Form Methods Syst Des (2018) 52:147–192
https://doi.org/10.1007/s10703-017-0294-7

Conditions of contracts for separating responsibilities in heterogeneous systems

Jonas Westman[1] · Mattias Nyberg[1]

Published online: 13 September 2017

Abstract A general, compositional, and component-based *contract* theory is proposed for modeling and specifying *heterogeneous systems*, characterized by consisting of parts from different domains, e.g. software, electrical and mechanical. Given a contract consisting of *assumptions* and a *guarantee*, clearly separated *conditions* on a component and its environment are presented where the conditions ensure that the guarantee is fulfilled—a *responsibility* assigned to the component, given that the environment fulfills the assumptions. The conditions are applicable whenever it cannot be ensured that the sets of ports of components are partitioned into inputs and outputs, and hence fully support scenarios where components, characterized by both causal and acausal models, are to be integrated by solely relying on the information of a contract. An example of such a scenario of industrial relevance is explicitly considered, namely a scenario in a supply chain where the development of a component is outsourced. To facilitate the application of the theory in practice, necessary properties of contracts are also derived to serve as sanity checks of the conditions. Furthermore, based on a graph that represents a structuring of a hierarchy of contracts, sufficient conditions to achieve *compositionality* are presented.

Keywords Contract · Heterogeneous systems · Architecture · Component · Specification · Composition

1 Introduction

The notion of *contracts* was first introduced in [1] as a pair of pre- and post-conditions [2] to be used in formal specification of software (SW). In more recent contract theory [3–

✉ Jonas Westman
jowestm@kth.se

Mattias Nyberg
matny@kth.se

1 KTH, Brinellvägen 83, 100 44 Stockholm, Sweden

5], developed within European project SPEEDS [6], the use of contracts is extended from formal specification of SW to serving as a central systems engineering philosophy to support the design of *heterogeneous systems* [7–9]. Heterogeneous systems are characterized by consisting of parts from multiple domains, e.g. SW, mechanical, electrical, etc. The use of theory [3–5] has been advocated in several contexts, e.g. Platform Based Design in [5], Model Based Design in [10], safety analyzes in [11], virtual integration and testing in [12], and for structuring safety requirements in [13–15].

With the intent to be able to model and specify heterogeneous systems, the theory in [3–5] considers a basic component-based modeling formalism that relies on a "*language-based abstraction where composition is by intersection.* [...] *No particular model of computation and communication is enforced, and continuous time dynamics such as those needed in physical system modeling is supported as well.*" [3]

More specifically, the theory in [3–5] is centered around the notion of a *component* M with a set of *ports* and an *implementation*, also denoted M, which is an *assertion*, i.e. a set of value sequences over the ports. A *contract* C for component M is a pair of assertions (A, G) where G represents an intended property such as a *requirement* [16–18] or a design constraint. The *responsibility* that *guarantee* G is fulfilled, is assigned to component M, given that certain *assumptions* A are fulfilled—a responsibility of the environment of the component. This clear separation of responsibilities, embodied by a contract, is a principle for seamless integration of components, a primary concern in heterogeneous systems development, characterized by complex supply chains distributed over multiple organizations [3,5].

To concretize the separation of responsibilities embodied by a contract in theory [3–5], contract satisfiability conditions are presented. Given that "*M and C have the same ports*" [3], component M *satisfies* C if $A \cap M \subseteq G$. This satisfaction relation ensures that the composition of component M and an environment component M_E where $M_E \subseteq A$ fulfills the guarantee, i.e. that $M_E \cap M \subseteq G$. Hence, given that contract C is limited to the set of ports of M, to ensure that guarantee G is fulfilled, conditions on component M and its environment M_E can be separated as $A \cap M \subseteq G$ and $M_E \subseteq A$, respectively.

However, guarantee G is trivially fulfilled if $M_E \cap M = \emptyset$, which is an undesirable case since it characterizes a contradiction if M_E and M are represented through logical formulas. In addition, allowing G to be trivially fulfilled means that a component where $A \cap M = \emptyset$ (and in particular where $M = \emptyset$) is always an acceptable solution, *regardless of how environment M_E is implemented*. Notably, the theory in [3–5] does not explicitly address this undesirable case, but the theory does propose an approach to further separate responsibilities, which indirectly leads to that non-trivial solution $M_E \cap M = \emptyset$ is avoided in the typical case. This approach is to partition the sets of ports of component M and its environment M_E into mirroring inputs and outputs, and enforcing the additional conditions that M and M_E are *receptive to their inputs*, i.e. that any values of the inputs are allowed by the implementations at any point in time.

From these basic concepts of theory [3–5], it can be observed that in order to ensure both that guarantee G is fulfilled $M_E \cap M \subseteq G$ and that trivial solution $M_E \cap M = \emptyset$ is avoided, it is necessary to enforce conditions $A \cap M \subseteq G$, $M_E \subseteq A$, and that M and M_E are receptive to their inputs. However, in order to enforce these conditions, the two following prerequisites apply:

(a) the sets of ports of component M and its environment M_E are partitioned into mirroring inputs and outputs; and
(b) contract $C = (A, G)$ is limited to the set of ports of component M.

Hence, trivial solution $M_E \cap M = \emptyset$ is avoided in the theory in [3–5] only through prerequisite (a), expressing that the causality of the ports are specified. However, considering typical models of the parts of a heterogeneous system, including not only SW, but also physical parts (mechanical, electrical, etc.), in addition to causal models, there would also be *acausal models* [19] where the causality of ports is unspecified, which is common in e.g. *Modelica* [20,21]. In fact, as stated in [19], when modeling physical parts, acausal models are well suited given that they reflect the physical structure of the parts and are also more reusable than causal models since the solution direction is not fixed. Hence, prerequisite (a) expresses a limitation of the theory in [3–5] and highlights the need for conditions on a component and its environment where these conditions ensure that the guarantee is *non-trivially fulfilled* $\emptyset \neq M_E \cap M \subseteq G$ whenever prerequisite (a) is not ensured.

Regarding prerequisite (b), while the theory in [3–5] only supports contracts that are limited to the sets of ports of components, there are at least two strong reasons why such a limitation needs to be relaxed in the context of specifying heterogeneous systems.

The first reason is to support a central principle in *requirements engineering* (RE) [16–18] where this principle expresses that requirements for a system should be expressed over ports that are not of the system, but rather in the environment [18]. In accordance with [18], systems are developed to make the behavior of the environment satisfactory and thus, all requirements on a system should be statements about the environment. An example of when this RE principle is made utterly explicit is the functional documents model [22,23] where, in the context of a SW system represented as a predicate SOF, there are four distinct collections of ports: m and c for quantities respectively monitored and controlled by peripheral devices attached to SOF; and i and o for input and output registers of SOF. SW system SOF and ports are shown in Fig. 1 as a rectangle filled with gray and boxes, respectively. In accordance with this model, *requirements* are described as a predicate $REQ(m, c)$, which is to be fulfilled by SOF given *environment assumptions* expressed as predicates: $IN(m, i)$ on how sensors transform monitored values to their internal representation; $OUT(o, c)$ on how actuators transform software outputs to controlled values; and $NAT(m, c)$ on the rest of the environment. Interpreted through contracts theory, these predicates form a contract $(IN(m, i) \wedge OUT(o, c) \wedge NAT(m, c), REQ(m, c))$ for SOF where *this contract is not limited to the set of ports of* SOF.

An example where this RE principle is manifested in an industrial case study can be found in [24] where ModelicaML [25] is used to specify and verify requirements on a subsystem of a fuel management system where the requirements express the end-to-end functionality of the fuel management system in general. Another example can be found in [26] where SysML [27] is used to specify requirements on an engine knock controller and where the requirements allocated to the controller explicitly refer to parts, such as the piston, which is not a port of the controller but rather a port in its environment.

The second reason why contracts that are not limited to component ports are needed is due to the fact that, in the area of functional safety [28,29], the associated risk of a component, is assessed in the context of how it affects its environment. Hence, in order to properly express a safety specification for a component using a contract, there is a need to refer to parts in the environment that the component is to be deployed in [14]. For example, in functional safety standard ISO 26262 [29], top-level safety requirements for an item, i.e. a system *within* the vehicle, are formulated in order to prevent or mitigate hazards, where these hazards

> *"shall be defined in terms of the conditions or behaviour that can be observed at the vehicle level"* [29].

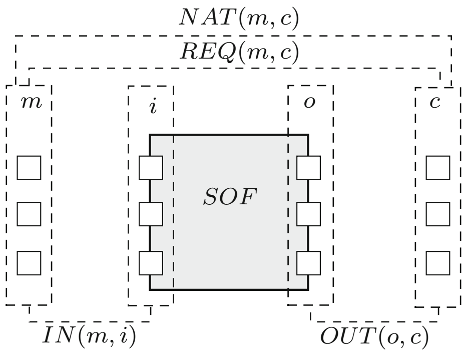


Fig. 1 A SW system SOF and its inputs i and outputs o; requirements $REQ(m,c)$; assumptions $IN(m,i)$, $OUT(o,c)$, and $NAT(m,c)$; and ports m and c in the environment of SOF

Hence, in order for a contract to capture the fact that one overall responsibility of the item is to mitigate or prevent the hazard, which extends beyond the ports of the item, a contract that is not limited to the set of ports of the item, is needed. This can be observed in industrial examples [13,14], and also in [30], where safety requirements that are not limited to the sets of ports of components are necessarily used in order to properly express safety specifications for components.

In the context of specifying heterogeneous systems, the two reasons above explain the importance of relaxing the limitation expressed in prerequisite (b) such that assumptions and guarantees can be specified, not only over the ports of a component, but also over ports in the environment of the component. Considering this, and the previous stated fact that prerequisite (a) also expresses a limitation of theory [3–5], it can be concluded that there is a need for conditions on a component and its environment where these conditions ensure that the guarantee is non-trivially fulfilled $\emptyset \neq M_E \cap M \subseteq G$, whenever prerequisite (a) cannot be ensured and regardless of whether prerequisite (b) holds or not.

Contributions As the main contribution of the present paper, a set of clearly separated conditions on a component and its environment are established where the conditions ensure that the guarantee is non-trivially fulfilled. These conditions are, in contrast to the conditions presented in [3], applicable whenever it cannot be ensured that prerequisites (a) and (b) hold or not. More specifically, the main contribution shows that in order for relation $\emptyset \neq M_E \cap M \subseteq G$ to hold, the respective conditions on the component and the environment can be separated as:

(i) $A \cap M \subseteq G$ and $A \cap G \subseteq M$; and
(ii) $M_E \subseteq A$ and $M_E \cap G \neq \emptyset$,

where M is limited to constraining only the set of ports of the component, but where A and G are not. Note that this includes the considered case in [3] where A and G only constrain the ports of the component. In fact, it is proven that condition (i) is a necessary and sufficient condition on component M to ensure that relation $\emptyset \neq M_E \cap M \subseteq G$ holds for each environment M_E that is such that condition (ii) holds. This means that conditions (i) and (ii) fully support any scenario where a component and its environment are developed in *complete isolation*; the only information that needs to be shared is a contract and the set of ports of the component where this set is *not* required to be partitioned into inputs and outputs. An example of such a scenario of industrial relevance is explicitly considered throughout the paper, namely a scenario in a supply chain where the development of a component is outsourced from a client to a supplier. In this scenario, the supplier is to ensure that condition

(i) on the component holds, while the client is to ensure that condition (ii) on the environment holds.

A second contribution considers the fact that the relaxation of the limitation expressed in the prerequisite (b) leads to increased expressiveness with respect to how a contract can be specified. However, the increased expressiveness is not unlimited since there are necessary restrictions on the ports constrained by the assumptions and the guarantee of a contract in order for conditions (i) and (ii) to hold. Therefore, to facilitate the specification of contracts in practice, conditions, called *scoping conditions*, are introduced to limit the set of ports that the assumptions and the guarantee of a contract can constrain; these scoping conditions ensure that the necessary restrictions are not violated without limiting expressiveness.

As a third contribution, considering that the contract properties *consistency* and *compatibility* are in [3,5] defined under the limitations expressed in prerequisites (a) and (b), revised definitions of these properties are presented where these limitations are relaxed. More specifically, in [3,5], the definitions of consistency and compatibility are based on the concept of receptivity to inputs, but have the overall aim to ensure that a contract is such that there in fact exist a component and an environment that are such their corresponding conditions, as defined in [3,5], hold. In contrast to the definitions in [3,5], which require prerequisite (a) and (b) to hold, the definitions of consistency and compatibility in the present paper do not. Instead, consistency and compatibility are defined as necessary properties of conditions (i) and (ii); more specifically, consistency and compatibility are defined respectively as: if there exists a component and an environment that are such that their corresponding conditions (i) and (ii) hold.

As a fourth and final contribution, as a basis for structuring a contract C and a set of contracts $\{C_i\}_{i=1}^{N}$ in parallel to a composition M of a set of components $\{M_i\}_{i=1}^{N}$ with the intent to establish that M is such that condition (i) holds with respect to C, a graph, called a *contracts composition structure*, is introduced. Based on a contracts composition structure and in accordance with the principle of *compositionality* [31,32], sufficient conditions are provided to ensure that: composition M of $\{M_i\}_{i=1}^{N}$ is such that condition (i) holds with respect to C if each component M_i is such that condition (i) holds with respect to C_i. Hence, the fourth contribution supports an indirect way of establishing that M is such that condition (i) holds with respect to C when a direct approach is not feasible due to e.g. the complexity of the composition.

The four contributions constitute a general contract theory for heterogeneous systems where all theorems and definitions are expressed in terms of the language-independent formalism of assertions. This generality allows the proposed theorems and definitions to be instantiated in more concrete theories/formalisms tailored for a specific purpose, e.g. formal verification in tools.

Related work Notably, in addition to [3–5], there are other general theories [33–59] for assume-guarantee reasoning. These theories and other related work will be described in more detail in Sect. 7, and the following will instead focus solely on arguing for the fact that the contributions of the present paper cannot be found in any of theories [3–5,33–59]. Since the second to fourth contribution are derived with the main contribution as a foundation, the main contribution alone will be considered as a basis of argumentation. Due to the fact that the part of the main contribution that specifically addresses the limitation expressed in prerequisite (a), is rather specific to assume-guarantee theories with a satisfaction relation $A \cap M \subseteq G$, namely [4,5,34–37,47,55] and one of the instantiated theories in [33], only these will be considered when arguing for this part of the main contribution. The part of the main contribution regarding the fact that conditions (i) and (ii) are applicable also when relaxing the limitation in prerequisite (b), is compared to this aspect of expressivity of other assume-guarantee theories, in general.

Considering [4,5,33–37,47,55], the theories [4,33,37] use the same type of approach as [3] to avoid trivial solution $M_E \cap M \neq \emptyset$ and only [34] provides an alternative approach. In [34], a game-theoretic approach [60] is considered where the component and its environment take turns in changing the values (called states) of variables in a fixed set. To avoid trivial solution $M_E \cap M \neq \emptyset$, the approach in [34] requires that each step (or transition) of a run also needs to be identified as an action taken by either the component or its environment. This means that the approach in [34] is not applicable in the case of conventional physical models (see e.g. Modelica [20,21]) where interactions are modeled, not by an explicit separation of actions taken by a component and its environment, but rather by equations that simultaneously constrain each other. In contrast to [34], conditions (i) and (ii) in the present paper are indeed applicable without the need for identifying steps as component or environment actions, and are hence applicable for conventional physical models.

Despite the fact that contracts are limited to the sets of ports of components in [3], there are assume-guarantee theories that do not consider such a limitation, namely [34,36,40,44,47,48,53–55] and the meta theory in [33]. However, in contrast to the present paper, in theories [36,44,47,48,53–55], the concept of ports is abstracted away by considering implementations, assumptions, and guarantees as formulas or abstract properties that are not bound to be specified over a certain structure such as a set of variables. In [34,40], assumptions, guarantees, and implementations are all expressed over a fixed set of variables, which means that if a subset of the fixed set is associated as the set of ports of a component, it could be argued that contracts that are not limited to the set of ports of the component are supported. However, in contrast to the present paper, the association of a variable as a port of a component is only established informally since none of theories [34,40] incorporate a means to enforce the condition that the component implementation can only constrain its ports and no other variables in the fixed set. This does not only mean that the second contribution of the present paper is not derivable from theories [33,34,36,40,44,47,48,53–55], but also something more fundamental; these theories consider a looser notion of contract satisfiability where no conditions are enforced on the set of ports that a component can or cannot constrain. Hence, these theories do not capture the additional design conditions enforced on the component as expressed by its set of ports, which is a fundamental principle in many works, e.g. [39] and the present paper.

However, out of theories [3–5,35,37–39,41–43,45,46,49–52,56–59] where ports[1] are explicitly considered, none of these fully relax the limitation that assumptions and guarantees must be limited to component ports. Many of these theories provide ways of extending assumptions and guarantees to be expressed over a greater set of ports, e.g. by using inverse projection [33]. However, despite being expressed over a greater set of ports, the extended assumptions and guarantee will still specify the same constraints as the non-extended assumptions and guarantee; in the extended case, the constraints are simply quantified over additional ports, which are, however, not constrained. In fact, the work in [39] is the only theory that allows assumptions to constrain ports in the environment of a component, but not guarantees. Thus, in contrast to [39], and also [3–5,35,37–39,41–43,45,46,49–52,56–59], the present paper allows both assumptions and guarantees to constrain ports that extend outside of the set

[1] A generalization of ports as defined in [3–5] is considered here where the concept of ports encompasses not just variables, but also *labels*, as long as they both are in a structure, e.g. a set, over which a component implementation is explicitly expressed. Hence, ports characterize inputs, outputs, messages, events, signals, actions, etc.

of ports of a component and, thus, fully relaxes the limitation that contracts must be limited to component ports.

While none of theories [3–5,35,37–39,41–43,45,46,49–52,56–59] fully relax the limitation that contracts must be limited to component ports, it can be argued, considering the particular aspect of relaxing such a limitation, that these theories could serve as equivalent formalisms to the one in the present paper if the set of ports of a component is allowed to simply include any port of the environment. These additional ports could further be labeled as "environment ports" to distinguish them from those inherent to the component. However, in addition to this, for these theories to serve as equivalent formalisms to the one in the present paper, there would also be a need to enforce the condition that the component implementation can only constrain the subset of its ports that are not labeled as environment ports. Notably, the theories [3–5,35,37–39,41–43,45,46,49–52,56–59] do not support a straightforward way to enforce such a condition since, analogous to [34,40], they do not incorporate a means to distinguish a port that an implementation constrains from one that it does not in the set over which the implementation is expressed. This can be contrasted with the present paper where such a condition is enforced at the foundation of the theory, influencing almost all definitions and theorems. Hence, considering the particular aspect of relaxing the limitation that contracts must be limited to component ports, it is clear that theories [3–5,35,37–39,41–43,45,46,49–52,56–59] do not serve as equivalent formalisms to the present paper and can neither be trivially extended to serve as such.

Organization of paper Based on a theoretical foundation in Sect. 2, *contracts* are introduced in Sect. 3 along with the main contribution, namely the derivation of conditions (i) and (ii). The basis is a scenario where a contract is used to outsource the development of a component. Considering conditions (i) and (ii), Sect. 4 presents necessary restrictions on the set of variables over which assumptions and guarantees are specified. Definitions of *consistency* and *compatibility* of a contract are established in Sect. 5 and Sect. 6 introduces *contracts composition structures* as a means to structure a hierarchy of contracts to achieve *compositionality*. Section 7 extends the related work in this section and Sect. 8 summarizes the paper and draws conclusions.

2 Assertions and elements

This section establishes concepts for modeling a heterogeneous system and its parts, and to be able to derive the contributions concerning contracts and their properties as will be presented in Sects. 3, 4, 5 and 6. The concepts presented in this section mainly draws inspiration from contract theory [3–5] developed in European research project SPEEDS [6]. Similarities between the concepts presented in this section and the ones presented in [3–5], as well as with other related work, are discussed briefly throughout this section and in more detail in Sect. 7.

2.1 Assertions and runs

Let $X = \{x_1, \ldots, x_N\}$ be a non-empty set of variables. Consider a pair (x_i, ξ_i) consisting of a variable x_i and a trajectory $\xi_i = \{(t, x_i(t))\}_{t \in T}$ of values of x_i over a time-window $\emptyset \neq T \subseteq \mathbb{R}_{\geq 0}$. For example, Fig. 2a shows a trajectory of values of the variable x_1. A set $\{(x_1, \xi_1), \ldots, (x_N, \xi_N)\}$ of such pairs with trajectories over the same time-window T, is called a *run* for X over T, denoted either $\omega_{X,T}$ or simply ω. For example, a run can be

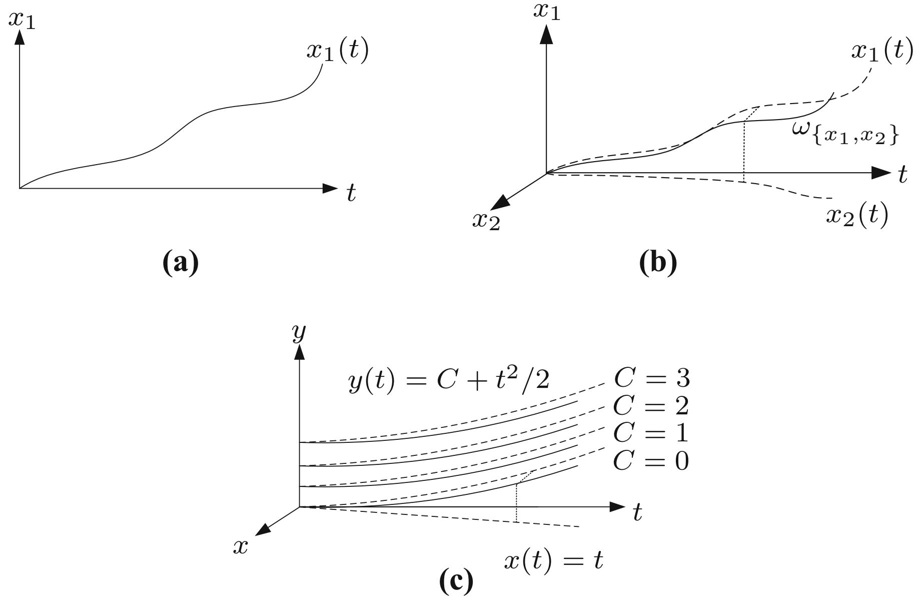


Fig. 2 In (**a**), a trajectory of values of x_1 is shown. In (**b**), a run $\omega_{\{x_1,x_2\},T}$ is shown, consisting of two pairs containing the trajectory shown in (**a**) and another trajectory of values of x_2. In (**c**), a subset of the runs that are solutions to differential equation $\frac{\mathrm{d}y}{\mathrm{d}t} = x(t)$, where $x(t) = t$ are shown

a *trace* [56,61–63] or an *execution* as presented in [36]. As a more illustrative example, a run $\omega_{\{x_1,x_2\},T}$ consisting of two pairs is shown in Fig. 2b as a solid line in three dimensions x_1, x_2, t where the trajectories of the pairs are also shown as two dashed lines. The trajectory of values of x_1 is the trajectory shown in Fig. 2a and the other trajectory consists of values of variable x_2.

In the following, a universal set of variables Ξ will be assumed where Ω will denote the set of all possible runs for Ξ over each time-window $\emptyset \neq T \subseteq \mathbb{R}_{\geq 0}$. An *assertion* W is a, possibly empty, subset of Ω, i.e. $\mathsf{W} \subseteq \Omega$. This notion corresponds to similar definitions in theories [3–5,14,64]. However, these theories consider assertions as sets of runs for dissimilar sets of variables *local* to the assertions, rather than sets of runs for a *universal* set of variables as in the present paper. The choice of considering each run in an assertion for a universal set of variables, i.e. the set Ξ, is inspired by [34,40] and allows to combine and compare assertions with the use of regular set operations (e.g. $\cup$, $\cap$) and set relations (e.g. $\subseteq$).

Note that rather than explicitly declaring its set of runs, an assertion can be expressed through constraints, e.g. by equations, inequalities, or logical formulas. For example, an assertion W' expressed through equation $u = v$, is the set of all runs in Ω where $u = v$ holds for each time point.

As a second example, consider that W'' is an assertion expressed through first order differential equation

$$\frac{\mathrm{d}y}{\mathrm{d}t} = x(t), \text{ where } x(t) = t \text{ and } t \in \mathbb{R}_{\geq 0}.$$

Hence, assertion W'' is the set of all possible runs for $\varXi$ over $\mathbb{R}_{\geq 0}$ where the runs are the solutions to the differential equation. Figure 2c shows a subset of these solutions in the dimensions x, y, and t.

As a third example of how an assertion can be expressed, consider that W''' is an assertion expressed through logic formula $a(t) = 0 \vee b(t) = 0$ where $t \in \{0, 1, \ldots, 10\}$ and where both variables a and b take values from $\{0, 1\}$. This means that assertion W''' is the set of all possible runs for $\varXi$ over discrete time-window $\{0, 1, \ldots, 10\}$ where, for each time-point t, at least one of a and b has value 0.

2.1.1 Projection of assertions

Given an assertion W and a set of variables X, the *projection* [14,33,65] of W onto X, written $proj_X(\mathsf{W})$, is the set obtained by removing each pair that does not contain a variable $x \in X$ from each run $\omega_{\varXi,T}$ in W, i.e.

$$proj_X(\mathsf{W}) = \{\omega_{X,T} \mid \omega_{\varXi,T} \in \mathsf{W} \text{ and } \omega_{X,T} = \{(x, \xi) \mid (x, \xi) \in \omega_{\varXi,T} \text{ and } x \in X\}\}. \quad (1)$$

Note that $proj_X(\emptyset) = \emptyset$ and $proj_\emptyset(\mathsf{W} \neq \emptyset) = \{\emptyset\}$.

Furthermore, relation (1) corresponds to the definition of projection in [14,33], while [65] defines projection as an operation on a single run instead of on a set of runs.

The *extended projection* of W onto X is denoted $\widehat{proj}_X(\mathsf{W})$ and is the assertion obtained by extending each run $\omega_{X,T}$ in $proj_X(\mathsf{W})$ with all possible runs for $\varXi \setminus X$ over T. That is,

$$\widehat{proj}_X(\mathsf{W}) = \{\omega \mid \omega \in \varOmega \text{ and } proj_X(\{\omega\}) \subseteq proj_X(\mathsf{W})\}. \quad (2)$$

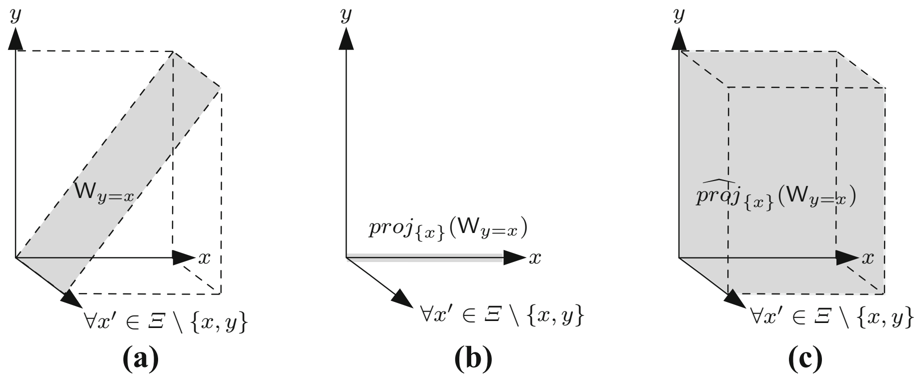
Note that $\widehat{proj}_\emptyset(\mathsf{W}) = \varOmega$ due to the fact that $proj_\emptyset(\mathsf{W}) = \{\emptyset\}$ and $proj_\emptyset(\{\omega\}) = \{\emptyset\}$ for each $\omega \in \varOmega$.

The type of operation used for extending $proj_X(\mathsf{W})$ to assertion $\widehat{proj}_X(\mathsf{W})$ is called *inverse projection* in [14,33]. Furthermore, if W is expressed through a logical formula P, then the extended projection of W onto $\varXi \setminus \{x_1, \ldots, x_N\}$ corresponds to the notion of *port elimination* [3] or *variable hiding* [66,67] through existential quantification, i.e. $\exists x_1, \ldots, x_N : P$.

As an example of projection and extended projection, consider an assertion expressed through equation $y = x$, where the assertion will be denoted as $\mathsf{W}_{y=x}$ for convenience. Assertion $\mathsf{W}_{y=x}$ is shown in Fig. 3a where the x'-axis can be the axis of any variable in $\varXi \setminus \{x, y\}$ and where only one single point in time is shown considering that $y = x$ is independent of time. The projection of $\mathsf{W}_{y=x}$ onto $\{x\}$, i.e. set of runs $proj_{\{x\}}\left(\mathsf{W}_{y=x}\right)$, is shown in Fig. 3b. If each run $\omega_{\varXi,T}$ in $proj_{\{x\}}\left(\mathsf{W}_{y=x}\right)$ is extended with all possible runs for $\varXi \setminus \{x\}$ over T, the extended projection of $\mathsf{W}_{y=x}$ onto $\{x\}$ is obtained. This assertion is shown in Fig. 3c and is in fact the assertion $\varOmega$.

2.1.2 Variables constrained by assertions

As previously presented, assertions are sets of runs for universal set of variables $\varXi$. However, as can be seen in previous examples, assertions can be expressed through constraints specified over a subset of $\varXi$. For example, equation $y = x$, through which the assertion $\mathsf{W}_{y=x}$ shown in Fig. 3a is expressed, is specified simply over set of variables $\{x, y\}$. For a given assertion, this section introduces a concept that distinguishes such as set of variables from set of variables $\varXi$.



y
$\mathsf{W}_{y=x}$
x
$\forall x' \in \Xi \setminus \{x,y\}$
(a)
$proj_{\{x\}}(\mathsf{W}_{y=x})$
(b)
$\widehat{proj}_{\{x\}}(\mathsf{W}_{y=x})$
(c)

Fig. 3 In **a**–**c**, an assertion $\mathsf{W}_{y=x}$, its projection onto $\{x\}$, and its extended projection onto $\{x\}$ are shown, respectively

The concept of such a set is not presented in [3–5], but is, on the other hand, an essential concept in the present paper. This difference between [3–5] and the present paper can be explained by the fact that while each assertion consists of runs for universal set of variables Ξ in the present paper, an assertion consists of runs for a set of variables local to the assertion in [3–5]. Considering that the use-cases in [3–5] show the clear intent that the set of local variables should be equal to the set of variables that are necessary and sufficient to express the assertion, no distinction between these two sets is required in [3–5]. In contrast, when considering assertions as defined in the present paper, while the runs of an assertion are for Ξ, the set of variables that are necessary and sufficient to express the assertion will, in the generic case, be a proper subset of Ξ, e.g. as exemplified with the assertion $\mathsf{W}_{y=x}$. Thus, to be able to directly refer to the subset of Ξ where this subset is necessary and sufficient to express an assertion, the concept of *the set of variables constrained by the assertion*, is introduced.

Definition 1 (*Variables constrained by assertion*) A variable x is constrained by an assertion W if

$$\widehat{proj}_{\Xi\setminus\{x\}}(\mathsf{W}) \neq \mathsf{W}.$$

Let X_{W} denote the set of variables constrained by W. □

Notably, in accordance with Definition 1, to find the set X_{W}, i.e. the set of variables constrained by W, there is a need to iterate through each variable $x \in \Xi$ to determine whether or not it holds that $\widehat{proj}_{\Xi\setminus\{x\}}(\mathsf{W}) \neq \mathsf{W}$. The following proposition provides an alternative approach for finding X_{W} without the need for iterating over Ξ.

Proposition 1 *Given an assertion* W*, a set of variables X is equal to X_{W} if and only if each variable in X is constrained by* W *and* $\widehat{proj}_X(\mathsf{W}) = \mathsf{W}$.

The proof of Proposition 1 is found in “Appendix A”.

In accordance with Proposition 1, there exists a unique set of variables X constrained by W such that $\widehat{proj}_X(\mathsf{W}) = \mathsf{W}$. This unique set of variables is actually set of variables X_{W} constrained by W.

As an example of how to use Proposition 1, consider assertion $\mathsf{W}_{y=x}$. As shown in Fig. 3a, c, the extended projection of $\mathsf{W}_{y=x}$ onto $\{x\}$ is not equal to $\mathsf{W}_{y=x}$, but rather a proper superset. The same holds for the extended projection of $\mathsf{W}_{y=x}$ onto $\{y\}$. This means that both variables in set $\{x, y\}$ are constrained by $\mathsf{W}_{y=x}$. However, not only that, since equality $\widehat{proj}_{\{x,y\}}(\mathsf{W}_{y=x}) = \mathsf{W}_{y=x}$ implies, in accordance with Proposition 1, that $\{x, y\}$ is in fact also *the* set of variables constrained by $\mathsf{W}_{y=x}$.

2.2 Elements

In this section, the concept of an *element* is introduced. Elements correspond to Heterogeneous Rich Components (HRCs) [4,68,69], as used in contract theory [3–5] of SPEEDS, in the sense that an element can represent any part of a heterogeneous system in general, such as a SW or physical part. However, an element can also serve as a *connector*, e.g. as described in Modelica [20,21], or as a functional or logical design entity in general, e.g. as a SysML block [27]. The term element is in the present paper chosen over the term component due to the fact that the concept of elements also encompasses connectors, which are in Modelica [20,21] and in theories such as [39], treated as separate entities from components.

Definition 2 (*Element*) An *element* $\mathbb{E}$ is an ordered pair (X, B) where:

(a) X is a non-empty set of variables, each called *a port variable*; and
(b) B is an assertion, called the *behavior* of $\mathbb{E}$ and where the set of variables constrained by B is a subset of X. □

In the typical case, an element represents a real world entity where the port variables represent tangible quantities of the entity from the perspective of an external observer to the entity. The behavior of the element captures the static and dynamic constraints that the entity imposes on the quantities, independent of its surroundings.

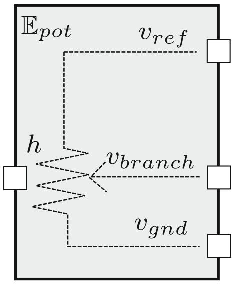
Definition 2 is of a general type, which means that conditions (a) and (b) hold regardless of the domain, e.g. mechanical, SW, etc., that is considered. However, in some domains, e.g. the SW domain, set of port variables X of an element (X, B) is typically partitioned into inputs X^{in} and outputs X^{out}. In accordance with [3,5], the partitioning of X into inputs and outputs enforces an additional condition on the behavior B, namely that B *is receptive to* X^{in}, i.e. that $\widehat{proj}_{X_{in}}(\mathsf{B})$ is the set of all possible runs for X^{in} over each time-window in $\{T|\omega_{\Xi,T} \in \mathsf{B}\}$.

As an illustrative example of an element, let $\mathbb{E}_{pot} = \left(X_{pot}, \mathsf{B}_{pot}\right)$ be an element representing a potentiometer. The element and its port variables are shown in Fig. 4 as a rectangle filled with gray and boxes on the edges of the rectangle, respectively. Port variables v_{ref}, v_{branch}, and v_{gnd} represent the reference, branch, and ground voltages, respectively. Furthermore, h represents the position $(0 - 100\%)$ of the 'slider' that moves over the resistor and branches the circuit. Given a representation where it is assumed that the branched circuit is connected to a resistance that is significantly larger than the resistance of the potentiometer, behavior B_{pot} can be expressed through equation $h = \frac{v_{branch} - v_{gnd}}{v_{ref} - v_{gnd}}$.

2.3 Composition of elements

This section describes how a set of elements $\{\mathbb{E}_1, \ldots, \mathbb{E}_N\}$ can be combined into a single element $\mathbb{E}$—*a composition of* $\{\mathbb{E}_1, \ldots, \mathbb{E}_N\}$. In accordance with [3–5], the underlying principle is to combine individual behaviors using *intersection* where the *sharing of port variables* between elements captures the interaction points between the elements. Sharing of port variables is also used in e.g. [58].

Fig. 4 An element $\mathbb{E}_{pot} = (X_{pot}, \mathsf{B}_{pot})$, representing a potentiometer



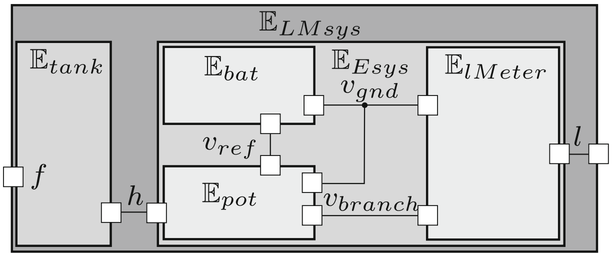


Fig. 5 A set of elements representing a "Level Meter system" and its parts

Prior to presenting a formal definition of composition of a set of elements, the concept is introduced by considering a set of elements representing the parts of a "Level Meter system" (LM-system) as shown in Fig. 5. LM-system $\mathbb{E}_{LMsys}$ consists of a tank $\mathbb{E}_{tank}$ and an electric-system $\mathbb{E}_{Esys}$, which further consists of potentiometer $\mathbb{E}_{pot}$ shown in Fig. 4, a battery $\mathbb{E}_{bat}$, and a level meter $\mathbb{E}_{lMeter}$. The sharing of a port variable between elements is shown either by a line connecting two or more boxes corresponding to the same port variable or by the appearance of the same box on edges of several rectangles.

In the LM-system, slider h is connected to a "floater", trailing level f in the tank. In this way, potentiometer $\mathbb{E}_{pot}$ is used as a level sensor to estimate the level in the tank. The estimated level is presented by level meter $\mathbb{E}_{lMeter}$ where l denotes the presented level.

Behaviors B_{bat}, B_{lMeter}, and B_{tank} of $\mathbb{E}_{bat}$, $\mathbb{E}_{lMeter}$, and $\mathbb{E}_{tank}$ are expressed through equations $v_{ref} - v_{gnd} = 5V$, $l = \frac{v_{branch} - v_{gnd}}{5V}$, and $h = f$, respectively. Intersection $\mathsf{B}'_{LMsys} = \mathsf{B}_{bat} \cap \mathsf{B}_{Lmeter} \cap \mathsf{B}_{tank} \cap \mathsf{B}_{pot}$ captures the combined constraints expressed by the behaviors of $\mathbb{E}_{bat}$, $\mathbb{E}_{lMeter}$, $\mathbb{E}_{tank}$, and $\mathbb{E}_{pot}$. Assertion B'_{LMsys} is also the behavior of an element $(X'_{LMsys}, \mathsf{B}'_{LMsys})$, called *the composition of* $\{\mathbb{E}_{bat}, \mathbb{E}_{lMeter}, \mathbb{E}_{tank}, \mathbb{E}_{pot}\}$ where $X'_{LMsys} = X_{bat} \cup X_{Lmeter} \cup X_{tank} \cup X_{pot}$.

While B'_{LMsys} captures the combined constraints expressed by the behaviors of $\mathbb{E}_{bat}$, $\mathbb{E}_{lMeter}$, and $\mathbb{E}_{tank}$, and $\mathbb{E}_{pot}$, the set of port variables that is constrained by the assertion B'_{LMsys} is a proper *superset* of set of port variables $X_{LMsys} = \{f, l\}$ of $\mathbb{E}_{LMsys}$ as shown in Fig. 5. In accordance with Definition 2, this means that B'_{LMsys} cannot be the behavior of $\mathbb{E}_{LMsys}$. Behavior B_{LMsys} of $\mathbb{E}_{LMsys}$ is instead the extended projection of B'_{LMsys} onto $\{f, l\}$, which, in accordance with relation (2), means that B_{LMsys} can be expressed through equation $l = f$. The element $\mathbb{E}_{LMsys}$ is also called *the composition of* $\{\mathbb{E}_{bat}, \mathbb{E}_{lMeter}, \mathbb{E}_{tank}, \mathbb{E}_{pot}\}$ *onto* X_{LMsys}. Correspondingly, element $\mathbb{E}_{Esys}$ is the composition of $\{\mathbb{E}_{bat}, \mathbb{E}_{lMeter}, \mathbb{E}_{pot}\}$ onto $X_{Esys} = \{h, l\}$.

Now that the concept of element composition has been introduced, the formal definition follows.

Definition 3 (*Composition of elements (onto a set of variables)*) The *composition of a set of elements* $\{(X_1, \mathsf{B}_1), \ldots, (X_N, \mathsf{B}_N)\}$ *onto a non-empty set* $X \subseteq \bigcup_{i=1}^{N} X_i$, is the element $(X, \widehat{proj}_X(\bigcap_{i=1}^{N} \mathsf{B}_i))$. In the case where $X = \bigcup_{i=1}^{N} X_i$, the composition of $\{(X_1, \mathsf{B}_1), \ldots, (X_N, \mathsf{B}_N)\}$ onto X, is simply called the composition of $\{(X_1, \mathsf{B}_1), \ldots, (X_N, \mathsf{B}_N)\}$. □

In accordance with Definition 3 and as previously indicated, in the case where $X = \bigcup_{i=1}^{N} X_i$, the composition of $\{(X_1, \mathsf{B}_1), \ldots, (X_N, \mathsf{B}_N)\}$ onto X, is simply called the composition of $\{(X_1, \mathsf{B}_1), \ldots, (X_N, \mathsf{B}_N)\}$. This case is in accordance with the definition of composition of HRCs in [3–5].

In the case where $X \subset \bigcup_{i=1}^{N} X_i$, Definition 3 combines composition as defined in [3–5] with *port elimination* [3], also called *variable hiding* [66,67]. As shown in the example in Fig. 5, this case allows representing hierarchical systems. For example, in [70], the overall concept of elements and how they compose were used for representing C-programs as architecture models.

Given a set of elements $\mathcal{E}$, *the environment of an element* $\mathbb{E} \in \mathcal{E}$, denoted $\mathbb{E}_{Env_{\mathcal{E}}(\mathbb{E})}$, is the composition of $\mathcal{E} \setminus \{\mathbb{E}\}$. As an example of the environment of an element, given a subset $\mathcal{E}_{LMsys} = \{\mathbb{E}_{bat}, \mathbb{E}_{tank}, \mathbb{E}_{pot}, \mathbb{E}_{lMeter}\}$ of the elements shown in Fig. 5, the environment $\mathbb{E}_{Env_{\mathcal{E}_{LMsys}}(\mathbb{E}_{lMeter})}$ of $\mathbb{E}_{lMeter}$ is the composition of $\{\mathbb{E}_{bat}, \mathbb{E}_{tank}, \mathbb{E}_{pot}\}$.

3 Conditions of contracts for separating responsibilities

Based on the concepts presented in Sect. 2, this section introduces the concept of a *contract* containing assumptions and a guarantee, and also *conditions* that ensure that the guarantee is fulfilled.

3.1 Contracts

As mentioned in Sect. 1, the notion of contracts was first introduced in [1] as a pair of pre and post-conditions [2] to be used in formal specification of SW interfaces. In recent contract theory [3–5], developed within SPEEDS [6], the use of contracts is extended from formal specification of SW to serving as a central philosophy in systems engineering to support the design of heterogeneous systems. One of the key challenges that triggered the extension of contracts is the increasingly complex development environment of heterogeneous systems, characterized by distributed OEM (Original Equipment Manufacturer)/supplier chains [3].

In the context of an OEM/supplier chain, in order for a global intended property to be fulfilled by a composition of a set of elements, the OEM needs to distribute the responsibilities of fulfilling local properties between different elements that are to be integrated into the composition. Considering that these elements are to be developed by different suppliers, clearly defined interfaces and the separation of responsibilities between the different elements are paramount in order to support seamless integration. A *contract* addresses such concerns by assigning the responsibility that a certain property is fulfilled, to an element in the form of a *guarantee*, given that certain *assumptions* are fulfilled—a responsibility of the *environment* of the element.

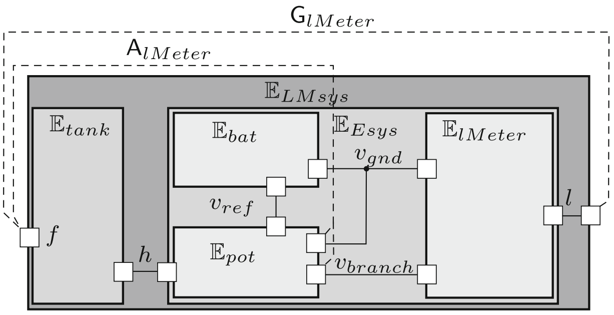


Fig. 6 A set of elements representing a "Level Meter system" and its parts, and the set of variables constrained by the assumption and the guarantee of a contract $\mathcal{C}_{lMeter} = (\{\mathsf{A}_{lMeter}\}, \mathsf{G}_{lMeter}, X_{lMeter})$

Although the discussion above focuses on the use of contracts for managing the complexity of OEM/supplier chains, the discussion can be generalized and is equally valid for any context where clear separations of responsibilities are desired.

A contract $(\{\mathsf{A}_i\}_{i=1}^N, \mathsf{G}, X)$ is a specification for an element $\mathbb{E}$ with a set of port variables X, expressing the intent that the behavior of the element is such that the guarantee G is fulfilled, given a set of elements containing $\mathbb{E}$ where its environment fulfills the assumptions in $\{\mathsf{A}_i\}_{i=1}^N$.

Definition 4 (*Contract*) A *contract* $\mathcal{C}$ is a tuple $(\mathscr{A}, \mathsf{G}, X)$, where

(i) G is an assertion, called *guarantee*;
(ii) $\mathscr{A}$ is a set of assertions $\{\mathsf{A}_i\}_{i=1}^N$ where each A_i is called an *assumption*; and
(iii) X is a set of variables. □

For the sake of readability, let $\mathsf{A}_{\mathscr{A}} = \bigcap_{j=1}^N \mathsf{A}_i$.

As an illustrative example of a contract, let $(\{\mathsf{A}_{lMeter}\}, \mathsf{G}_{lMeter}, X_{lMeter})$ be a contract $\mathcal{C}_{lMeter}$ where the set of port variables constrained by A_{lMeter} and G_{lMeter} are connected with dashed lines in Fig. 6. Guarantee G_{lMeter}, specified by equation $l = f$, expresses the intent that the indicated level, displayed by the meter, corresponds to the level in the tank. In the context of set of elements $\mathcal{E}_{LMsys} = \{\mathbb{E}_{bat}, \mathbb{E}_{tank}, \mathbb{E}_{pot}, \mathbb{E}_{lMeter}\}$, the responsibility that guarantee G_{lMeter} is fulfilled, is assigned to $\mathbb{E}_{lMeter}$, but only if the voltage measured between v_{branch} and v_{gnd} maps to a specific level in the tank, i.e. only if the environment $\mathbb{E}_{Env_{\mathcal{E}_{LMsys}}}(\mathbb{E}_{lMeter})$ fulfills the assumption A_{lMeter}, specified by equation $f = \frac{v_{branch} - v_{gnd}}{5V}$.

In general assume-guarantee theories [3,4,35,37–39,41–43,45,46,49–52,56–59] where ports are explicit, contract $\mathcal{C}_{lMeter}$ is not supported since assumption A_{lMeter} and guarantee G_{lMeter} are not specified only over the set of ports of $\mathbb{E}_{lMeter}$, i.e. over the voltage connections v_{branch} and v_{gnd}, and the indicated level l. In contrast to these theories, the present paper supports specifying a contract for $\mathbb{E}_{lMeter}$ where both A_{lMeter} and G_{lMeter} constrain port variables that are not in the set of port variables of $\mathbb{E}_{lMeter}$ as exemplified in Fig. 6. Note that the present paper also supports contracts that are limited to the set of ports of elements as in other assume-guarantee theories.

Contract $\mathcal{C}_{lMeter}$ can for example be used in a scenario where an OEM develops $\mathbb{E}_{lMeter}$ in-house while the development of elements $\mathbb{E}_{bat}$, $\mathbb{E}_{pot}$, and $\mathbb{E}_{tank}$ are outsourced to suppliers. The responsibility that the overall intended functionality of the LM-system, as expressed by G_{lMeter}, is fulfilled, is assigned to $\mathbb{E}_{lMeter}$ with the meaning that the OEM is not only responsible for ensuring the development of $\mathbb{E}_{lMeter}$, but also its successful integration with elements $\mathbb{E}_{bat}$, $\mathbb{E}_{tank}$, and $\mathbb{E}_{pot}$ into the composition of $\mathcal{E}_{LMsys}$ that fulfills G_{lMeter}. A successful integration of the elements can be ensured by the OEM, given that assumption A_{lMeter} is fulfilled. This is a responsibility of environment $\mathbb{E}_{Env_{\mathcal{E}_{LMsys}}(\mathbb{E}_{lMeter})}$ that is to be developed by the suppliers.

Note that assumption A_{lMeter} specifies a *property* that is intended to be fulfilled by the environment, it does not specify a *particular* structure of the environment except that f, v_{gnd}, v_{branch} are ports in the environment. That is, this assumption, and also assumptions of contracts in general, do not specify which or how many elements are in the environment, which sets of ports they have, or the total set of ports that are in the environment.

In the previously introduced example, the fact that contract $\mathcal{C}_{lMeter}$ is not limited to the set of ports of $\mathbb{E}_{lMeter}$ captures the intent of having the OEM being responsible over the integration of the elements in $\mathcal{E}_{LMsys}$. However, in general, contracts that are not limited to the set of ports of elements can be used to capture types of responsibility separation other than that of integration. As another example of a type of responsibility separation that such contracts can capture, consider the contract shown in Fig. 1. In this case, the fact that both assumptions and guarantees constrain port variables not of SOF does not mean that the developer of SOF is responsible for its integration with HW devices. Rather, this contract captures the perspective that the SW developer is responsible for developing SOF with the overall aim to fulfill REQ; the surrounding HW is simply considered as an enabler to realize this overall aim. An additional example of responsibility separation was also mentioned in Sect. 1, namely that when contracts are used to separate responsibilities of fulfilling overall safety properties.

3.2 Conditions on element and environment

Given a set of elements $\mathcal{E}$ and a contract $(\mathscr{A}, \mathsf{G}, X)$ for an element $\mathbb{E} = (X, \mathsf{B}) \in \mathcal{E}$, this section proposes the following respective conditions on element $\mathbb{E}$ and on its environment $\mathbb{E}_{Env_{\mathcal{E}}(\mathbb{E})}$:

(i) $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}$ and $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B}$; and
(ii) $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$ and $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G} \neq \emptyset$.

As will be shown in this section, these conditions ensure that *the guarantee is fulfilled*, which can expressed as

$$\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{B} \subseteq \mathsf{G}. \tag{3}$$

In fact, not only that, but condition (i) on the element $\mathbb{E}$ actually ensures that relation $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{B} \subseteq \mathsf{G}$ holds for *each set of elements* where the environment of $\mathbb{E}$ is such that condition (ii) holds. Furthermore, conditions (i) and (ii) also ensure that the trivial solution where $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{B} = \emptyset$, is avoided. That is, these conditions actually ensure that *the guarantee is non-trivially fulfilled*, i.e.

$$\emptyset \neq \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{B} \subseteq \mathsf{G}. \tag{4}$$

As will be shown, condition $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B}$ is needed to ensure relation (4) since $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G}$ can simply consist of one run, and this run can possibly be any run in $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G}$.

In contrast to [3–5] where trivial solution $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{B} = \emptyset$ is avoided only in the case where set of ports X is partitioned into inputs and outputs, the conditions that will presented in this section ensure that the trivial solution is avoided even when this is not the case. The conditions that will be presented also hold when the assumptions and the guarantee are not limited to constraining port variables in X; a case that is prohibited in [3–5].

In particular, as mentioned in Sect. 1, conditions (i) and (ii) support the case where an element and its environment are developed in *complete isolation* and where the only information is shared is a contract. In order to get a better understanding of this case, a scenario in the context of an OEM/supplier chain as previously presented, is examined. In the scenario, a contract $\mathcal{C} = (\mathscr{A}, \mathsf{G}, X)$ is used to outsource the development of an element $\mathbb{E} = (X, \mathsf{B})$. Specifically, the scenario can be described in three phases:

(I) a contract $\mathcal{C}$ is handed from the OEM to a supplier;
(II) the supplier develops an element $\mathbb{E} = (X, \mathsf{B})$ that is handed to the OEM; and
(III) the OEM integrates element $\mathbb{E}$ with a set of elements $\{\mathbb{E}_i\}_{i=1}^N$ into the composition of $\mathcal{E} = \{\mathbb{E}\} \cup \{\mathbb{E}_i\}_{i=1}^N$.

As expressed in phases (I–II), the development of the element $\mathbb{E}$ is guided only by the information available in contract $\mathcal{C}$, i.e. without access to the environment $\mathbb{E}_{Env_{\mathcal{E}}(\mathbb{E})}$ of $\mathbb{E}$. Therefore, in order for the composition of $\mathcal{E}$ in phase (III) to be such that relation (4) holds with respect to $\mathcal{C}$, conditions must be enforced on the element $\mathbb{E}$ such that relation (4) holds not just for the set $\mathcal{E}$, but rather for *each set of elements* containing $\mathbb{E}$ where the environment is such that certain conditions hold. The conditions on the element is to be enforced by the supplier, while the conditions on the environment is to be ensured by the OEM.

As will be shown in the following sections, conditions (i) and (ii) are, in fact, instantiations of such conditions for the element and its environment, respectively. The subset of conditions (i) and (ii) that ensure that relation $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{B} \subseteq \mathsf{G}$ holds will first be derived in Sect. 3.2.1, followed by the remaining subset that ensure that also relation $\emptyset \neq \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{B}$ holds in Sect. 3.2.2.

3.2.1 Conditions ensuring guarantee is fulfilled

As previously mentioned, in the context of $\mathcal{E}$, the responsibility that the guarantee is fulfilled, is assigned to $\mathbb{E}$, given that *the environment of* $\mathbb{E}$ *fulfills the assumptions*. This means that it must hold that

$$\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}. \tag{5}$$

Supposing that relation (5) holds, it follows that relation (3) holds if

$$\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}. \tag{6}$$

Note that if $\mathscr{A} = \emptyset$, then relation (6) simplifies to $\mathsf{B} \subseteq \mathsf{G}$. Condition (6) on the element has previously been identified in e.g. [3–5,33,34,64].

Relation (6) is a *sufficient*, but not necessary condition for relation (3) to hold considering a *specific* set of elements where the environment is such that relation (5) holds. As expressed in the following proposition, relation (6) is also a *necessary* condition in order for relation (3) to hold for *each set of elements* where the environment is such that relation (5) holds.

Proposition 2 *Consider a contract* $\mathcal{C} = (\mathscr{A}, \mathsf{G}, X)$ *and an element* $\mathbb{E} = (X, \mathsf{B})$. *It holds that* $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}$, *if and only if* $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{B} \subseteq \mathsf{G}$ *for each set of elements* $\mathcal{E} \ni \mathbb{E}$ *where* $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$.

Proof Consider a contract $\mathcal{C} = (\mathscr{A}, \mathsf{G}, X)$ and an element $\mathbb{E} = (X, \mathsf{B})$.

For the if-only case, assume that $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}$. Consider an arbitrary set of elements $\mathcal{E} \ni \mathbb{E}$ where $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$. Relations $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}$ and $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$ imply that $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{B} \subseteq \mathsf{G}$. Since $\mathcal{E}$ was chosen arbitrarily, it means that relation $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{B} \subseteq \mathsf{G}$ also holds for each set of elements $\mathcal{E} \ni \mathbb{E}$ where $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$.

For the if case, assume that relation $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{B} \subseteq \mathsf{G}$ holds for each set of elements $\mathcal{E} \ni \mathbb{E}$ where $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$. Assume that $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \nsubseteq \mathsf{G}$, which will be shown to lead to a contradiction. Assume a set of elements $\mathcal{E} \ni \mathbb{E}$ where $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} = \mathsf{A}_{\mathscr{A}}$, which means that it follows that $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} = \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{B} \subseteq \mathsf{G}$. This contradicts the fact that $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \nsubseteq \mathsf{G}$, which means that it must rather hold that $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}$, which concludes the proof. □

Proposition 2 expresses that relation (6) on the element $\mathbb{E}$ is a necessary and sufficient condition in order to obtain an element and its environment in phase (III) such that relation (3) holds in general, given that relation (5) on the environment holds. However, relation (3) trivially holds if the behavior of the composition of the element and the environment is empty, which would imply that relation (4) does not hold. Therefore, relation (4) does not follow from relations (5) and (6), which means that additional conditions must be imposed on the environment and on the element in order to ensure that the trivial solution is avoided. These conditions will be examined in Sect. 3.2.2, which now follows.

3.2.2 Conditions ensuring non-triviality

This section presents contract conditions, additional to those established in Sect. 3.2.1, in order to ensure that the guarantee of a contract is non-trivially fulfilled in a context such as the considered OEM/supplier scenario. Notably, this context assumes that: (a) it cannot be ensured that the set of ports of an element are partitioned into inputs and outputs; and (b) the element and its environment are developed in complete isolation. In another context, e.g. when inputs and outputs are well-defined, these additional conditions may not always be applicable. Such other contexts, and the corresponding contract conditions that are then applicable, will be discussed in Sect. 7.1. The rest of this section, as well as Sects. 4–6, will focus only on contexts characterized by (a) and (b).

Consider a contract $\mathcal{C} = (\mathscr{A}, \mathsf{G}, X)$ and an element $\mathbb{E} = (X, \mathsf{B})$, and assume a set of elements $\mathcal{E}$ containing $\mathbb{E}$ such that its environment is such that $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G} = \emptyset$. Notably, if the behavior of the composition of the element and its environment is non-empty, i.e. if $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{B} \neq \emptyset$, then it must follow that $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{B} \nsubseteq \mathsf{G}$ since none of the runs in $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})}$ are in G. Hence, in order for relation (4) to hold, the environment must be such that $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G} \neq \emptyset$. This insight is summarized in the following proposition.

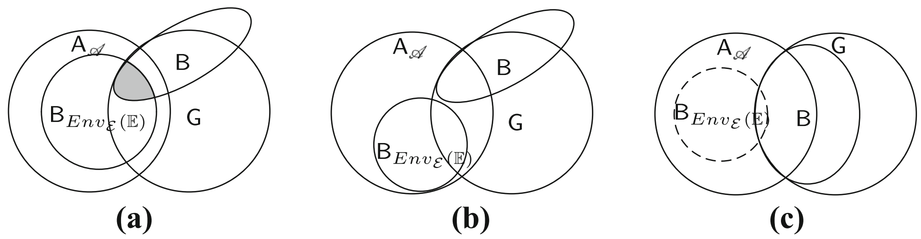
Proposition 3 *Given a contract* $\mathcal{C} = (\mathscr{A}, \mathsf{G}, X)$ *and a set of elements* $\mathcal{E}$ *containing an element* $\mathbb{E} = (X, \mathsf{B})$*, it holds that* $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G} \neq \emptyset$ *if* $\emptyset \neq \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{B} \subseteq \mathsf{G}$.

Proof Consider a contract $\mathcal{C} = (\mathscr{A}, \mathsf{G}, X)$ and a set of elements $\mathcal{E}$ containing an element $\mathbb{E} = (X, \mathsf{B})$. Assume that $\emptyset \neq \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{B} \subseteq \mathsf{G}$. Intersecting both sides of this relation with $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})}$, yields $\emptyset \neq \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{B} \subseteq \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G}$. This implies that $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G} \neq \emptyset$. □

Now that necessary condition $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G} \neq \emptyset$ on the environment of $\mathbb{E}$ has been identified in order for relation (4) to hold, a complementary sufficient condition on the element $\mathbb{E}$ is examined in order to ensure that relation (4) holds.

Consider that relation $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}$ on the element $\mathbb{E}$ holds. As shown in the Venn diagram in Fig. 7a, if $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$ and $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G} \neq \emptyset$, it is possible that relation (4) holds. However, as shown in Fig. 7b, this is not true for all cases. In fact, since $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G}$ can simply consist of one run, and this run can possibly be any run in $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G}$, in order to ensure that $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{B} \neq \emptyset$, it must hold that $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B}$ as shown in Fig. 7c.

These insights are now summarized in the following theorem.



$\mathsf{A}_{\mathscr{A}}$
B
$\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})}$
G
(a)
(b)
(c)

Fig. 7 In **a**, a Venn diagram shows a case where $\emptyset \neq \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{B} \subseteq \mathsf{G}$ holds and **b** shows a case where this does not hold. In **c**, a Venn diagram is shown where $\emptyset \neq \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{B} \subseteq \mathsf{G}$ for each set $\mathcal{E}$ where $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$ and $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G} \neq \emptyset$

Theorem 1 *Given a contract $\mathcal{C} = (\mathscr{A}, \mathsf{G}, X)$ and an element $\mathbb{E} = (X, \mathsf{B})$ where $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}$, it holds that*

$$\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B},$$

if and only if $\emptyset \neq \mathsf{B} \cap \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{G}$ for each set of elements $\mathcal{E}$ containing $\mathbb{E}$ where $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G} \neq \emptyset$ and $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$.

Now follows a clarification of Theorem 1 using quantifiers. Given a contract $\mathcal{C} = (\mathscr{A}, \mathsf{G}, X)$ and an element $\mathbb{E} = (X, \mathsf{B})$ where $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}$:

$$\begin{aligned}&\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B} \iff \\ &\quad (\forall \mathcal{E} \ni \mathbb{E} : ((\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G} \neq \emptyset \wedge \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}) \implies \emptyset \neq \mathsf{B} \cap \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{G})).\end{aligned}$$

Proof Consider a contract $\mathcal{C} = (\mathscr{A}, \mathsf{G}, X)$ and an element $\mathbb{E} = (X, \mathsf{B})$ such that $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}$.

For the if-only part, assume that $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B}$. Furthermore, consider a set of elements $\mathcal{E}$ containing $\mathbb{E}$ where $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$ and $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G} \neq \emptyset$. Relations $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B}$ and $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$ imply that $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G} \subseteq \mathsf{B}$. This and the fact that $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G} \neq \emptyset$ imply that $\emptyset \neq \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G} \subseteq \mathsf{B}$. Intersecting both sides of this relation with $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})}$ yields $\emptyset \neq \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G} \subseteq \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{B}$. This implies that $\mathsf{B}_{Env_{\mathscr{A}}(\mathbb{E})} \cap \mathsf{B} \neq \emptyset$. This and since relations $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}$ and $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$ imply that $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{B} \subseteq \mathsf{G}$, it can be concluded that $\emptyset \neq \mathsf{B} \cap \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{G}$. Considering that $\mathcal{E}$ was chosen arbitrarily, it follows that relation $\emptyset \neq \mathsf{B} \cap \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{G}$ also holds for each set of elements $\mathcal{E}$ containing $\mathbb{E}$ where $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$ and $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G} \neq \emptyset$. This completes the if-only part of the proof.

For the if part, assume that for each set of elements $\mathcal{E}$ containing $\mathbb{E}$ where $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$ and $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G} \neq \emptyset$, it follows that $\emptyset \neq \mathsf{B} \cap \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{G}$. Assume that $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \nsubseteq \mathsf{B}$, which will be shown to lead to a contradiction. This means that there exists a run ω such that $\omega \in \mathsf{A}_{\mathscr{A}} \cap \mathsf{G}$ and $\omega \notin \mathsf{B}$. Furthermore, assume that there exists a set of elements $\mathcal{E}$ containing $\mathbb{E}$ where $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} = \{\omega\}$. This and the fact that $\omega \in \mathsf{A}_{\mathscr{A}} \cap \mathsf{G}$ imply that both relations $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$ and $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G} \neq \emptyset$ hold. As was assumed, this means that it follows that $\emptyset \neq \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{B} \subseteq \mathsf{G}$; however, this is a contradiction since $\omega \notin \mathsf{B}$ implies that $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{B} = \emptyset$. It follows that $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \nsubseteq \mathsf{B}$ cannot be true, which means that it must hold that $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B}$, which concludes the proof. □

Given that relation $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}$ on the element $\mathbb{E}$ holds, Theorem 1 expresses necessary and sufficient condition $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B}$ on element $\mathbb{E}$ such that, for each set of elements $\mathcal{E} \ni \mathbb{E}$ where $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$ and $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G} \neq \emptyset$, the composition of $\mathcal{E}$ non-trivially fulfills G. The condition expressed in Theorem 1 holds regardless of the considered domain, e.g. SW, electrical, mechanical, etc, and does not require the set of port variables of the element to be partitioned into inputs and outputs.

As a technical remark, given that conditions $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}$, $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B}$, and $\mathsf{G} \cap \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \neq \emptyset$ hold, it can be noted that condition $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$ can actually be relaxed to $\mathsf{G} \cap \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$. However, this relaxed condition can actually always be embedded in condition $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$. This can be understood by considering that relaxed condition $\mathsf{G} \cap \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$ can be rewritten as $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}} \cup \overline{\mathsf{G}}$ where $\overline{\mathsf{G}}$ is the complement of G. This means that while condition $\mathsf{G} \cap \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$ is indeed more relaxed than $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$, the two conditions are, in fact, equally expressive.

Theorem 1 quantifies over sets of elements, and, thus, also over various environments of element $\mathbb{E}$. To clarify the respective conditions on $\mathbb{E}$ and its environment $\mathbb{E}_{Env_{\mathcal{E}}(\mathbb{E})}$ in a *specific set of elements*, the following corollary simplifies Theorem 1 by removing the quantification over sets of elements.

Corollary 1 *Given a contract* $\mathcal{C} = (\mathscr{A}, \mathsf{G}, X)$ *and a set of elements* $\mathcal{E}$ *containing an element* $\mathbb{E} = (X, \mathsf{B})$*, it holds that* $\emptyset \neq \mathsf{B} \cap \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{G}$ *if both the following conditions hold:*

(i) element $\mathbb{E}$ *is such that*

$$\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}, \textit{ and} \tag{7}$$

$$\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B}; \tag{8}$$

ii environment $\mathbb{E}_{Env_{\mathcal{E}}(\mathbb{E})}$ *of* $\mathbb{E}$ *is such that*

$$\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}, \textit{ and} \tag{9}$$

$$\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G} \neq \emptyset. \tag{10}$$

Proof Trivially follows from Theorem 1. □

In the context of the scenario presented in the beginning of this section, Corollary 1 specifies condition (i) that the supplier needs to meet and condition (ii) that the OEM needs to meet in order to ensure that the integration of $\mathbb{E}$ with the elements in $\{\mathbb{E}_i\}_{i=1}^{N}$ in phase (III) results in that the guarantee is non-trivially fulfilled by the composition of $\{\mathbb{E}\} \cup \{\mathbb{E}_i\}_{i=1}^{N}$, i.e. that relation (4) holds.

As a conclusion to this section, it is examined whether element $\mathbb{E}_{lMeter}$ and its environment $\mathbb{E}_{Env_{\mathcal{E}_{LMsys}}(\mathbb{E}_{lMeter})}$ in the context of the set of elements $\mathcal{E}_{LMsys} = \{\mathbb{E}_{bat}, \mathbb{E}_{tank}, \mathbb{E}_{pot}, \mathbb{E}_{lMeter}\}$ shown in Fig. 5 are such that respective conditions (i) and (ii) of Corollary 1 holds with respect to the contract $\mathcal{C}_{lMeter}$ shown in Fig. 6. As expressed in condition (i) of Corollary 1, it must hold that

$$\mathsf{A}_{lMeter} \cap \mathsf{B}_{lMeter} \subseteq \mathsf{G}_{lMeter}, \text{ and} \tag{11}$$

$$\mathsf{A}_{lMeter} \cap \mathsf{G}_{lMeter} \subseteq \mathsf{B}_{lMeter}. \tag{12}$$

Furthermore, as expressed in condition (ii) of Corollary 1, it must hold that

$$\mathsf{B}_{Env_{\mathcal{E}_{LMsys}}(\mathbb{E}_{lMeter})} \subseteq \mathsf{A}_{lMeter}, \text{ and} \tag{13}$$

$$\mathsf{B}_{Env_{\mathcal{E}_{LMsys}}(\mathbb{E}_{lMeter})} \cap \mathsf{G}_{lMeter} \neq \emptyset. \tag{14}$$

By applying the operation of intersection, $\mathsf{A}_{lMeter} \cap \mathsf{B}_{Lmeter}$ yields an assertion specified by equations $f = (v_{branch} - v_{gnd})/5V$ and $l = (v_{branch} - v_{gnd})/5V$. Due to the fact that equation $f = l$, which specifies G_{lMeter}, can be obtained by combining the equations specifying $\mathsf{A}_{lMeter} \cap \mathsf{B}_{Lmeter}$, relation (11) holds. Furthermore, since $\mathsf{A}_{lMeter} \cap \mathsf{G}_{lMeter}$ is an assertion specified by equations $l = f$ and $f = (v_{branch} - v_{gnd})/5V$, which can be combined into equation $l = (v_{branch} - v_{gnd})/5V$ that specifies B_{lMeter}, relation (12) also holds.

In accordance with Sect. 2.3, the behavior of $Env_{\mathscr{A}_{LMsys}}(\mathbb{E}_{lMeter})$ is the intersection of the behaviors of $\mathbb{E}_{tank}$, $\mathbb{E}_{Esys}$, and $\mathbb{E}_{pot}$, which are specified by equations $f = h$, $h = (v_{branch} - v_{gnd})/(v_{branch} - v_{gnd})$, and $v_{branch} - v_{gnd} = 5V$. Due to the fact that equation $f = (v_{branch} - v_{gnd})/5V$ can be obtained by combining these equations, relation (13) holds. Furthermore, due to the fact that $\mathsf{B}_{Env_{\mathcal{E}_{LMsys}}(\mathbb{E}_{lMeter})}$ does not constrain l, it holds that $\mathsf{B}_{Env_{\mathcal{E}_{LMsys}}(\mathbb{E}_{lMeter})} \cap \mathsf{G}_{lMeter}$ is non-empty, i.e. relation (14) also holds.

Considering that relations (11)–(14) hold, Corollary 1 implies that guarantee G_{lMeter} is non-trivially fulfilled by the composition of $\mathcal{E}_{LMsys}$.

4 Scoping conditions for specifying contracts

Section 3 presented conditions on an element and its environment where these conditions ensure that the guarantee of a given contract is non-trivially fulfilled. Previous general assume-guarantee theories [3,4,35,37–39,41–43,45,46,49–52,56–59] that explicitly consider ports, do not allow guarantees and assumptions to be specified over ports that are not in the set of ports of an element. This means that the present paper is strictly more expressive than previous assume-guarantee theories with respect to how a contract can be specified. However, the increased expressiveness is not unlimited since there are necessary restrictions on the set of port variables constrained by the assumptions and the guarantee of a contract in order for an element and its environment to be such that conditions (i) and (ii) of Corollary 1 hold. Therefore, to facilitate the specification of contracts in practice, this section introduces conditions, called *scoping conditions*, which ensure that these restrictions are not violated without limiting expressiveness.

In order to introduce and motivate these scoping conditions, a definition and two propositions, will first be presented. The definition, which now follows, characterizes a necessary condition in order for conditions (i) and (ii) of Corollary 1 to hold.

Definition 5 An assertion W *restricts* a variable x if W constrains x and there does not exist an assertion W' where $x \notin X_{\mathsf{W}'}$ and $\emptyset \neq \mathsf{W}' \subseteq \mathsf{W}$. □

In accordance with Definition 5, if an assertion W restricts x, then it is necessary for an assertion W' to constrain x in order to non-trivially fulfill W.

As a first example, guarantee G_{Esys}, specified through equation $l = f$, restricts both l and f. Consider an architecture containing an element $\mathbb{E}$ where $\mathsf{B} \cap \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})}$ does not constrain both l and f. In accordance with Definition 5, it does not hold that $\emptyset \neq \mathsf{B} \cap \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{G}_{Esys}$. Formulated differently, the fact that G_{Esys} restricts a port variable that is not constrained by $\mathsf{B} \cap \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})}$ means conditions (i) and (ii) of Corollary 1 will not both hold for $\mathbb{E}$ and $Env_{\mathcal{E}}(\mathbb{E})$, respectively.

In the first example, the port variables constrained by G_{Esys} are also the port variables it restricts; however, in general, an assertion can constrain a port variable without restricting it. As a second example, consider an assertion $\mathsf{W}_{x>0 \Rightarrow y=0}$ expressed through logical formula $x > 0 \Rightarrow y = 0$. The assertion $\mathsf{W}_{x>0 \Rightarrow y=0}$ constrains y, but it does not restrict it since in the case of e.g. an assertion $\mathsf{W}_{x=0}$ expressed through equation $x = 0$, it does indeed hold that $\emptyset \neq \mathsf{W}_{x=0} \subseteq \mathsf{W}_{x>0 \Rightarrow y=0}$ despite the fact that $\mathsf{W}_{x=0}$ does not constrain y.

As shown in the second example, under the conditions that an assertion W constrains a variable x, but does not restrict it, then it is possible that there exists a case where another assertion W'' does not constrain x, but where it still holds that $\emptyset \neq \mathsf{W}'' \subseteq \mathsf{W}$. As will be shown in the following proposition, under such conditions, it holds that there exists another assertion W' that does not constrain x, but where $\emptyset \neq \mathsf{W}'' \subseteq \mathsf{W}'$ holds regardless of the specific runs that are in W''. This means that if the intent is that W is to specify an assertion that is to non-trivially fulfill W and not constrain x, then it is possible to replace W with an assertion W' that does not constrain x, but that still specifies the exact same assertions.

Proposition 4 *Given a set of variables X and an assertion W, there exists an assertion W' where $X_{\mathsf{W}'} \subseteq X$ such that for each assertion W'' where $X_{\mathsf{W}''} \subseteq X$:*

$$\emptyset \neq \mathsf{W}'' \subseteq \mathsf{W}' \textit{ if and only if } \emptyset \neq \mathsf{W}'' \subseteq \mathsf{W}.$$

Lemma 1 *Given two assertions W and W' where $\mathsf{W} \cup \mathsf{W}' \neq \emptyset$, it holds that $X_{\mathsf{W} \cup \mathsf{W}'} \subseteq X_{\mathsf{W}} \cup X_{\mathsf{W}'}$.*

The proof of Lemma 1 is found in “Appendix A”. The proof of Proposition 4 follows.

Proof Assume that W' is the union of each assertion W'' where $X_{\mathsf{W}''} \subseteq X$ and $\emptyset \neq \mathsf{W}'' \subseteq \mathsf{W}$. The rest of the proof trivially follows from Lemma 1. □

Note that in order for there to exist an assertion W'' where $X_{\mathsf{W}''} \subseteq X$ and $\emptyset \neq \mathsf{W}'' \subseteq \mathsf{W}$, in accordance with Definition 5, it is necessary that W does not restrict variables that are not in X.

Given previously presented assertion $\mathsf{W}_{x>0 \Rightarrow y=0}$ and set $\{x\}$. In accordance with Proposition 4, there exists an assertion W' where $X_{\mathsf{W}'} \subseteq \{x\}$ such that for each assertion W'' where $X_{\mathsf{W}''} \subseteq \{x\}$, it holds that $\emptyset \neq \mathsf{W}'' \subseteq \mathsf{W}_{x>0 \Rightarrow y=0}$ if and only if $\emptyset \neq \mathsf{W}'' \subseteq \mathsf{W}'$. For example, W' can be expressed through the inequality $x \leq 0$.

Proposition 5 *Given two assertions W and W' where $\mathsf{W} \cap \mathsf{W}' \neq \emptyset$, it holds that $X_{\mathsf{W} \cap \mathsf{W}'} \subseteq X_{\mathsf{W}} \cup X_{\mathsf{W}'}$.*

The proof of Proposition 5 is found in “Appendix A”.

Definition 5 and Propositions 4 and 5 will now be applied on two examples to motivate the need and the basis for the scoping rules that will be proposed for specifying contracts. In the following two examples, the considered use case is to establish that the guarantee of a contract is non-trivially fulfilled in a set of elements containing an element with a set of port variables X_{Esys} and where X_{tank} is the set of port variables of the environment of this element.

What will be shown in the two examples is that, regardless of the specific runs in the assumptions and guarantee of a contract, it is indeed necessary that the assumptions and guarantee do not restrict variables in $X_{Esys} \cup X_{tank}$ in order for condition (i) and (ii) of Corollary 1 to hold. Furthermore, it will also be shown that if the assumptions and guarantee constrain port variables in $X_{Esys} \cup X_{tank}$ without restricting them, then the assumptions and guarantee can be reformulated to constrain a subset of $X_{Esys} \cup X_{tank}$ and still specify the exact same element and environment behaviors constraining a subset of X_{Esys} and X_{tank}, respectively.

Example 1a Consider a contract $\mathcal{C}'_{Esys} = (\{\mathsf{A}'_{Esys}\}, \mathsf{G}'_{Esys}, X_{Esys})$ as shown in Fig. 8a.

Assume that conditions (i) and (ii) of Corollary 1 hold respectively for $\mathbb{E}_{Esys}$ and $\mathbb{E}_{tank}$ with respect to $\mathcal{C}'_{Esys}$. From condition (ii), it follows that $\emptyset \neq \mathsf{B}_{tank} \subseteq \mathsf{A}'_{Esys}$. As shown

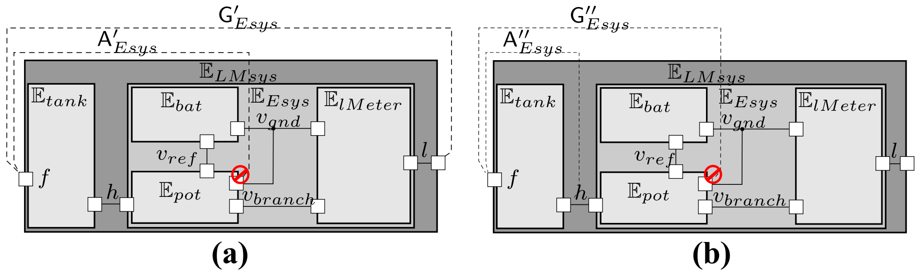


Fig. 8 In **a** and **b**, two contracts $(\{\mathsf{A}'_{Esys}\}, \mathsf{G}'_{Esys}, X_{Esys})$ and $(\{\mathsf{A}''_{Esys}\}, \mathsf{G}''_{Esys}, X_{Esys})$ are shown where none of them are scope-compliant with respect to X_{tank}

in Fig. 8a, port variable v_{gnd} is constrained by A'_{Esys}, but v_{gnd} is not a port variable of $\mathbb{E}_{tank}$, i.e. the environment of $\mathbb{E}_{Esys}$. In accordance with Definition 2, this means that v_{gnd} is not constrained by B_{tank}. This means, in accordance with Definition 5 and considering that $\emptyset \neq \mathsf{B}_{tank} \subseteq \mathsf{A}'_{Esys}$, that A'_{Esys} does not restrict v_{gnd}. That is, the fact that condition (ii) holds, implies that A'_{Esys} does not restrict v_{gnd}.

Furthermore, considering X_{tank} as given, in accordance with Proposition 4, there exists an assertion A^{new}_{Esys} that constrains a subset of X_{tank} and where, for each element $(X_{tank}, \mathsf{B}'_{tank})$, it holds that $\emptyset \neq \mathsf{B}'_{tank} \subseteq \mathsf{A}'_{Esys}$ if and only if $\emptyset \neq \mathsf{B}'_{tank} \subseteq \mathsf{A}^{new}_{Esys}$.

This example shows that it is necessary that assumption $\mathsf{A}_{\mathscr{A}}$ does not restrict any port variable that is not in $X_{Env_{\mathcal{E}}(\mathbb{E})}$. Furthermore, if $\mathsf{A}_{\mathscr{A}}$ does constrain a port variable $x \notin X_{Env_{\mathcal{E}}(\mathbb{E})}$ without restricting it, then it is possible to replace A'_{Esys} with an assertion A^{new}_{Esys} that does not constrain x, but that still specifies the exact same behaviors constraining a subset of X_{tank}. □

Example 1b In Fig. 8a, a contract $\mathcal{C}''_{Esys} = (\{\mathsf{A}''_{Esys}\}, \mathsf{G}''_{Esys}, X_{Esys})$ is shown.

Similar to Example 1a, assume that conditions (i) and (ii) of Corollary 1 hold respectively for $\mathbb{E}_{Esys}$ and $\mathbb{E}_{tank}$ with respect to $\mathcal{C}''_{Esys}$. Conditions (i) and (ii) imply that $\emptyset \neq \mathsf{A}''_{Esys} \cap \mathsf{B}_{Esys} \subseteq \mathsf{G}''_{Esys}$. As shown in Fig. 8b, the port variable v_{gnd} is constrained by G''_{Esys}, but v_{gnd} is not a port variable of $\mathbb{E}_{Esys}$. This and considering that $v_{gnd} \notin X_{\mathsf{A}''_{Esys}}$ imply, in accordance with Definition 2 and Proposition 5, that v_{gnd} is not constrained by $\mathsf{A}''_{Esys} \cap \mathsf{B}_{Esys}$. It follows, in accordance with Definition 5 and considering that $\emptyset \neq \mathsf{A}''_{Esys} \cap \mathsf{B}_{Esys} \subseteq \mathsf{G}''_{Esys}$, that G''_{Esys} does not restrict v_{gnd}. That is, the fact that condition (i) and (ii) hold implies that G''_{Esys} does not restrict v_{gnd}.

In addition, considering set $X_{\mathsf{A}''_{Esys}} \cup X_{Esys}$ as given, in accordance with Propositions 4 and 5, there exists an assertion G^{new}_{Esys} that constrains a subset of $X_{\mathsf{A}''_{Esys}} \cup X_{Esys}$ and where, for each element $(X_{Esys}, \mathsf{B}'_{Esys})$, it holds that $\emptyset \neq \mathsf{A}''_{Esys} \cap \mathsf{B}'_{Esys} \subseteq \mathsf{G}''_{Esys}$ if and only if $\emptyset \neq \mathsf{A}''_{Esys} \cap \mathsf{B}'_{Esys} \subseteq \mathsf{G}^{new}_{Esys}$. That is, similar to Example 1a, it is possible to replace G''_{Esys} with an assertion G^{new}_{Esys} that does not constrain v_{gnd}, but that still specifies the exact same behaviors constraining a subset of $X_{\mathsf{A}''_{Esys}} \cup X_{Esys}$. □

Consider a contract $(\mathscr{A}, \mathsf{G}, X)$ and a set of port variables $X_{Env_{\mathcal{E}}(\mathbb{E})}$. As expressed in Example 1a, in order for condition (ii) of Corollary 1 to hold for an element $(X_{Env_{\mathcal{E}}(\mathbb{E})}, \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})})$, it is necessary that assumptions $\mathsf{A}_{\mathscr{A}}$ do not restrict any port variable that is not in $X_{Env_{\mathcal{E}}(\mathbb{E})}$. Similarly, as expressed in Example 1b, in order for condition (i) of Corollary 1 to hold for an element (X, B), it is necessary that guarantee G does not restrict any port variable that is not in $X \cup X_{\mathsf{A}_{\mathscr{A}}}$.

Furthermore, consider that $\mathsf{A}_{\mathscr{A}}$ and G constrain port variables that are not in respective sets $X_{Env_{\mathcal{E}}(\mathbb{E})}$ and $X \cup X_{\mathsf{A}_{\mathscr{A}}}$, but where $\mathsf{A}_{\mathscr{A}}$ and G do not restrict these port variables. As indicated in Examples 1a and 1b, in such a case, it is in fact redundant to constrain such port variables. That is, these examples indicate that $\mathsf{A}_{\mathscr{A}}$ and G could be reformulated to *not constrain such port variables* and yet *specify the same element and environment behavior*.

These indications are formalized in the following theorem.

Theorem 2 *Given a contract* $(\mathscr{A}, \mathsf{G}, X)$ *and set of variables* $X_{Env_{\mathcal{E}}(\mathbb{E})}$*, there exists a contract* $(\mathscr{A}', \mathsf{G}', X)$ *where:*

(a) $X_{\mathsf{A}_{\mathscr{A}'}} \subseteq X_{Env_{\mathcal{E}}(\mathbb{E})}$*; and*
(b) $X_{\mathsf{G}'} \subseteq X_{Env_{\mathcal{E}}(\mathbb{E})} \cup X$

such that for each set of elements containing an element (X, B) *and where pair* $(X_{Env_{\mathcal{E}}(\mathbb{E})}, \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})})$ *is the environment of* (X, B)*:*

(i') $\mathsf{A}_{\mathscr{A}'} \cap \mathsf{B} \subseteq \mathsf{G}'$ *and* $\mathsf{A}_{\mathscr{A}'} \cap \mathsf{G}' \subseteq \mathsf{B}$*, and*
(ii') $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}'}$ *and* $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G}' \neq \emptyset$*,*

if and only if

(i) $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}$ *and* $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B}$*, and*
(ii) $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$ *and* $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G} \neq \emptyset$*.*

The proof of Theorem 2 is found in "Appendix A".

Consider the task of specifying a contract $(\mathscr{A}, \mathsf{G}, X)$ with the intent that condition (i) and (ii) are to hold respectively for an element (X, B) and its environment $(X_{Env_{\mathcal{E}}(\mathbb{E})}, \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})})$ in a set of elements. As expressed in Theorem 2, for such a task, it is sufficient to only allow contracts to be specified in accordance with conditions (a) and (b); that is, no expressiveness is lost by only allowing such contracts. In the following, given a contract $\mathcal{C} = (\mathscr{A}, \mathsf{G}, X)$ and a set of variables $X_{Env_{\mathcal{E}}(\mathbb{E})}$, conditions (a) and (b) will be called *scoping conditions for* $\mathcal{C}$ *and* $X_{Env_{\mathcal{E}}(\mathbb{E})}$.

Suppose that either or both of scoping conditions (a) and (b) are violated for a given contract $\mathcal{C} = (\mathscr{A}, \mathsf{G}, X)$ and a set of variables $X_{Env_{\mathcal{E}}(\mathbb{E})}$. For example, consider that $X_{\mathsf{A}_{\mathscr{A}'}} \setminus X_{Env_{\mathcal{E}}(\mathbb{E})} = \{x\}$. Notably, either $\mathsf{A}_{\mathscr{A}'}$ restricts x or it constrains it without restricting it. As previously mentioned and in accordance with Definition 5, the former case ensures that conditions (i) and (ii) of Corollary 1 cannot hold. Considering the latter case, in accordance with Theorem 2, constraining x is indeed redundant. Thus, the scoping conditions of Theorem 2 constitute as checks that either detect the violation of conditions (i) and (ii) or redundantly constrained port variables.

Note that in accordance with Sect. 2.1.2, the set of variables constrained by an assertion can only be derived from the runs that the assertion contains. However, as indicated in Sect. 2.1, in practice, an assertion would typically be expressed through a constraint, explicitly specified over a set of variables, e.g. as the assertion $\mathsf{W}_{y=x}$ shown in Fig. 3a. Assuming the generic case when the set of variables over which the constraint is specified is equal to the set of variables constrained by the assertion, the fact that scoping conditions (a) and (b) hold for a contract $\mathcal{C} = (\mathscr{A}, \mathsf{G}, X)$ and a set of port variables $X_{Env_{\mathcal{E}}(\mathbb{E})}$, can actually be established by considering only X, $X_{Env_{\mathcal{E}}(\mathbb{E})}$, and the sets of port variables over which assumptions $\mathsf{A}_{\mathscr{A}}$ and guarantee G are specified.

To illustrate the use of conditions (a) and (b) of Theorem 2, consider again Examples 1a and 1b shown in Fig. 8. Scoping condition (a) is violated for contract $\mathcal{C}'_{Esys}$ and X_{tank} due to the fact that $X_{\mathsf{A}_{Esys}} \nsubseteq X_{tank}$. Furthermore, considering contract $\mathcal{C}''_{Esys}$ and set of port variables X_{tank}, scoping condition (b) does not hold since $X_{\mathsf{G}''_{Esys}} \nsubseteq X_{tank} \cup X_{Esys}$. Thus, checking whether scoping conditions (a) and (b) hold or not, can be done without explicitly considering the runs that are in these assumptions and guarantees. As previously mentioned, the violation of the scoping conditions means that either conditions (i) and (ii) of Corollary 1 are violated or that more port variables are constrained than what is necessary.

5 Contract properties consistency and compatibility

Section 4 described necessary restrictions on the sets of port variables constrained by assumptions and the guarantee of a contract, in order for conditions (i) and (ii) of Corollary 1 to hold. In contrast to Sect. 4, this section presents sufficient and necessary conditions for the *existence of a set of elements* where an element and its environment is such that conditions (i) and (ii) of Corollary 1 holds with respect to the contract. If such a set of elements exists, then the contract is said to be *consistent* and *compatible*.

In order to get a better understanding of when the properties consistency and compatibility are relevant, the scenario presented in Sect. 3.2 is examined. Considering phase (I) of the scenario, the expectation of the OEM when handing over contract $\mathcal{C} = (\mathscr{A}, \mathsf{G}, X)$ to the supplier, is that the supplier will deliver an element in phase (II) such that condition (i) of Corollary 1 holds with respect to $\mathcal{C}$. However, in order for the supplier to be able to meet this expectation from the OEM, contract $\mathcal{C}$ needs to be such that there actually exists an element (X, B) that is such that condition (i) of Corollary 1 holds. If such an element exists, the contract $\mathcal{C}$ will be referred to as a *consistent* contract.

Furthermore, in phase (III), the OEM also has the intent of integrating element $\mathbb{E}$ delivered by the supplier with a set of elements $\{\mathbb{E}_i\}_{i=1}^{N}$ such that the composition of $\{\mathbb{E}\} \cup \{\mathbb{E}_i\}_{i=1}^{N}$ non-trivially fulfills the guarantee G. However, in order for this to be possible, there needs to exists at least one set of elements containing $\mathbb{E}$ where the environment of the element is such that condition (ii) of Corollary 1 holds with respect to $\mathcal{C}$. If such a set of elements exists, then the contract $\mathcal{C}$ will be referred to as a *compatible* contract.

Now that the concepts of consistency and compatibility have been introduced in the context of a scenario, formal definitions follow. Considering that the definitions quantify over elements and set of elements, complementary sufficient and necessary conditions that can be established to hold on the contract alone, will also be presented.

Definition 6 (*Consistent contract*) A contract $(\mathscr{A}, \mathsf{G}, X)$ is *consistent* if there exists an element $\mathbb{E} = (X, \mathsf{B})$ such that

(a) $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}$, and
(b) $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B}$. □

Definition 6 corresponds to an instantiation of the abstract definition of consistency in the meta theory of contracts in [33] using condition (i) of Corollary 1. Definition 6 is also closely related to definitions of consistency and compatibility in [3,5], and to the definition of *realizability* in [71], but where Definition 6, in contrast to the definitions in [3,5,71], considers a context where contract $(\mathscr{A}, \mathsf{G}, X)$ is not necessarily limited to set of port variables X and where X does not need to be partitioned into inputs and outputs.

A sufficient and necessary condition of Definition 6 now follows.

Theorem 3 *A contract* $(\mathscr{A}, \mathsf{G}, X)$ *is* consistent *if and only if*

$$\mathsf{A}_{\mathscr{A}} \cap \widehat{proj}_X(\mathsf{A}_{\mathscr{A}} \cap \mathsf{G}) \subseteq \mathsf{G}.$$

Lemma 2 *Given two assertions* W *and* W'*, and a set of variables* X*, it holds that* $\widehat{proj}_X(\mathsf{W}) \subseteq \widehat{proj}_X(\mathsf{W}')$*, if* $\mathsf{W} \subseteq \mathsf{W}'$.

The proof of Lemma 2 is found in "Appendix A". The proof of Theorem 3 now follows.

Proof Consider a contract $\mathcal{C} = (\mathscr{A}, \mathsf{G}, X)$.

For the only-if part, assume that $\mathcal{C}$ is consistent. In accordance with Definition 6, this means that there exists an element $\mathbb{E} = (X, \mathsf{B})$ such that $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}$ and $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B}$. In accordance with Lemma 2, this means that $\widehat{proj}_X(\mathsf{A}_{\mathscr{A}} \cap \mathsf{G}) \subseteq \widehat{proj}_X(\mathsf{B})$. This and the fact that $\widehat{proj}_X(\mathsf{B}) = \mathsf{B}$ in accordance with Definition 2 and Proposition 1, it follows that $\widehat{proj}_X(\mathsf{A}_{\mathscr{A}} \cap \mathsf{G}) \subseteq \mathsf{B}$. This and relation $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}$ imply that $\mathsf{A}_{\mathscr{A}} \cap \widehat{proj}_X(\mathsf{A}_{\mathscr{A}} \cap \mathsf{G}) \subseteq \mathsf{G}$.

For the if part, assume that relation $\mathsf{A}_{\mathscr{A}} \cap \widehat{proj}_X(\mathsf{A}_{\mathscr{A}} \cap \mathsf{G}) \subseteq \mathsf{G}$ holds. Assume that $\mathbb{E} = (X, \mathsf{B})$ is an element where $\mathsf{B} = \widehat{proj}_X(\mathsf{A}_{\mathscr{A}} \cap \mathsf{G})$, which means that $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}$. In accordance with relations (1) and (2), it holds that $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \widehat{proj}_X(\mathsf{A}_{\mathscr{A}} \cap \mathsf{G})$. This and considering that $\mathsf{B} = \widehat{proj}_X(\mathsf{A}_{\mathscr{A}} \cap \mathsf{G})$ imply that $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B}$, which concludes the proof. □

In contrast to Definition 6, Theorem 3 supports a way of establishing whether a contract is consistent or not without the need for iterating through each element with a set of port variables X in order to determine if there exists an element $\mathbb{E} = (X, \mathsf{B})$ that is such that condition (i) of Corollary 1 holds with respect to $\mathcal{C}$.

Definition 7 (*Compatible contract*) A contract $(\mathscr{A}, \mathsf{G}, X)$ is *compatible* if there exists an element $\mathbb{E} = (X, \mathsf{B})$ and a set of elements $\mathcal{E}$ containing $\mathbb{E}$, such that

(a) $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G} \neq \emptyset$, and
(b) $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$. □

Definition 7 corresponds to an instantiation of the abstract definition of compatibility in the meta theory of contracts in [33] using condition (ii) of Corollary 1. Definition 7 is also closely related to the definitions of compatibility in [3,5]. However, in contrast to [3,5], Definition 7 considers a context where the contract $(\mathscr{A}, \mathsf{G}, X)$ is not necessarily limited to set of port variables X and where X does not need to be partitioned into inputs and outputs.

A sufficient and necessary condition of compatibility now follows.

Theorem 4 *A contract* $(\mathscr{A}, \mathsf{G}, X)$ *is* compatible *if and only if* $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \neq \emptyset$.

Proof Consider a contract $(\mathscr{A}, \mathsf{G}, X)$.

For the if part, assume that $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \neq \emptyset$. This implies that there exists at least one run ω in $\mathsf{A}_{\mathscr{A}}$ that is also in $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G}$. Assume that $\mathcal{E}_0$ is a set of elements containing an element $\mathbb{E} = (X, \mathsf{B})$ such that $\mathsf{B}_{Env_{\mathcal{E}_0}(\mathbb{E})} = \{\omega\}$. This implies that there exists a set of elements containing an element $\mathbb{E} = (X, \mathsf{B})$, such that relations (i) and (ii) of Definition 7 hold.

For the if-only part, assume that $(\mathscr{A}, \mathsf{G}, X)$ is compatible. In accordance with Definition 7, this means that there exists a set of elements $\mathcal{E}$ containing an element $\mathbb{E} = (X, \mathsf{B})$ such that $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G} \neq \emptyset$ and $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$. It trivially follows that $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \neq \emptyset$, which completes the proof. □

In contrast to Definition 7, Theorem 4 supports a way of establishing whether a contract $\mathcal{C} = (\mathscr{A}, \mathsf{G})$ is compatible or not, without the need for iterating through each set of elements containing an element with a set of port variables X in order to determine if there exists a set of elements where the environment of an element $\mathbb{E} = (X, \mathsf{B})$ is such that condition (ii) of Corollary 1 holds.

As a conclusion to this section, a scenario is examined where a supplier wants to outsource the development of a level meter by the use of contract $\mathcal{C}_{lMeter}$ shown in Fig. 6. In order for the supplier and the client to complete phases (I–III) in the scenario described in Sect. 3.2 such that relation (4) holds, contract $\mathcal{C}_{lMeter}$ must be consistent and compatible.

In order for this to be the case, Theorems 3 and 4 express that it is necessary and sufficient that:

$$\mathsf{A}_{lMeter} \cap \mathsf{G}_{lMeter} \neq \emptyset; \text{ and} \tag{15}$$

$$\mathsf{A}_{lMeter} \cap \widehat{proj}_{X_{lMeter}}(\mathsf{A}_{lMeter} \cap \mathsf{G}_{lMeter}) \subseteq \mathsf{G}_{lMeter}. \tag{16}$$

Considering that $\mathsf{A}_{lMeter} \cap \mathsf{G}_{lMeter}$ is an assertion specified by equations $f = \frac{v_{branch} - v_{gnd}}{5}$ and $l = f$, which obviously have intersecting solutions, relation (15) holds. The extended projection of this assertion onto set of variables $X_{lMeter} = \{l, v_{branch}, v_{gnd}\}$, i.e. $\widehat{proj}_{X_{lMeter}}(\mathsf{A}_{lMeter} \cap \mathsf{G}_{lMeter})$, is specified by equation $l = \frac{v_{branch} - v_{gnd}}{5}$. The intersection of A_{lMeter} and assertion $\widehat{proj}_X(\mathsf{A}_{lMeter} \cap \mathsf{G}_{lMeter})$ yields an assertion specified by equations $f = \frac{v_{branch} - v_{gnd}}{5}$ and $l = \frac{v_{branch} - v_{gnd}}{5}$, which can be combined into equation $l = f$. This means that relation (16) holds. Therefore, it can be concluded that contract $\mathcal{C}_{lMeter}$ is consistent and compatible and is, thus, an appropriate specification to be used for outsourced development.

6 Hierarchical structuring of contracts

Consider a pair $(\mathcal{C}, \{\mathcal{C}_i\}_{i=1}^N)$, characterizing a *two-level contract hierarchy* such that $\mathcal{C} = (\mathscr{A}, \mathsf{G}, X)$ is a contract at the first level and $\{\mathcal{C}_i = (\mathscr{A}_i, \mathsf{G}_i, X_i)\}_{i=1}^N$ is a set of contracts where $X \subseteq \bigcup_{i=1}^N X_i$, at the second level. This section establishes the following property of two level contract hierarchy $(\mathcal{C}, \{\mathcal{C}_i\}_{i=1}^N)$: for each set of elements $\{(X_i, \mathsf{B}_i)\}_{i=1}^N$ where each element (X_i, B_i) is such that condition (i) of Corollary 1 holds with respect to $(\mathscr{A}_i, \mathsf{G}_i, X_i)$, the composition (X, B) of $\{(X_i, \mathsf{B}_i)\}_{i=1}^N$ onto X is such that condition (i) of Corollary 1 holds with respect to $\mathcal{C}$. If this property holds, then two-level contract hierarchy $(\mathcal{C}, \{\mathcal{C}_i\}_{i=1}^N)$ is said to be *proper*. Establishing that a contract hierarchy of an arbitrary number of levels is proper, amounts to establishing that each pair of adjoining levels in the contract hierarchy is proper.

Definition 8 (*Proper contract hierarchy*) Given a contract $\mathcal{C} = (\mathscr{A}, \mathsf{G}, X)$ and a set of contracts $\{\mathcal{C}_i = (\mathscr{A}_i, \mathsf{G}_i, X_i)\}_{i=1}^N$ where $X \subseteq \bigcup_{i=1}^N X_i$, *the two-level contract hierarchy* $(\mathcal{C}, \{\mathcal{C}_i\}_{i=1}^N)$ *is proper* if, for each set of elements $\{(X_i, \mathsf{B}_i)\}_{i=1}^N$:

$$(\forall i : \mathsf{A}_{\mathscr{A}_i} \cap \mathsf{B}_i \subseteq \mathsf{G}_i \text{ and } \mathsf{A}_{\mathscr{A}_i} \cap \mathsf{G}_i \subseteq \mathsf{B}_i) \Longrightarrow \mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G} \text{ and } \mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B},$$

where (X, B) is the composition of $\{(X_i, \mathsf{B}_i)\}_{i=1}^N$ onto X. □

Definition 8 is in accordance with the general principle of *compositionality* [31,32] since the fact that element (X, B) is such that relations $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}$ and $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B}$ hold can be inferred from establishing that each element (X_i, B_i) is such that relations $\mathsf{A}_{\mathscr{A}_i} \cap \mathsf{B}_i \subseteq \mathsf{G}_i$

and $\mathsf{A}_{\mathscr{A}_i} \cap \mathsf{G}_i \subseteq \mathsf{B}_i$ hold. The compositional approach of indirectly establishing that (X, B) is such that condition (i) of Corollary 1 holds with respect to $\mathcal{C}$ is needed when a direct approach is not feasible due to e.g. the complexity of (X, B).

Despite considering dissimilar conditions than those presented in Corollary 1 in the present paper, Definition 8 corresponds, in essence, to the definitions of *dominance* in [39,72] where [72] offers a minor extension to the definitions in [35,44]. However, in contrast to [72] where ports are not considered and to [39] where guarantees must be limited to the set of ports of a component, Definition 8 considers a context where port variables are explicit, but where contracts are not necessarily limited to the sets of port variables of elements.

A special case of Definition 8 is when a contract hierarchy is of the form $((\mathscr{A}, \mathsf{G}, X), \{(\mathscr{A}', \mathsf{G}', X)\})$. In accordance with Definitions 3 and 8, in order for such a contract hierarchy to be proper, it means that condition (i) of Corollary 1 must hold with respect to $(\mathscr{A}, \mathsf{G}, X)$ for each element (B, X) that is such that condition (i) of Corollary 1 holds with respect to $(\mathscr{A}', \mathsf{G}', X)$. Notably, this special case of Definition 8 corresponds to an instantiation of the abstract definition of *refinement* in the meta theory of contracts in [33] using condition (i) of Corollary 1. The notion of refinement, defined as a special case of of Definition 8, is further compared to refinement as defined in other contract theories in Sect. 7.

In order to provide further understanding of when Definition 8 is relevant, a scenario is presented. The scenario is in the context of an OEM/supplier chain as described in Sect. 3, but the principles are equally valid for any design context where clear separations of responsibilities are desired, also within a single company. Specifically, the scenario consists of three phases:

(I') the OEM establishes a two-level contract hierarchy $(\mathcal{C} = (\mathscr{A}, \mathsf{G}, X), \{\mathcal{C}_i\}_{i=1}^N)$ where each contract $\mathcal{C}_i = (\mathscr{A}_i, \mathsf{G}_i, X_i)$ is handed from the OEM to a supplier and where $X \subseteq \bigcup_{i=1}^N X_i$;

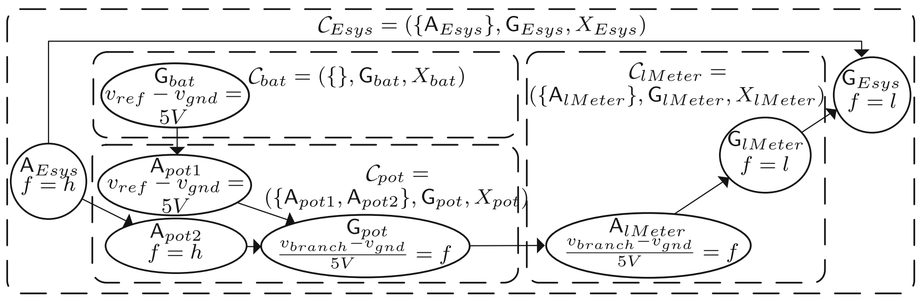
(II') each supplier develops an element $\mathbb{E}_i = (X_i, \mathsf{B}_i)$ such that condition (i) of Corollary 1 holds with respect to $\mathcal{C}_i$; and

(III') the OEM integrates the set of elements $\{\mathbb{E}_i\}_{i=1}^N$ into an element $\mathbb{E} = (X, \mathsf{B})$ that is the composition of $\{\mathbb{E}_i\}_{i=1}^N$ onto X and is such that condition (i) of Corollary 1 holds with respect to $\mathcal{C}$.

In order to enable an integration of the set of elements $\{\mathbb{E}_i\}_{i=1}^N$ into element $\mathbb{E}$ in phase (III'), the fact that element $\mathbb{E}$ is such that condition (i) of Corollary 1 holds with respect to $\mathcal{C}$ needs to follow from the completion of phase (II'). In order for this to be the case, in phase (I'), two-level contract hierarchy $(\mathcal{C}, \{\mathcal{C}_i\}_{i=1}^N)$ needs to proper.

In order to find a two-level contract hierarchy $(\mathcal{C}, \{\mathcal{C}_i\}_{i=1}^N)$ that is proper, a graph, called a *composition structure*, is introduced in Sect. 6.1. Based on a composition structure, a theorem that expresses sufficient conditions for a contract hierarchy to be proper is presented in Sect. 6.2. Despite considering dissimilar formalisms than the present paper, the sufficient conditions that will be presented for a contract hierarchy to be proper corresponds, in essence, to the *compositional proof step* [73] in assume-guarantee theories such as [34,47,51], and also to sufficient conditions of dominance in [39]. However, in contrast to [34,39,47,51], the sufficient conditions in Sect. 6.2 are based on the concept of a graph, or more specifically, a composition structure.

The proposed graph-based approach supports the approach of establishing that *any* two-level contract $(\mathcal{C}, \{\mathcal{C}_i\}_{i=1}^N)$ is proper directly. Another alternative, but more indirect, way of establishing that $(\mathcal{C}, \{\mathcal{C}_i\}_{i=1}^N)$ is proper is to: first, find a contract $\mathcal{C}' = (\mathscr{A}', \mathsf{G}', X)$ such that $(\mathcal{C}', \{\mathcal{C}_i\}_{i=1}^N)$ is proper and there exists no other contract $\mathcal{C}'' = (\mathscr{A}'', \mathsf{G}'', X)$ that refines $\mathcal{C}'$ (a special case of Definition 8 as previously mentioned) such that $(\mathcal{C}'', \{\mathcal{C}_i\}_{i=1}^N)$ is proper; and



$\mathcal{C}_{Esys} = (\{\mathsf{A}_{Esys}\}, \mathsf{G}_{Esys}, X_{Esys})$
G_{bat} $v_{ref} - v_{gnd} = 5V$
$\mathcal{C}_{bat} = (\{\}, \mathsf{G}_{bat}, X_{bat})$
$\mathcal{C}_{lMeter} = (\{\mathsf{A}_{lMeter}\}, \mathsf{G}_{lMeter}, X_{lMeter})$
G_{Esys} $f = l$
G_{lMeter} $f = l$
A_{Esys} $f = h$
A_{pot1} $v_{ref} - v_{gnd} = 5V$
$\mathcal{C}_{pot} = (\{\mathsf{A}_{pot1}, \mathsf{A}_{pot2}\}, \mathsf{G}_{pot}, X_{pot})$
A_{pot2} $f = h$
G_{pot} $\frac{v_{branch} - v_{gnd}}{5V} = f$
A_{lMeter} $\frac{v_{branch} - v_{gnd}}{5V} = f$

Fig. 9 A composition structure of two-level contract hierarchy ($\mathcal{C}_{Esys}$, $\{\mathcal{C}_{pot}$,$\mathcal{C}_{bat}$,$\mathcal{C}_{lMeter}\}$)

second, check whether $\mathcal{C}'$ refines $\mathcal{C}$. In accordance with the meta theory in [33], the contract $\mathcal{C}'$ in such an approach corresponds to an instantiation of *parallel composition* [3–5,37,44,56] of $\{\mathcal{C}_i\}_{i=1}^N$ using condition (i) of Corollary 1.

Notably, the indirect way does not require checking that any two-level contract ($\mathcal{C}$, $\{\mathcal{C}_i\}_{i=1}^N$) is proper, but rather only the special case of refinement. The indirect way does, however, require computing parallel composition, which is not required for the more direct approach. The rest of this section will focus only on the graph-based approach and on supporting the direct, rather than the indirect, approach. The main reason for doing so is that the indirect approach requires a way of checking refinement anyways and this is supported by the graph-based approach. That is, the graph-based approach provides a partial foundation for enabling the indirect approach, while the inverse is not true.

6.1 Composition structures of contract hierarchies

Prior to presenting the formal definition of a composition structure, the concept is introduced in an informal manner by structuring a two-level contract hierarchy ($\mathcal{C}_{Esys}$, $\{\mathcal{C}_{pot}$,$\mathcal{C}_{bat}$,$\mathcal{C}_{lMeter}\}$). The contracts express specifications for the elements of the electric-system of an LM-system, e.g. the one shown in Fig. 5.

Consider the assumptions and the guarantee of each contract in set of contracts $\{\mathcal{C}_{Esys}$,$\mathcal{C}_{pot}$,$\mathcal{C}_{bat}$,$\mathcal{C}_{lMeter}\}$ being structured as nodes in a directed graph, as shown in Fig. 9 where the boxes with rounded corners and dashed edges have been added to also show the two-level contract hierarchy. The set of incoming arcs to a guarantee G from a set of assumptions $\mathscr{A}$, represents that $\mathscr{A}$ and G are in the same contract, e.g. the arc from A_{Esys} to guarantee G_{Esys} represents contract ($\{\mathsf{A}_{Esys}\}$, G_{Esys}, X_{Esys}).

The set of incoming arcs to an assumption A from a set of assertions $\{\mathsf{W}_i\}_{i=1}^N$ where W_i is either an assumption or a guarantee represents the intent $\bigcap_{i=1}^N \mathsf{W}_i \subseteq \mathsf{A}$. For example, the arc to A_{pot2} from assumption A_{Esys}, represents the intent of $\mathsf{A}_{Esys} \subseteq \mathsf{A}_{pot2}$.

The set of incoming arcs to a guarantee G from a set of guarantees $\{\mathsf{G}_j\}_{j=1}^M$ represents the intent of $\bigcap_{j=1}^M \mathsf{G}_j \subseteq \mathsf{G}$. For example, the arc from the guarantee G_{lMeter} to G_{Esys}, represents the intent of $\mathsf{G}_{lMeter} \subseteq \mathsf{G}_{Esys}$.

Now that the concept of a composition structure has been introduced informally, the formal definition follows.

Definition 9 (*Composition structure of contract hierarchy*) Given a contract $\mathcal{C} = (\mathscr{A}, \mathsf{G}, X)$ and a set of contracts $\{\mathcal{C}_i = (\mathscr{A}_i, \mathsf{G}_i, X_i)\}_{i=1}^N$ where $X \subseteq \bigcup_{i=1}^N X_i$, a *composition structure* $\mathfrak{D}$ *of two-level contract hierarchy* ($\mathcal{C}$, $\{\mathcal{C}_i\}_{i=1}^N$) is a Directed Acyclic Graph (DAG), such that:

(a) the guarantees G_i, the assumptions in each $\mathscr{A}_i$, the assumptions in $\mathscr{A}$, and guarantee G are the nodes in $\mathfrak{D}$;
(b) G has no successors;
(c) each assumption in $\mathscr{A}$ has no predecessor;
(d) at least one G_i is a direct predecessor of G;
(e) each assumption in each $\mathscr{A}_i$ has at least one predecessor;
(f) the assumptions in $\mathscr{A}_i$ are the only direct predecessors of each G_i;
(g) G_i is the only direct successor of each assumption in each $\mathscr{A}_i$; and
(h) G is a direct successor of each assumption in $\mathscr{A}$. □

As described in the beginning of this section and in Definition 9, a composition structure represents a structuring of a two-level contract hierarchy as expressed in condition (a) where: the set of incoming arcs to a guarantee from a set of assumptions represents that the assumptions and the guarantee are in the same contract as expressed in conditions (f) and (h); the intent is that at least one guarantee G_i or an assumption in $\mathscr{A}$ should be a subset of each assumption in $\mathscr{A}_i$, as expressed in conditions (b) and (g), and condition (e) in particular; and the intent is that at least one guarantee G_i should be a subset of the guarantee G, as expressed in condition (d). Condition (c), and further also conditions (b), (f), and (g), disallow the existence of any other arcs from those not already mentioned above.

Out of general assume-guarantee theories [3,4,33–59], only [48] considers a graph-based approach for structuring contracts. However, in contrast to [48] where the aim of the structuring is to be able to verify a set of contracts with *circular dependencies* (see Remark 1 in the end of this section), a composition structure represents a structuring of a two-level contract hierarchy in general.

Apart from general assume-guarantee theories, composition structures do have a lot in common with Goal Oriented Requirements Engineering (GORE) models, see e.g. I* [74] or KAOS [75] or [76] for a survey, where [74,75] draw on ideas presented in [77–79]. The main difference is, again, that while a composition structure represents a structuring of a two-level contract hierarchy in general, GORE models are more specific since the use of assumptions, also called *expectations*, in GORE models are strictly limited to top-level specifications that split the responsibilities between a *SW system* and its environment. Furthermore, a similar concept to a contract structure is presented in [80,81] based on Bayesian networks, but with the specific focus to model failure propagation.

Given a composition structure $\mathfrak{D}$ of a contract hierarchy $(\mathcal{C}, \{\mathcal{C}_i\}_{i=1}^N)$, in accordance with Definition 9, each assumption and each guarantee in $\mathcal{C}$ and $\mathcal{C}_i$ are nodes in $\mathfrak{D}$. There are, however, cases when re-using an assumption or a guarantee would be preferred [15], e.g. if two guarantees rely on the same assumption or if a guarantee is equal to an assumption. In practice, such a case can be represented by either the use of a single node or to label one node as a copy of another.

Remark 1 (Circular reasoning) Since a composition structure is a directed *acyclic* graph where the assumptions and guarantees are the nodes, the use of circular argumentation is avoided. Note that circularity can be resolved in other ways, e.g. by introducing assumptions about the computational model [34] or the timing model [48]. See also [82,83] for more discussions on such matters. □

6.2 Sufficient conditions for proper contract hierarchy

This section presents sufficient conditions for a two-level contract hierarchy $(\mathcal{C} = (\mathscr{A}, \mathsf{G}, X), \{\mathcal{C}_i = (\{\mathsf{A}_{ij}\}_{j=1}^{M_i}, \mathsf{G}_i, X_i)\}_{i=1}^N)$ to be proper in accordance with Definition 8, based on the concept of a composition structure. The sufficient conditions supports a way of establishing that $(\mathcal{C}, \{\mathcal{C}_i\}_{i=1}^N)$ is proper, without having to iterate through each possible set of elements $\{(X_i, \mathsf{B}_i)\}_{i=1}^N$ to determine that their composition onto X is such that condition (i) of Corollary 1 holds with respect to $\mathcal{C}$, if each element (X_i, B_i) is such that condition (i) of Corollary 1 holds with respect to $(\mathscr{A}_i, \mathsf{G}_i, X_i)$.

Consider a composition structure of $\mathfrak{D}$ of $(\mathcal{C}, \{\mathcal{C}_i\}_{i=1}^N)$. Let $dPred()$ denote a function that takes a node W in $\mathfrak{D}$ as input and returns the set of all nodes that are direct predecessors of W. As presented in Sect. 6.1, the composition structure $\mathfrak{D}$ represents the intent that

$$\bigcap\nolimits_{\mathsf{W} \in dPred(\mathsf{G}) \cap \{\mathsf{G}_i\}_{i=1}^N} \mathsf{W} \subseteq \mathsf{G}, \text{ and} \tag{17}$$

$$\bigcap\nolimits_{\mathsf{W} \in dPred(\mathsf{A}_{ij})} \mathsf{W} \subseteq \mathsf{A}_{ij}, \text{ for each } i, j. \tag{18}$$

As an example, the composition structure of $(\mathcal{C}_{Esys}, \{\mathcal{C}_{pot}, \mathcal{C}_{bat}, \mathcal{C}_{lMeter}\})$ in Fig. 9 represents the intent that relation $\mathsf{G}_{lMeter} \subseteq \mathsf{G}_{Esys}$ holds, and furthermore that relations $\mathsf{G}_{bat} \subseteq \mathsf{A}_{pot1}$, $\mathsf{A}_{Esys} \subseteq \mathsf{A}_{pot2}$, $\mathsf{G}_{pot} \subseteq \mathsf{A}_{lMeter}$ hold.

However, as will be shown in the following illustrative Examples 2a, 2b, and 2c, the fact that relations (17) and (18) hold for $\mathfrak{D}$ does not imply that $(\mathcal{C}, \{\mathcal{C}_i\}_{i=1}^N)$ is proper. As a quick overview, Examples 2a and 2b will show that properness cannot be ensured if either the assumptions in $\mathsf{A}_{\mathscr{A}}$ or guarantee G of contract $\mathcal{C}$ constrains port variables in set $\bigcup_{i=1}^N X_i \setminus X$. Example 2c shows the need to introduce an additional condition for the specific purpose of ensuring that relation $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B}$ (of condition (i) of Corollary 1) holds for the composition (X, B) of each set of elements $\{(X_i, \mathsf{B}_i)\}_{i=1}^N$ where (X_i, B_i) is such that condition (i) of Corollary 1 holds with respect to $\mathcal{C}_i = (\mathscr{A}_i, \mathsf{G}_i, X_i)$.

Example 2a Consider that a composition structure $\mathfrak{D}_a$ of a two level contract hierarchy $(\mathcal{C}'_{Esys}, \{\mathcal{C}_{pot}, \mathcal{C}'_{bat}, \mathcal{C}_{lMeter}\})$ is the resulting composition structure from making the following modifications to the composition structure and two-level contract hierarchy shown in Fig. 9: in contract $\mathcal{C}'_{bat} = (\{\}, \mathsf{G}'_{bat}, X_{bat})$, it holds that $\mathsf{G}'_{bat} = \Omega$, instead of being specified by equation $v_{ref} - v_{gnd} = 5V$; A'_{Esys} in contract $\mathcal{C}'_{Esys} = (\{\mathsf{A}'_{Esys}\}, \mathsf{G}_{Esys}, X_{Esys})$ is specified by equations $f = h$ and $v_{ref} - v_{gnd} = 5V$, instead of only $f = h$; and that an outgoing arc has been added from A'_{Esys} to A_{pot1}. In accordance with relation (18), the intent is that $\mathsf{A}'_{Esys} \cap \Omega \subseteq \mathsf{A}_{pot1}$, which is equal to $\emptyset \neq \mathsf{A}'_{Esys} \subseteq \mathsf{A}_{pot1}$. In accordance with Definition 5, due to the fact that A_{pot1} restricts $\{v_{ref}, v_{gnd}\}$, A_{Esys} also needs to restrict $\{v_{ref}, v_{gnd}\}$ in order for relation (18) to hold. Note that neither v_{ref} nor v_{gnd} are in X_{Esys}.

Now consider set of elements $\{\mathbb{E}'_{bat}, \mathbb{E}_{pot}, \mathbb{E}_{lMeter}\}$ where $\mathbb{E}_{pot}$ and $\mathbb{E}_{lMeter}$ are shown in Fig. 5 and where $\mathbb{E}'_{bat}$ is the modification of $\mathbb{E}_{bat}$ such that the behavior of $\mathbb{E}'_{bat}$ is equal to Ω, instead of being specified by equation $v_{ref} - v_{gnd} = 5V$. It trivially holds that elements $\mathbb{E}'_{bat}, \mathbb{E}_{pot}, \mathbb{E}_{lMeter}$ are such that condition (i) of Corollary 1 holds with respect to $\mathcal{C}'_{bat}$, $\mathcal{C}_{pot}$, and $\mathcal{C}_{lMeter}$. Furthermore, it can easily be realized that relations (17) and (18) hold for $\mathfrak{D}_a$. However, since relation $v_{ref} - v_{gnd} = 5V$ is ensured by A'_{Esys}, rather than by the behavior of $\mathbb{E}'_{bat}$, the behavior of the composition $\mathbb{E}'_{sys}$ of $\{\mathbb{E}'_{bat}, \mathbb{E}_{pot}, \mathbb{E}_{lMeter}\}$ onto X_{Esys} is Ω, rather than being specified by $l = h$. This means that $\mathbb{E}'_{sys}$ is not such that relation (7) of condition (i) of Corollary 1 holds with respect to $\mathcal{C}'_{Esys}$, i.e. it does not hold that $\mathsf{A}'_{Esys} \cap \mathsf{B}'_{Esys} \subseteq \mathsf{G}'_{Esys}$. Thus, the composition $\mathbb{E}'_{Esys}$ of $\{\mathbb{E}'_{bat}, \mathbb{E}_{pot}, \mathbb{E}_{lMeter}\}$ onto X_{Esys} is not such that condition (i) of Corollary 1 holds with respect to $\mathcal{C}'_{Esys}$. □

As shown in Example 2a, the fact that it is necessary for A'_{Esys} to restrict port variables in $X_{bat} \cup X_{pot} \cup X_{lMeter} \setminus X_{Esys}$, in order for relation (18) to hold, means that even if relations (17) and (18) do hold, $(\mathcal{C}'_{Esys}, \{\mathcal{C}_{pot}, \mathcal{C}'_{bat}, \mathcal{C}_{lMeter}\})$ is not proper. Notably, regardless if is necessary or not for A'_{Esys} to restrict port variables in $X_{bat} \cup X_{pot} \cup X_{lMeter} \setminus X_{Esys}$, if A'_{Esys} does restrict such variables, then in accordance with Definition 5, there does not exists an element that non-trivially fulfills A'_{Esys}. That is, the fact that A'_{Esys} restricts variables in $X_{bat} \cup X_{pot} \cup X_{lMeter} \setminus X_{Esys}$ is undesirable all together.

Considering the general case with a composition structure $\mathfrak{D}$, a sufficient condition for ensuring that $\mathsf{A}_{\mathscr{A}}$ does not restrict port variables in $\bigcup_{i=1}^{N} X_i \setminus X$ is to ensure that $\mathsf{A}_{\mathscr{A}}$ does not constrain any port variable in $\bigcup_{i=1}^{N} X_i \setminus X$, i.e.

$$(X_{\mathsf{A}_{\mathscr{A}}} \setminus X) \cap \bigcup_{i=1}^{N} X_i = \emptyset. \tag{19}$$

Notably, while ensuring that $\mathsf{A}_{\mathscr{A}}$ does not restrict port variables in $\bigcup_{i=1}^{N} X_i \setminus X$, in accordance with Proposition 4, it is also the case that relation (19) can be enforced *without loosing expressiveness*. However, the fact that relation 19 holds in combination with relations (17)–(18), as will be shown by the following illustrative Examples 2b and 2c, is not sufficient to ensure that $(\mathcal{C}, \{\mathcal{C}_i\}_{i=1}^{N})$ is proper; thus, additional conditions are presented in the following.

Example 2b Suppose that a composition structure $\mathfrak{D}_b$ of a two level contract hierarchy $(\mathcal{C}''_{Esys}, \{\mathcal{C}_{pot}, \mathcal{C}_{bat}, \mathcal{C}''_{lMeter}\})$ is the resulting composition structure from making the following modifications to the composition structure and two-level contract hierarchy shown in Fig. 9: both guarantees G''_{lMeter} and G''_{Esys} in $\mathcal{C}''_{Esys} = (\{\mathsf{A}_{Esys}\}, \mathsf{G}''_{Esys}, X_{Esys})$ and $\mathcal{C}''_{lMeter} = (\{\mathsf{A}_{lMeter}\}, \mathsf{G}''_{lMeter}, X_{lMeter})$ are specified by equations $\frac{v_{branch} - v_{gnd}}{5} = f$ and $f = l$, instead of $f = l$. Considering the composition of the set of elements $\{\mathbb{E}_{bat}, \mathbb{E}_{pot}, \mathbb{E}_{lMeter}\}$ onto X_{Esys}, i.e. element $\mathbb{E}_{Esys}$ shown in Fig. 5, due to the fact that $\mathsf{A}_{Esys} \cap \mathsf{B}_{Esys}$ is specified by equations $f = h$ and $l = h$, it holds that $\mathsf{A}_{Esys} \cap \mathsf{B}_{Esys}$ is non-empty and constrains exactly $\{f, l, h\}$. In accordance with Definition 5, from the fact that G''_{Esys} does indeed restrict both v_{branch} and v_{gnd}, where neither of these are in $\{f, l, h\}$, it follows that relation (7) of condition (i) of Corollary 1 does not hold, i.e. that $\mathsf{A}_{Esys} \cap \mathsf{B}_{Esys} \nsubseteq \mathsf{G}''_{Esys}$. □

Example 2c Consider that a composition structure $\mathfrak{D}_b$ of a two level contract hierarchy $(\mathcal{C}_{Esys}, \{\mathcal{C}_{pot}, \mathcal{C}_{bat}, \mathcal{C}'''_{lMeter}\})$ is the resulting composition structure from making the following modifications to the composition structure and two-level contract hierarchy shown in Fig. 9: guarantee G'''_{Esys} in the contract $\mathcal{C}'''_{lMeter} = (\{\mathsf{A}_{lMeter}\}, \mathsf{G}'''_{lMeter}, X_{lMeter})$ is specified by relation $l - 0.1 \leq f \leq l + 0.1$, instead of equation $f = l$. Considering element $\mathbb{E}_{Esys}$, since B_{Esys} is specified by equation $l = f$ and $\mathsf{A}_{Esys} \cap \mathsf{G}'''_{Esys}$ by relation $l - 0.1 \leq f \leq l + 0.1$ and equation $h = f$, relation (8) of condition (i) of Corollary 1 does not hold. That is, it does not hold that $\mathsf{A}_{Esys} \cap \mathsf{G}'''_{Esys} \subseteq \mathsf{B}_{Esys}$. □

In both Examples 2b and 2c, it trivially holds that elements $\mathbb{E}_{bat}$, $\mathbb{E}_{pot}$, and $\mathbb{E}_{lMeter}$ are such that condition (i) of Corollary 1 hold with respect to the contracts containing their sets of port variables. Furthermore, it can easily be realized that relations (17) and (18) hold for both $\mathfrak{D}_b$ and $\mathfrak{D}_c$. However, since neither one of relations $\mathsf{A}_{Esys} \cap \mathsf{B}_{Esys} \subseteq \mathsf{G}''_{Esys}$ and $\mathsf{A}_{Esys} \cap \mathsf{G}'''_{Esys} \subseteq \mathsf{B}_{Esys}$ hold, it does not follow that the composition $\mathbb{E}_{Esys}$ of $\{\mathbb{E}_{bat}, \mathbb{E}_{pot}, \mathbb{E}_{lMeter}\}$ onto X_{Esys} is such that condition (i) of Corollary 1 holds with respect to any of $\mathcal{C}''_{Esys}$ or $\mathcal{C}'''_{Esys}$. Hence, despite the fact that relations (17) and (18) hold for $\mathfrak{D}_b$ and $\mathfrak{D}_c$, neither $(\mathcal{C}''_{Esys}, \{\mathcal{C}_{pot}, \mathcal{C}_{bat}, \mathcal{C}''_{lMeter}\})$ nor $(\mathcal{C}_{Esys}, \{\mathcal{C}_{pot}, \mathcal{C}_{bat}, \mathcal{C}'''_{lMeter}\})$ are proper.

For a two level contract hierarchy $((\mathscr{A}, \mathsf{G}, X), \{(\{\mathsf{A}_{ij}\}_{j=1}^{M_i}, \mathsf{G}_i, X_i)\}_{i=1}^{N})$, similar to Examples 2a, 2b shows that if G restricts a variable in $\bigcup_{i=1}^{N} X_i \setminus (X \cup X_{\mathsf{A}_{\mathscr{A}}})$, then $(\mathcal{C}, \{\mathcal{C}_i\}_{i=1}^{N})$

is not proper. In accordance with Proposition 4, a sufficient condition to avoid this case, but without loosing expressiveness, is $(X_{\mathsf{G}} \setminus X_{\mathsf{A}_{\mathscr{A}}}) \cap \bigcup_{i=1}^{N} X_i = \emptyset$. This and relation (19) imply that

$$(X_{\mathsf{G}} \setminus X) \cap \bigcup_{i=1}^{N} X_i = \emptyset. \tag{20}$$

Example 2c, on the other hand, shows that it is necessary for G to be a subset of the extended projection of the guarantees G_i onto $\Xi \setminus (\bigcup_{i=1}^{N} X_i \setminus X)$. That is, it is necessary that $\mathsf{G} \subseteq \widehat{proj}_{\Xi \setminus (\bigcup_{i=1}^{N} X_i \setminus X)}(\bigcap_{i=1}^{N} \mathsf{G}_i)$. However, given that this holds, as well relations (19) and (20), it follows that G must be equal to $\widehat{proj}_{\Xi \setminus (\bigcup_{i=1}^{N} X_i \setminus X)}(\bigcap_{i=1}^{N} \mathsf{G}_i)$, rather than a subset, i.e. that

$$\mathsf{G} = \widehat{proj}_{\Xi \setminus (\bigcup_{i=1}^{N} X_i \setminus X)}(\bigcap_{i=1}^{N} \mathsf{G}_i). \tag{21}$$

This can be realized by considering that $\widehat{proj}_{\Xi \setminus (\bigcup_{i=1}^{N} X_i \setminus X)}(\bigcap_{i=1}^{N} \mathsf{G}_i) \subseteq \mathsf{G}$, which follows from the fact that relations (20) and (19) respectively imply that $\mathsf{G} = \widehat{proj}_{\Xi \setminus (\bigcup_{i=1}^{N} X_i \setminus X)}(\mathsf{G})$ in accordance with Proposition 1 and

$$\bigcap_{i=1}^{N} \mathsf{G}_i \subseteq \bigcap\nolimits_{\mathsf{W} \in dPred(\mathsf{G}) \cap \{\mathsf{G}_i\}_{i=1}^{N}} \mathsf{W} \subseteq \mathsf{G}.$$

To characterize a composition structure where relations (17)–(21) hold, the concept of a *proper composition structure* is introduced.

Definition 10 (*Proper composition structure*) Given a composition structure $\mathfrak{D}$ of a two-level contract hierarchy

$$((\mathscr{A}, \mathsf{G}, X), \{(\{\mathsf{A}_{ij}\}_{j=1}^{M_i}, \mathsf{G}_i, X_i)\}_{i=1}^{N})$$

consisting of a contract $(\mathscr{A}, \mathsf{G}, X)$ and set of contracts $\{(\{\mathsf{A}_{ij}\}_{j=1}^{M_i}, \mathsf{G}_i, X_i)\}_{i=1}^{N}$ where $X \subseteq \bigcup_{i=1}^{N} X_i$, *the composition structure* $\mathfrak{D}$ *is proper*, if:

(i) $\bigcap_{\mathsf{W} \in dPred(\mathsf{G}) \cap \{\mathsf{G}_i\}_{i=1}^{N}} \mathsf{W} \subseteq \mathsf{G}$;
(ii) $\bigcap_{\mathsf{W} \in dPred(\mathsf{A}_{ij})} \mathsf{W} \subseteq \mathsf{A}_{ij}$ for each i, j;
(iii) $((X_{\mathsf{G}} \cup X_{\mathsf{A}_{\mathscr{A}}}) \setminus X) \cap \bigcup_{i=1}^{N} X_i = \emptyset$; and
(iv) $\mathsf{G} = \widehat{proj}_{\Xi \setminus (\bigcup_{i=1}^{N} X_i \setminus X)}(\bigcap_{i=1}^{N} \mathsf{G}_i)$. □

Condition (i) and (ii) of Definition 10 are relations (17) and (18), respectively. Condition (iii) of Definition 10 combines relations (19) and (20) into a single condition. Considering Example 2a, due to the fact that

$$((X_{\mathsf{G}_{Esys}} \cup X_{\mathsf{A}'_{Esys}}) \setminus X_{Esys}) \cap (X_{pot} \cup X_{bat} \cup X_{lMeter}) = \{v_{ref}, v_{gnd}\} \neq \emptyset\ ,$$

condition (iii) of Definition 10 ensures that the case highlighted in Example 2a is avoided. Furthermore, condition (iii) also ensures that the case highlighted in Example 2b is avoided considering that

$$((X_{\mathsf{G}''_{Esys}} \cup X_{\mathsf{A}_{Esys}}) \setminus X_{Esys}) \cap (X_{pot} \cup X_{bat} \cup X_{lMeter}) = \{v_{branch}, v_{gnd}\} \neq \emptyset.$$

Condition (iv) of Definition 10 is relation (21). Considering Example 2c, it holds that $\Xi \setminus \{v_{ref}, v_{gnd}, v_{branch}\} = \Xi \setminus ((X_{pot} \cup X_{bat} \cup X_{lMeter}) \setminus X_{Esys})$, which means that

$$\widehat{proj}_{\Xi\setminus((X_{pot}\cup X_{bat}\cup X_{lMeter})\setminus X_{Esys})}(\mathsf{G}'''_{lMeter} \cap \mathsf{G}_{bat} \cap \mathsf{G}_{pot})$$

and G_{Esys} are specified by equation $f = l$ and relation $l - 0.1 \leq f \leq l + 0.1$, respectively. It follows that

$$\mathsf{G}_{Esys} \nsubseteq \widehat{proj}_{\Xi\setminus((X_{pot}\cup X_{bat}\cup X_{lMeter})\setminus X_{Esys})}(\mathsf{G}'''_{lMeter} \cap \mathsf{G}_{bat} \cap \mathsf{G}_{pot})\ ,$$

which means that condition (iv) of Definition 10 ensures that the case highlighted in Example 2c is avoided.

A theorem that presents sufficient conditions of a two-level contract hierarchy being proper now follows.

Theorem 5 *Given a contract* $\mathcal{C} = (\mathscr{A}, \mathsf{G}, X)$ *and a set of contracts* $\{\mathcal{C}_i\}_{i=1}^N$ *where* $\mathcal{C}_i = (\mathscr{A}_i, \mathsf{G}_i, X_i)$ *and* $X \subseteq \bigcup_{i=1}^N X_i$, *two-level contract hierarchy* $(\mathcal{C}, \{\mathcal{C}_i\}_{i=1}^N)$ *is proper if there exists a proper composition structure of* $(\mathcal{C}, \{\mathcal{C}_i\}_{i=1}^N)$.

The proof of Theorem 5 is found in “Appendix A”.

To illustrate the use of Theorem 5, consider the scenario with phases (I’–III’) and, more specifically, that the two-level contract hierarchy established in phase (I’) is $(\mathcal{C}_{Esys}, \{\mathcal{C}_{pot}, \mathcal{C}_{bat}, \mathcal{C}_{lMeter}\})$. As previously mentioned, in order to complete phase (III’), this two-level contract hierarchy needs to be proper. As expressed in Theorem 5, a sufficient condition for this is that there exists a proper composition structure of $(\mathcal{C}_{Esys}, \{\mathcal{C}_{pot}, \mathcal{C}_{bat}, \mathcal{C}_{lMeter}\})$. Hence, in accordance with Theorem 5, if e.g. the composition structure of $(\mathcal{C}_{Esys}, \{\mathcal{C}_{pot}, \mathcal{C}_{bat}, \mathcal{C}_{lMeter}\})$ in Fig. 9 is proper, then $(\mathcal{C}_{Esys}, \{\mathcal{C}_{pot}, \mathcal{C}_{bat}, \mathcal{C}_{lMeter}\})$ must also be proper. The following will examine if the composition structure in Fig. 9 is indeed proper, i.e. if it is in accordance with Definition 10.

As shown in the composition structure of $(\mathcal{C}_{Esys}, \{\mathcal{C}_{pot}, \mathcal{C}_{bat}, \mathcal{C}_{lMeter}\})$ in Fig. 9, it trivially holds that conditions (i) and (ii) of Definition 10 hold. Furthermore, considering that the union of sets of variables $\{f, l\}$ and $\{f, h\}$, respectively constrained by G_{Esys} and A_{Esys}, and set $(X_{pot} \cup X_{bat} \cup X_{lMeter}) \setminus X_{Esys} = \{v_{ref}, v_{gnd}, v_{branch}\}$ are disjoint, i.e. that $\{f, l, h\} \cap \{v_{ref}, v_{gnd}, v_{branch}\} = \emptyset$, condition (iii) of Definition 10 is met. Finally, since the extended projection of $\mathsf{G}_{lMeter} \cap \mathsf{G}_{bat} \cap \mathsf{G}_{pot}$ onto $\Xi \setminus \{v_{ref}, v_{gnd}, v_{branch}\} = \Xi \setminus ((X_{pot} \cup X_{bat} \cup X_{lMeter}) \setminus X_{Esys})$ is equal to G_{Esys}, condition (iv) of Definition 10 also holds. This means that the composition structure shown in Fig. 9 is proper.

Due to the fact the composition structure in Fig. 9 is proper, it follows that $(\mathcal{C}_{Esys}, \{\mathcal{C}_{pot}, \mathcal{C}_{bat}, \mathcal{C}_{lMeter}\})$ also is proper in accordance with Theorem 5. Thus, in accordance with Definition 8, for any set of elements with sets of port variables $X_{pot}, X_{bat}, X_{lMeter}$ where these elements are developed by the suppliers in phase (II’) such that condition (i) of Corollary 1 holds with respect to contracts $\mathcal{C}_{pot}$, $\mathcal{C}_{bat}$, and $\mathcal{C}_{lMeter}$, it follows that the composition of such a set of elements is such that condition (i) of Corollary 1 holds with respect to $\mathcal{C}_{Esys}$. For example, consider that the suppliers develop elements $\mathbb{E}_{bat}$, $\mathbb{E}_{pot}$, and $\mathbb{E}_{lMeter}$ in phase (II’). It trivially holds that each element $\mathbb{E}_i \in \{\mathbb{E}_{bat}, \mathbb{E}_{pot}, \mathbb{E}_{lMeter}\}$ is such that condition (i) of Corollary 1 holds with respect to $\mathcal{C}_i \in \{\mathcal{C}_{pot}, \mathcal{C}_{bat}, \mathcal{C}_{lMeter}\}$. Due to this and the fact that $(\mathcal{C}_{Esys}, \{\mathcal{C}_{pot}, \mathcal{C}_{bat}, \mathcal{C}_{lMeter}\})$ is proper, it automatically follows that $\mathbb{E}_{Esys}$ is such that condition (i) of Corollary 1 holds with respect to $\mathcal{C}_{Esys}$, as desired in phase (III’).

7 Discussion and related work

This section extends the discussions in Sects. 1–6 regarding proposed conditions (i) and (ii) of Corollary 1 and relations between these conditions and other concepts in the present paper with similar conditions and concepts in other work.

7.1 Conditions of contracts

As shown in Sect. 3.2, conditions (i) and (ii) of Corollary 1 with respect to a contract $\mathcal{C} = (\mathscr{A}, \mathsf{G}, X)$, or more specifically conditions $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}$ and $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B}$ on an element $\mathbb{E} = (X, \mathsf{B})$ and conditions $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$ and $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G} \neq \emptyset$ on environment $Env_{\mathcal{E}}(\mathbb{E})$, can be derived from a context characterized by the OEM/supplier scenario described in Sect. 3.2. More specifically, these conditions are derived to ensure that guarantee G is non-trivially fulfilled in a context subject to the following criteria:

(a) it cannot be ensured that the set of ports of element $\mathbb{E}$ are partitioned into inputs and outputs; and
(b) element $\mathbb{E}$ and its environment $Env_{\mathcal{E}}(\mathbb{E})$ are developed in complete isolation.

Note that contract $\mathcal{C}$ is not necessarily limited to the set of ports of $\mathbb{E}$.

Conditions $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}$ and $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$ have already been established in previous contract theories, e.g. [3,5]. As mentioned in Sect. 1, in a context where criterion (a) does not hold, i.e. when the set of ports of the element are partitioned into inputs and outputs, the non-trivial solution can typically be avoided by complementing conditions $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}$ and $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$ with additional conditions on *receptivity* [3,5] of input ports.

In comparison to theories such as [3–5,37] where the only conditions, additional to $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}$ and $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$, are conditions solely applicable when criterion (a) does not hold, condition $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B}$, proposed in the present paper, might appear too strict. This is due to the fact that relation $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B}$, in combination with $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}$, requires that $\mathsf{A} \cap \mathsf{B} = \mathsf{A} \cap \mathsf{G}$ while the input and output case allows B to be such that $\mathsf{A} \cap \mathsf{B} \subseteq \mathsf{A} \cap \mathsf{G}$. This additional implementation flexibility of the element is in the input and output case achieved solely through the constraint that the environment must be receptive to output ports, thus allowing the element to enforce stronger constraints on outputs than specified by guarantee G. When relaxing this constraint, as needed whenever criterion (a) holds, Theorem 1 shows that $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B}$ is indeed not too strong, despite the fact that less implementation flexibility of the element is given in comparison to the input and output case.

Analogous to the case where criterion (a) does not hold, conditions $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B}$ and $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G} \neq \emptyset$ might also be too strict in a context where criterion (b) does not hold, i.e. when $\mathbb{E}$ and $Env_{\mathcal{E}}(\mathbb{E})$ are not developed in isolation from each other. An example of such a case is when both $\mathbb{E}$ and $Env_{\mathcal{E}}(\mathbb{E})$ are developed within the same company. Due to the fact that the team that develops $\mathbb{E}$ has full access to $Env_{\mathcal{E}}(\mathbb{E})$, non-trivial solution $\mathsf{B} \cap \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \neq \emptyset$ is avoided by composing the elements in a trial-and-error fashion, relying on the expertise of the in-house development teams to make small modifications to the behaviors when needed. Therefore, in order to ensure that the guarantee is non-trivially fulfilled in such a context, it is sufficient that respective $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}$ and $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$ on the element and the environment hold.

Notably, in the present paper, conditions (i) and (ii) of Corollary 1, as well as the contract conditions and properties presented in Sects. 4–6 are all derived from the industrially relevant OEM/supplier scenario. Another way to derive contract conditions and properties is to first define a *refinement preorder on contracts*, and then use this relation to derive further conditions and properties of contracts. As previously mentioned in Sect. 6, in the context of the proposed contract conditions in the present paper, refinement is simply a special case of Definition 8, i.e. a proper contract hierarchy of the form $((\mathscr{A}, \mathsf{G}, X), \{(\mathscr{A}', \mathsf{G}', X)\})$.

Refinement has the property that if relation (4) holds for a set of elements $\mathcal{E}$ containing an element $\mathbb{E} = (X, \mathsf{B})$ that is such that condition (i) holds with respect to a contract $(\mathscr{A}, \mathsf{G}, X)$, then relation (4) will also hold if $\mathbb{E}$ is replaced with an element $\mathbb{E}' = (X, \mathsf{B}')$ that is such that condition (i) of Corollary 1 holds with respect to a contract $(\mathscr{A}', \mathsf{G}', X)$ that refines $(\mathscr{A}, \mathsf{G}, X)$. This property of refinement is called *independent implementability* in [84]. It is also stated in [84] that independent implementability can only be ensured in a context where ports are partitioned into inputs and outputs. Notably, this statement is indeed true when refinement is defined to mean that $\mathsf{A}_{\mathscr{A}} \subseteq \mathsf{A}_{\mathscr{A}'}$ and $\mathsf{G}' \subseteq \mathsf{G}$, as in e.g. [3,5] and corresponding to $\mathsf{A}_{\mathscr{A}} \rightarrow \mathsf{A}_{\mathscr{A}'}$ and $\mathsf{G}' \rightarrow \mathsf{G}$ in [84]. Notably, these conditions allow that $\mathsf{A} \cap \mathsf{G}' \subsetneq \mathsf{A} \cap \mathsf{G}$, which means that the conditions are actually tailored for the input-output case since relation $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E}')} \nsubseteq \mathsf{A} \cap \mathsf{G} \setminus (\mathsf{A} \cap \mathsf{G}')$, necessary for relation (3) to hold when $\mathbb{E}$ is replaced with $\mathbb{E}'$, is ensured solely through the constraint that the environment must be receptive to output ports. If this constraint is relaxed, as required when inputs and outputs are not considered, conditions $\mathsf{A}_{\mathscr{A}} \subseteq \mathsf{A}_{\mathscr{A}'}$ and $\mathsf{G}' \subseteq \mathsf{G}$ do not ensure the property of independent implementability. However, this property is indeed ensured if refinement is defined as a proper contract hierarchy of the form $((\mathscr{A}, \mathsf{G}, X), \{(\mathscr{A}', \mathsf{G}', X)\})$ as in the present paper, instead of being defined to mean conditions $\mathsf{A}_{\mathscr{A}} \subseteq \mathsf{A}_{\mathscr{A}'}$ and $\mathsf{G}' \subseteq \mathsf{G}$ as in [3,5,84]. This can be understood by considering the fact that sufficient and necessary conditions for a contract hierarchy $((\mathscr{A}, \mathsf{G}, X), \{(\mathscr{A}', \mathsf{G}', X)\})$ to be proper are $\mathsf{A}_{\mathscr{A}} \subseteq \mathsf{A}_{\mathscr{A}'}$, $\mathsf{G}' \subseteq \mathsf{G}$, and $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G}' = \mathsf{A}_{\mathscr{A}} \cap \mathsf{G}$. Condition $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G}' = \mathsf{A}_{\mathscr{A}} \cap \mathsf{G}$, additional to the conditions for refinement proposed in [3,5,84], caters to the relaxation of the constraint applicable only when inputs and outputs are considered.

7.2 Contracts and their properties, compositionality, elements, and runs

In Sects. 1–6, a vast number of general assume-guarantee theories [4,33–59] have been referred to, without a proper introduction. Therefore, the following two paragraphs of this section is dedicated to describing the contexts and applications of these general theories, as well as related concrete theories. The rest of this section compares the present paper with these theories and other related work, focusing on technical matters that have not been previously discussed in the present paper.

As mentioned in Sect. 3, the notion of contracts was first introduced in [1] to be used as formal specification in object-oriented programming. Since then, the use of contracts has been extended to component-based design [85] and a contract theory for analog systems have been proposed in [38,86]. Contracts have also been introduced in formalisms *Behavior Interaction Priority (BIP)* [87] and *refinement calculus* [88], in [39,89] and [40], respectively. Furthermore, in European research project SPEEDS [6], a contract theory [3–5] was introduced as a means to meet the challenges in the design of heterogeneous systems [7–9]. Similar work to [3–5] is presented in [46] and in [35] with tool support [90], and also in a more applied setting in [91,92]. The use of theory [3–5] has been advocated in [10–15] and the use of contracts in general has been proposed for analyzes integration [93] and as a means to achieve functional safety in [94,95] and also in [96] with tool support [97]. Contract theory has also been extended both with *modalities* [98] in [12,41] and to a stochastic setting in [37,42]. Meta theories of contracts have been established in [33,36], and in [44] with refinement [45]. Theory [44] also clarifies the distinction between contracts and *specification* *theories*, e.g. [56,99,100] that extend interface automata [101,102] and where [56] is shown to also support assume-guarantee reasoning in [103].

More generally than contracts, assume-guarantee reasoning can be traced back to two independent theories [104,105] and [50] concerning *compositional* [31,32] proof methods for concurrent programs. However, the ideas in [50,104,105] can be traced even further back to proof methods [2,106,107] for sequential programs and non-compositional proof methods [108–110] for concurrent programs. Since the conception of [50,104,105], several theories that extend the ideas in [104,105] and [50], have emerged, such as e.g. [34,48] and [49,111], respectively. Furthermore, assume-guarantee reasoning has also been used in formal verification, see e.g. [59,112,113] or [114] for an overview. Moreover, automatic techniques for assume-guarantee reasoning have been proposed, see e.g. [115] or [116] for a survey. Given that the two approaches [104,105] and [50] are the same in principle, meta theories [47,51–54], and [54] as an extension of [117], have been introduced to unify [104,105] and [50]. General assume-guarantee theories are also presented in [57,58], and in [43] based on [118,119]. Furthermore, with inspiration from [48,120], how to perform compositional verification on architecture models is described in [121].

As previously mentioned in Sect. 6, both properties *refinement* and *parallel composition* can be derived from Definition 8 in accordance with the meta theory in [33]. As further shown in [33] and also in [44], these properties also allow deriving other properties of contracts, namely *conjunction* [3–5,33,37,56] and *quotient* [33,56]. While both parallel composition and conjunction merge a set of contracts into a single contract, the latter is only applicable when the contracts contain the same set of port variables and concerns the case when the contracts are specified for different *viewpoints* [122,123]. Quotient computes a missing contract in a contract hierarchy to achieve compositionality.

In Sect. 2.2, the concept of an element that essentially corresponds to a HRC [68,69] as used in [3–5], was introduced. The main difference is that a HRC can have several *implementations*, i.e. behaviors, which means that an element corresponds to an implementation of a HRC, rather than to a HRC itself. An element in the present paper is in that sense more similar to a component as defined in [65] that is inspired by the *tagged signal model* [124] and *interface theory* [125].

In accordance with [3–5] and also with [14,64], contracts and behaviors of elements are in the present paper both defined by relying on the concept of assertions as a set of runs. The concept of runs is, in turn, largely inspired by the works in [36,126,127] that generalize the concept of *traces* [61–63] to behaviors that are independent of a particular model of computation. Notably, two assertions are equivalent if they have the same runs, i.e. they are *trace-equivalent* [128], which means that assertions are limited to a weaker form of equivalence than e.g. observation equivalence [129] or equivalence through alternating simulation [130], which can be verified on labeled and alternating transition systems, respectively. However, as shown in [128], for deterministic models, trace equivalence means observation equivalence, and vice versa.

8 Conclusion

This paper has presented a general compositional contract theory for modeling and specifying *heterogeneous systems*. As the main contribution, given a contract for an element representing any part of a heterogeneous system, e.g. SW, mechanical, or electrical part, Corollary 1 presented clearly separated *conditions* (i) and (ii) on the element and its environment where the conditions ensure that the guarantee is non-trivially fulfilled by the composition of the element and the environment.

In contrast to similar conditions of other general assume-guarantee theories [3–5,33–59], while explicitly considering the set of port variables of an element, conditions (i) and (ii) require neither that this set is partitioned into inputs and outputs nor that the assumptions and guarantee must be specified over this set of port variables. The former means that the causality of the port variables can remain unspecified, which is common and recommended practice when modeling physical parts. The latter allows assigning the *responsibility* of fulfilling a global property to the element; this is needed in order to properly express safety specifications for the element, e.g. in accordance with ISO 26262.

The ability to assign the *responsibility* of fulfilling a global property to an element, increases the expressiveness with respect to how a contract can be specified. To facilitate the specification of contracts in practice, scoping conditions were introduced that limit the set of port variables over which the assumptions and the guarantee of a contract are specified. These scoping conditions ensure that certain necessary properties of conditions (i) and (ii) are are not violated without limiting expressiveness. Notably, these conditions can be checked, not only for the cases where the assumptions and the guarantee are specified using formal notation, but also when they are specified using *semi-formal notation*, e.g. as free text with formal references to port variables of elements. Hence, considering a tool where these checks are automatically performed, feedback to a user specifying a contract can be given, both when semi-formal and formal notations are used.

In the context of a scenario where a contract is used to outsource the development of an element, necessary contract properties *consistency* and *compatibility* were presented. Complementary necessary and sufficient conditions of these properties were also introduced where these conditions are easier to enforce in practice (e.g. by a tool) than their corresponding definitions. Furthermore, as a basis for structuring contracts in parallel to an hierarchical composition of a set of elements, a graph, called a *composition structure*, was introduced. Based on a composition structure, sufficient conditions to achieve *compositionality* was presented. Note that proving that the sufficient conditions hold, requires specifying the assumptions and the guarantees using formal notation. However, regardless if the assumptions and guarantees are specified using formal, semi-formal, or *informal notation*, e.g. as free text, the conditions for structuring the overall intended relations between the assumptions and guarantees as a composition structure, still apply. This means that, regardless of the level of formalization used in the specifications, support for structuring a contract hierarchy can be given in the form of a tool that enforces these conditions.

Considering the concepts presented above, they are all general in both the senses that they are relevant to any developer of heterogeneous system parts, and that they rely on a general set-theoretic formalism. Due to the generality of the theory, it is essentially applicable in any context, and it can also be instantiated by more concrete theories whenever needed. Moreover, the theory is tightly coupled with practical application where introduced definitions have been both well motivated by industrial needs and/or scenarios, and complemented with necessary and sufficient conditions that can more easily be enforced in practice, e.g. by tools implementing the theory. As previously mentioned, the concrete support that can be given by such tools, is not limited to the cases where formal notations are used, but also when semi-formal and informal notations are used. Hence, not only does the presented theory constitute a general compositional contract theory for modeling and specifying heterogeneous systems, but the theory is indeed also accommodated for providing concrete support for developing such systems in practice.

Acknowledgements This work is a result from a collaborative effort with automotive manufacturer *Scania CV AB*, and has received funding under grant agreement no.2011 − 04446 from *Fordonsstrategisk Forskning och Innovation* (FFI)—a partnership between the Swedish government and automotive industry for joint funding of research, innovation and development concentrating on Climate & Environment and Safety

A Appendix

This section presents proofs of Propositions 1 and 5, Lemmas 1 and 2, and Theorems 2 and 5.

Proposition 1 *Given an assertion* W*, a set of variables* X *is equal to* X_{W} *if and only if each variable in* X *is constrained by* W *and* $\widehat{proj}_X(\mathsf{W}) = \mathsf{W}$.

Proof Consider an assertion W.

For the if part, assume that there exists a set of variables X constrained by W such that $\widehat{proj}_X(\mathsf{W}) = \mathsf{W}$. Assume $X \neq X_{\mathsf{W}}$, will be shown to lead to a contradiction. This means that either: (i) $X = \emptyset$ and $X_{\mathsf{W}} \neq \emptyset$ or (ii) there exists a variable $x \in X$ that is not also in X_{W}. From assuming that case (i) holds, in accordance with relation (2), it follows that $\mathsf{W} = \Omega$ since $\widehat{proj}_{X=\emptyset}(\mathsf{W}) = \Omega$ and $\widehat{proj}_X(\mathsf{W}) = \mathsf{W}$. This and the fact that $X_{\mathsf{W}} \neq \emptyset$ imply, in accordance with Definition 1, that there exists a variable $x \in X_{\mathsf{W}}$ such that $\widehat{proj}_{\Xi\setminus\{x\}}(\Omega) \neq \Omega$. This is a contradiction since, in accordance with Sect. 2.1 and relation (2), Ω is indeed obtained if the set of all runs for $\Xi \setminus \{x\}$ is extended with all runs for x. Hence, case (i) must be false, and it must either be that case (ii) holds or that assumption $X \neq X_{\mathsf{W}}$ is false. Assuming that case (ii) holds directly yields a contradiction since this would mean that there exists a variable $x \notin X_{\mathsf{W}}$ such that $\widehat{proj}_{\Xi\setminus\{x\}}(\mathsf{W}) \neq \mathsf{W}$, which is not in accordance with Definition 1. This means that the assumption that $X \neq X_{\mathsf{W}}$ must be false and thus, it follows that $X = X_{\mathsf{W}}$.

For the if-only part, it suffices to show that $\widehat{proj}_{X_{\mathsf{W}}}(\mathsf{W}) = \mathsf{W}$, since W constrains each variable in X_{W} in accordance with Definition 1. Assume that $\widehat{proj}_{X_{\mathsf{W}}}(\mathsf{W}) \neq \mathsf{W}$, which will be shown to lead to a contradiction. This implies that that there exists a run $\omega_{\Xi,T} \in \mathsf{W}$ and a pair $(x \notin X_{\mathsf{W}}, \xi')$ such that a run $\omega'_{\Xi,T}$ is not in W if $\omega'_{\Xi,T}$ is obtained by replacing the pair $(x, \xi) \in \omega_{\Xi,T}$ with (x, ξ'). In accordance with relation (2), this means that $\widehat{proj}_{\Xi\setminus\{x\}}(\mathsf{W}) \neq \mathsf{W}$ and thus, W constrains x in accordance with Definition 1. However, this contradicts the fact that $x \notin X_{\mathsf{W}}$, which means that the assumption that $\widehat{proj}_{X_{\mathsf{W}}}(\mathsf{W}) \neq \mathsf{W}$ must be false, and it rather must hold that $\widehat{proj}_{X_{\mathsf{W}}}(\mathsf{W}) = \mathsf{W}$, which completes the proof. □

Lemma 1 *Given two assertions* W *and* W' *where* $\mathsf{W} \cup \mathsf{W}' \neq \emptyset$*, it holds that* $X_{\mathsf{W}\cup\mathsf{W}'} \subseteq X_{\mathsf{W}} \cup X_{\mathsf{W}'}$.

Proof In accordance with Proposition 1, it holds that $\mathsf{W}\cup\mathsf{W}' = \widehat{proj}_{X_{\mathsf{W}}}(\mathsf{W}) \cup \widehat{proj}_{X_{\mathsf{W}'}}(\mathsf{W}')$. Furthermore, in accordance with relation (2), it holds that $\widehat{proj}_{X_{\mathsf{W}}}(\mathsf{W})$ and $\widehat{proj}_{X_{\mathsf{W}'}}(\mathsf{W}')$ are obtained by extending $proj_{X_{\mathsf{W}}}(\mathsf{W})$ and $proj_{X_{\mathsf{W}'}}(\mathsf{W}')$ with all possible runs for $\Xi \setminus X_{\mathsf{W}}$ and $\Xi \setminus X_{\mathsf{W}'}$, respectively. Notably, both these set of runs are extended with all possible runs for $\Xi \setminus (X_{\mathsf{W}} \cup X_{\mathsf{W}'})$. This means, since $\mathsf{W} \cup \mathsf{W}' = \widehat{proj}_{X_{\mathsf{W}}}(\mathsf{W}) \cup \widehat{proj}_{X_{\mathsf{W}'}}(\mathsf{W}')$, that it follows that $\mathsf{W} \cup \mathsf{W}'$ is obtained by extending $proj_{X_{\mathsf{W}}\cup X_{\mathsf{W}'}}(\mathsf{W} \cup \mathsf{W}')$ with all possible runs for $\Xi \setminus (X_{\mathsf{W}} \cup X_{\mathsf{W}'})$. In accordance with Definition 1, this means that $X_{\mathsf{W} \cup \mathsf{W}'} \subseteq X_{\mathsf{W}} \cup X_{\mathsf{W}'}$. □

Proposition 5 *Given two assertions* W *and* W' *where* $\mathsf{W} \cap \mathsf{W}' \neq \emptyset$*, it holds that* $X_{\mathsf{W} \cap \mathsf{W}'} \subseteq X_{\mathsf{W}} \cup X_{\mathsf{W}'}$.

Proof Given two assertions W and W' where $\mathsf{W} \cap \mathsf{W}' \neq \emptyset$, in accordance with relation (1), for each run $\omega \in \mathsf{W} \cap \mathsf{W}'$, it holds that $proj_{X_{\mathsf{W}} \cup X_{\mathsf{W}'}}(\{\omega\})$ is in both $proj_{X_{\mathsf{W}} \cup X_{\mathsf{W}'}}(\mathsf{W})$ and $proj_{X_{\mathsf{W}} \cup X_{\mathsf{W}'}}(\mathsf{W}')$. This and since $\widehat{proj}_{X_{\mathsf{W}} \cup X_{\mathsf{W}'}}(\mathsf{W}) = \mathsf{W}$ and $\widehat{proj}_{X_{\mathsf{W}} \cup X_{\mathsf{W}'}}(\mathsf{W}) = \mathsf{W}'$ in accordance with Proposition 1 and Sect. 2.1.2, imply that each run in $\widehat{proj}_{X_{\mathsf{W}} \cup X_{\mathsf{W}'}}(\{\omega\})$ is in both W and W'. Overall, this means that each run $\omega_{X_{\mathsf{W}} \cup X_{\mathsf{W}'}, T}$ in $proj_{X_{\mathsf{W}} \cup X_{\mathsf{W}'}}(\mathsf{W} \cap \mathsf{W}')$ can be extended with any run for $\Xi \setminus (X_{\mathsf{W}} \cup X_{\mathsf{W}'})$ over T, and the obtained run will also be in $\mathsf{W} \cap \mathsf{W}'$. In accordance with relation (2), this means that $\widehat{proj}_{X_{\mathsf{W}} \cup X_{\mathsf{W}'}}(\mathsf{W} \cap \mathsf{W}') = \mathsf{W} \cap \mathsf{W}'$. In accordance with Proposition 1 and Sect. 2.1.2, it follows that $X_{\mathsf{W} \cap \mathsf{W}'} \subseteq X_{\mathsf{W}} \cup X_{\mathsf{W}'}$. □

Lemma 2 *Given two assertions* W *and* W'*, and set of variables* X*, it holds that* $\widehat{proj}_X(\mathsf{W}) \subseteq \widehat{proj}_X(\mathsf{W}')$*, if* $\mathsf{W} \subseteq \mathsf{W}'$.

Proof Given two assertions W and W' and a set of variables X, consider that $\mathsf{W} \subseteq \mathsf{W}'$. In accordance with relation (1), the set of runs $proj_X(\mathsf{W})$ and $proj_X(\mathsf{W}')$ are obtained by removing each pair $(x \notin X, \xi)$ from each run in W and W', respectively. This and since $\mathsf{W} \subseteq \mathsf{W}'$, it follows that each pair that is removed from W' to obtain $proj_X(\mathsf{W}')$, is also removed from W to obtain $proj_X(\mathsf{W})$. Hence, it holds that $proj_X(\mathsf{W}) \subseteq proj_X(\mathsf{W}')$. In accordance with relation (2), assertions $\widehat{proj}_X(\mathsf{W})$ and $\widehat{proj}_X(\mathsf{W}')$ are obtained by extending each run $\omega_{\Xi,T}$ in $proj_X(\mathsf{W})$ and $proj_X(\mathsf{W}')$ with all possible runs for $\Xi \setminus X$ over T. This and since $proj_X(\mathsf{W}) \subseteq proj_X(\mathsf{W}')$, it follows that each run $\omega_{\Xi,T}$ that is extended in $proj_X(\mathsf{W})$ with all possible runs for $\Xi \setminus X$ over T to obtain $\widehat{proj}_X(\mathsf{W})$, is also extended in $proj_X(\mathsf{W})$ with all possible runs for $\Xi \setminus X$ over T to obtain $\widehat{proj}_X(\mathsf{W}')$. Hence, it holds that $\widehat{proj}_X(\mathsf{W}) \subseteq \widehat{proj}_X(\mathsf{W}')$. □

Theorem 2 *Given a contract* $(\mathscr{A}, \mathsf{G}, X)$ *and set of variables* $X_{Env_{\mathcal{E}}(\mathbb{E})}$*, there exists a contract* $(\mathscr{A}', \mathsf{G}', X)$ *where:*

a) $X_{\mathsf{A}_{\mathscr{A}'}} \subseteq X_{Env_{\mathcal{E}}(\mathbb{E})}$*; and*
b) $X_{\mathsf{G}'} \subseteq X_{Env_{\mathcal{E}}(\mathbb{E})} \cup X$

such that for each set of elements containing an element (X, B) *and where the environment of* (X, B) *is pair* $(X_{Env_{\mathcal{E}}(\mathbb{E})}, \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})})$*:*

(i') $\mathsf{A}_{\mathscr{A}'} \cap \mathsf{B} \subseteq \mathsf{G}'$ *and* $\mathsf{A}_{\mathscr{A}'} \cap \mathsf{G}' \subseteq \mathsf{B}$*, and*
(ii') $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}'}$ *and* $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G}' \neq \emptyset$*,*

if and only if

(i) $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}$ *and* $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B}$*, and*
(ii) $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$ *and* $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G} \neq \emptyset$.

Proof Consider a contract $(\mathscr{A}, \mathsf{G}, X)$ and a set of variables $X_{Env_{\mathcal{E}}(\mathbb{E})}$. First, let $(\mathscr{A}', \mathsf{G}', X)$ be a contract with a first property that $\mathsf{A}_{\mathscr{A}'}$ is the union of the behavior of each element $(X_{Env_{\mathcal{E}}(\mathbb{E})}, \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})})$ where $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}}$ and $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G} \neq \emptyset$. This means, in accordance with Lemma 1 and Definition 2, it holds that condition (a) holds, i.e. $X_{\mathsf{A}_{\mathscr{A}'}} \subseteq X_{Env_{\mathcal{E}}(\mathbb{E})}$.

Generally, this first property also implies that

$$\forall(X_{Env_{\mathcal{E}}(\mathbb{E})}, \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})}) \text{ where } \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}} \text{ and } \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G} \neq \emptyset : \\ \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}'} \subseteq \mathsf{A}_{\mathscr{A}}. \tag{22}$$

As a second property, consider that G' is the union of each intersection $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B}$ where (X, B) is an element where $\emptyset \neq \mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}$ and $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B}$. Given the fact that $X_{\mathsf{A}_{\mathscr{A}'}} \subseteq X_{Env_{\mathcal{E}}(\mathbb{E})}$ and in accordance with Lemma 1, Proposition 5, and Definition 2, it follows that condition (b) holds, i.e. $X_{\mathsf{G}'} \subseteq X_{Env_{\mathcal{E}}(\mathbb{E})} \cup X$. Generally, this also implies that

$$\forall(X, \mathsf{B}) \text{ where } \mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G} \text{ and } \mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B} : \ \mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}' \subseteq \mathsf{G}. \tag{23}$$

Now, for the if case, assume that conditions (i) and (ii) hold with respect to $(\mathscr{A}', \mathsf{G}', X)$ for an arbitrary set of elements $\mathcal{E}$ containing an element (X, B) and where pair $(X_{Env_{\mathcal{E}}(\mathbb{E})}, \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})})$ is the environment of (X, B). Condition (ii) and relation (22) imply that

$$\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \subseteq \mathsf{A}_{\mathscr{A}'} \subseteq \mathsf{A}_{\mathscr{A}}. \tag{24}$$

Condition (i) and relation (23) imply that $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}' \subseteq \mathsf{G}$. This and relation (24) imply that

$$\mathsf{A}_{\mathscr{A}'} \cap \mathsf{B} \subseteq \mathsf{G}' \subseteq \mathsf{G}. \tag{25}$$

Furthermore, in accordance with Theorem 1, conditions (i) and (ii) imply that $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{B} \neq \emptyset$. This and relations (24) and (25) imply that $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G}' \neq \emptyset$. Finally, relation $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B}$ of condition (i) and the fact that $\mathsf{A}_{\mathscr{A}'} \subseteq \mathsf{A}_{\mathscr{A}}$ and $\mathsf{G}' \subseteq \mathsf{G}$, as expressed in relations (24) and (25), respectively, imply that $\mathsf{A}_{\mathscr{A}'} \cap \mathsf{G}' \subseteq \mathsf{B}$. Relations (24), (25), $\mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})} \cap \mathsf{G}' \neq \emptyset$ and $\mathsf{A}_{\mathscr{A}'} \cap \mathsf{G}' \subseteq \mathsf{B}$ imply that conditions (i') and (ii') hold.

For the if-only case, assume that conditions (i') and (ii') hold for an arbitrary set of elements $\mathcal{E}$ containing an element (X, B) and where pair $(X_{Env_{\mathcal{E}}(\mathbb{E})}, \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})})$ is the environment of (X, B). From the first and second property of contract $(\mathscr{A}', \mathsf{G}', X)$, it directly follows that (X, B) and $(X_{Env_{\mathcal{E}}(\mathbb{E})}, \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})})$ are such condition (i) and (ii) hold, respectively.

Thus, given contract $(\mathscr{A}, \mathsf{G}, X)$ and set of variables $X_{Env_{\mathcal{E}}(\mathbb{E})}$, it can be concluded that there exists a contract $(\mathscr{A}', \mathsf{G}', X)$ where conditions (a) and (b) hold such that for each set of elements containing an element (X, B) and where pair $(X_{Env_{\mathcal{E}}(\mathbb{E})}, \mathsf{B}_{Env_{\mathcal{E}}(\mathbb{E})})$ is the environment of (X, B), conditions (i') and (ii') hold if and only if conditions (i) and (ii) hold. □

Theorem 5 *Given a contract $\mathcal{C} = (\mathscr{A}, \mathsf{G}, X)$ and a set of contracts $\{\mathcal{C}_i\}_{i=1}^N$ where $\mathcal{C}_i = (\mathscr{A}_i, \mathsf{G}_i, X_i)$ and $X \subseteq \bigcup_{i=1}^N X_i$, the two-level contract hierarchy $(\mathcal{C}, \{\mathcal{C}_i\}_{i=1}^N)$ is proper if there exists a proper composition structure of $(\mathcal{C}, \{\mathcal{C}_i\}_{i=1}^N)$.*

Proof Given a contract $\mathcal{C} = (\mathscr{A}, \mathsf{G}, X)$ and a set of contracts $\{\mathcal{C}_i\}_{i=1}^N$ where $\mathcal{C}_i = (\mathscr{A}_i, \mathsf{G}_i, X_i)$ and $X \subseteq \bigcup_{i=1}^N X_i$, assume that there exists a proper composition structure of the contract hierarchy $(\mathcal{C}, \{\mathcal{C}_i\}_{i=1}^N)$. In accordance with Definition 8, assume that there exists an arbitrary set of elements $\{(X_i, \mathsf{B}_i)\}_{i=1}^N$ such that $\mathsf{A}_{\mathscr{A}_i} \cap \mathsf{B}_i \subseteq \mathsf{G}_i$ and $\mathsf{A}_{\mathscr{A}_i} \cap \mathsf{G}_i \subseteq \mathsf{B}_i$ holds for each i.

With the intent to first show that the composition (X, B) of $\{(X_i, \mathsf{B}_i)\}_{i=1}^N$ onto X is such that $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}$ holds, consider the fact that $\mathsf{A}_{\mathscr{A}_i} \cap \mathsf{B}_i \subseteq \mathsf{G}_i$ holds for each i. In accordance with Definition 9, this and since each assertion that is a direct predecessor of an assumption in $\mathscr{A}_i$ is either a guarantee G_j or an assumption in $\mathscr{A}$, it follows that $\mathsf{A}_{\mathscr{A}} \cap \bigcap_{i=1}^N \mathsf{B}_i \subseteq \bigcap_{i=1}^N \mathsf{G}_i$ in accordance with condition (ii) of Definition 10. This and condition (i) of Definition 10 imply that $\mathsf{A}_{\mathscr{A}} \cap \bigcap_{i=1}^N \mathsf{B}_i \subseteq \mathsf{G}$. In accordance with Sect. 2.1 and Proposition 1, this and considering that condition (iii) implies that neither $\mathsf{A}_{\mathscr{A}}$ nor G constrains any subset of $\bigcup_{i=1}^{N} X_i \setminus X$, it follows that $\mathsf{A}_{\mathscr{A}} \cap \widehat{proj}_X(\bigcap_{i=1}^{N} \mathsf{B}_i) \subseteq \mathsf{G}$ due to the fact that $X = \bigcup_{i=1}^{N} X_i \setminus (\bigcup_{i=1}^{N} X_i \setminus X)$. In accordance with Definition 3, this means that it holds that $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}$.

With the intent to now show that the composition (X, B) of $\{(X_i, \mathsf{B}_i)\}_{i=1}^{N}$ onto X also is such that $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B}$ holds, consider the fact that $\mathsf{A}_{\mathscr{A}_i} \cap \mathsf{G}_i \subseteq \mathsf{B}_i$ holds for each i. In accordance with Definition 9, this and since each assertion that is a direct predecessor of an assumption in $\mathscr{A}_i$ is either a guarantee G_i or an assumption in $\mathscr{A}$, it trivially follows that $\mathsf{A}_{\mathscr{A}} \cap \bigcap_{i=1}^{N} \mathsf{G}_i \subseteq \bigcap_{i=1}^{N} \mathsf{B}_i$ in accordance with condition (ii) of Definition 10. This and given that it holds that $\bigcap_{i=1}^{N} \mathsf{B}_i \subseteq \widehat{proj}_X(\bigcap_{i=1}^{N} \mathsf{B}_i)$ in accordance with relations (1) and (2), it follows that $\mathsf{A}_{\mathscr{A}} \cap \bigcap_{i=1}^{N} \mathsf{G}_i \subseteq \mathsf{B}$ in accordance with Definition 3. In accordance with Sect. 2.1 and Proposition 1, this and due to the fact that Definition 3 and condition (iii) of Definition 10 imply that neither $\mathsf{A}_{\mathscr{A}}$ nor B can constrain any non-empty subset of $\bigcup_{i=1}^{N} X_i \setminus X$, it holds that $\mathsf{A}_{\mathscr{A}} \cap \widehat{proj}_{\Xi \setminus (\bigcup_{i=1}^{N} X_i \setminus X)}(\bigcap_{i=1}^{N} \mathsf{G}_i) \subseteq \mathsf{B}$. This and condition (iv) of Definition 10 imply that $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B}$.

Since the set of elements $\{(X_i, \mathsf{B}_i)\}_{i=1}^{N}$ was chosen arbitrarily, it holds that the composition of *each set of elements* $\{(X_i, \mathsf{B}_i)\}_{i=1}^{N}$ onto X is such that relations $\mathsf{A}_{\mathscr{A}} \cap \mathsf{B} \subseteq \mathsf{G}$ and $\mathsf{A}_{\mathscr{A}} \cap \mathsf{G} \subseteq \mathsf{B}$ hold, if each element (X_i, B_i) is such that $\mathsf{A}_{\mathscr{A}_i} \cap \mathsf{B}_i \subseteq \mathsf{G}_i$ and $\mathsf{A}_{\mathscr{A}_i} \cap \mathsf{G}_i \subseteq \mathsf{B}_i$ hold. This means that $(\mathcal{C}, \{\mathcal{C}_i\}_{i=1}^{N})$ is proper in accordance with Definition 8, which completes the proof. □

References

1. Meyer B (1992) Applying "design by contract". IEEE Comput 25:40–51
2. Hoare CAR (1969) An axiomatic basis for computer programming. Commun ACM 12(10):576–580
3. Benveniste A et al (2008) Multiple viewpoint contract-based specification and design. In: Boer FS et al (eds) Formal methods for components and object. Springer, Berlin, pp 200–225
4. Benveniste A, Caillaud B, Passerone R (2009) Multi-viewpoint state machines for rich component models. In: Nicolescu G, Mosterman P (eds) Model-based design for embedded systems. Taylor & Francis, Boca Raton, pp 487–518
5. Sangiovanni-Vincentelli A L, Damm W, Passerone R (2012) Taming Dr. Frankenstein: contract-based design for cyber-physical systems. Eur J Control 18(3):217–238
6. SPEEDS (2006–2009) SPEculative and exploratory design in systems engineering. http://www.speeds.eu.com/
7. Henzinger T, Sifakis J (2007) The discipline of embedded systems design. Computer 40(10):32–40
8. Lee E (2008) Cyber physical systems: design challenges. In: 11th IEEE international symposium on object oriented real-time distributed computing (ISORC), pp 363–369
9. Rawat D B, Rodrigues J J, Stojmenovic I (2015) Cyber-physical systems: from theory to practice. CRC Press, Boca Raton
10. Baumgart A et al (2011) A model-based design methodology with contracts to enhance the development process of safety-critical systems. In: Software technologies for embedded and ubiquitous systems. Volume 6399 of Lecture Notes in Computer Science. Springer, Berlin, pp 59–70
11. Damm W, Josko B, Peikenkamp T (2009) Contract based ISO CD 26262 safety analysis. In: SAE Technical Paper. SAE International. doi:10.4271/2009-01-0754
12. Damm W, Hungar H, Josko B, Peikenkamp T, Stierand I (2011) Using contract-based component specifications for virtual integration testing and architecture design. In: 2011 Design, automation test in Europe. DATE'11, pp 1–6. doi:10.1109/DATE.2011.5763167
13. Westman J, Nyberg M (September 2013) A reference example on the specification of safety requirements using ISO 26262. In: Roy M (ed) Proceedings of workshop DECS (ERCIM/EWICS workshop on dependable embedded and cyber-physical systems) of the 32nd international conference on computer safety, reliability and security, France , p NA
14. Westman J, Nyberg M, Törngren M (2013) Structuring safety requirements in ISO 26262 using contract theory. In: Bitsch F, Guiochet J, Kaniche M (eds) Computer safety, reliability, and security, vol 8153. Lecture Notes in Computer Science. Springer, Berlin, pp 166–177

15. Westman J, Nyberg M (Jan 2015) Extending contract theory with safety integrity levels. In: IEEE 15th international symposium on high-assurance systems engineering (HASE) 2015. Springer, Berlin
16. Cheng BHC, Atlee JM (2007) Research directions in requirements engineering. In: Future of software engineering, 2007. FOSE '07. IEEE Computer Society, Washington, DC, pp 285–303. doi:10.1109/FOSE.2007.17
17. Hull MEC, Jackson K, Dick J (eds) (2011) Requirements engineering, 3rd edn. Springer, New York
18. Zave P, Jackson M (1997) Four dark corners of requirements engineering. ACM Trans Softw Eng Methodol 6(1):1–30
19. Fritzson P (2011) Introduction to modeling and simulation of technical and physical systems with Modelica. Wiley, New York
20. Fritzson P, Engelson V (1998) Modelica—a unified object-oriented language for system modeling and simulation. In: Jul E (ed) ECOO'98—object-oriented programming, vol 1445. Lecture Notes in Computer Science. Springer, Berlin, pp 67–90
21. Fritzson P (2014) Principles of object-oriented modeling and simulation with modelica 3.3: a cyber-physical approach. Wiley, New York
22. van Schouwen AJ, Parnas DL, Madey J (Jan 1993) Documentation of requirements for computer systems. In: [1993] Proceedings of the IEEE international symposium on requirements engineering, pp 198–207
23. Parnas DL, Madey J (1995) Functional documents for computer systems. Sci Comput Program 25(1):41–61
24. Liang F et al (2012) Model-based requirement verification : a case study. In: Proceedings of the 9th international Modelica conference, pp 263–268
25. Schamai W et al (2009) Towards unified system modeling and simulation with ModelicaML: modeling of executable behavior using graphical notations. In: 7th Modelica conference 2009, University Electronic Press
26. Boulanger J-L, Dao VQ (July 2008) Requirements engineering in a model-based methodology for embedded automotive software. In: IEEE international conference on research, innovation and vision for the future, 2008. RIVF 2008, pp 263–268
27. Friedenthal S, Moore A, Steiner R (2008) A practical guide to SysML: systems modeling language. Morgan Kaufmann Publishers Inc., San Francisco
28. IEC 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems (2010)
29. ISO 26262: "Road vehicles-Functional safety" (2011)
30. Izosimov V, Ingelsson U, Wallin A (2012) Requirement decomposition and testability in development of safety-critical automotive components. In: Ortmeier F, Daniel P (eds) Computer safety, reliability, and security, vol 7612. Lecture Notes in Computer Science. Springer, Berlin, pp 74–86
31. de Roever W, Langmaack H, Pnueli A (1998) Compositionality: the significant difference. Springer, New York
32. Hooman J, de Roever WP (1986) The quest goes on: a survey of proofsystems for partial correctness of CSP. In: de Bakker JW, de Roever WP, Rozenberg G (eds) Current trends in concurrency, overviews and tutorials. Springer, Berlin, pp 343–395
33. Benveniste A et al (November 2012) Contracts for system design. Rapport de recherche RR-8147, INRIA
34. Abadi M, Lamport L (1993) Composing specifications. ACM Trans Program Lang Syst 15(1):73–132
35. Cimatti A, Tonetta S (2015) Contracts-refinement proof system for component-based embedded systems. Sci Comput Program 97(Part 3):333–348. Object-Oriented Programming and Systems (OOPS 2010) Modeling and Analysis of Compositional Software (papers from fEUROMICROg SEAA'12)
36. Negulescu R (2000) Process spaces. In: Proceedings of the 11th international conference on concurrency theory. CONCUR '00. Springer, London, pp 199–213
37. Delahaye B, Caillaud B, Legay A (2011) Probabilistic contracts: a compositional reasoning methodology for the design of systems with stochastic and/or non-deterministic aspects. Form. Methods Syst. Des. 38(1):1–32
38. Sun X et al (July 2009) Contract-based system-level composition of analog circuits. In: Design automation conference, 2009. DAC '09. 46th ACM/IEEE, pp 605–610
39. Quinton S, Graf S (Nov. 2008) Contract-based verification of hierarchical systems of components. In: Sixth IEEE international conference on software engineering and formal methods, 2008. SEFM '08, pp 377–381
40. Back R-J, von Wright J (2000) Contracts, games, and refinement. Inf Comput 156(1–2):25–45
41. Goessler G, Raclet J-B (2009) Modal contracts for component-based design. In: Proceedings of the 2009 7th IEEE international conference on software engineering and formal methods. SEFM '09, Washington, DC, USA. IEEE Computer Society, pp 295–303

42. Goessler G, Xu D, Girault A (2012) Probabilistic contracts for component-based design. Form Methods Syst Des 41(2):211–231
43. Broy M (2011) Towards a theory of architectural contracts: schemes and patterns of assumption/promise based system specification. In: Broy M, Leuxner C, Hoare T (eds) Software and systems safety—specification and verification. NATO Science for Peace and Security Series—D: information and communication security, vol 30. IOP Press, Amsterdam, pp 33–87
44. Bauer S et al (2012) Moving from specifications to contracts in component-based design. In: Lara J, Zisman A (eds) Fundamental approaches to software Engineering, vol 7212. Lecter Notes in Computer Science. Springer, Berlin, pp 43–58
45. Bauer SS, Hennicker R, Legay A (2014) A meta-theory for component interfaces with contracts on ports. Sci Comput Program 91:70–89
46. Le TTH et al (2015) A tag contract framework for modeling heterogeneous systems. Sci Comput Program 115:225–246
47. Maier P (2001) A set-theoretic framework for assume-guarantee reasoning. In: Orejas F, Spirakis P, van Leeuwen J (eds) Automata, languages and programming, vol 2076. Lecture Notes in Computer Science. Springer, Berlin, pp 821–834
48. Mcmillan KL (1999) Circular compositional reasoning about liveness. In: Advances in hardware design and verification: IFIP WG10.5 international conference on correct hardware design and verification methods (CHARME 99), vol 1703 of Lecture Notes in Computer Science. Springer, Berlin, pp 342–345
49. Abadi M, Lamport L (1995) Conjoining specifications. ACM Trans Program Lang Syst 17(3):507–535
50. Misra J, Chandy K (1981) Proofs of networks of processes. IEEE Trans Softw Eng SE–7(4):417–426
51. Cau A, Collette P (1996) Parallel composition of assumption-commitment specifications. Acta Inf 33(2):153–176
52. Xu Q, Cau A, Collette P (1994) On unifying assumption-commitment style proof rules for concurrency. In: Jonsson B, Parrow J (eds) CONCUR'94: concurrency theory, vol 836. Lecture Notes in Computer Science. Springer, Berlin, pp 267–282
53. Viswanathan M, Viswanathan R (2001) Foundations for circular compositional reasoning. In: Orejas F, Spirakis P, van Leeuwen J (eds) Automata, languages and programming, vol 2076. Lecture Notes in Computer Science. Springer, Berlin, pp 835–847
54. Tsay Y-K (2000) Compositional verification in linear-time temporal logic. In: Tiuryn J (ed) Foundations of software science and computation structures, vol 1784. Lecture Notes in Computer Science. Springer, Berlin, pp 344–358
55. Amla N et al (2003) Abstract patterns of compositional reasoning. In: Amadio R, Lugiez D (eds) CONCUR 2003–concurrency theory, vol 2761. Lecture Notes in Computer Science. Springer, Berlin, pp 431–445
56. Chilton C, Jonsson B, Kwiatkowska M (2014) An algebraic theory of interface automata. Theor Comput Sci 549:146–174
57. Tripakis S et al (2011) A theory of synchronous relational interfaces. ACM Trans Program Lang Syst 33(4):14:1–14:41
58. Alur R, Henzinger T (1999) Reactive modules. Form Methods Syst Des 15(1):7–48
59. Grumberg O, Long DE (1994) Model checking and modular verification. ACM Trans Program Lang Syst 16(3):843–871
60. Davis M (1961) Infinite games with perfect information. University of California, Berkeley
61. Dill DL (1988) Trace theory for automatic hierarchical verification of speed-independent circuits. In: Proceedings of the fifth MIT conference on Advanced research in VLSI. MIT Press, Cambridge, pp 51–65
62. Wolf ES (1996) Hierarchical models of synchronous circuits for formal verification and substitution. PhD thesis, Stanford University, Stanford, CA, USA UMI Order No. GAX96-12052
63. Brookes SD, Hoare CAR, Roscoe AW (1984) A theory of communicating sequential processes. J ACM 31(3):560–599
64. Westman J, Nyberg M (2014) Environment-centric contracts for design of cyber-physical systems. In: Dingel J et al (eds) Model-driven engineering languages and systems. Volume 8767 Lecture Notes in Computer Science, vol 8767. Springer, Berlin, pp 218–234
65. Simko G et al (2014) Towards a theory for cyber-physical systems modeling. In: Proceedings of the 4th ACM SIGBED international workshop on design, modeling, and evaluation of cyber-physical systems. CyPhy '14. ACM, New York, pp 56–61
66. Lamport L (1989) A simple approach to specifying concurrent systems. Commun ACM 32(1):32–45
67. Abadi M, Lamport L (1991) The existence of refinement mappings. Theor Comput Sci 82(2):253–284
68. Josko B, Ma Q, Metzner A (01 2008) Designing embedded systems using heterogeneous rich components. In: Proceedings of the INCOSE international symposium 2008

69. Damm W (June 2005) Controlling speculative design processes using rich component models. In: . Fifth international conference on application of concurrency to system design, 2005. ACSD 2005, pp 118–119
70. Westman J, Nyberg M, Gustavsson J, Gurov D (2017) Formal architecture modeling of sequential non-recursive C programs. Sci Comput Program 146:2–27
71. Gacek A et al (2015) Towards realizability checking of contracts using theories. In: Havelund K, Holzmann G, Joshi (eds) NASA formal methods. Volume 9058 of Lecture Notes in Computer Science. Springer, Berlin, pp 173–187
72. Le TTH, Passerone R (Oct 2014) Refinement-based synthesis of correct contract model decompositions. In: 2014 Twelfth ACM/IEEE international conference on formal methods and models for Codesign (MEMOCODE), pp 134–143
73. de Roever W-P (1998) The need for compositional proof systems: a survey. In: de Roever W-P, Langmaack H, Pnueli A (eds) Compositionality: the significant difference, vol 1536. Lecture Notes in Computer Science. Springer, Berlin, pp 1–22
74. Yu E (Jan 1997) Towards modelling and reasoning support for early-phase requirements engineering. In: Proceedings of the third IEEE international symposium on requirements engineering, 1997, pp 226–235
75. van Lamsweerde A, Letier E (2004) From object orientation to goal orientation: a paradigm shift for requirements engineering. In: Wirsing M, Knapp A, Balsamo S (eds) Radical innovations of software and systems engineering in the future, vol 2941. Lecture Notes in Computer Science. Springer, Berlin, pp 325–340
76. Lapouchnian A (205) Goal-oriented requirements engineering: an overview of the current research. Technical report, University of Toronto
77. Jackson M (1995) The world and the machine. In: Proceedings of the 17th international conference on software engineering. ICSE ’95. ACM, New York, pp 283–292
78. Jackson M (1995) Software requirements and specifications: a lexicon of practice principles and prejudices. ACM Press/Addison-Wesley Publishing Co., New York
79. Parnas DL (1995) Functional documents for computer systems. Sci Comput Program 25:41–61
80. Nyberg M (Oct 2013) Failure propagation modeling for safety analysis using causal Bayesian networks. In: 2013 Conference on control and fault-tolerant systems (SysTol), pp 91–97
81. Nyberg M, Westman J (Sept 2015) failure propagation modeling based on contracts theory. In: Dependable computing conference (EDCC), 2015 Eleventh European, pp 108–119
82. Namjoshi KS, Trefler RJ (2010) On the completeness of compositional reasoning methods. ACM Trans Comput Logic 11(3):1–22
83. Maier P (2003) compositional circular assume-guarantee rules cannot be sound and complete. In: Gordon A (ed) Foundations of software science and computation structures, vol 2620. Lecture Notes in Computer Science. Springer, Berlin, pp 343–357
84. Doyen L et al (2008) Interface theories with component reuse. In: Proceedings of the 8th ACM international conference on embedded software. EMSOFT ’08. ACM, New York, pp 79–88
85. Giese H (2000) Contract-based component system design. In: Thirty-third annual Hawaii international conference on system sciences (HICSS-33). IEEE Press, Maui
86. Sun X (May 2011) Compositional design of analog systems using contracts. PhD thesis, EECS Department, University of California, Berkeley
87. Bliudze S, Sifakis J (2007) The algebra of connectors: structuring interaction in BIP. In: Proceedings of the 7th ACM and IEEE international conference on embedded software. EMSOFT ’07. ACM, New York, pp 11–20
88. Back R-JJ, Akademi A, Wright JV (1998) Refinement calculus: a systematic introduction, 1st edn. Springer, New York
89. Graf S, Quinton S (2007) Contracts for BIP: hierarchical interaction models for compositional verification. In: Proceedings of the 27th IFIP WG 6.1 international conference on formal techniques for networked and distributed systems. FORTE ’07. Springer, Berlin, pp 1–18
90. Cimatti A, Dorigatti M, Tonetta S (Nov 2013) OCRA: a tool for checking the refinement of temporal contracts. In: 2013 IEEE/ACM 28th international conference on automated software engineering (ASE), pp 702–705
91. Derler P et al (2013) Cyber-physical system design contracts. In: ICCPS ’13: ACM/IEEE 4th international conference on cyber-physical systems
92. Törngren M et al (Aug 2012) Design contracts for cyber-physical systems: making timing assumptions explicit. Technical Report UCB/EECS-2012-191, EECS Department, University of California, Berkeley
93. Ruchkin I et al. (2014) Contract-based integration of cyber-physical analyses. In: Proceedings of the 14th international conference on embedded software. EMSOFT ’14. ACM, New York, pp 23:1–23:10

94. Bate I, Hawkins R, McDermid J (2003) A contract-based approach to designing safe systems. In: Proceedings of the 8th Australian workshop on safety critical system and software, vol 33. SCS ’03, Australian Computer Society, Inc., pp 25–36
95. Arts T, Dorigatti M, Tonetta S (2014) Making implicit safety requirements explicit. In: Bondavalli A, Di Giandomenico F (eds) Computer safety, reliability, and security. Volume 8666 of Lecture Notes in Computer Science. Springer, Berlin, pp 81–92
96. Soderberg A, Johansson R (Nov 2013) Safety contract based design of software components. In: 2013 IEEE international symposium on software reliability engineering workshops (ISSREW), pp 365–370
97. Soderberg A, Vedder B (Nov 2012) Composable safety-critical systems based on pre-certified software components. In: 2012 IEEE 23rd international symposium on software reliability engineering workshops (ISSREW), pp 343–348
98. Larsen KG (1990) Modal specifications. In: Sifakis J (ed) automatic verification methods for finite state systems, vol 407. Lecture Notes in Computer Science. Springer, Berlin, pp 232–246
99. David A et al (2010) Timed I/O automata: a complete specification theory for real-time systems. In: Proceedings of the 13th ACM international conference on hybrid systems: computation and control. HSCC ’10. ACM, New York, pp 91–100
100. Raclet J-B et al (2011) A modal interface theory for component-based design. Fundam Inf 108(1–2):119–149
101. de Alfaro L, Henzinger TA (2001) Interface automata. SIGSOFT Softw Eng Notes 26(5):109–120
102. Lynch NA, Tuttle MR (1989) An introduction to input/output automata. CWI Q 2:219–246
103. Chilton C, Jonsson B, Kwiatkowska M (2014) Compositional assumeguarantee reasoning for input/output component theories. Sci Comput Program 91(Part A):115–137. Special Issue on Formal Aspects of Component Software (Selected Papers from FACS12)
104. Jones CB (September 1983) Specification and design of (parallel) programs. In: Mason REA (ed) Information processing 83. Volume 9 of IFIP Congress Series. Paris, France, IFIP, North-Holland, pp 321–332
105. Jones CB (1983) Tentative steps toward a development method for interfering programs. ACM Trans Program Lang Syst 5(4):596–619
106. Floyd RW (1967) Assigning meanings to programs. In: Schwartz JT (ed) Mathematical aspects of computer science. Volume 19 of proceedings of symposia in applied mathematics. American Mathematical Society, Providence, pp 19–32
107. Dijkstra EW (1975) Guarded commands, nondeterminacy and formal derivation of programs. Commun ACM 18(8):453–457
108. Owicki S, Gries D (1976) An axiomatic proof technique for parallel programs I. Acta Inf 6(4):319–340
109. Ashcroft EA (1975) Proving assertions about parallel programs. J Comput Syst Sci 10(1):110–135
110. Lamport L (1977) Proving the correctness of multiprocess programs. IEEE Trans Softw Eng 3(2):125–143
111. Pnueli A (1985) In transition from global to modular temporal reasoning about programs. In: Apt K (ed) Logics and models of concurrent systems, vol 13. NATO ASI Series. Springer, Berlin, pp 123–144
112. Alur R et al (1998) MOCHA: modularity in model checking. In: Hu A, Vardi M (eds) Computer aided verification, vol 1427. Lecture Notes in Computer Science. Springer, Berlin, pp 521–525
113. Gurov D, Huisman M, Sprenger C (2008) Compositional verification of sequential programs with procedures. Inf Comput 206(7):840–868
114. Kupferman O, Vardi MY (2000) An automata-theoretic approach to modular model checking. ACM Trans Program Lang Syst 22(1):87–128
115. Păsăreanu CS et al (2008) Learning to divide and conquer: applying the L* algorithm to automate assume-guarantee reasoning. Form Methods Syst Des 32(3):175–205
116. Cobleigh JM, Avrunin GS, Clarke LA (2008) Breaking up is hard to do: an evaluation of automated assume-guarantee reasoning. ACM Trans Softw Eng Methodol 17(2):7:1–7:52
117. Jonsson B, Yih-Kuen T (1996) Assumption/guarantee specifications in linear-time temporal logic. Theor Comput Sci 167(12):47–72
118. Broy M (1997) Compositional refinement of interactive systems. J ACM 44(6):850–891
119. Broy M, Stølen K (2001) Specification and development of interactive systems: focus on streams, interfaces, and refinement. Springer, New York
120. Hammond J, Rawlings R, Hall A (2001) Will it work? [Requirements engineering]. In: Proceedings. Fifth IEEE international symposium on requirements engineering, 2001, pp 102–109
121. Cofer D et al (2012) Compositional verification of architectural models. In: Proceedings of the 4th international conference on NASA formal methods. NFM’12. Springer, Berlin, pp 126–140

122. Persson M et al (2013) A characterization of integrated multi-view modeling in the context of embedded and cyber-physical systems. In: Proceedings of the eleventh ACM international conference on embedded software. EMSOFT '13. IEEE Press, Piscataway, pp 10:1–10:10
123. Törngren M et al (2014) Integrating viewpoints in the development of mechatronic products. Mechatronics 24(7):745–762 1. Model-Based Mechatronic System Design 2. Model Based Engineering
124. Lee E, Sangiovanni-Vincentelli A (1998) A framework for comparing models of computation. IEEE Trans Comput Aided Des Integr Circuits Syst 17(12):1217–1229
125. de Alfaro L, Henzinger TA (2001) Interface theories for component-based design. In: Henzinger TA, Kirsch CM (eds) Embedded software, vol 2211. Lecture Notes in Computer Science. Springer, Berlin, pp 148–165
126. Passerone R (2004) Semantic foundations for heterogeneous systems. PhD thesis, University of California, Berkeley AAI3146975
127. Burch J, Passerone R, Sangiovanni-Vincentelli A (2001) Overcoming heterophobia: modeling concurrency in heterogeneous systems. In: Proceedings. 2001 international conference on application of concurrency to system design, 2001, pp 13–32
128. Engelfriet J (1985) Determinancy (observation equivalence = trace equivalence). Theor Comput Scie 36:21–25
129. Milner R (1982) A calculus of communicating systems. Springer, New York
130. Alur R et al (1998) Alternating refinement relations. In: In Proceedings of the ninth international conference on concurrency theory (CONCUR98), volume 1466 of LNCS. Springer, Berlin, pp 163–178