurlyeq_{\mathbf{B}'} A'_2$, then $\gamma(A'_1) \preccurlyeq_{\mathbf{B}} \gamma(A'_2)$.*

Intuitively, Property (1) of Theorem 1 provides results on the existence of bounds. Given A in the concrete formalism, $\gamma \circ \alpha(A)$ acts as a trivial upper bound for A. Analogously, given A' in the abstract formalism, $\alpha \circ \gamma(A')$ is a trivial lower bound. Property (2) describes the effect of applying α and γ to *multiple* (sets of) behaviors in each formalism, stating that the ordering between them is preserved upon abstraction and concretization. However, in the converse direction, $\alpha(A_1) \preccurlyeq_{\mathbf{B}'} \alpha(A_2)$ need not imply $A_1 \preccurlyeq_{\mathbf{B}} A_2$. For example, consider the Galois connection ψ_1 between $\mathbb{R}$ to $\mathbb{Z}$ defined above, and let $r_1 = 2.8, r_2 = 2.5$. By inspection, we obtain $\alpha_u(r_1) \leq \alpha_u(r_2)$ in the abstract formalism. However, in the concrete formalism, $r_1 \not\leq r_2$. Galois connections may not preserve the outcome of checking the ordering $\leq$ from the abstract to the concrete formalism [29]. *Conservative approximations* [27]–[29] were introduced to transfer behaviors across sets $\mathbf{B}$ and $\mathbf{B}'$, while guaranteeing that ordering is preserved in either direction.

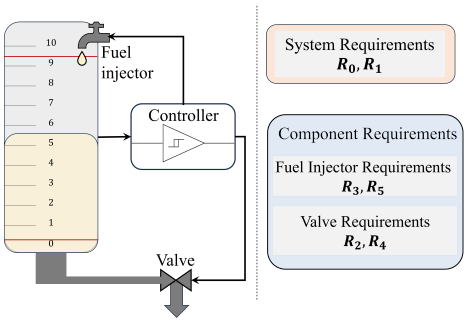


Fig. 1: Tank control problem with heterogeneous requirements.

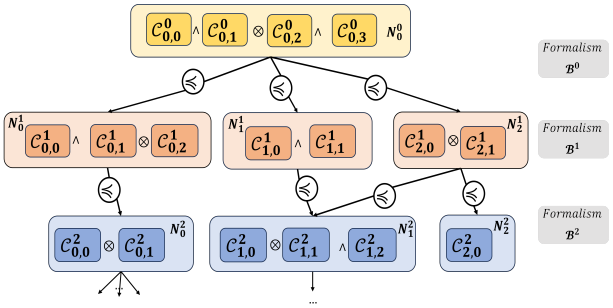


Fig. 2: Illustration of an HHCN $\mathcal{G}$.

A conservative approximation Ψ relies on *two* abstraction functions $\Psi = (\alpha_l, \alpha_u)$ to transfer behaviors from $\mathbf{B}$ to $\mathbf{B}'$.

Definition 2 (Conservative Approximation)**.** *Let* $(\mathbf{B}, \preccurlyeq_{\mathbf{B}})$ *and* $(\mathbf{B}', \preccurlyeq_{\mathbf{B}'})$ *be as in Definition 1. A pair of functions* $\alpha_l : \mathbf{B} \to \mathbf{B}'$ *and* $\alpha_u : \mathbf{B} \to \mathbf{B}'$ *forms a conservative approximation if* $\forall A_1, A_2 \in \mathbf{B}$*,* $\alpha_u(A_1) \preccurlyeq_{\mathbf{B}'} \alpha_l(A_2)$ *implies* $A_1 \preccurlyeq_{\mathbf{B}} A_2$*.*

The notion of conservative approximation has been studied in relation to Galois connections [29], producing an equivalent definition in terms of a pair of Galois connections.

Definition 3 (Conservative Approximation [29])**.** *Let* $(\mathbf{B}, \preccurlyeq_{\mathbf{B}})$ *and* $(\mathbf{B}', \preccurlyeq_{\mathbf{B}'})$ *be as in Definition 1. Let* $\langle \alpha_u, \gamma_u \rangle$ *be a Galois connection from* $\mathcal{B}$ *to* $\mathcal{B}'$ *and* $\langle \gamma_l, \alpha_l \rangle$ *be a Galois connection from* $\mathcal{B}'$ *to* $\mathcal{B}$*.* $\Psi = (\alpha_l, \alpha_u)$ *is a* conservative approximation *from* $\mathcal{B}$ *to* $\mathcal{B}'$ *if and only if, for all* $A' \in \mathbf{B}'$*,* $\gamma_u(A') \preccurlyeq_{\mathbf{B}} \gamma_l(A')$ *and for all* $A \in \mathbf{B}$*,* $\alpha_l(A) \preccurlyeq_{\mathbf{B}'} \alpha_u(A)$*.*

To illustrate these definitions, consider a conservative approximation from $\mathbb{R}$ to $\mathbb{Z}$ [29], written $\Psi^{\mathbb{Z}}_{\mathbb{R}}$, composed of two Galois connections $\psi_1 = \langle \alpha_u, \gamma_u \rangle$ and $\psi_2 = \langle \gamma_l, \alpha_l \rangle$. Let $\alpha_u : \mathbb{R} \to \mathbb{Z}$, $\alpha_u(r) = \lceil r \rceil$, $\alpha_l : \mathbb{R} \to \mathbb{Z}$, $\alpha_l(r) = \lfloor r \rfloor$, $\gamma_u(s) = \gamma_l(s) = s$, where $r \in \mathbb{R}$, $s \in \mathbb{Z}$. We can see that $\forall r \in \mathbb{R}$, $\alpha_l(r) \leq \alpha_u(r)$ and $\forall s \in \mathbb{Z}, \gamma_u(s) \leq \gamma_l(s)$. Therefore, both conditions of Definition 3 are satisfied. Moreover, as required by Definition 2, for any $r_1, r_2 \in \mathbb{R}$, if $\alpha_u(r_1) \leq \alpha_l(r_2)$, then it is always true that $r_1 \leq r_2$. In this paper, we use conservative approximations to define abstract and concrete contract embeddings. We will prove that resorting to conservative approximations is instrumental in defining contract abstractions and concretizations that retain the outcome of refinement checking independently of the formalism, abstract or concrete, in which the check is performed.

IV. Requirement Modeling and Traceability

We begin with an example to motivate our approach.

Example 1. *We consider a control system for filling a tank with fuel, as illustrated in Figure 1, and the following system requirement set* $\mathcal{R} = \{R_0, \ldots, R_5\}$*:*

R_0*: If the fuel level is less than* 5*, the valve shall be* off*.*

R_1*: If the fuel level is less than* 5*, the fuel injector shall be* on*.*

$R_{2(3)}$*: The voltage applied to the valve (fuel injector) in the* off *condition shall be between* V^{off}_{min} *and* V^{off}_{max}*.*

$R_{4(5)}$*: The voltage applied to the valve (fuel injector) in the* on *condition shall be between* V^{on}_{min} *and* V^{on}_{max}*.*

For simplicity, we assume that $\mathcal{R}$ *applies to the fuel level and the* on/off *status of the valve and the fuel injector at each instant. Additionally, we assume that the fuel level is measured in steps of* 1 m*.*

The requirements in Example 1 can be expressed using different formalisms. For example, at the system level, they may be concisely captured using arithmetic constraints over discrete (integer) variables. At the component level, we may use inequality constraints over continuous (real) variables, such as the height of the fuel column and the actuation voltages applied to the valve and fuel injector. In general, we assume that system requirements can be expressed with a predefined number of formalisms $\{\mathcal{B}^0, \ldots, \mathcal{B}^{L-1}\}$, e.g., chosen by the designer to represent the system behaviors at different levels of abstraction with sufficient accuracy, while making their analysis tractable. We introduce contract networks (CNs) to formalize sets of interrelated requirements.

Definition 4 (Contract Network)**.** *A* contract network N *is defined recursively as follows:*

- *A* contract term *is an expression* $\mathcal{C}_N := \mathcal{C} \,|\, \mathcal{C}_{N,1} \wedge \mathcal{C}_{N,2} \,|\, \mathcal{C}_{N,1} \otimes \mathcal{C}_{N,2}$ *where* $\mathcal{C}$ *is a contract and* $\mathcal{C}_N, \mathcal{C}_{N,1}, \mathcal{C}_{N,2}$ *are contract terms.*
- *The support of a contract term* $\mathcal{C}_N$*, denoted by* $supp(\mathcal{C}_N)$ *is the set of contracts used to define the term. The support is defined recursively as follows:* $supp(\mathcal{C}) = \{\mathcal{C}\}$*,* $supp(\mathcal{C}_{N,1} \wedge \mathcal{C}_{N,2}) = supp(\mathcal{C}_{N,1} \otimes \mathcal{C}_{N,2}) = supp(\mathcal{C}_{N,1}) \cup supp(\mathcal{C}_{N,2})$*.*
- *A* contract network *is a pair* $N = (\mathbb{C}_N, \mathcal{C}_N)$ *where* $\mathcal{C}_N$ *is a contract term,* $\mathbb{C}_N$ *is its support, and each contract in the support is used in the term exactly once.*

Intuitively, a CN is a set of contracts connected via conjunction and composition operations, used to represent the combination of different operational viewpoints on the same component (conjunction) or requirements associated with different components (composition). We now define heterogeneous hierarchical contract networks, which formalize a requirement hierarchy in terms of refinement relations between the contract terms of pairs of contract networks.

Definition 5 (Heterogeneous Hierarchical Contract Network)**.** *A* Heterogeneous Hierarchical Contract Network (HHCN) *with* L *levels is a graph* $\mathcal{G} = (\mathbf{N}, \mathbf{E})$*, where* $\mathbf{N}$ *is a set of*

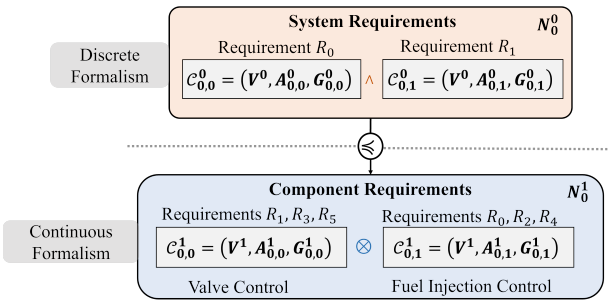


Fig. 3: HHCN $\mathcal{G}_{tank}$ for the system in Example 1.

contract networks and $\mathbf{E}$ *is a set of directed edges, such that the following properties hold:*

- $\mathbf{N}$ *is partitioned into* L *nonempty subsets* $\mathbf{N}^0, \ldots, \mathbf{N}^{L-1}$*, where* $\mathbf{N}^0$ *is a singleton set, i.e.,* $\mathbf{N}^0 = \{N_0^0\}$.
- $\mathbf{E}$ *consists only of edges between sets* $\mathbf{N}^l$ *and* $\mathbf{N}^{l+1}$*,* $l = 0, \ldots, L-2$*, that is,* $\mathbf{E} = \{(N_i^l, N_j^{l+1}) : N_i^l \in \mathbf{N}^l, N_j^{l+1} \in \mathbf{N}^{l+1}, l = 0, \ldots, L-2\}$*. Edge* (N_i^l, N_j^{l+1}) *denotes that the contract term of* N_i^l refines *that of* N_j^{l+1}.

We denote by $\mathbb{C}_{N,i}^l$ *the support set of contract network* $N_i^l \in \mathbf{N}^l$ *defined in formalism* $\mathcal{B}^l$.

An illustration of an HHCN is shown in Figure 2. Intuitively, an HHCN captures a layered set of requirements, associated with different levels of abstraction. The operations $\{\wedge, \otimes\}$ between contracts in $\mathcal{C}_N^l$ may be used to capture the system architecture and the allocation of the system specifications to the system components.

We now construct the HHCN for Example 1 as shown in Figure 3. The system-level requirements are organized into the contract network N_0^0 over the integer variables $V^0 = \{h_d, \mu_v, \mu_f\}$, where h_d is the fuel level, and μ_v and μ_f encode the *on* and *off* states of the valve and fuel injector, respectively. Requirement R_0 ensures that, whenever $h_d \in \{0, \ldots, 4\}$, the valve remains *off*, i.e., $\mu_v = 0$. Thus, we write contract $\mathcal{C}_{0,0}^0 = (V^0, A_{0,0}^0, G_{0,0}^0)$ where $A_{0,0}^0 := (h_d \in \{0, \ldots, 4\})$ and $G_{0,0}^0 := (\mu_v = 0)$. We encode the *on* state of the fuel injector with $\mu_f = 5$. As a result, $\mathcal{C}_{0,1}^0$ has the assumptions and guarantees $A_{0,1}^0 := (h_d \in \{0, \ldots, 4\})$ and $G_{0,1}^0 := (\mu_f = 5)$. $\mathcal{C}_{0,0}^0$ and $\mathcal{C}_{0,1}^0$ must be simultaneously satisfied by the system; therefore, they combine via conjunction.

At the component level, we define contracts capturing the requirements on the valve and fuel injection control in N_0^1 over the variables $V^1 = \{h_c, \mathrm{V}_v, \mathrm{V}_f\}$, including the physical height of the fuel column, the voltage applied to the valve and the voltage applied to the fuel injector (in Volts), respectively. To model R_2 and R_3, we take $V_{min}^{off} = 0$ and $V_{max}^{off} = 1$. To model R_4 and R_5, we take $V_{min}^{on} = 5$ and $V_{max}^{on} = 6$. As shown in Figure 3, we model the requirement on the valve controller with contract $\mathcal{C}_{0,0}^1 = (V^1, A_{0,0}^1, G_{0,0}^1)$, where $A_{0,0}^1 := (0 \leq h_c < 5)$ and $G_{0,0}^1 := (0 \leq \mathrm{V}_v < 1)$. The requirement on the fuel injection controller is captured by contract $\mathcal{C}_{0,1}^1 = (V^1, A_{0,1}^1, G_{0,1}^1)$, where $A_{0,1}^1 := (0 \leq h_c < 5)$ and $G_{0,1}^1 := (5 \leq \mathrm{V}_f < 6)$. Contracts modeling the valve controller and the fuel injection controller describe the operation of two parallel control loops; therefore, they combine by composition.

The behavior of the overarching control algorithm is encoded by contract term $\mathcal{C}_{N,0}^0 := \mathcal{C}_{0,0}^0 \wedge \mathcal{C}_{0,1}^0$. Similarly, contract term $\mathcal{C}_{N,0}^1 := \mathcal{C}_{0,0}^1 \otimes \mathcal{C}_{0,1}^1$ specifies the behaviors of the fuel injection and valve controllers. We ensure the existence of feasible implementations by requiring that the assumptions and guarantees of $\mathcal{C}_{N,0}^0$ and $\mathcal{C}_{N,0}^1$ are non-empty, therefore, the contracts are *consistent* and *compatible*. For traceability to hold, we would like to conclude that $\mathcal{C}_{N,0}^1$ *refines* $\mathcal{C}_{N,0}^0$. Generalizing, we state the *HHCN traceability validation problem* as follows.

Problem 1 (HHCN Traceability Validation)**.** *Given an HHCN* $\mathcal{G}$*,* $\forall\, l \in \{0, \ldots, L-1\}, \forall\, i, j \in \{0, \ldots, |\mathbf{N}^l| - 1\}$*, prove the following properties:*

- *The contract term* $\mathcal{C}_{N,i}^l$ *is* consistent *and* compatible*;*
- *If* $E = (N_i^l, N_j^{l+1}) \in \mathbf{E}$*, then* $\mathcal{C}_{N,j}^{l+1}$ *refines* $\mathcal{C}_{N,i}^l$*, where* $\mathcal{C}_{N,i}^l$ *and* $\mathcal{C}_{N,j}^{l+1}$ *are the contract terms corresponding to the CNs* N_i^l *and* N_j^{l+1}*. In other words, all refinement relations hold along the edges of* $\mathcal{G}$*.*

If the contract networks in $\mathbf{N}^{L-1}$ reduce to a representation of the system implementation, then traceability validation includes the *system verification problem*, stated as follows.

Problem 2 (System Requirement Satisfaction)**.** *Given a system implementation* M*, verify whether it satisfies all the contract terms in* $\mathbf{N}^{L-1}$*, i.e., verify that* $\forall\ \mathcal{C}_{N,i}^{L-1},\ i \in \{0, \ldots, |\mathbf{N}^{L-1}| - 1\},\ M \models \mathcal{C}_{N,i}^{L-1}$.

Traceability validation and system verification when all the contracts in an HCN are expressed using the same formalism have been explored in the literature [9], [10], [32]. In the sequel, we focus on the problem of establishing refinement relationships across levels of abstraction associated with heterogeneous formalisms.

V. Contract Embeddings

Contract embeddings help establish refinement relationships between contract terms on adjacent levels of an HHCN. As shown in Figure 4, a conservative approximation Ψ is used to define an *abstract embedding* $\curlywedge_\Psi$ and a *concrete embedding* $\curlyvee_\Psi$. These are used to map a concrete contract $\mathcal{C}_N$ upwards in the abstract formalism, and to map an abstract contract $\mathcal{C}'_N$ downwards in the concrete formalism, respectively, so that we can transform the refinement checking problem between heterogeneous contracts into a standard refinement checking problem between homogeneous contracts.

A. Abstract and Concrete Embeddings

We define a conservative approximation Ψ by first identifying the sets $\mathbf{B}$ and $\mathbf{B}'$ (see Definition 3) that are used to construct contract terms in formalisms $\mathcal{B}$ and $\mathcal{B}'$.

Consider the set of variables V of a contract term $\mathcal{C}_N$. We call $\mathrm{S}(V)$ the *value space* of these variables, i.e., the set of all possible values of the variables in V. Next, we define the universe of possible behaviors $\mathbf{B}(V)$ *supported* by the value space $\mathrm{S}(V)$. Intuitively, $\mathbf{B}(V)$ is a *set of sets*, i.e., the set containing all possible sets of behaviors which may be used to write contracts in the formalism $\mathcal{B}$.

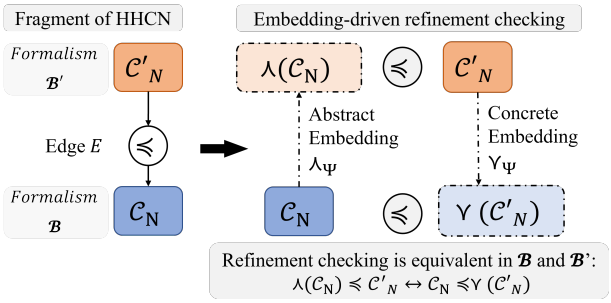


Fig. 4: Embedding-driven refinement checking for traceability.

Given two contracts $\mathcal{C}_1 = (V, A_1, G_1)$ and $\mathcal{C}_2 = (V, A_2, G_2)$ such that $A_1, A_2, G_1, G_2 \in \mathbf{B}(V)$, the assumptions and guarantees of $\mathcal{C}_\wedge = \mathcal{C}_1 \wedge \mathcal{C}_2$ and $\mathcal{C}_\otimes = \mathcal{C}_1 \otimes \mathcal{C}_2$ should also be representable in $\mathbf{B}(V)$. Therefore, we require $\mathbf{B}(V)$ to be closed under *unions* and *intersections*, that is, $\forall A_1, A_2 \in \mathbf{B}(V)$, $A_1 \cap A_2 \in \mathbf{B}(V)$ and $A_1 \cup A_2 \in \mathbf{B}(V)$. Moreover, we require $\emptyset \in \mathbf{B}(V)$. Define $\Omega = \bigcup_{A \in \mathbf{B}(V)} A$ as the collection of all possible behaviors in formalism $\mathcal{B}$. We impose the condition that $\mathbf{B}(V)$ be closed under *complements* ($^-$) with respect to Ω, i.e., $\forall A \in \mathbf{B}(V)$, $\Omega \setminus A \in \mathbf{B}(V)$. In particular, as a σ-algebra possesses these desirable properties [33], possible choices of $\mathbf{B}(V)$ include σ-algebras defined over the sets of behaviors in formalism $\mathcal{B}$.

Two elements in $\mathbf{B}(V)$ and $\mathbf{B}(V')$ can be compared with the *subset* relation $\subseteq$, which is a partial order. Between two partially ordered sets $(\mathbf{B}(V), \subseteq)$, $(\mathbf{B}(V'), \subseteq)$, we can define functions $\alpha_u, \alpha_l : \mathbf{B}(V) \to \mathbf{B}(V')$ and $\gamma_u, \gamma_l : \mathbf{B}(V') \to \mathbf{B}(V)$ which satisfy Definition 3 to create a conservative approximation Ψ and use it to transfer contracts across $\mathcal{B}$ and $\mathcal{B}'$. Further details on this construction may be found in the literature [9].

Example 2. *In the HHCN $\mathcal{G}_{tank}$ for Example 1, the value space of $V^0 = \{h_d, \mu_v, \mu_f\}$ is $\mathrm{S}(V^0) = \mathbb{Z} \times \mathbb{V} \times \mathbb{V}$, where $\mathbb{V} = \{0, 5\}$ is the set of the possible values for μ_v and μ_f. The value space of $V^1 = \{h_c, \mathrm{V}_v, \mathrm{V}_f\}$ is, instead, $\mathrm{S}(V^1) = \mathbb{R}^3$. These spaces support the set of possible behaviors $\mathbf{B}(V^0) = 2^{\mathbb{Z} \times \mathbb{V} \times \mathbb{V}}$ and $\mathbf{B}(V^1) = \sigma(\mathbb{R}^3)$, i.e., the Borel sigma algebra on $\mathbb{R}^3$ [33]. Between $(\mathbf{B}(V^0), \subseteq)$ and $(\mathbf{B}(V^1), \subseteq)$ we can construct a conservative approximation Ψ_C^D by defining functions $\alpha_u, \alpha_l, \gamma_u, \gamma_l$ as follows [29], [34]:*

$$\begin{aligned} \alpha_u(A) &:= g(A), \\ \alpha_l(A) &:= g(A) \setminus g(B(V^1) \setminus A), \\ \gamma_u(A') = \gamma_l(A') &:= g^{-1}(A'), \end{aligned} \tag{2}$$

where $g = \lfloor . \rfloor$ is the floor function, $g(A)$ and $g^{-1}(A)$ are the image *and* preimage *of A under g, and $B(V^1) \subseteq \mathbf{B}(V^1)$ is the set of all the behaviors whose image under g is contained in $g(A)$, i.e., $B(V^1) = \bigcup_X \{X \subseteq \mathbf{B}(V^1) : g(X) \subseteq g(A)\}$. By Definition 3, $\Psi_C^D = (\alpha_l, \alpha_u)$ is a conservative approximation between the continuous set $\mathbf{B}(V^1)$ and the discrete set $\mathbf{B}(V^0)$.*

In Example 2, many behaviors in $\mathbf{B}(V^1)$ may map to a single behavior in $\mathbf{B}(V^0)$. For example, $A_1 = [1.9, 3)$ and $A_2 = [1.7, 3)$ map to the same set $\alpha_u(A_1) = \alpha_u(A_2) = \{1, 2\}$. However, the converse is not true. Therefore, $\mathcal{B}^0$ is the *abstract* formalism and $\mathcal{B}^1$ is the *concrete* formalism.

Given a conservative approximation Ψ between the set of possible behaviors in the concrete domain $\mathbf{B}(V)$ and the abstract domain $\mathbf{B}(V')$, we define *abstract and concrete embeddings* between contract terms $\mathcal{C}_N$ and $\mathcal{C}'_N$.

Definition 6 (Contract Embeddings)**.** *Let $\mathcal{C}_N = (V, A, G)$ and $\mathcal{C}'_N = (V', A', G')$ be contract terms such that $A, G \in \mathbf{B}(V)$ and $A', G' \in \mathbf{B}(V')$. Let $\Psi = (\alpha_l, \alpha_u)$ be a conservative approximation consisting of Galois connections $\psi_1 = \langle \alpha_u, \gamma_u \rangle$, defined from $\mathbf{B}(V)$ to $\mathbf{B}(V')$, and $\psi_2 = \langle \gamma_l, \alpha_l \rangle$, defined from $\mathbf{B}(V')$ to $\mathbf{B}(V)$. The* abstract embedding *of $\mathcal{C}_N$, written $\curlywedge_\Psi(\mathcal{C}_N)$, is given by*

$$\curlywedge_\Psi(\mathcal{C}_N) = (V', \alpha_l(A), \alpha_u(G)). \tag{3}$$

The concrete embedding *of $\mathcal{C}'$, written $\curlyvee_\Psi(\mathcal{C}')$, is given by*

$$\curlyvee_\Psi(\mathcal{C}'_N) = (V, \gamma_l(A'), \gamma_u(G')). \tag{4}$$

For brevity, we write $\curlyvee_\Psi$ and $\curlywedge_\Psi$ as $\curlyvee$ and $\curlywedge$, unless required by the context. Contract embeddings preserve the outcome of refinement checking across formalisms, that is, the refinement checking operation, performed either in the abstract formalism $\mathcal{B}'$ or the concrete formalism $\mathcal{B}$, as shown in Figure 4, produces the same outcome. This property is proved in the following result.

Theorem 2. *Let $\mathcal{C}_N$ and $\mathcal{C}'_N$ be two contract terms as in Definition 6. The abstract embedding of $\mathcal{C}_N$ refines $\mathcal{C}'_N$ if and only if $\mathcal{C}_N$ refines the concrete embedding of $\mathcal{C}'_N$. That is,*

$$\curlywedge(\mathcal{C}_N) \preceq \mathcal{C}'_N \longleftrightarrow \mathcal{C}_N \preceq \curlyvee(\mathcal{C}'_N). \tag{5}$$

Proof. From Definition 6, we obtain $\curlywedge(\mathcal{C}_N) = (V', \alpha_l(A), \alpha_u(G))$ and $\curlyvee(\mathcal{C}'_N) = (V, \gamma_l(A'), \gamma_u(G'))$. Thus, by definition, $\curlywedge(\mathcal{C}) \preceq \mathcal{C}'$ means $\alpha_l(A) \supseteq A'$, $\alpha_u(G) \subseteq G'$. By Definition 3, since $\langle \gamma_l, \alpha_l \rangle$ is a Galois connection from $\mathbf{B}(V')$ to $\mathbf{B}(V)$, we have $\forall A' \in \mathbf{B}(V')$, $A \in \mathbf{B}(V)$:

$$\gamma_l(A') \subseteq A \leftrightarrow A' \subseteq \alpha_l(A). \tag{6}$$

Similarly, $\langle \alpha_u, \gamma_u \rangle$ forms a Galois connection from $\mathbf{B}(V)$ to $\mathbf{B}(V')$. Therefore, $\forall G \in \mathbf{B}(V)$, $G' \in \mathbf{B}(V')$, we have

$$\alpha_u(G) \subseteq G' \leftrightarrow G \subseteq \gamma_u(G'). \tag{7}$$

Therefore, combining these two equations, we obtain $(\alpha_l(A) \supseteq A', \alpha_u(G) \subseteq G') \leftrightarrow (A \supseteq \gamma_l(A'), G \subseteq \gamma_u(G'))$, i.e., $\curlywedge(\mathcal{C}_N) \preceq \mathcal{C}'_N \longleftrightarrow \mathcal{C}_N \preceq \curlyvee(\mathcal{C}'_N)$. □

We illustrate this theorem on our running example and show the traceability validation of the requirement set $\mathcal{R}$.

Example 3. *Consider checking the refinement between contracts $\mathcal{C}^0_{N,0}$ and $\mathcal{C}^1_{N,0}$, shown in Figure 3, computed as follows:*

$A^0_{N,0} = (h_d \in \{0, \ldots, 4\})$	$A^1_{N,0} = (0 \leq h_c < 5)$
$G^0_{N,0} = (h_d \in \{0, \ldots, 4\}) \to$ $((\mu_v = 0) \wedge (\mu_f = 5))$	$G^1_{N,0} = (0 \leq h_c < 5) \to$ $\big((0 \leq \mathrm{V}_v < 1) \wedge (5 \leq \mathrm{V}_f < 6)\big)$

We perform refinement checking in both the formalisms $\mathcal{B}^0$ and $\mathcal{B}^1$, assuming an obvious mapping of variables in V^1 to variables in V^0: h_c to h_d, V_v to μ_v, and V_f to μ_f.

In the discrete formalism, we compute $\alpha_l(A^1_{N,0})$ *and* $\alpha_u(G^1_{N,0})$. *Let* $A_h = [0,5)$. *We have* $g(A_h) = \lfloor [0,5) \rfloor = \{0,\dots,4\}$ *and, from* (2), $\alpha_l(A^1_{N,0}) = g(A_h) \backslash g(B(V^1) \backslash A_h)$. *We have* $B(V^1) = \bigcup_X \{X \subseteq \mathbf{B}(V^1) : g(X) \subseteq \{0,\dots,4\}\} = [0,5)$, *that is,* $B(V^1) = [0,5)$ *is the largest possible interval whose image under the floor function equals* $\{0,\dots,4\}$. *We then obtain* $g(B(V^1) \backslash A_h) = g([0,5) \backslash [0,5)) = g(\emptyset) = \emptyset$. *Therefore,* $\alpha_l(A^1_{N,0}) = \{0,\dots,4\} \backslash \emptyset = \{0,\dots,4\}$.

Next, we find $\alpha_u(G^1_{N,0})$. *From the definition above, the expression for* $G^1_{N,0}$ *is equivalent to* $h_c \notin [0,5) \vee (\mathrm{V}_v \in [0,1) \wedge \mathrm{V}_f \in [5,6))$. *By applying* α_u *to each term in* $G^1_{N,0}$ *and noting that the image distributes under the disjunction of sets, we obtain:*

$$\begin{aligned}&\alpha_u(h_c \in (-\infty,0) \cup [5,\infty)) = h_d \in \lfloor (-\infty,0) \rfloor \cup \lfloor [5,\infty) \rfloor \\ &= h_d \in \{\dots,-2,-1\} \cup \{5,6,\dots\} = h_d \notin \{0,\dots,4\}.\end{aligned}$$

We can compute $\alpha_u(\mathrm{V}_v \in [0,1) \wedge \mathrm{V}_f \in [5,6))$ *in a similar way. Combining these expressions, we obtain* $\alpha_u(G^1_{N,0}) = (h_d \notin \{0,\dots,4\}) \vee \big((\mu_v = 0) \wedge (\mu_f = 5)\big)$ *and, for the abstract embedding of the contract* $\mathcal{C}^1_{N,0}$,

$$\begin{aligned}&\alpha_l(A^1_{N,0}) = (h_d \in \{0,\dots,4\}) \\ &\alpha_u(G^1_{N,0}) = (h_d \notin \{0,\dots,4\}) \vee \big((\mu_v = 0) \wedge (\mu_f = 5)\big).\end{aligned} \tag{8}$$

Because $A^0_{N,0} = (h_d \in \{0,\dots,4\})$ *and* $G^0_{N,0}$ *can be written as* $G^0_{N,0} = (h_d \notin \{0,\dots,4\}) \vee \big((\mu_v = 0) \wedge (\mu_f = 5)\big)$, *after variable mapping, we observe that* $\alpha_l(A^1_{N,0}) \supseteq A^0_{N,0}$ *and* $\alpha_u(G^0_{N,1}) \subseteq G^0_{N,0}$, *which prove that* $\curlywedge(\mathcal{C}^1_{N,0}) \preceq \mathcal{C}^0_{N,0}$.

In the converse direction, we have $\gamma_u = \gamma_l = g^{-1}(A')$ *for any set* $A' \in \mathcal{B}^0$. *To compute* $\gamma_l(A^0_{N,0})$, *the preimage of the set* $A'_{h_d} = \{0,\dots 4\}$ *under the floor function is* $\gamma_l(A'_{h_d}) = [0,5)$, *therefore,* $\gamma_l(A^0_{N,0}) = (0 \le h_c < 5)$. *We can compute* $\gamma_u(G^0_{N,0})$ *in a similar way, by noting the ranges* $A'_{h_d} = \{0,\dots,4\}$, $A'_v = \{0\}$, *and* $A'_f = \{5\}$, *for the variables in* $G^0_{N,0}$. *Thus, we obtain* $\gamma_l(A'_{h_d}) = [0,5)$, $\gamma_l(A'_v) = [0,1)$, *and* $\gamma_l(A'_f) = [5,6)$. *Combining these expressions, we find* $\gamma_u(G^0_{N,0}) = (h_c \notin [0,5)) \vee (\mathrm{V}_v \in [0,1) \wedge \mathrm{V}_f \in [5,6))$. *Again, we can observe that* $A^1_{N,0} \supseteq \gamma_l(A^0_{N,0})$ *and* $G^1_{N,0} \subseteq \gamma_u(G^0_{N,0})$, *i.e.,* $\mathcal{C}^1_{N,0} \preceq \curlyvee(\mathcal{C}^0_{N,0})$. *Finally, we obtain* $\curlywedge(\mathcal{C}^1_{N,0}) \preceq \mathcal{C}^0_{N,0} \leftrightarrow \mathcal{C}^1_{N,0} \preceq \curlyvee(\mathcal{C}^0_{N,0})$, *consistently with Theorem 2. The traceability between system requirements and component-level requirements is validated.*

We note that the traceability validation problem can be solved algorithmically and can be automated. Contract embeddings need not be computed every time, but can be instantiated from a library of predefined functions and encoding templates, which enhances the scalability of the approach. Further, the refinement checking problems can be formulated as satisfiability modulo theory (SMT) problems for appropriate theories, e.g., the theory of linear integer arithmetic [35] for checking $\curlywedge(\mathcal{C}^1_{N,0}) \preceq \mathcal{C}^0_{N,0}$ *in this example.*

B. Properties of Contract Embeddings

The bidirectional implication in (5) resembles (1). Moreover, as discussed below, contract embeddings enjoy a set of properties that are analogous to the properties stated for a Galois connection in Theorem 1. We then term the pair of embeddings $\langle \curlywedge, \curlyvee \rangle$ in (5) a *quasi-Galois connection* between contracts in formalisms $\mathcal{B}$ and $\mathcal{B}'$. In fact, in this section, we show that such a quasi-Galois connection does *not* coincide, in general, with a Galois connection. We start by recalling the properties of the Galois connections associated with a conservative approximation, used in our embeddings.

Proposition 1. *Let* $\Psi = (\alpha_l, \alpha_u)$ *be a conservative approximation composed of Galois connections* $\psi_1 = \langle \alpha_u, \gamma_u \rangle$, *from* $\mathbf{B}(V)$ *to* $\mathbf{B}(V')$, *and* $\psi_2 = \langle \gamma_l, \alpha_l \rangle$, *from* $\mathbf{B}(V')$ *to* $\mathbf{B}(V)$. *Then,* $\forall A \in \mathbf{B}(V)$ *and* $\forall A' \in \mathbf{B}(V')$,

- α_u, α_l, γ_u, *and* γ_l *are all monotone functions;*
- $A \subseteq \gamma_u \circ \alpha_u(A)$ *and* $\alpha_u \circ \gamma_u(A') \subseteq A'$;
- $A' \subseteq \alpha_l \circ \gamma_l(A')$ *and* $\gamma_l \circ \alpha_l(A) \subseteq A$.

Proof. The proof of the first two statements proceeds straightforwardly from Theorem 1. The third statement may be obtained by renaming the functions in Theorem 1 and noting that $\psi_2 = \langle \gamma_l, \alpha_l \rangle$ is defined from $\mathbf{B}(V')$ to $\mathbf{B}(V)$. □

Once we define functions $\alpha_l, \alpha_u, \gamma_l, \gamma_u$ which constitute a conservative approximation Ψ, we can construct contract embeddings $\curlywedge, \curlyvee$ with the following properties.

Theorem 3. *The following properties hold for* $\curlywedge$ *and* $\curlyvee$:

1) $\curlywedge$ *and* $\curlyvee$ *preserve contract consistency and compatibility, i.e., if* $\mathcal{C}_N$ *is consistent (compatible), then so is* $\curlywedge(\mathcal{C}_N)$. *The same holds for* $\curlyvee(\mathcal{C}'_N)$.
2) $\curlywedge$ *and* $\curlyvee$ *are* monotone, *that is, if* $\mathcal{C}_{N,1} \preceq \mathcal{C}_{N,2}$, $\mathcal{C}'_{N,1} \preceq \mathcal{C}'_{N,2}$, *then* $\curlywedge(\mathcal{C}_{N,1}) \preceq \curlywedge(\mathcal{C}_{N,2})$ *and* $\curlyvee(\mathcal{C}'_{N,1}) \preceq \curlyvee(\mathcal{C}'_{N,2})$.
3) $\mathcal{C}_N \preceq \curlyvee(\curlywedge(\mathcal{C}_N))$ *and* $\curlywedge(\curlyvee(\mathcal{C}'_N)) \preceq \mathcal{C}'_N$.

Proof. Property (1) can be proved by using the property of Galois connections [31]. If A is a nonempty set, then $\alpha_u(A)$ is nonempty. Similar statements can be made for $\alpha_l, \gamma_l, \gamma_u$. Therefore, if the assumptions A are nonempty, the contract is compatible, and so is the abstract embedding $\curlywedge(\mathcal{C})$. Similarly, if the guarantees G are nonempty, then the contract is consistent as is $\curlywedge(\mathcal{C})$.

We can prove (2) by noting that $\alpha_u, \gamma_u, \gamma_l$, and α_l are all monotone functions (by Theorem 1), that is, they preserve set inclusion. Thus, by $\mathcal{C}_{N,1} \preceq \mathcal{C}_{N,2}$, we obtain $A_1 \supseteq A_2$ and $G_1 \subseteq G_2$. Then, by monotonicity, we obtain $\alpha_l(A_1) \supseteq \alpha_l(A_2)$ and $\alpha_u(G_1) \subseteq \alpha_u(G_2)$, that is, $\curlywedge(\mathcal{C}_1) \preceq \curlywedge(\mathcal{C}_2)$. Similarly, $\mathcal{C}'_1 \preceq \mathcal{C}'_2 \to \curlyvee(\mathcal{C}'_1) \preceq \curlyvee(\mathcal{C}'_2)$.

To prove (3), note that $\curlywedge(\curlyvee(\mathcal{C}_N)) = (V, \gamma_l(\alpha_l(A)), \gamma_u(\alpha_u(G)))$. From Proposition 1, we get $\gamma_l \circ \alpha_l(A) \subseteq A$ and $G \subseteq \gamma_u \circ \alpha_u(G)$. Therefore, by definition, $\mathcal{C} \preceq \curlyvee(\curlywedge(\mathcal{C}))$. Similarly, $\curlywedge(\curlyvee(\mathcal{C}'_N)) = (V', \alpha_l(\gamma_l(A')), \alpha_u(\gamma_u(G')))$. From Proposition 1, we get $\alpha_l(\gamma_l(A')) \supseteq A$ and $\alpha_u(\gamma_u(G')) \subseteq G'$, and therefore, $\curlywedge(\curlyvee(\mathcal{C}'_N)) \preceq \mathcal{C}'_N$. □

Despite the resemblance between (5) and (1) and between Theorem 3 and Theorem 1, $\langle \curlywedge, \curlyvee \rangle$ in (5) only form a *quasi-Galois connection*, that is, for the functions $\alpha_l, \alpha_u, \gamma_l, \gamma_u$, it is necessary that at least one of $\alpha_u \neq \alpha_l$ and $\gamma_u \neq \gamma_l$ hold. If we require $\alpha_u = \alpha_l = \alpha$ and $\gamma_u = \gamma_l = \gamma$ simultaneously, leading to only one Galois connection $\langle \alpha, \gamma \rangle$ to construct contract abstractions and concretizations of the

form $\bar{\alpha}(\mathcal{C}_N) = (V', \alpha(A), \alpha(G))$ and $\bar{\gamma} = (V, \gamma(A'), \gamma(G'))$, respectively, then Theorems 2 and 3 may not hold, in general, and $\langle \bar{\alpha}, \bar{\gamma} \rangle$ may not satisfy (5). To prove this result formally, we start by recalling the following basic result stating that inverse relations between sets preserve set inclusion [36].

Proposition 2. *Let $\mathcal{S}$ and $\mathcal{S}'$ be any two sets. Let $f_1 \subseteq \mathcal{S} \times \mathcal{S}'$ and $f_2 \subseteq \mathcal{S} \times \mathcal{S}'$ be two relations, and f_1^{-1} and f_2^{-1} be the corresponding inverse relations. If $f_1 \subseteq f_2$, then, $f_1^{-1} \subseteq f_2^{-1}$.*

We now state a corollary of this proposition.

Proposition 3. *Let $\alpha_l : \mathbf{B}(V) \to \mathbf{B}(V')$ and $\alpha_u : \mathbf{B}(V) \to \mathbf{B}(V')$ be functions such that $\forall A \in \mathbf{B}(V), \alpha_l(A) \subseteq \alpha_u(A)$. If both α_l, α_u are also* bijections*, then, $\forall A' \in \mathbf{B}(V')$, we have $\alpha_l^{-1}(A') \subseteq \alpha_u^{-1}(A')$. Similarly, let $\gamma_u, \gamma_l : \mathbf{B}(V') \to \mathbf{B}(V)$ be such that $\forall A' \in \mathbf{B}(V'), \gamma_u(A') \subseteq \gamma_l(A')$. If both γ_u, γ_l are also bijections, then, $\forall A \in \mathbf{B}(V)$, $\gamma_u^{-1}(A) \subseteq \gamma_l^{-1}(A)$.*

We are now ready to prove the main result of this section, i.e., a quasi-Galois connection on contracts coincides with a Galois connection only under very stringent conditions on the constituent functions and the sets of behaviors $\mathbf{B}(V)$ and $\mathbf{B}(V')$. Specifically, requiring that the quasi-Galois connection only use two functions, α and γ, is equivalent to requiring the existence of an isomorphism between $\mathbf{B}(V)$ and $\mathbf{B}(V')$.

Theorem 4. *Let a conservative approximation $\Psi = (\alpha_l, \alpha_u)$ from $\mathcal{B}$ to $\mathcal{B}'$ be composed of Galois connections $\psi_1 = \langle \alpha_u, \gamma_u \rangle$ and $\psi_2 = \langle \gamma_l, \alpha_l \rangle$. Then, we obtain that $\alpha_u = \alpha_l = \alpha$ and $\gamma_u = \gamma_l = \gamma$ simultaneously if and only if all of these functions are bijections and $\alpha_u^{-1} = \gamma_u$, $\gamma_u^{-1} = \alpha_u$, $\alpha_l^{-1} = \gamma_l$, and $\gamma_l^{-1} = \alpha_l$ hold. Moreover, because α and γ are monotone, $\alpha_u = \alpha_l$ and $\gamma_u = \gamma_l$ hold simultaneously if and only if both $\alpha : \mathbf{B}(V) \to \mathbf{B}(V')$ and $\gamma : \mathbf{B}(V') \to \mathbf{B}(V)$ are* isomorphisms*.*

Proof. From Definition 3, if $\alpha_u = \alpha_l = \alpha$ and $\gamma_u = \gamma_l = \gamma$, we get that both $\langle \alpha, \gamma \rangle$ from $\mathcal{B}$ to $\mathcal{B}'$ and $\langle \gamma, \alpha \rangle$ from $\mathcal{B}'$ to $\mathcal{B}$ are Galois connections. Then, from Proposition 1, we obtain $A' \subseteq \alpha \circ \gamma(A')$ and $\alpha \circ \gamma(A') \subseteq A'$, that is, $\forall A' \in \mathbf{B}(V'), A' = \alpha \circ \gamma(A')$. In a similar way, we get $A \subseteq \gamma \circ \alpha(A)$ and $\gamma \circ \alpha(A) \subseteq A$, thus, $\forall A \in \mathbf{B}(V), \gamma \circ \alpha(A) = A$. From the properties of the composition of images under a function [37] and the relation $\forall A' \in \mathbf{B}(V'), A' = \alpha \circ \gamma(A')$, we obtain that α is an injection and γ is a surjection. From the other relation $\forall A \in \mathbf{B}(V), \gamma \circ \alpha(A) = A$, we obtain that γ is an injection and α is a surjection. As α, γ are both injections and surjections, they are bijections. Thus, they are both invertible. Then, $A' = \alpha \circ \gamma(A')$ implies that $\alpha = \gamma^{-1}$ and $\gamma \circ \alpha(A) = A$ implies $\gamma = \alpha^{-1}$. Moreover, because the functions α, γ are both monotone and bijective, they are an isomorphism.

In the converse direction, from Proposition 3, note that $\gamma_u^{-1}(A) \subseteq \gamma_l^{-1}(A)$. Because we are given $\gamma_l^{-1} = \alpha_l$ and $\gamma_u^{-1} = \alpha_u$, we get $\alpha_u(A) \subseteq \alpha_l(A)$. However, from Definition 3, $\alpha_l(A) \subseteq \alpha_u(A)$. Therefore, we can conclude that $\forall A \in \mathbf{B}(V)$, $\alpha_u(A) = \alpha_l(A)$. In other words, $\alpha_u = \alpha_l = \alpha$. We can use a similar argument to show that $\gamma_u = \gamma_l = \gamma$. □

In summary, the contract embeddings form a quasi-Galois connection, which coincides with a Galois connection if and only if there exists an isomorphism between the possible set of behaviors in the abstract and concrete formalisms. A pair of distinct Galois connections, composing a conservative approximation is *necessary* to define non-trivial contract abstractions and concretizations. We exploit embeddings to solve the traceability validation and system verification problems.

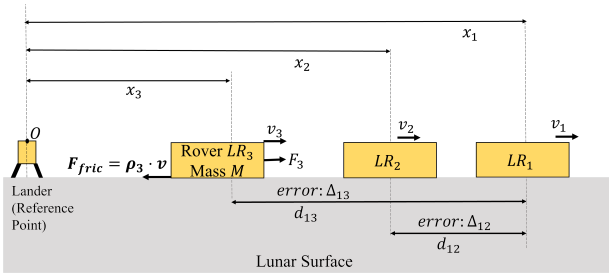


Fig. 5: Illustration of a multi-agent lunar mission

C. HHCN Traceability Validation

We solve Problem 1 by first checking the consistency and compatibility of each contract network in the HHCN $\mathcal{G}$ using methods developed in the literature. In fact, on each level of abstraction, depending on the nature of the formalism, we can reason about contract consistency, compatibility, and refinement, using different techniques, e.g., model checking [13], [38], simulation-based falsification [32], [39], or satisfiability modulo theories [21], [35]. Next, we validate the cross-level refinement by constructing the abstract or concrete embeddings and checking refinement between contract terms on adjacent levels in the formalism (abstract or concrete) suitable to the problem at hand.

For example, given an HHCN $\mathcal{G}$ and contract network $N_k^{L-1} \in \mathbf{N}^{L-1}$, we can abstract it to the level $L-2$ by computing the abstract embedding of the contract term $\mathcal{C}_{N,k}^{L-1}$ given by $\curlywedge_{\Psi_{L-1}^{L-2}}(\mathcal{C}_{N,k}^{L-1})$. Then, we check that this abstract embedding refines the contract terms of the parents of N_k^{L-1} using tools suitable for the formalism $\mathcal{B}^{L-2}$. If refinement is established for every parent-child pair, then traceability is guaranteed, as shown in Example 3. A similar approach can be used to solve Problem 2.

VI. Case Study

We consider a requirement hierarchy inspired from future lunar missions featuring multi-agent, autonomous rovers [40]. As shown in Figure 5, three identical rovers, LR_1, LR_2, LR_3, with mass M, drive in a straight line on the lunar surface led by rover LR_1. The mission-level requirements, shown in Figure 6, stipulate each rover to autonomously drive for a predetermined distance D from their starting position in a straight line to their destination. The team leader LR_1 is ahead of the other two followers, LR_2 and LR_3, and serves as the reference point for the inter-rover distances d_{12} and d_{13}. The requirements also specify the driving direction of the rovers and require them to stop upon reaching their destinations.

These mission-level requirements are specialized into the *system kinematic requirements*, which impose an upper bound

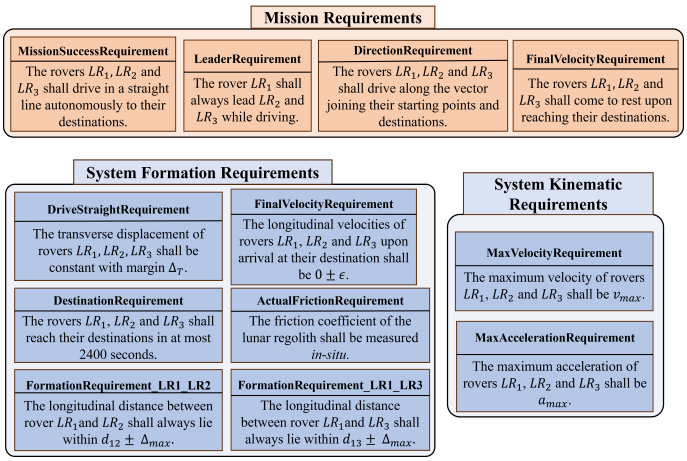


Fig. 6: Requirement hierarchy including mission-level and system-level layers in a multiagent lunar mission [40].

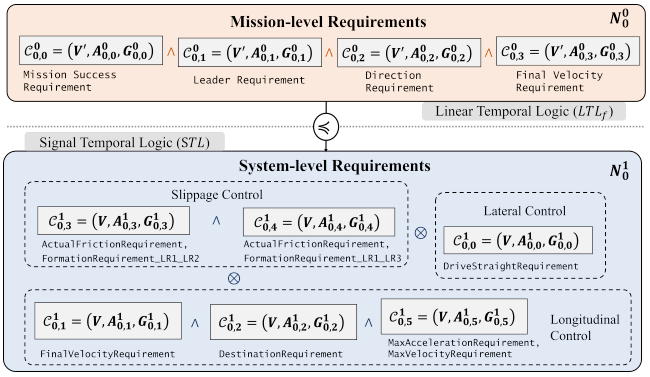


Fig. 7: HHCN $\mathcal{G}_{rover}$ for the requirement set $\mathcal{R}$ in Figure 6.

on the velocity and acceleration of each rover, and the *system formation requirements*, which specify constraints on the inter-rover coordination. For example, the formation requirements constrain the permissible error margin Δ_{max} in the inter-rover separation distances d_{12} and d_{13}. We denote the set of all requirements by $\mathcal{R}$. In our case study, $D = 5\ m$, $\Delta_{max} = 0.25\ m$, $\Delta_T = 0.25\ m$, $d_{12} = 4\ m$, $d_{13} = 6\ m$, $M = 25\ kg$.

A. Modeling the Rover Dynamics and Control

As shown in Figure 5, each rover is modeled as a block on a rough surface, with coefficient of friction ρ_i. The force F applied by the wheels causes a longitudinal acceleration while the friction offered by the lunar regolith is described by the equation $F_{fric} = -\rho \cdot v$, where v is the longitudinal velocity, resulting in linear time-invariant dynamics. Each rover's architecture consists of a plant, evolving according to the rover's dynamics, a local controller, and an actuator. Measurement of the rover states $\mathbf{x}_i, i = 1, 2, 3$, is carried out by error-free sensors. Further, we assume that the state $\mathbf{x}_i$ is instantaneously communicated to all rovers with zero delay at an update frequency equal to the controller frequency. The force F has a maximum limit F_{max}.

The control task of rover LR_1 is to drive to its destination in a straight line, whereas the task of LR_2 and LR_3 is to drive straight, while following LR_1 at longitudinal distances d_{12} and d_{13}, respectively. A local controller for each rover can be designed, for example, using the LQR approach [41].

B. Modeling Requirements with an HHCN

As shown in Figure 6, the `MissionRequirements` specify the goals of the system in terms of desirable configurations at discrete points, such as the start and end states of the rovers. At this level, such configurations can be modeled by a set of propositional variables such as $dest_i$, which models "LR_i is at its destination." Requirements such as "*eventually* LR_i is at its destination" can be well represented in linear temporal logic with finite traces (LTL_f) [42], [43]. For each rover, LTL_f formulas can be defined over a set of time-varying propositional variables $V' = \{dest_i, dir_i, stop_i, lead_i, dom_i, i \in \{1, 2, 3\}\}$, where each proposition is derived from `MissionRequirements` in Figure 6. We introduce additional propositional variables dom_i to signify that the requirements apply only when the lunar rovers arrive on the lunar surface.

On the other hand, the system-level requirements, including the formation and kinematic requirements, involve functions of the continuously evolving system state variables $\mathbf{x}_i = (x_i, y_i, v_i, z_i)^T, i \in \{1, 2, 3\}$, where x_i is the longitudinal displacement of the rovers, y_i is their transverse displacement, and v_i, z_i are their longitudinal and transverse velocities, respectively. These requirements are better specified in a continuous-time formalism, such as bounded-time signal temporal logic (STL) [44]. In this case, STL formulas are defined over the set $V = \bigcup_{i=1}^{3}\{x_i, y_i, v_i, z_i\}$. The semantics of LTL_f [43] and STL [44] define the set of allowed behaviors as *traces* at the mission and system levels, respectively. Multiple continuous-time traces can produce the same discrete-time trace, meaning that LTL_f can be considered as the abstract formalism $\mathcal{B}'$ and STL as the concrete formalism $\mathcal{B}$. We construct an HHCN $\mathcal{G}_{rover}$ from the requirement set $\mathcal{R}$ as shown in Figure 7.

1) Modeling mission-level requirements with LTL_f contracts: We express the LTL_f contracts over the set of variables V'. Because the requirements apply only when the propositional variables dom_i are asserted, we include the LTL_f formula $\mathcal{G}\,(\wedge_{i=1}^{3} dom_i)$, where $\mathcal{G}$ is used for *globally*, in the assumptions of every mission-level contract. The guarantees of the contract capturing `MissionSuccessRequirement` are given by $\wedge_{i=1}^{3}\, \mathcal{F}\, dest_i$, where $dest_i$ is asserted when the i^{th} rover reaches its destination and $\mathcal{F}$ stands for *eventually*. The guarantees of `DirectionRequirement` are denoted by $\mathcal{G}\,(\wedge_{i=1}^{3} dir_i)$, where dir_i is asserted when the i^{th} rover drives in the correct direction. `FinalVelocityRequirement` can be formalized with guarantees $\mathcal{G}\,\big(\wedge_{i=1}^{3}(dest_i \rightarrow stop_i)\big)$, that is, the i^{th} rover must always stop once it reaches its destination. Finally, the guarantees of `LeaderRequirement` can be written $\mathcal{G}\ lead_1$, i.e., LR_1 always leads the formation.

2) Modeling system-level requirements with STL contracts: We adopt bounded-time STL, where the mission horizon $T = 2400\,s$ is specified by `DestinationRequirement` in Figure 6. The state of each rover $\mathbf{x}_i$ should lie in a set

$\mathcal{D} = \{\mathbf{x} : x \in [0, 10], y \in [-5, +5], v \in [-0.2, 0.2], z \in [-0.01, +0.01]\}$. The assumptions $\mathcal{G}_{[0,T]} \wedge_{i=1}^{3} (\mathbf{x}_i \in \mathcal{D})$ are incorporated in every contract in N_0^1. Other assumptions are conjoined with this formula as necessitated by the context.

a) System formation requirements: The guarantees of contract $\mathcal{C}_{0,0}^1$ for `DriveStraightRequirement` are given by $G_{0,0}^1 := \mathcal{G}_{[0,T]} \wedge_{i=1}^{3} (|y_i - y_{i,init}| \leq \Delta_T)$, where $\Delta_T = 0.1\,m$. $y_{i,init}$ is the initial position of the i^{th} rover in the transverse direction. Intuitively, the guarantee of $\mathcal{C}_{0,0}^1$ formalizes the notion that rovers LR_1, LR_2, and LR_3 must not drift from their initial positions along the transverse direction by a margin greater than Δ_T. The `FinalVelocityRequirement` can be formalized with the contract $\mathcal{C}_{0,1}^1$ guarantees $G_{0,1}^1 := \mathcal{F}_{[0,T-t]}\mathcal{G}_{[0,t]} \wedge_{i=1}^{3} (|v_i| < \epsilon)$ stating that eventually the velocities of all the rovers should be less than the tolerance $\epsilon = 10^{-6}\ m/s$ for a period of t seconds continuously, signifying that the rovers have stopped. The `DestinationRequirement` can be modeled by contract $\mathcal{C}_{0,2}^1$ with guarantees $G_{0,2}^1 := \wedge_{i=1}^{3} \mathcal{F}_{[0,T]}(x_i \geq D + x_{i,init})$, i.e., each rover must drive a distance D from its initial state.

We further model `FormationRequirement_LR_1_LR_2` and `FormationRequirement_LR_1_LR_3` by noting that the formation, i.e., the inter-rover separation, depends on the different coefficients of friction ρ_i between each rover and the lunar regolith. Thus, we include the *in-situ* measurements of ρ_i given by the `ActualFrictionRequirement` in the assumptions. The assumptions for contracts $\mathcal{C}_{0,3}^1$ and $\mathcal{C}_{0,4}^1$ modeling the two formation requirements are then given by $A_{0,3}^1 := \mathcal{G}_{[0,T]}(\wedge_{i=1}^{3}(\mathbf{x}_i \in \mathcal{D})) \wedge (\rho_2 = \rho_{m,2})$ and $A_{0,4}^1 := \mathcal{G}_{[0,T]}(\wedge_{i=1}^{3}(\mathbf{x}_i \in \mathcal{D})) \wedge (\rho_3 = \rho_{m,3})$, respectively, where $\rho_{m,i}$, with $i \in \{2, 3\}$, is the measured friction coefficient of the i^{th} rover. The guarantees $G_{0,3}^1$ and $G_{0,4}^1$ capture the distance constraints, namely, $G_{0,3}^1 := \mathcal{G}_{[0,T]}\big(|x_1 - x_2 - d_{12}| \leq \Delta_{max}\big)$ and $G_{0,4}^1 := \mathcal{G}_{[0,T]}\big(|x_1 - x_3 - d_{13}| \leq \Delta_{max}\big)$.

b) System kinematic requirements: We encode the `SystemKinematicRequirements` with a single contract $\mathcal{C}_{0,5}^1$. The maximum acceleration applied by the rover wheels is a property of the hardware actuators which we can capture by introducing the constraint $(a \leq a_{max})$ throughout the mission, i.e., $A_{0,5}^1 := \mathcal{G}_{[0,T]}(\wedge_{i=1}^{3}(\mathbf{x}_i \in \mathcal{D})) \wedge (a \leq a_{max})$, where $a_{max} = 0.005\ m/s^2$. The guarantees of $\mathcal{C}_{0,5}^1$ require that the maximum velocity requirement be always satisfied, i.e., $G_{0,5}^1 := \mathcal{G}_{[0,T]} \wedge_{i=1}^{3} (v_i \leq v_{max})$, where $v_{max} = 10\ cm/s$.

3) Constructing $\mathcal{G}_{rover}$: As shown in Figure 7, the mission-level contracts $\mathcal{C}_{0,0}^0, \ldots, \mathcal{C}_{0,3}^0$ apply to the mission planner; therefore, we use the conjunction operation to combine them into contract term $\mathcal{C}_{N,0}^0$ of contract network N_0^0, i.e., $\mathcal{C}_{N,0}^0 = \wedge_{i=0}^{3} \mathcal{C}_{0,i}^0$. Contracts $\mathcal{C}_{0,0}^1, \ldots, \mathcal{C}_{0,5}^1$ on the system level can be categorized into 3 groups. $\mathcal{C}_{0,3}^1$ and $\mathcal{C}_{0,4}^1$ specify the *slippage control* and can be conjoined to ensure correct spacing between rovers. $\mathcal{C}_{0,0}^1$ specifies the *lateral control*. Finally, $\mathcal{C}_{0,1}^1, \mathcal{C}_{0,2}^1$ and $\mathcal{C}_{0,5}^1$ can be conjoined to specify the *longitudinal control*. The contract term of the network N_0^1 is given by composition $\mathcal{C}_{N,0}^1 = (\mathcal{C}_{0,0}^1) \otimes (\mathcal{C}_{0,3}^1 \wedge \mathcal{C}_{0,4}^1) \otimes (\mathcal{C}_{0,1}^1 \wedge \mathcal{C}_{0,2}^1 \wedge \mathcal{C}_{0,5}^1)$.

C. Mappings Between Continuous and Discrete Domains

We first characterize the value space $\mathrm{S}(V')$ and the set of possible behaviors $\mathbf{B}(V')$ at the mission level. Each variable in V' is discrete-time and Boolean-valued, with values in $\mathbb{B} = \{0, 1\}$. Thus, $\mathrm{S}(V') = \mathbb{B}^{15}$. Next, to identify the universe of possible behaviors $\mathbf{B}(V')$, we note that each behavior at the mission level can be described with discrete-time *traces*. Therefore, $\mathbf{B}(V')$ must contain all the sets of discrete-time traces corresponding to the LTL_f formulas which may be used to write contracts. To ensure that $\mathbf{B}(V')$ is closed under unions, intersections, and complements, we select $\mathbf{B}(V')$ as the σ-algebra over all possible discrete-time traces, i.e., the product σ-algebra [33]. Similarly, the value space for the system-level requirements is $\mathrm{S}(V) = \mathbb{R}^{|V|} = \mathbb{R}^{12}$. In a manner analogous to the discrete-time formalism, we choose $\mathbf{B}(V)$ as the product σ-algebra on $\mathrm{S}(V)$. We now define a conservative approximation $\Psi_{DT}^{CT} = (\alpha_l, \alpha_u)$ between these domains.

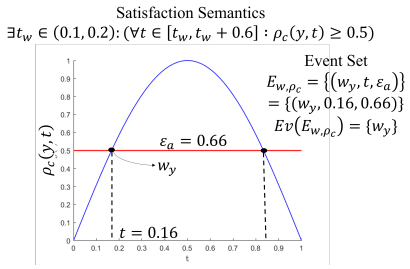


Fig. 8: Illustration of the construction of the event set E for the trace ρ_c satisfying $\mu_{STL} = \mathcal{F}_{(0.1,0.2)}\mathcal{G}_{[0,0.6)}(y \geq 0.5)$.

We interpret the satisfaction of an LTL_f formula μ_{LTL} over discrete-time traces. A discrete time trace ρ_d is a function $\rho_d : \mathbb{N} \to \mathrm{S}(V')$ that maps a time index k to a vector of values for the variables V'. For $v' \in V'$ and $k \in \mathbb{N}$, we denote by $\rho_d(v')$ the projection of the trace on the variable v', and by $\rho_d(v', k)$ the value of v' at k. A trace ρ_d satisfies μ_{LTL} if a set of *events* implied by the satisfaction semantics of LTL_f occur as k varies between 0 and the mission horizon N. For example, the LTL_f formula $\mathcal{G}\ dom$ is satisfied by a trace ρ_d if $\rho_d(dom, k) = 1\ \forall k \in \{0, \ldots, N-1\}$.

Given a trace ρ_d, we define the event set $E'_{e,\rho_d} = \{(e_{v'}, k) | v' \in V', k \in \mathbb{N}, \rho_d(v', k) = 1\}$. $e_{v'}$ is said to be the *symbolic label* for the event "v' is asserted." For example, if $\mu_{LTL} = \mathcal{G}\ dom$, we can write $E'_{e,\rho_d} = \{(e_{dom}, 0), (e_{dom}, 1), \ldots, (e_{dom}, N-1)\}$, where ρ_d is the discrete-time trace which satisfies μ_{LTL}. The event label set is the set of all labels defined as $Ev(E'_{e,\rho_d}) = \{e_{v'} | \exists v' \in V', k \in \mathbb{N}$ s.t. $(e_{v'}, k) \in E'_{e,\rho_d}\}$. For example, given $\mu_{LTL} = \mathcal{G}\ dom$ and ρ_d which satisfies μ_{LTL}, $Ev(E'_{e,\rho_d}) = \{e_{dom}\}$.

Similarly, the satisfaction of an STL formula μ_{STL} is defined over continuous-time traces. A continuous time trace ρ_c is a function $\rho_c : \mathbb{R}_{\geq 0} \to \mathrm{S}(V)$ that maps the continuous time to a vector of values for the variables V. For $v \in V$ and $t \in \mathbb{R}_{\geq 0}$, $\rho_c(v)$ is the projection of the trace on v, and $\rho_c(v, t)$ is the value of v at time t. ρ_c satisfies an STL formula μ_{STL} if a set of events implied by the satisfaction semantics of STL occur as t varies between 0 and the mission horizon T. For example, as shown in Figure 8, $\rho_c(y, t) = sin(\pi t)$ satisfies $\mu_{STL} = \mathcal{F}_{(0.1,0.2)}\mathcal{G}_{[0,0.6)}(y \geq 0.5)$.

Given a trace ρ_c, an *event threshold* $\Gamma_v \in \mathbb{R}$, and a time t such that $\exists \varepsilon_a, \varepsilon_b : 0 < \varepsilon_b < t, \varepsilon_a > 0$ such that $\forall \tau \in$

$[t-\varepsilon_b, t] : \rho_c(v,t) < \Gamma_v$ and $\forall \tau \in [t, t+\varepsilon_a] : \rho_c(v,t) \geq \Gamma_v$, i.e., if ρ_c crosses the event threshold at time t, then we say that an *event* has occurred at time t with duration (at least) ε_a. Analogous to the discrete-time case, we define the event set $E_{w,\rho_c} = \{(w_v, t, \varepsilon_a) \mid v \in V, t \in \mathbb{R}_{\geq 0}, \forall \tau \in [t, t+\varepsilon_a] : \rho_c(v,\tau) \geq \Gamma_v\}$. w_v is called the symbolic label for the event "$\rho(v)$ crosses the event threshold Γ_v." For example, in Figure 8, ρ_c crosses the event threshold $\Gamma_y = 0.5$ at time $t = 0.16$ for a duration of $\varepsilon_a = 0.66$, therefore, $E_{w,\rho_c} = \{(w_y, 0.16, 0.66)\}$. Similarly to the discrete-time case, we define the event label set as $Ev(E_{w,\rho_c}) = \{w_v | \exists v \in V, t \in \mathbb{R}_{\geq 0}, \varepsilon_a > 0, \text{ s.t. } (w_v, t, \varepsilon_a) \in E_{w,\rho_c}\}$. For example, in Figure 8, $Ev(E_{w,\rho_c}) = \{w_y\}$. We leverage a correspondence between the event sets E_{w,ρ_c} and E'_{e,ρ_d} to define Ψ^{DT}_{CT} below.

Definition 7. *Consider two event sets E and E'. Let the* event label mapping P_E *be a bijective map $\mathrm{P}_E : Ev(E) \to Ev(E')$. Let T and N be the continuous and discrete time horizons, respectively. We partition the interval $H = [0, T)$ into N sub-intervals $H = \bigcup_{k=0}^{N-1} \tau_k$, such that, for all k, the length of each interval is equal, i.e., $||\tau_k||_1 = ||\tau_{k+1}||_1$ and each interval is non-overlapping with its neighbors, namely, $\tau_k \cap \tau_{k+1} = \emptyset$. Intuitively, each interval τ_k corresponds to a single discrete-time step k. $\Psi^{DT}_{CT} = (\alpha_u, \alpha_l)$ is defined as:*

$$\alpha_u(E) = \bigcup_k \{(\mathrm{P}_E(w_i), k) : (\exists (w_i, t, \epsilon_a) \in E : [t, t+\epsilon_a] \cap \tau_k \neq \emptyset)\}$$

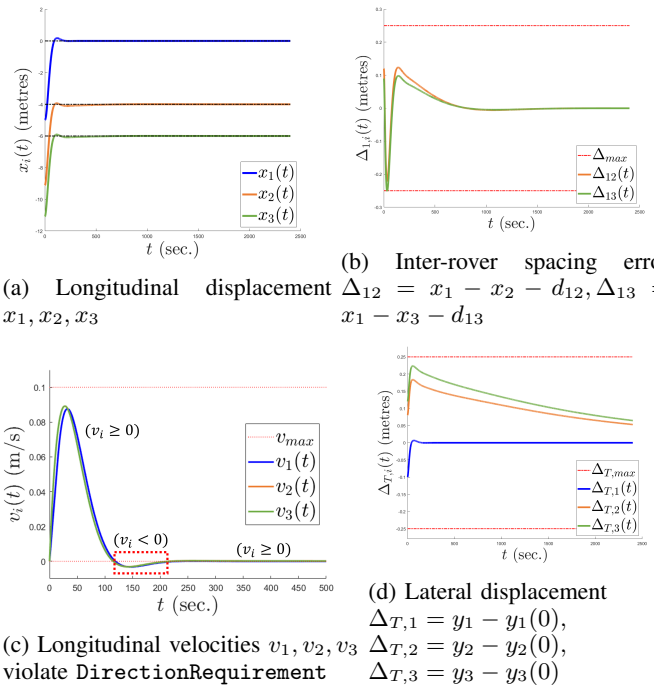
$$\alpha_l(E) = \bigcup_k \{(\mathrm{P}_E(w_i), k) : (\exists (w_i, t, \epsilon_a) \in E : [t, t+\epsilon_a] \supseteq \tau_k)\}.$$

We adopt this conservative approximation to validate the traceability for $\mathcal{G}_{rover}$. By Theorem 2, it suffices to check any one of the relations $\lambda(\mathcal{C}_N) \preceq \mathcal{C}'_N$ or $\mathcal{C}_N \preceq \gamma(\mathcal{C}'_N)$. In our case, we check $\lambda(\mathcal{C}_N) \preceq \mathcal{C}'_N$ by showing that $\lambda(\mathcal{C}_N) \not\preceq \mathcal{C}'_N$ is infeasible, thus reducing refinement checking to a *satisfiability* problem in the abstract formalism. We attempt to solve the satisfiability problem using a falsification method. That is, we falsify $\alpha_l(A) \supseteq A'$ or $\alpha_u(G) \subseteq G'$. From Section VI-B3, $\mathcal{C}'_N = \mathcal{C}^0_{N,0}$ and $\mathcal{C}_N = \mathcal{C}^1_{N,0}$. Consequently, the falsification problem reduces to searching for continuous traces which satisfy $\mathcal{C}^1_{N,0}$ but whose representations in the discrete domain do not satisfy $\mathcal{C}^0_{N,0}$. Through simulation, we identify a trace ρ_c, associated with an event set E, such that $\rho_c \models \mathcal{C}_N$, i.e., $\rho_c \subseteq G_N$. Using the conservative approximation Ψ^{DT}_{CT}, we compute $\alpha_l(E)$, which corresponds to a trace ρ_d on the discrete level. If $\rho_d \not\models \mathcal{C}'_N$, i.e., $\rho_d \not\subseteq G'_N$, we conclude $G_N \not\subseteq G'_N$, and consequently, $\lambda(\mathcal{C}_N) \not\preceq \mathcal{C}'_N$.

D. HHCN Traceability Validation

We simulate the continuous-time behaviors of the 3 rovers using MATLAB. As shown in Figure 9, the trace ρ_c satisfies the guarantees of $\mathcal{C}_N$ when the rovers drive in the desired region $\mathcal{D}$, i.e., the contract term $\mathcal{C}_N$ is consistent and compatible. Consequently, the system behaviors shown in Figure 9 satisfy the requirements at the system level.

To check the traceability, we resort to the method given in Section VI-C. To instantiate the conservative approximation in Definition 7, we use the continuous-time mission horizon $T = 2400\ s$ and the discrete-time horizon $N = 1000$ from



(a) Longitudinal displacement x_1, x_2, x_3

(b) Inter-rover spacing error $\Delta_{12} = x_1 - x_2 - d_{12}, \Delta_{13} = x_1 - x_3 - d_{13}$

(c) Longitudinal velocities v_1, v_2, v_3 violate `DirectionRequirement`

(d) Lateral displacement $\Delta_{T,1} = y_1 - y_1(0)$, $\Delta_{T,2} = y_2 - y_2(0)$, $\Delta_{T,3} = y_3 - y_3(0)$

Fig. 9: Simulation results: Continuous-time trace ρ^c for the autonomous multirover lunar mission.

Section VI-B. Therefore, we get $\tau_0 = [0, 2.4), \ldots, \tau_{999} = [2397.6, 2400)$.

Next, we construct the event mapping P_E. In the sequel, $\rho_c(x_i, t)$ is the position of the i^{th} rover at time t and $\rho_c(v_i, t)$ the velocity of the i^{th} rover at time t. To define the event sets on the continuous-time level, we define the event label $w_{dom,i}$ for the event $(\rho_c(x_i,t), \rho_c(v_i,t)) \in \mathcal{D}$ at some time t, i.e., the i^{th} rover is within the set $\mathcal{D}$ defined in Section VI-B. We define $w_{dest,i}$ as the event label for the threshold crossing $\rho_c(x_i,t) - D - x_{i,init} \geq 0$, i.e., $w_{dest,i}$ signifies the arrival of the i^{th} rover at its destination. $w_{dir,i}$ is the event label which encodes the driving direction of the rovers and is asserted if and only if the rovers have a non-negative longitudinal velocity, i.e., if and only if $\rho_c(v_i,t) \geq 0$. To encode the arrival of the rovers at the destination, we define $w_{stop,i}$ as the threshold crossing $|\rho_c(v_i,t)| \leq \epsilon$. Finally, we label w_{lead} as the event corresponding to $((\rho_c(x_1,t) - \rho_c(x_2,t) \geq 0) \wedge (\rho_c(x_1,t) - \rho_c(x_3,t) \geq 0))$, capturing the event that rover LR_1 be ahead of LR_2 and LR_3. On the discrete-time level, we define e_{dom_i} as the event when the i^{th} rover arrives on the lunar surface, i.e., e_{dom_i} occurs if and only if dom_i is asserted. Similarly, we define e_{dest_i} as the event of arrival of the i^{th} rover at its destination, i.e., when $dest_i$ is asserted. Similarly, e_{dir_i} occurs when the rovers drive in the correct direction, i.e., dir_i is asserted. Then, the event label mapping P_E is given by $\mathrm{P}_E(w_{dom,i}) = e_{dom_i}$, $\mathrm{P}_E(w_{dest,i}) = e_{dest_i}$, $\mathrm{P}_E(w_{dir,i}) = e_{dir_i}$, $P_E(w_{stop,i}) = e_{stop_i}$, and $\mathrm{P}_E(w_{lead}) = e_{lead_1}$.

On the continuous-time level, between $t = 115\ s$ and $t = 230\ s$, we see that the longitudinal velocities of all the 3 rovers become negative (red rectangle in Figure 9c). Therefore, the continuous-time event set corresponding to the trace $\rho_c(v_i)$ is $E_{dir} = \{(w_{dir,1}, 0, 115), (w_{dir,2}, 0, 115), (w_{dir,3}, 0, 115),$

$(w_{dir,1}, 230, 2170), (w_{dir,2}, 230, 2170), (w_{dir,3}, 230, 2170)\}$. By computing $\alpha_l(E_{dir})$ from Definition 7, we find that $\alpha_l(E_{dir}) = \{(e_{dir_1}, 0), (e_{dir_2}, 0), (e_{dir_3}, 0), \ldots, (e_{dir_1}, 46), (e_{dir_2}, 46), (e_{dir_3}, 46), (e_{dir_1}, 96), (e_{dir_2}, 96), (e_{dir_3}, 96), \ldots, (e_{dir_1}, 999), (e_{dir_2}, 999), (e_{dir_3}, 999)\}$. However, on the discrete-time level, the guarantee of $\mathcal{C}'_N$ contains the LTL_f formula $\mathcal{G}(\wedge_{i=1}^{3} dir_i)$. As discussed in Section VI-C, the only satisfying trace $\rho_d(dir_i)$ for this formula corresponds to the event set $E'_{dir} = \{(e_{dir_1}, 0), (e_{dir_2}, 0), (e_{dir_3}, 0), \ldots, (e_{dir_1}, 999), (e_{dir_2}, 999), (e_{dir_3}, 999)\}$. As a result, we find that $\alpha_l(E_{dir}) \neq E'_{dir}$, i.e., the mapped representation of the continuous-time event set E_{dir} is not equal to the discrete-time event set E'_{dir}. Therefore, we have constructed a trace ρ_c which satisfies $\mathcal{C}_N$, but its mapping *violates* $\mathcal{C}'_N$. We can conclude that $\lambda(\mathcal{C}_N) \not\preceq \mathcal{C}'_N$, i.e., requirement traceability is violated for the HHCN $\mathcal{G}_{rover}$. To summarize, HHCN-based requirement validation can assist the rigorous analysis of the requirements, enabling rigorous cross-level refinement checking across different levels of abstraction.

VII. Conclusions

We addressed the formal modeling and analysis of requirement hierarchies by introducing heterogeneous hierarchical contract networks (HHCNs). We formalized the problem of requirement traceability validation in terms of refinement checking between contract terms on adjacent levels of an HHCN. For cross-level refinement checking, we leveraged conservative approximations to introduce abstract and concrete embeddings of contracts, which ensure that refinement is preserved independently of the formalism. We applied our method to a case study derived from autonomy requirements for a lunar rover mission. In the future, we plan to further explore the software implementation of contract embeddings and their connection to abstraction-based verification methods in symbolic control and abstract interpretation.

References

[1] S. R. Hirshorn *et al.*, "NASA systems engineering handbook," Tech. Rep. SP-2016-6105, Natl. Aeronautics and Space Admin., 2017.

[2] R. Amini *et al.*, "FRESCO: A framework for spacecraft systems autonomy," in *IEEE Aerospace Conf.*, pp. 1–18, 2021.

[3] L. Fesq *et al.*, "Extended mission technology demonstrations using the Asteria spacecraft," in *IEEE Aerospace Conf.*, pp. 1–11, 2019.

[4] E. Hull *et al.*, *DOORS: A Tool to Manage Requirements*, pp. 187–204. Springer, 2002.

[5] G. Kotonya and I. Sommerville, "Requirements engineering with viewpoints," *Software Engineering Journal*, vol. 11, no. 1, pp. 5–18, 1996.

[6] A. Terry Bahill and S. J. Henderson, "Requirements development, verification, and validation exhibited in famous failures," *Systems engineering*, vol. 8, no. 1, pp. 1–14, 2005.

[7] D. H. Stamatis, *Failure mode and effect analysis*. Quality Press, 2003.

[8] I. A. Nesnas *et al.*, "Autonomy for space robots: Past, present, and future," *Current Robotics Reports*, vol. 2, no. 3, pp. 251–263, 2021.

[9] A. Benveniste *et al.*, "Contracts for system design," *Foundations and Trends in Electron. Des. Autom.*, vol. 12, no. 2-3, pp. 124–400, 2018.

[10] P. Nuzzo *et al.*, "A platform-based design methodology with contracts and related tools for the design of cyber-physical systems," *Proc. IEEE*, vol. 103, no. 11, pp. 2104–2132, 2015.

[11] P. Nuzzo *et al.*, "CHASE: Contract-based requirement engineering for cyber-physical system design," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 839–844, IEEE, 2018.

[12] C. Oh *et al.*, "ARACHNE: Automated validation of assurance cases with stochastic contract networks," in *Int. Conf. Computer Safety, Reliability, and Security*, pp. 65–81, Springer, 2022.

[13] A. Cimatti, M. Dorigatti, and S. Tonetta, "OCRA: A tool for checking the refinement of temporal contracts," in *IEEE/ACM Int. Conf. Automated Software Engineering (ASE)*, pp. 702–705, IEEE, 2013.

[14] P. Nuzzo and A. L. Sangiovanni-Vincentelli, *Hierarchical System Design with Vertical Contracts*, pp. 360–382. Springer, 2018.

[15] J. Day, R. Rasmussen, and I. A. Nesnas, "Principles for architecting autonomous systems," Tech. Rep. 20230005685, Jet Propulsion Laboratory, National Aeronautics and Space Administration, 2022.

[16] L. Lamport, *Specifying systems: the TLA+ language and tools for hardware and software engineers*. Addison-Wesley Professional, 2002.

[17] K. Robinson, "System modelling & design using event-B," *The University of New South Wales*, 2012.

[18] R.-J. Back and J. Wright, *Refinement calculus: a systematic introduction*. Springer Science & Business Media, 2012.

[19] Y. Yu, P. Manolios, and L. Lamport, "Model checking TLA+ specifications," in *Adv. Res. Working Conf. Correct Hardware Design and Verification Methods*, pp. 54–66, Springer, 1999.

[20] J.-R. Abrial *et al.*, "Rodin: An open toolset for modelling and reasoning in event-B," *Int. J. software tools for technology transfer*, vol. 12, pp. 447–466, 2010.

[21] T. E. Wang *et al.*, "Hierarchical contract-based synthesis for assurance cases," in *NASA Formal Methods Symp.*, pp. 175–192, Springer, 2022.

[22] C. Ptolemaeus, ed., *System Design, Modeling, and Simulation using Ptolemy II*. Ptolemy.org, 2014.

[23] F. Balarin *et al.*, "Metropolis: an integrated electronic system design environment," *Computer*, vol. 36, no. 4, pp. 45–52, 2003.

[24] E. A. Lee and A. Sangiovanni-Vincentelli, "A framework for comparing models of computation," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, pp. 1217–1229, 1998.

[25] A. Rajhans *et al.*, "Supporting heterogeneity in cyber-physical systems architectures," *IEEE Trans. Automatic Control*, vol. 59, no. 12, pp. 3178–3193, 2014.

[26] P. Cousot and R. Cousot, "Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints," in *ACM symp. Principles of Prog. Lang.*, pp. 238–252, 1977.

[27] R. Passerone *et al.*, "Conservative approximations for heterogeneous design," in *Proc. ACM Int. Conf. Embedded Soft.*, pp. 155–164, 2004.

[28] R. Passerone and A. L. Sangiovanni-Vincentelli, *Approximating Behaviors in Embedded System Design*, pp. 721–742. Springer, 2008.

[29] R. Passerone *et al.*, "Refinement preserving approximations for the design and verification of heterogeneous systems," *Formal Methods in System Design*, vol. 31, pp. 1–33, 2007.

[30] A. Benveniste, D. Nickovic, and T. Henzinger, "Compositional contract abstraction for system design," tech. rep., INRIA, 2014.

[31] K. Denecke, M. Erné, and S. L. Wismath, *Galois connections and applications*, vol. 565. Springer Science & Business Media, 2013.

[32] P. Nuzzo *et al.*, "A contract-based methodology for aircraft electric power system design," *IEEE Access*, vol. 2, pp. 1–25, 2014.

[33] D. L. Cohn, *Measure theory*, vol. 5. Springer, 2013.

[34] J. R. Burch, *Trace algebra for automatic verification of real-time concurrent systems*. PhD thesis, Carnegie Mellon University, 1992.

[35] L. De Moura and N. Bjørner, "Z3: An efficient SMT solver," in *Int. conf. Tools and Algorithms for the Construction and Analysis of Systems*, pp. 337–340, Springer, 2008.

[36] ProofWiki, "Inverse of subset of relation is subset of inverse." https://proofwiki.org/wiki/Inverse_of_Subset_of_Relation_is_Subset_of_Inverse, 2024. Accessed: 2024-03-31.

[37] J. S. Rose, *A course on group theory*. Courier Corporation, 1994.

[38] A. Cimatti *et al.*, "Nusmv 2: An opensource tool for symbolic model checking," in *Int. Conf. Computer Aided Verification*, pp. 359–364, Springer, 2002.

[39] L. Mangeruca *et al.*, "Formalization and completeness of evolving requirements using contracts," in *Int. Symp. Industrial Embedded Systems*, pp. 120–129, 2013.

[40] "Cooperative autonomous distributed robotic exploration (CADRE)." https://www.jpl.nasa.gov/missions/cadre, 2023. Accessed: 2024-07-31.

[41] B. Friedland, *Control system design: an introduction to state-space methods*. Courier Corporation, 2012.

[42] R. Gerth *et al.*, "Simple on-the-fly automatic verification of linear temporal logic," in *Int. Conf. Protocol Specification, Testing and Verification*, pp. 3–18, Springer, 1995.

[43] G. De Giacomo and M. Y. Vardi, "Linear temporal logic and linear dynamic logic on finite traces," in *International Joint conference on Artificial Intelligence*, pp. 854–860, ACM, 2013.

[44] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *Int. Conf. Formal Modeling and Analysis of Timed Systems*, pp. 92–106, Springer, 2010.