Verifying Contract-Based Specifications of Product Lines using Description Logic

Damir Nešić[1] and Mattias Nyberg[1,2]

[1] Royal Institute of Technology, Stockholm, Sweden
[2] Scania CV AB, Södertalje, Sweden
{damirn,matny}@kth.se

Abstract. The complexity of critical systems is constantly increasing and if developed as *Product Lines* (PLs), the number of possible system configuration can be huge. Consequently, assuring system properties such as safety or security is increasingly difficult. *Assurance cases* are used often to argue that a system is safe or secure and *Contract-Based Specification* models are a promising foundation for assurance case argumentation. This paper defines a method for *Description Logic* (DL) based verification of the well-formedness constraints of an arbitrary CBS model of a PL. The paper presents the DL encoding of arbitrary CBS model, the DL encoding of the well-formedness constraints, and shows how the verification of these constraints can be reduced to satisfiability verification of the corresponding *knowledge base*. In order to validate the presented approach, a small, but real, industrial PL was expressed as a CBS model, implemented as an OWL ontology, and an off-the-shelf reasoner was used to verify if the CBS model is well-formed.

1 Introduction

The complexity and heterogeneity of critical systems from domains such as automotive or aerospace is constantly increasing [15]. Furthermore, in order to increase the portfolio of products, critical systems are designed by following the *Product Line Engineering* (PLE) paradigm [2], which facilitates the development of a *Product Line* (PL) of similar systems such that a high number of distinct system configurations is possible. As a consequence, assuring that the complete PL satisfies properties like safety or security is increasingly difficult and in the case of a very high number of configurations even not feasible [16, 24, 38].

Development of critical systems is regulated by various standards [19, 20], and often the proof of compliance is embodied in a form of an *assurance case*. An assurance case is a *"a reasoned and compelling argument, supported by a body of evidence, that a system [...] will operate as intended for a defined application in a defined environment"* [1]. Developing the assurance case argumentation-structure and identifying the appropriate evidences are both difficult tasks. The required *"body of evidence"* are concrete engineering artifacts, such as results of safety analyses, specifications, test cases etc. These are usually maintained and evolved by different tools, which in an enterprise setting often leads to inconsistencies among these artifacts. As argued in [5, 27], *Semantic Web* [32],

and *Linked Data* [9] are promising approaches for *cleaning* and *integrating* such heterogeneous data with the purpose of creating a reliable source of knowledge, e.g. a source of assurance case evidences.

Difficulties regarding the assurance case argumentation-structure stem from the fact that arguing that *a system is safe* or *secure* can only be done based on the methods used to engineer the system. Typically, a variety of methods with different semantics and levels of formality are used to specify, design, implement, and verify a PL of systems. Consequently, constructing a *"reasoned and compelling"* argumentation-structure is challenging and most often manual. One of the few approaches that can serve as a unifying framework for specification, design, and input for verification of complex, heterogeneous PLs is the *Contract-Based Specification* (CBS) framework [8, 30, 37] and its PLE extension [26]. As briefly discussed in [8], and as shown in [34] with basic CBS concepts, a CBS model of a system can serve as the foundation for assurance case argumentation.

In order to use a CBS model for this purpose, the model must conform to various constraints which ensure sound and complete argumentation. Although these constraints can be verified using various technologies, the present paper introduces an approach based on *Description Logic* (DL) [4]. Because DL is the foundation of the *Web Ontology Language* (OWL) [23], which is a part of the *Semantic Web* stack, the aim of the paper is to enable the verification of CBS models, and indirectly the assurance case arguments, with technologies that are tightly coupled with technologies that can be used for assurance-case evidence access. Consequently, this should reduce the effort of automating the prevailingly manual process of assurance case construction. The results show that using DL for the verification of CBS models is possible, and also that formalizing the semantics of PLE and CBS concepts in DL is intuitive. However, the results also show that the *open-world assumption* and the lack of support for meta-modeling, incur overhead in the encoding of CBS models as a DL *Knowledge Base* (KB).

Literature on using DL and DL-based reasoners for analyzing contract-based models is rare. Somewhat related work is [22, 39] where the issue of automatically brokering contracts between *web services* is treated. However the notion of a contract and its use is different and less formal compared to CBS frameworks. Most notable non DL-based work regarding analysis of contract-based models is the MICA tool [11]. While the present paper focuses on verifying CBS models of PLs independent of the formalism for expressing contracts, the MICA tool focuses on fine-grained analyses of CBS models expressed using *modal interface theory*. Interestingly, OWL has been extensively used in PLE as a tool for the analysis of the so-called *variability models*, which represent the commonality and variability of a PL. Work in [31] encodes a set of Simulink [13] objects that represent PL variations, as a DL KB and uses an off-the-shelf reasoners to analyze the possible product configurations for various defects [7]. Approaches in [14, 28, 36] provide the encoding of a particular type of a variability model, the so-called *feature model* (FM) [6], as a DL KB and also use off-the-shelf reasoners to detect similar product configuration defects. The approach in [10] performs a similar task but instead of an FM, the considered variability model is an OVM model [29]. The present paper also encodes an arbitrary FM as a DL KB but because the purpose

of the FM encoding is to support the analysis of CBS models, the DL semantics of FM constructs is different and the encoding is more compact.

Paper Structure. Section 2 summarizes the concepts underlying PLE, CBS and the constraints to be verified against an arbitrary CBS model. Section 3 defines the semantics of presence conditions in terms of CBS concepts. Section 4 presents the encoding of the PLE and CBS concepts as a DL KB and it is followed by Section 5 which exemplifies how an arbitrary CBS model can be verified against the defined constraints. Finally, Section 6 concludes the paper.

2 Background

The idea behind PLE is to declare features [2] which represent characteristics of each product in a PL, and express them, and their mutual dependencies, in a variability model which is most often a *Feature Model* (FM). This representation of a PL is a part of the so-called *problem space* [3]. Given an FM, development artifacts are labeled with feature-based formulas referred to as *presence conditions*. The representation of a PL in terms of artifacts labeled with presence conditions is referred to as the *solution space* [3]. By selecting features from an FM, a particular *product configuration* is selected, and the set of artifacts that implement the selected product configuration are those whose presence conditions evaluate to true for the given feature selection. The remainder of this section presents FMs as the solution space and CBS models, as the problem space representation of a PL. From here on, a feature is considered to be a Boolean variable.

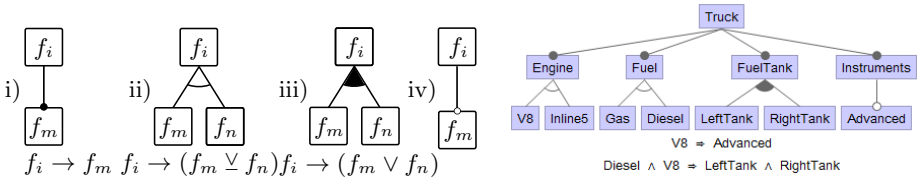
2.1 Feature models and presence conditions

Definition 1 (Feature Model). *A* feature model $\mathcal{V}$ *is a pair* $\mathcal{V} = (\mathcal{F}, \mathfrak{C})$ *where* $\mathcal{F} = \{f_1, \ldots, f_n\}$ *is a set of features and* $\mathfrak{C}$ *is a set of Boolean constraints over the features in* $\mathcal{F}$*.*

FMs define graphical syntax for a set of Boolean constraints in order to structure the features into a tree, where each *parent-*f_i to *child-*f_m relation corresponds to the expression $f_m \rightarrow f_i$. Figure 1a shows the FM graphical syntax used in the present paper, together with corresponding propositional expressions [12]. Figure 1b shows the FM running example, created using the FeatureIDE tool [21]. The syntax in Figure 1a is read as: i) f_m is a *mandatory* feature, ii) f_m and f_n are a *group* of *alternative* features where $\veebar$ stands for *exclusive or*, iii) f_m and f_n are a group of *or* features, and iv) f_m is an *optional* feature. Note that Figure 1b contains additional Boolean constraints, where the constraint such as *V8* $\rightarrow$ *Advanced* is read as *requires*.

Definition 2 (Product Configuration). *Given a feature model* $\mathcal{V} = (\mathcal{F}, \mathfrak{C})$*, a* product configuration*, denoted as* $\mathcal{P}_C$*, is a set of feature* value-assignments *such that each* $f_i \in \mathcal{F}$ *is assigned with a value true or false. A product configuration for which each Boolean constraint in* $\mathfrak{C}$ *evaluates to true is* valid*.*

A product configuration represents, in terms of features, all individual systems that are identically configured. As discussed previously, the feature representation is mapped to the representation in terms of concrete artifacts by labeling the artifacts with *presence conditions*.



(a) Used FM graphical syntax. (b) FM running example.

Fig. 1: Used FM constructs and the FM running-example.

Definition 3 (Presence Condition). *A* presence condition φ *is a Boolean expression over features in* $\mathcal{F}$*, defined as* $\varphi ::= f_i | \neg(\varphi) | (\varphi \wedge \varphi) | (\varphi \vee \varphi)$.

2.2 Contract-Based Specification and Design

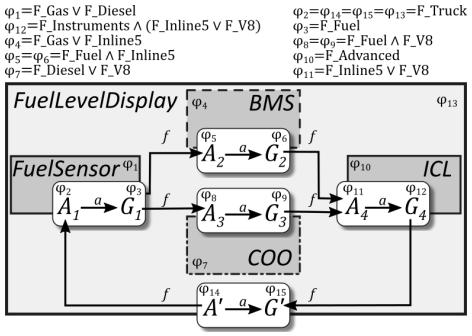
The basic concepts of CBS are shown in Figure 2a. Given these, the central concept of a Contract is defined as following.

Definition 4 (Contract). *A Contract* K *is an ordered pair of specifications, denoted* (A, G)*, where* A *is called an Assumption,* G *is called a Guarantee, and* A *and* G *are in the* assumption of *relation, denoted* assumptionOf(A, G).

A component C satisfies a contract $K = (A, G)$ if for each component C' such that implements(C', A), the component C'' is such that implements(C'', G) where composedOf(C'', C') and composedOf(C'', C). Components composed of other components are referred to as Composite while components not composed of other components are referred to as Atomic. The expectation that a component C shall satisfy the contract K is indicated by *allocating* a contract to a component, denoted as allocatedTo(K, C). Finally, in order to allow PLs, function Φ labels each Specification and each Component with a presence condition φ. Figure 2b shows the running example which is the *Fuel Level Display system* (FLD), a real system present in product of Scania CV AB.

Basic CBS concepts
Component C - Physical or logical component
Specification S - Requirement on C behaviour
implements(C, S) - C behaviour conforms to S
fulfills(S_i, S_j) - $\forall C$.implements$(C, S_i) \rightarrow$ implements(C, S_j)
composedOf(C, C_1) - C_1 comprises C
[a] Detailed descriptions can be found in [26].

(a) Basic CBS concepts



(b) FLD system running example.

Fig. 2: Basic CBS concept (left) and the running CBS model example (right)

The composite component FLD is composed of atomic components FuelSensor, COO, short for *COOrdinator*, BMS, short for *Body Management System*, and ICL,

short for *Instrumentation CLuster*. Relations fulfills are shown as arrows labeled *f*, relations assumptionOf are shown as arrows labeled *a*, and white rectangles represent contracts. Overlaying a contract over a component represents the allocatedTo relation. The intended functionality of the FLD system is to display the current *fuel level* on the ICL display and guarantee that the displayed value corresponds to the *actual* fuel level. Guarantee G' captures this specification under the assumption A' that the ignition key is turned on.

The guarantees of contracts allocated to FuelSensor, COO, BMS and ICL specify that the *fuel level* measurement in volts, its conversion to liters, and its display on a $[0 - max]$ scale, correspond to the *actual* fuel level, while assuming that the power-supply is nominal, that the information on the communication buss is updated regularly etc. Consequently, if G_3 is implemented, i.e. the displayed value corresponds to the actual one, then G' is implemented, i.e. G_3 fulfills G'. The dashed border of BMS and COO indicates that these components are not present in each product configuration, i.e. COO is responsible for volts to liters conversions in product configurations with *diesel* fuel, while BMS, performs the same conversion in product configurations with *gas* fuel. In other words, the CBS model in Figure 2b defines the solution space of a PL of FLD systems.

If a CBS model of a PL conforms to certain constraints, then it is referred to as a *specification structure*.

Definition 5 (Specification Structure). *Given a set of contracts $K_i = (A_i, G_i)$, where each A_i and G_i is unique, such that each K_i is allocated to a single atomic component C_i, a contract $K' = (A', G')$ allocated to the composite component C', which is composed of atomic components C_i, an edge-labeled directed graph $\mathfrak{D} = (\mathcal{N}, \mathcal{E})$ is a* specification structure *if:*

i) each node $n \in \mathcal{N}$ is either an assumption A_i or A' or a guarantee G_i or G',
ii) each edge $e = (n_i, n_j) \in \mathcal{E}$ corresponds to either a single assumptionOf(S_i, S_j) *or a single* fulfills(S_i, S_j) *relation,*
iii) for each edge $e = (n_i, n_j) \in \mathcal{E}$ it holds that $n_i \neq n_j$,
iv) for each assumptionOf(S_i, S_j) *relation, S_i is an assumption, S_j is a guarantee and the ordered pair (S_i, S_j) is a contract,*
v) for each edge fulfills(S_i, S_j)*, if*
  a) S_i is a guarantee then S_i is a guarantee of a contract K_i and S_j is either an assumption A_j or the guarantee G',
  b) S_i is an assumption then S_i is the assumption A' and S_j is any A_i.

Definition 5 allows circular specification structures, or presence conditions of an assumption A and a guarantee G of a contract (A, G) such that in some product configurations A and G do not simultaneously apply. To avoid such cases, a *proper* specification structure is defined. The entailment between presence conditions φ_i and φ_j is denoted as $\varphi_i \models_{\mathfrak{C}} \varphi_j$ as a short hand for $(\mathfrak{C}, \varphi_i) \models \varphi_j$, i.e. entailment must hold under the constraints from $\mathfrak{C}$. For example, in Figure 2b, it holds that $\varphi_2 \not\models \varphi_1$, but when constraints Truck $\rightarrow$ Fuel and Fuel $\rightarrow$ Gas $\veebar$ Diesel from Figure 1b are considered, then the entailment holds, i.e. $\varphi_2 \models_{\mathfrak{C}} \varphi_1$.

Definition 6 (Proper Specification Structure). *Given a feature model $\mathcal{V} = (\mathcal{F}, \mathfrak{C})$, a specification structure is* proper *if:*

i) *for each assumption A_i of a contract K_i, there exists a specification S_k such that* $\mathsf{fulfills}(S_k, A_i)$*, where S_k is either A' or G_i,*
ii) *there exists a guarantee G_i such that* $\mathsf{fulfills}(G_i, G')$*,*
iii) *the graph of $\mathfrak{D}$ is acyclic,*
iv) *for each contract (A_i, G_i), it holds that $\Phi(A_i) \models_{\mathfrak{C}} \Phi(G_i)$ and $\Phi(G_i) \models_{\mathfrak{C}} \Phi(A_i)$,*
v) *for each contract (A_i, G_i) allocated to a component C_i it holds that $\Phi(A_i) \models_{\mathfrak{C}} \Phi(C_i)$ and $\Phi(G_i) \models_{\mathfrak{C}} \Phi(C_i)$,*
vi) *for the composite component C' composed of atomic components C_i, for each C_i it holds that $\Phi(C_i) \models_{\mathfrak{C}} \Phi(C')$,*
vii) *for each contract $K_i = (A_i, G_i)$ allocated to an atomic component C_i and for each specification S_k such that* $\mathsf{fulfills}(S_k, A_i)$ *it holds that $\Phi(A_i) \models_{\mathfrak{C}} \bigvee_{\mathsf{fulfills}(S_k, A_i)} \Phi(S_k)$,*
viii) *for the contract $K' = (A', G')$ allocated to the composite component C' and for each guarantee G_i of a contract C_i such that* $\mathsf{fulfills}(G_i, G')$ *it holds that $\Phi(G') \models_{\mathfrak{C}} \bigvee_{\mathsf{fulfills}(G_i, G')} \Phi(G_i)$.*

The following theorem presents the key compositional idea of the CBS [26]. Note that in the following theorem, symbol $\models$ means entailment between S_i, S_j irregardless of the formalisms used to express S_i and S_j.

Theorem 1. *Given a feature model $\mathcal{V}$, and a specification structure $\mathfrak{D}$, if*

i) *$\mathfrak{D}$ is proper,*
ii) *for each relation* $\mathsf{fulfills}(S_i, S_j)$*, it holds that $S_i \models S_j$,*
iii) *each atomic component C_i* $\mathsf{satisfies}$ *the contract allocated to it,*

then, for each valid product configuration $\mathcal{P}_{C_i}$, the composite component C' $\mathsf{satisifes}$ *the contract K'.*

Less formally, if a CBS model of a PL is a proper specification structure, and if each pair of specifications in $\mathsf{fulfills}$ relation entail each other, then by verifying that each component of each PL product configuration satisfies the contract allocated to it, it can be inferred that each product configuration satisfies the contract allocated to it. The remainder of the paper shows how to verify that an arbitrary CBS model, is a *proper specification structure*.

3 Semantics of presence conditions

This and the following sections present the contribution of the paper. In order to be able to verify that an arbitrary CBS model is a proper specifications structure, it is necessary to define a common semantic interpretation of the constructs representing the problem space and the solution space.

Because an FM and a CBS model jointly represent the PL design, a potentially infinite number of product individuals p can be created by instantiating the PL design. From this it follows that each product configuration corresponds to a set of products individuals. Furthermore, each p can be represented as a $\mathsf{Component}$, either atomic or composite. For example, each individual Scania product, which is a truck or a bus, is a $\mathsf{Composite}$ component $\mathsf{composedOf}$ an individual of the FLD system, which is also a $\mathsf{Composite}$ component.

Let Config be a function which given a product individual p returns its product configuration $\mathcal{P}_C$. Moreover, let Eval be a function that takes a presence condition φ and a product configuration $\mathcal{P}_C$ and returns the value of φ for the given $\mathcal{P}_C$.

Definition 7 (Presence condition semantics). *The semantics of a presence condition φ, denoted as $[\![\varphi]\!]$, is $[\![\varphi]\!] = \{p|\ \mathsf{Eval}(\varphi, \mathsf{Config}(p)) = true\}$.*

A presence condition φ corresponds to a set of individual products to which the artifact labeled with φ applies. This interpretation of presence conditions corresponds to the concept of *product groups* [25].

4 Encoding an FM and a CBS model as Tbox axioms

This section presents the encoding of an arbitrary FM and an arbitrary CBS model as a set of Tbox axioms. In order to provide a formal encoding, a CBS model is represented as a tuple $\mathcal{M} = (\mathcal{O}, \mathcal{R}, [\![\cdot]\!]^T)$, where $\mathcal{O}$ is a set of objects, $\mathcal{R}$ is a set of pairs representing relations, and $[\![\cdot]\!]^T$ is a function which returns the type of an object or a relation. For instance, a fragment of the example from Figure 2b would be $\{\mathsf{G}_1, \mathsf{A}_2\} \subseteq \mathcal{O}$, $(G_1, A_2) \in \mathcal{R}$, and $G_1^T = \mathsf{Specification}$, $A_1^T = \mathsf{Specification}$, $(G_1, A_1)^T = \mathsf{fulfills}$.

Regarding DL notation, let N_C, N_R and N_I be mutually disjoint sets of *concept names*, *role names*, and *individual names*, respectively. In the remainder of the paper we use the DL notation and the semantics defined in [4].

4.1 General CBS entities and relations as Tbox axioms

Before defining the encoding of a FM or a model $\mathcal{M}$, the Tbox is complemented with concepts and roles that correspond to general CBS entities and relations and these are shown in Table 1. Roles $R_{hasS}, R_{hasA}, R_{hasG}$, correspond to the *containment* relation between a specification structure and specifications, denoted $\mathsf{hasSpecification}(\mathfrak{D}, \mathsf{S_i})$, and the containment relation between contracts and corresponding assumptions and guarantees, denoted $\mathsf{hasAssumption}(K, A)$ and $\mathsf{hasGuarantee}(K, G)$. The role R_{comp}, short for *comprises*, represents the relation inverse to composedOf, and roles $R_{aPartOf}, R_{gPartOf}, R_{fullBy}$ are inverse to roles $R_{hasA}, R_{hasG}, R_{fulls}$, respectively. As will be shown later, expressing several conditions from Definition 5 and Definition 6 will require these roles.

According to the discussion in Section 3, the Tbox is complemented with the concept $D_P \in N_C$ which represents all individual products, and the axiom $D_P \sqsubseteq D_C$ in order to capture the fact that each individual product is a component. Moreover, because specifications are partitioned into assumptions and guarantees, the axiom $D_S \equiv D_{S_A} \sqcup D_{S_G}$ is added to the Tbox. Similarly, because components are partitioned into Atomic and Composite components, the axiom $D_C \equiv D_{C_A} \sqcup D_{C_C}$ is added to the Tbox.

Finally, Definition 4 and condition (iii) from Definition 5 can be encoded as Tbox axioms independently of the provided FM or a model $\mathcal{M}$. The axiom $D_K \sqsubseteq = 1\, R_{hasA}.D_{S_A} \sqcap = 1\, R_{hasG}.D_{S_G}$ corresponds to Definition 4, and axioms $\exists\, R_{assOf}.Self \sqsubseteq \bot$ *and* $\exists\, R_{fulls}.Self \sqsubseteq \bot$ correspond to condition (iii) of Definition 5. The semantics of the construct $\exists R.Self$ is defined in [17].

Table 1: Concepts and roles corresponding to CBS concepts and relations.

(a) Concepts $D_x \in N_C$

CBS entity	concept	Inclusion
Spec. Struct.	D_{SS}	
Specification	D_S	
Assumption	D_{S_A}	$D_{S_A} \sqsubseteq D_S$
Guarantee	D_{S_G}	$D_{S_G} \sqsubseteq D_S$
Component	D_C	
Composite Comp.	D_{C_C}	$D_{C_C} \sqsubseteq D_C$
Atomic Comp.	D_{C_A}	$D_{C_A} \sqsubseteq D_C$
Contract	D_K	
Disjoint: $D_{S_A} \sqcap D_{S_G} \sqsubseteq \bot, D_{C_C} \sqcap D_{C_A} \sqsubseteq \bot$		
Disjoint: D_{SS}, D_S, D_C, D_K are pairwise disjoint		

(b) Roles $R_x \in N_R$

Role	Domain	Range	Inverse
R_{fulls}	d:D_S	r:D_S	
R_{assOf}	d:D_{S_A}	r:D_{S_G}	
R_{alcTo}	d:D_K	r:D_C	
R_{comp}	d:D_{C_A}	r:D_{C_C}	
R_{hasA}	d:D_K	r:D_{S_A}	
R_{hasG}	d:D_K	r:D_{S_G}	
R_{hasS}	d:D_{SS}	r:D_S	
$R_{gPartOf}$	d:D_{S_G}	r:D_K	R_{hasG}
$R_{aPartOf}$	d:D_{S_A}	r:D_K	R_{hasA}
R_{fullBy}	d:D_S	r:D_S	R_{fulls}

[a] Actual axioms available in [26].

4.2 Arbitrary FM as Tbox axioms

In order to verify conditions (iv)-(viii) from Definition 6, it is necessary to consider the FM. Several publications [14, 28, 36] have encoded an FM as a set of Tbox axioms that capture the FM graph in order to verify properties [7] of the possible product configurations. The present paper encodes the FM differently and the semantics of each feature f is a set of product individuals whose product configuration $\mathcal{P}_C$ is such that $(f = \mathit{true}) \in \mathcal{P}_C$. Consequently, the semantics of concepts based on features, e.g. presence conditions or product configurations, is also set of product individuals, just as the intuition suggests. The Tbox encoding of an arbitrary FM $\mathcal{V} = (\mathcal{F}, \mathfrak{C})$ is shown in Table 2.

Table 2: FM as Tbox axioms.

FM construct	Tbox encoding
Feature f_i	$D_{f_i} \in N_C$, $D_{f_i} \sqsubseteq D_P$
Parent f_i - child f_m	$D_{f_m} \sqsubseteq D_{f_i}$
Parent f_i - *mandatory* child feature f_m	$D_{f_i} \sqsubseteq D_{f_m}$
Parent f_i - *optional* child feature f_m	-
Parent f_i - *alternative* feature group $f_1, \ldots, f_n$	$D_{f_i} \equiv \sqcup_{j=1}^{n} D_{f_j}$,$D_{f_j} \sqcap D_{f_{k \neq j}} \sqsubseteq \bot$
Parent f_i - *or* feature group $f_1, \ldots, f_n$	$D_{f_i} \equiv \sqcup_{j=1}^{n} D_{f_j}$
Feature f_i *requires* feature f_j	$D_{f_i} \sqsubseteq D_{f_j}$

Encoding arbitrary Boolean constraints in $\mathfrak{c} \in \mathfrak{C}$ is exemplified on the running example from Figure 1b. Consider the arbitrary constraint $\mathit{Diesel} \wedge \mathit{V8} \rightarrow \mathit{LeftTank} \wedge \mathit{RightTank}$. The corresponding Tbox axiom is $D_{Truck} \sqsubseteq \neg(D_{Diesel} \sqcap D_{V8}) \sqcup (D_{LeftTank} \sqcap D_{RightTank})$, where D_{Truck} is the root of the FM.

4.3 Arbitrary model $\mathcal{M}$ as Tbox axioms

Given an arbitrary model $\mathcal{M} = (\mathcal{O}, \mathcal{R}, [\![\cdot]\!]^T)$, the Tbox encoding consists of the encoding of: objects in $\mathcal{O}$, relations in $\mathcal{R}$, presence conditions of objects in $\mathcal{O}$, the absence of relations because of the underlying *open-world assumption*, and inverse relation described in the beginning of Section 4.1. The encoding is exemplified

by using particular types of objects and relations but the encoding principles are similar for all other types of objects and relations.

a) **Encoding objects.** For each $o_i \in \mathcal{O}$ such that $o_i^T = \mathsf{Guarantee}$, the Tbox is complemented with axioms $D_{o_i} \in N_C$, $D_{o_i} \sqsubseteq D_{S_G}$, $D_{S_G} \equiv \sqcup_i D_{o_i}$, and for each D_{o_i} and $D_{o_{j\neq i}}$, $D_{o_i} \sqcap D_{o_j} \sqsubseteq \bot$.
b) **Encoding relations.** For each $o_i \in \mathcal{O}$ such that $o_i^T = \mathsf{Guarantee}$ and each pair $(o_i, o_1), \ldots, (o_i, o_n) \in \mathcal{R}$ where $(o_i, o_1)^T = \cdots = (o_i, o_n)^T = \mathsf{fulfills}$, the Tbox is complemented with axioms $D_{o_i} \sqsubseteq \sqcap_{j=1}^{n} = 1\, R_{fulls}.D_{o_j}$, and $D_{o_i} \sqsubseteq \forall R_{fulls}.(\sqcup_{j=1}^{n} D_{o_j})$.
c) **Encoding presence conditions.** For each $o_i \in \mathcal{O}$ labeled with a presence condition $\Phi(o_i)$, the Tbox is complemented with axioms $D_{\Phi(o_i)} \in N_C$,$D_{\Phi(o_i)} \sqsubseteq D_P$, and $D_{\Phi(o_i)} \equiv \Phi_{DL}(o_i)$ where $\Phi_{DL}(o_i)$ is the DL encoding of the presence condition according to the same principles as for arbitrary FM constraints.
d) **Encoding the absence of relations.** For each $o_i \in \mathcal{O}$ such that $o_i^T = \mathsf{Guarantee}$ and $D_{o_i} \sqsubseteq D_S$, where D_S is the domain restriction of role R_{fulls}, if there is no $o_j \in \mathcal{O}$ such that $(o_i, o_j) \in \mathcal{R}$ and $(o_i, o_j)^T = \mathsf{fulfills}$, the Tbox is complemented with the axiom $D_{o_i} \sqsubseteq \neg(\exists\, R_{fulls}.D_S)$ where D_S is the range restriction of the role R_{fulls}.
e) **Encoding inverse relations.** Because each pair $(o_i, o_j) \in \mathcal{R}$ implicitly defines its inverse pair (o_j, o_i), the relations inverse to $\mathsf{fulfills}$, $\mathsf{hasAssumption}$, $\mathsf{hasGuarantee}$ can be constructed, and the encoding principle (b) can be used to encode the relations represented by roles R_{fullBy}, $R_{aPartOf}$, and $R_{gPartOf}$.

Using the running example, we exemplify the usage of the encoding principles. Object $\mathsf{FuelSensor}$ from Figure 2b is encoded as $D_{\mathsf{FuelSensor}} \sqsubseteq D_{C_A}$ and declared to be pairwise disjoint with each of the three remaining concepts $D_{\mathsf{BMS}}, D_{\mathsf{COO}}$, and D_{ICL}. As an example of relation encoding, consider the pair $(A_1, G_1)^T = \mathsf{assumptionOf}$ whose encoding is $D_{A_1} \sqsubseteq= 1\, R_{assOf}.D_{G_1}$ and $D_{A_1} \sqsubseteq \forall R_{assOf}.D_{G_1}$. Note that the reason for encoding relations using both *value restrictions* and number restrictions, is because *value restriction* does not guarantee the existence of the relation. As an example of a presence condition encoding, consider $\Phi(A_2) = \varphi_5 = \mathsf{Truck} \wedge \mathsf{V8}$ which is encoded as $D_{\Phi(A_2)} \equiv D_{Truck} \sqcap D_{V8}$. Finally, for an example of the absence of a relation, consider assumption A_1. Because each $\mathsf{Assumption}$ is a $\mathsf{Specification}$, and D_S which corresponds to $\mathsf{Specification}$ is the domain restriction of the role R_{fulls}, because there is no $\mathsf{Specification}$ S_k such that $(A_1, S_k)^T = \mathsf{fulfills}$, then the Tbox is complemented with the axiom $D_{A_1} \sqsubseteq \neg(\exists\, R_{fulls}.D_S)$ where D_S is the range restriction of R_{fulls}.

The presence condition semantics defined in Definition 7 is preserved by the corresponding Tbox encoding due to the following theorem.

Theorem 2. *Given a presence condition φ expressed over features $f_1, \ldots, f_m$, and corresponding concepts D_φ and $D_{f_1}, \ldots, D_{f_m}$, if $D_\varphi \equiv \varphi_{DL}$, where φ_{DL} is obtained by replacing each feature f_i from φ with D_{f_i}, each $\wedge$ with $\sqcap$, each $\vee$ with $\sqcup$, and preserving each $\neg$, then it holds that $[\![\varphi]\!] = D_\varphi^{\mathcal{I}}$.*

The presence conditions semantics in Definition 7 is defined under the the *closed-world assumption*, i.e. only the explicitly created product individuals p can be

elements of $[\![\varphi]\!]$. For example, there are thousands of configurations of the FLD system and only some configurations are instantiated into product individuals p. Consequently, $[\![\varphi]\!]$ can only be analyzed with respect to the created p. Given Theorem 2 and the the *open-world assumption* of DL, reasoning about $D^{\mathcal{I}}_{\varphi}$ is possible without asserting a single *named individual* in the Abox.

5 Verifying that $\mathcal{M}$ is a proper specification structure

This section presents the encoding of the constraints from Definition 5 and Definition 6 as Tbox axioms. Note that it is assumed that constraint iii) from Definition 6 is verified elsewhere.

The constraint from Definition 5 and Definition 6 can be partitioned into two groups; the ones about presence condition, and the unrelated to presence condition. The latter group includes constraints (i), (ii), (iv), (v) from Definition 5 and (i) and (ii) from Definition 6. The former group includes constraints (iv)-(viii) from Definition 6. Here the encoding of a single constraint from both groups is presented while the remaining ones can be found in [26]. An example of a constraint unrelated to presence conditions is constraint (v)-a) from Definition 5.

Definition 5(v)-a: *For each edge* $\mathsf{fulfills}(S_i, S_j)$ *if* S_i *is a guarantee then* S_i *is a guarantee of a contract* K_i *and* S_j *is either an assumption* A_j *or the guarantee* G'. The constraint is encoded by a concept $D_{5(v)-a}$ where the expression to the left of $\sqcap$ oprator defines the conditions on S_i and the expression to right defines conditions on S_j.

$$\begin{aligned} D_{5(v)-a} \equiv & (\exists\, R_{gPartOf}.(\exists\, R_{alcTo}.(\exists\, R_{comp}.D_{C'}))) \sqcap \\ & (\forall\, R_{fulls}.((\exists\, R_{aPartOf}.(\exists\, R_{alcTo}.(\exists\, R_{comp}.D_{C'}))) \sqcup D_{G'})) \end{aligned}$$

Given a model $\mathcal{M}$, for each $\{o_i, o_j\} \subseteq \mathcal{O}$ such that $o_i^T = \mathsf{Guarantee}$, and $(o_i, o_j)^T = \mathsf{fulfills}$, it should be verified that the concept corresponding to D_{o_i} is subsumed by $D_{5(v)-a}$, i.e. $\mathcal{K} \models D_{o_i} \sqsubseteq D_{5(v)-a}$. In accordance with [18], verifying that a KB entails the above inclusion is equivalent to verifying that the KB is not satisfiable if the Tbox is complemented with a fresh concept $D_{Fresh} \equiv D_{o_i} \sqcap \neg(D_{5(v)-a})$.

An example of a constraint related to presence condition is constraint (vii) from Definition 6.

Definition 6(vii): *For each contract* $K_i = (A_i, G_i)$ *allocated to an atomic component* C_i *and for each specification* S_k *such that* $\mathsf{fulfills}(S_k, A_i)$ *it holds that* $\Phi(A_i) \models_{\mathfrak{C}} \bigvee_{\mathsf{fulfills}(S_k, A_i)} \Phi(S_k)$. Given a model $\mathcal{M}$, for each set of pairs $\{(o_1, o_i), \cdots, (o_n, o_i)\} \subseteq \mathcal{R}$ such that $o_i^T = \mathsf{Assumption}$, $(o_1, o_i)^T = \ldots = (o_n, o_i)^T = \mathsf{fulfills}$, and $\Phi(o_i), \Phi(o_1), \ldots, \Phi(o_n)$ are the corresponding presence conditions, it should be verified that $\mathcal{K} \models D_{\Phi(o_i)} \sqsubseteq \sqcup_{j=1}^{n} D_{\Phi(o_j)}$. Note that because an FM $\mathcal{V} = (\mathcal{F}, \mathfrak{C})$ is already encoded as a set of Tbox axioms, verification that the above inclusion holds is with respect to the constraints in $\mathfrak{C}$. Verifying the entailment of the above inclusion is done in the same way as for $D_{5(v)-a}$.

Similarly to the constraints above, for an arbitrary model $\mathcal{M}$, each condition from Definition 5 and Definition 6 can be formulated as an inclusion axiom $D_i \sqsubseteq D_j$ and verified by establishing that $\mathcal{K} \models D_i \sqsubseteq D_j$. Let $\mathfrak{R}$ be a set of all inclusion axioms required to establish that $\mathcal{M}$ is a proper specification structure.

Theorem 3. *Given a model $\mathcal{M}$, a corresponding knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, and the set $\mathfrak{R}$, if $\forall \mathfrak{r} \in \mathfrak{R}.\mathcal{K} \models \mathfrak{r}$, then $\mathcal{M}$ is a proper specification structure.*

5.1 Implementation

In order to validate the presented encoding, the running example was manually implemented as an OWL ontology using the Protege [35] tool and HermiT [33] reasoner was used to perform the reasoning. The resulting ontology was $\mathcal{SRIQ}$ DL and it contained 463 axioms with 45 additional axioms needed to verify if the running example is a proper specification structure. As can be verified from Figure 2b, the analysis showed that $\Phi(\mathsf{A_4})$ and $\Phi(\mathsf{G_4})$ do not entail $\Phi(\mathsf{ICL})$, thus violating constraint (v) of Def. 6, i.e. the running example is not a proper specification structure. The resulting ontology is available online[3].

6 Conclusion

Compliance with standards regulating the development of critical systems is often shown using *assurance cases*. Assurance cases are comprised of two main parts, the *arguments* about the intended system properties and the *evidences* which support the arguments. As discussed in Section 1, *Contract-Based Specification* (CBS) frameworks have shown to be useful as the basis of assurance case arguments but only if they conform to a number of *constraints*. The present paper has presented a DL-based method for the verification of these constraints against an arbitrary CBS model of a *product line* of systems. Although other technologies, besides DL, can be used to perform such verifications, DL was used because DL is the foundation of OWL which is part of the Semantic Web stack and Semantic Web is a promising approach for *cleaning*, *integrating*, and enabling *uniform access* to the evidences required for an assurance case. Because some constraints require verification of *propositional entailment* it was not possible to encode a CBS model as a set of Abox assertions, and the constraints to which a CBS model must conform to as a set of Tbox axioms. Instead, both the CBS model and the CBS constraints were encoded as Tbox axioms and a new DL concept was added for each CBS model-element that was in the scope of a constraint. In other words, the number of additional DL concepts needed for the verification of a CBS model is linear in the size of the CBS model. A drawback of using DL and the underlying *open-world assumption* was the need to explicitly declare negative facts about the given CBS model, thus incurring an overhead compared to technologies working under the *closed-world assumption*. Still, each of the several types of verifications was possible to encode as the Tbox satisfiability problem and routinely checked by an off-the-shelf reasoner. Furthermore, the presented DL semantics of PL concepts *feature*, *product configuration*, and *presence condition* naturally captures the intuition that each of these concepts represents subsets of the set of all possible product individuals. Future work includes applying the presented method on a CBS model of a full-scale FLD system PL.

[3] https://goo.gl/8G5Wec

References

1. Goal structuring notation community standard version 2 (Jan 2018), https://scsc.uk/r141B:1?t=1
2. Apel, S., Batory, D., Kästner, C., Saake, G.: Feature-oriented software product lines. Springer-Verlag, Berlin-Heidelberg, Germany (2016)
3. Apel, S., Kästner, C.: An overview of feature-oriented software development (2009)
4. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
5. Bansal, S.K.: Towards a semantic extract-transform-load (etl) framework for big data integration. In: 2014 IEEE International Congress on Big Data. pp. 522–529 (2014)
6. Batory, D.: Feature models, grammars, and propositional formulas. In: International Conference on Software Product Lines. pp. 7–20 (2005)
7. Benavides, D., Segura, S., Ruiz Cortés, A.: Automated analysis of feature models 20 years later: A literature review. Information Systems **35**(6) (2010)
8. Benveniste, A., Caillaud, B., Nickovic, D., Passerone, R., Raclet, J.B., Reinkemeier, P., Sangiovanni-Vincentelli, A., Damm, W., Henzinger, T.A., Larsen, K.G.: Contracts for system design. Foundations and Trends® in Electronic Design Automation (2-3), 124–400 (2018)
9. Bizer, C., Heath, T., Berners-Lee, T.: Linked data: The story so far. In: Semantic services, interoperability and web applications: emerging concepts, pp. 205–227. IGI Global (2011)
10. Braun, G., Pol'la, M., Buccella, A., Cecchi, L., Fillottrani, P., Cechich, A.: A dl semantics for reasoning over ovm-based variability models. In: Description Logic workshop (2017)
11. Caillaud, B.: A modal interface compositional analysis library (Oct 2011), www.irisa.fr/s4/tools/mica
12. Czarnecki, K., Wasowski, A.: Feature diagrams and logics: There and back again. In: Proceedings of the International Software Product Line Conference. pp. 23–34. SPLC '07 (2007)
13. Dabney, J.B., Harman, T.L.: Mastering simulink. Pearson (2004)
14. Fan, S., Zhang, N.: Feature model based on description logics. In: Gabrys, B., Howlett, R.J., Jain, L.C. (eds.) Knowledge-Based Intelligent Information and Engineering Systems (2006)
15. Gunes, V., Peter, S., Givargis, T., Vahid, F.: A survey on concepts, applications, and challenges in cyber-physical systems. KSII Transactions on Internet & Information Systems **8**(12) (2014)
16. Halin, A., Nuttinck, A., Acher, M., Devroey, X., Perrouin, G., Baudry, B.: Test them all, is it worth it? Assessing configuration sampling on the JHipster Web development stack. Empirical Software Engineering (Jul 2018)
17. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible sroiq. In: Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning '06. pp. 57–67 (2006)
18. Horrocks, I., Patel-Schneider, P.: Reducing owl entailment to description logic satisfiability. Web Semantics: Science, Services and Agents on the World Wide Web **1**(4), 345 – 357 (2004)
19. ISO61508: Functional Safety. standard, The International Electrotechnical Commission, Geneva, CH (2010)

20. ISO26262: Road vehicles - Functional safety. standard, International Organization for Standardization, Geneva, CH (2011)
21. Kastner, C., Thum, T., Saake, G., Feigenspan, J., Leich, T., Wielgorz, F., Apel, S.: Featureide: A tool framework for feature-oriented software development. In: Proceedings of the 31st International Conference on Software Engineering. pp. 611–614 (2009)
22. Liu, H., Li, Q., Gu, N., Liu, A.: Modeling and reasoning about semantic web services contract using description logic. In: 2008 The Ninth International Conference on Web-Age Information Management. pp. 179–186 (July 2008)
23. McGuinness, D.L., Van Harmelen, F., et al.: Owl web ontology language overview. W3C recommendation **10**(10) (2004)
24. Mukelabai, M., Nešić, D., Maro, S., Berger, T., Steghöfer, J.P.: Tackling combinatorial explosion:a study of industrial needs and practices for analyzing highly configurable systems. In: Proceedings of ASE'18 (2018)
25. Nešić, D., Nyberg, M.: Modeling product-line legacy assets using multi-level theory. In: Proceedings of International Systems and Software Product Lines Conference '17 - Volume B. pp. 89–96 (2017)
26. Nešić, D., Nyberg, M.: Contract-based specification and description-logic-based validation of product lines. Technical report, Royal Institute of Technology, Stockholm, Sweden (2018)
27. Nešić, D., Nyberg, M.: Applying multi level modeling to data integration in product line engineering. In: Proceedings MODELS Satelite events: International Workshop on Multi-Level Modeling '17. pp. 235–242 (2017)
28. Noorian, M., Ensan, A., Bagheri, E., Boley, H., Biletskiy, Y.: Feature model debugging based on description logic reasoning. In: DMS (2011)
29. Pohl, K., Böckle, G., van der Linden, F.J.: Software Product Line Engineering. Foundations, Principles, and Techniques. Springer (2005)
30. Quinton, S., Graf, S.: Contract-based verification of hierarchical systems of components. In: Proceedings IEEE International Conference on Software Engineering and Formal Methods '08. pp. 377–381 (2008)
31. Ryssel, U., Ploennigs, J., Kabitzsch, K.: Reasoning of feature models from derived features. SIGPLAN Not. **48**(3), 21–30 (Sep 2012)
32. Shadbolt, N., Berners-Lee, T., Hall, W.: The semantic web revisited. IEEE Intelligent Systems pp. 96–101 (2006)
33. Shearer, R., Motik, B., Horrocks, I.: Hermit: A highly-efficient owl reasoner. In: OWLED. vol. 432, p. 91 (2008)
34. Sljivo, I., Gallina, B., Carlson, J., Hansson, H.: Generation of safety case argument-fragments from safety contracts. In: Proceedings International Conference on Computer Safety, Reliability, and Security '14. pp. 170–185 (2014)
35. Stanford Center for Biomedical Informatics Research: Protege ontology editor (2016), https://protege.stanford.edu/
36. Wang, H.H., Li, Y.F., Sun, J., Zhang, H., Pan, J.: Verifying feature models using owl. Web Semantics **5**(2), 117–129
37. Westman, J., Nyberg, M.: Conditions of contracts for separating responsibilities in heterogeneous systems. Formal Methods in System Design **52**(2), 147–192 (Apr 2018)
38. Wozniak, L., Clements, P.: How automotive engineering is taking product line engineering to the extreme. In: SPLC '15 (2015)
39. Zou, J., Wang, Y., Lin, K.J.: A formal service contract model for accountable saas and cloud services. In: 2010 IEEE International Conference on Services Computing. pp. 73–80 (July 2010)