






# Formal Foundations of Consistency in Model-Driven Development

Romain Pascual<sup>(✉)</sup> , Bernhard Beckert , Mattias Ulbrich ,  
Michael Kirsten , and Wolfram Pfeifer 

Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany  
`{romain.pascual, beckert, ulbrich, kirsten, wolfram.pfeifer}@kit.edu`

**Abstract.** Models are abstractions used to precisely represent specific aspects of a system in order to make work easier for engineers. This separation of concerns naturally leads to a proliferation of models, and thus to the challenge of ensuring that all models actually represent the same system. We can study this problem by considering that the property is abstracted as a relation between models called consistency. Yet, the exact nature of this relation remains unclear in the context of cyber-physical systems, as such models are heterogeneous and may not be formally described. Therefore, we propose a formal foundation for consistency relations, by (1) providing a set-theoretical description of the virtual single underlying model (V-SUM) methodology, (2) relating consistency to model transformations, and (3) studying the connection between consistency of models and their semantics. In particular, we show that a relation on the semantic spaces of models can be reflected as a relation on models and that this semantics forms a lattice, such that a canonical semantics can be derived from a consistency relation. Our findings lay the foundation for a formal reasoning about precise notions of consistency.

**Keywords:** Model-driven development · Model consistency · Model semantics · Formal foundations · Cyber-physical systems

## 1 Introduction

Model-driven development (MDD) [1, 17] offers a mechanism to overcome the increasing complexity of systems, where each model corresponds to a pragmatic purpose-oriented abstraction of the system [29, p. 131–133]. MDD enables modularization via concern separation, with each model typically dedicated to some specific aspect of the system. As a result, the complete system description is given by an interconnected set of models with complex dependencies between them. With an increase in both the number of models and their complexity, handling the interconnections between these models becomes a key challenge, usually referred to as consistency management [23, 28], i.e., detecting or repairing inconsistencies.

There are two standard ways to define consistency and reason about it. The first approach, mostly used in software engineering, is to assert that models are

consistent when derived from (usually empty) base models by a sequence of consistency-preserving transformations. Therefore, consistency is defined by the set of possible model transformations, such that a (set of) model(s) is consistent whenever belonging to the language associated with the grammar underlying the transformations. This rule-based approach demonstrates the process-oriented nature of ensuring consistency. The second approach to defining consistency assumes that inconsistencies may have a more complex manifestation, which can only be grasped via semantic reasoning. This approach typically defines consistency as the satisfiability of a set of constraints. This logic-based approach emphasizes that consistency relates to the conformance to some desired specifications or industry standards.

In this work, we propose a unified view of these two consistency paradigms within a set-theoretic framework. We aim to establish a formal basis for defining and reasoning about model consistency, bridging the gap between rule-based consistency and semantic constraints. More precisely, we formalize consistency-related notions in the context of the virtual single underlying model (V-SUM) methodology [20]. Then, consistency basically is a relation between the models of possibly distinct meta-models, i.e., models that do not necessarily adhere to the same formalism. In particular, we discuss some foundational aspects that arise when dealing with heterogeneous models for Cyber-Physical Systems (CPS) [22], where we consider consistency as an encoding for *joint realizability* [6].

More precisely, the end goal in system engineering is to ensure that the system can be built and will satisfy some constraints. However, extensively trying to build variations of the system for testing is usually costly and time-consuming, especially when dealing with CPS. Therefore, the realizability of the system is usually approximated by considering a relation over the models describing it. We call this relation *consistency* and propose to study its formal foundations. We illustrate the various introduced notions with two examples, one familiar to formal methods experts, namely consistency between Linear Temporal Logic (LTL) formulae and Büchi automata, and one that is closer to engineering concerns, namely models for the brake system of a car.

*Paper outline.* Sect. 2 presents the context of our work, more precisely, the virtual single underlying approach in model-driven development, further explaining the need to reason about model consistency. Sect. 3 further details how consistency is considered within database science and model-driven development. Sect. 4 describes our motivating examples, namely linear temporal logic and Büchi automata on the one side, and a simplified description of a car’s brake system. Sect. 5 proposes a set-theoretic formalization of the virtual single underlying approach based on an abstract notion of consistency viewed as a relation between models. Sects. 6 and 7 investigate the relations between the notion of consistency and rule-based modification of models, resp. semantics. More precisely, Sect. 6 discusses model transformations with an emphasis on consistency preservation rules as a practical definition of consistency, while Sect. 7 shows that abstracting semantics as a mapping into some semantic space yields a lattice that we can relate to consistency.

## 2 The V-SUM Approach in Model-Driven Development

Model-driven development (MDD) [1, 17] makes models, i.e., system descriptions, the primary artifact of the development process. Models may encode the structure, behavior, or some functionalities of the system, while the development process revolves around creating and editing models. MDD embodies the concept of abstraction, which is fundamental in computer science. While several models may be used to describe a system, Atkinson et al. [2] propose to use a *single underlying model* as the central comprehensive model of a system, such that all user-maintained models become views, i.e., projections from the single underlying model. Extending this idea, also the requirements, the design, and the code artifacts become projections from the single underlying model. In particular, this approach reduces redundancy within the single underlying model to ensure the consistency of all its views by construction.

As single underlying models correspond to monolithic system descriptions, they suffer from practical usability issues such as incompatibility with pre-existing (externally defined) languages and standards, poor maintainability, and reusability due to missing modularity. The VITRUVIUS<sup>1</sup> approach [20] proposes to virtualize the single underlying model. A virtual single underlying model, or V-SUM, is a single underlying model that is not monolithic but consists of several coupled models, where ‘coupled’ means that semantic relationships are provided as mappings between the models. Essentially, such a model acts as a single underlying model for the views but internally consists of several models, which can all have their specific meta-model. Thus, a V-SUM corresponds to a tuple of models associated with consistency relations. More precisely, Klare et al. [20] build consistency from rules for pairs of meta-models by considering some a priori conditions on meta-models. A V-SUM is then deemed consistent if all its pairs of models fulfill all required consistency rules. Theoretically, consistency preservation rules are derived from the consistency rules, intuitively allowing consistency to be restored by triggering further modifications on the models that constitute the V-SUM whenever one is modified. Further explanations are given in Sect. 6.2.

Klare et al. [20] define V-SUMs using notations from the Meta-Object Facility (MOF) for (meta-)models. However, other standards and notations are needed in the context of CPS design. Rather than adding other specific standards and notations, we strive to be agnostic to the description of models and, therefore, to the encoding of model consistency. Additionally, the VITRUVIUS approach can be considered as a refinement of the single underlying model method with a strong practical side, justifying the seemingly complex notions and importance of enabling modifications of the models. Here, we are interested in the foundations of consistency and not (yet) in consistency management, with the agenda of subsequently enabling semantic reasoning for the consistency of V-SUMs. As a result, we want to keep the discussion open, i.e., not only about the notions

---

<sup>1</sup> VITRUVIUS is the view-centric engineering using a virtual single underlying model.

of consistency given by the structure of the models, but rather for any generic consideration of consistency.

### 3 Related Work

To the best of our knowledge, existing research about consistency is domain-dependent, and works that formally reason about consistency assume additional information about what is being analyzed with respect to the consistency notion, e.g., models are represented in UML [23]. Nonetheless, these domain-dependent studies view consistency as a relation, which will be our starting definition in Sect. 5. In the following, we briefly review consistency in database science and model-driven development.

The issue of consistency has long been discussed in the area of data quality in database science. In that context, inconsistencies are known as data anomalies [18], a subject particularly relevant in computer-aided software engineering (CASE), such as in the Fujaba tool [25]. Initially developed for UML and Java round-trip engineering, Fujaba was later extended into a general, plugin-based CASE platform. This platform supports the integration of transformations and other tools to maintain the consistency of various engineering artifacts. Standard techniques for data integration and repair [4] include integrity constraints, functional dependencies, and data deduplication strategies [11]. These methods are typically designed for relational databases and aim to formulate consistent queries over inconsistent data. However, repairing inconsistencies is often challenging due to the size of the databases or the lack of necessary permissions or expertise among the individuals responsible for data integration.

Within MDD, the question of model consistency is often related to the question of model synchronization [14, 15, 33]. In particular, model transformations, i.e., rule-based approaches to describe the evolution of models, allow for the consistent propagation of changes between models. A bidirectional transformation (BX) synchronizes two models by propagating updates between them. More precisely, a *lens* is an asymmetric BX where one model (the view) is determined by another model (the source) such that modifying the source gets reflected in the view by the lens [5, 12]. Thus, bidirectional transformations provide a bidirectional specification of consistency and its repair for a pair of meta-models [31].

## 4 Motivating Examples

### 4.1 Linear Temporal Logic and Büchi Automata

While our primary focus is the consistency of CPS models, we will also use Linear Temporal Logic (LTL) and Büchi automata as a running example. These concepts are well understood within the formal methods' community, ensuring that we can leverage the reader's intuition without the need to understand domain-specific models. In addition, we reexamine familiar concepts from the perspective

of model-driven engineering. Thus, some examples will appear counterintuitive to highlight such variations.

Here, we fix the vocabulary and notation associated with LTL and Büchi automata and refer the reader to standard textbooks for additional information [8]. LTL is an extension of propositional logic that allows the specification of temporal relations between events, first introduced to formally verify computer programs [26]. Its temporal operators include *next*, written **X**, *until* **U**, *sometimes*  $\diamond$ , and *always*  $\square$ . If we assume an infinite set  $\mathcal{P}$  containing all propositional variables, then we obtain the set of all LTL formulae, which we denote by  $\mathcal{F}_{LTL}$ . The semantics of an LTL formula is given by the set of traces that satisfy the formula, where a trace is an infinite sequence of states, and a state is a propositional valuation that indicates which variables are considered true in that state. A nondeterministic Büchi automaton is a nondeterministic finite automaton reading  $\omega$ -words instead of finite ones and accepting those that visit accepting states infinitely many times. The transition function is typically governed by an alphabet, which is considered to be given by valuation functions over a finite set of propositional variables to describe temporal statements. In the sequel, we write  $\mathcal{B}$  for the set of all Büchi automata where the transition function is built over the alphabet of propositional valuation functions.

The connection between LTL and Büchi automata is well understood. Both an LTL formula and a Büchi automaton define a set of traces (i.e., infinite sequences of propositional valuations) that make the formula true, respectively, that are accepted by the automaton. Therefore, we can ask whether an LTL formula and an automaton have the same trace semantics, i.e., define the same  $\omega$ -language. In fact, since Büchi automata are more expressive than LTL, there is an automaton for each formula, but not vice versa. We can also describe a system as an automaton and some properties with a formula. Checking whether the execution of the system fulfills the property then amounts to finding a trace that is accepted by the automaton but does not satisfy the property (i.e., a counter-example). In this sense, we are interested in whether the intersection of the trace semantics of the negated property and the automaton representing the system is nonempty. These are examples of consistency relations.

## 4.2 A Model-Driven Description of a Car’s Brake System

In the context of CPS design, the system description is usually broken down into various models for the different aspects and components of the system. We introduce a simplified set of (meta) models describing a car’s brake system. The purpose is not to be exhaustive with the set of models nor to provide all details needed for manufacturing such a system, but to use the example to highlight pragmatical concerns arising in MDD, especially model interactions. We consider a brake system for which the whole model consists of a mechanical model, a hydraulic model, a thermal model, a system control model, and a software model.

*Mechanical Model.* The mechanical model describes the brake system’s physical components and behavior. It contains information about all components between

the brake pedal and the rotors, e.g., the master cylinder, the brake lines, the calipers, and the brake pads. This model thus encodes the physical response of pressing the brake pedal: it pushes a piston within the cylinder, forcing brake fluid into the brake lines and actioning the calipers, which press the brake pads onto the rotors, ensuring that the wheels are slowed down by friction. Such a model typically uses a geometric CAD description of the various parts annotated with semantic information.

*Hydraulic Model.* The hydraulic model describes fluid dynamics, such as pressure distribution and fluid flow, relying on a finite-element system description to allow for numerical simulations. The model encodes information about the hydraulic brake fluid, as well as some geometric properties of some components. It can be used to compute the pressure increase in the calipers based on the pressure increase in the cylinder and the resulting force applied to the brake pads.

*Thermal Model.* The thermal model describes heat generation within the brake pads and the rotors, as well as heat dissipation through sink mechanisms. It ensures that the material used can withstand the temperature increase without losing integrity.

*System Control Model.* The system control model contains the electronic parts of the brake system, e.g., the sensors, the electronic control unit (ECU), and the actuators. Its purpose is to manage and optimize the braking process, especially with electronic or anti-lock braking systems. In fact, when the brake pedal is pressed, the sensors and the ECU adjust the brake pressure to prevent wheel lockup. This model typically consists of block diagrams representing the equations of the state space of the brake system.

*Software Model.* The software model corresponds to the code run on the ECU. For instance, it contains the C source code for data acquisition to retrieve values from the sensors, signal processing, and slip ratio calculation to determine the potential sliding of the vehicle and feedback loop to modulate the braking force to prevent lockup.

*Model Relations.* Several relations exist between these models. For example, the mechanical and hydraulic models share the geometric description of some parts of the brake system, while providing data the thermal model needs to compute the heat generation. Similarly, the software model relies on data retrieved via control model sensors and performs computations that should be linked to the hydraulic and thermal models. These connections naturally entail consistency relations, where some values and properties must be shared between models. The VITRUVIUS approach [20] defines consistency preservation rules that describe how the modification of a value or, more generally, the transformation of a model should be propagated to the other models. Note that consistency needs not be a binary relation. For instance, the dynamics between the mechanics (especially the piston positions) and the hydraulics (more precisely the forces and

the pressure distribution) change drastically when the brake fluid undergoes a phase shift (which heavily depends on the involved temperatures). A consistency relation capturing adequate braking behavior must therefore inherently involve three models: the mechanical, the hydraulic and the thermal model.

In the following, we use  $M_{mech}$ ,  $M_{hydr}$ ,  $M_{thml}$ ,  $M_{ctrl}$ ,  $M_{soft}$  to denote the associated meta-models.

## 5 Set-Theoretic Foundations of the V-SUM Approach

The goal of this section is to introduce a set-theoretic counterpart to the definitions of V-SUM by Klare et al. [20], abstracting from consistency rules on pairs of (meta-)models, i.e., from a specific notion of consistency, and even abstracting from a specific notion of models. We define a *model* as a purpose-driven representation of a system. For now, we restrict the discussion to models as atomic entities, leaving aside the possibility of decomposing a model into model elements. As a result, we consider consistency notions as relations between models without considering more fine-grained notions that would further specify the relations between ‘parts’ (in a loose sense) of the model.

In MDD, the set of syntactically admissible models of a certain kind is described by a meta-model, serving a similar purpose as a formal grammar does for a formal language. We do not consider how meta-models describe their set of models and assume that a meta-model is the set of its possible models.

**Definition 1 (Meta-model).** *A meta-model  $M$  is the set of its well-formed models  $m \in M$ .*

*Example 1.* In this sense, an LTL formula and a Büchi automaton are models, while the set  $\mathcal{F}_{LTL}$  of LTL formulae and the set  $\mathcal{B}$  of Büchi automata where the transition function is built over valuation functions are meta-models. Note that (LTL) formulae are typically defined over a given signature. Here, we consider the propositional variables used in an LTL formula to be part of the model. In that sense, a model contains a declarative part that details the set of allowed propositional variables (its signature) and a content part consisting of the actual formula. These two parts also extend to Büchi automata.

**Definition 2 (V-SUM meta-model).** *A V-SUM meta-model is a pair  $\mathcal{M} = (M, CR)$  where  $M = M_1 \times \dots \times M_n$  is a Cartesian product of a finite number  $n$  of meta-models equipped with a consistency relation  $CR \subseteq M$ . The meta-models  $M_1, \dots, M_n$  are called the components of  $\mathcal{M}$ .*

Note that our definition of a V-SUM meta-model generalizes the standard binary conceptualization of consistency in model-driven engineering, effectively allowing an arbitrary number of meta-models to be taken into account for the consistency relation. Additionally, the consistency is voluntarily independent of specific meta-models and possible (already) existing relations between them.

*Example 2.* As a basic example, we can consider a V-SUM meta-model composed of  $\mathcal{F}_{LTL}$  and  $\mathcal{B}$ . We then need a consistency relation between LTL formulae and Büchi automata. Such a relation can be semantic as well as purely syntactic. For instance, we may define that the two models are consistent if and only if they are equivalent (i.e., recognize the same set of traces), or we may define that they are consistent if and only if they are co-satisfiable, i.e., the intersection of their trace semantics is non-empty. However, we can also define a weaker syntactic consistency relation, where any two models are consistent if they use the same subset of propositional variables  $P \subset \mathcal{P}$ .

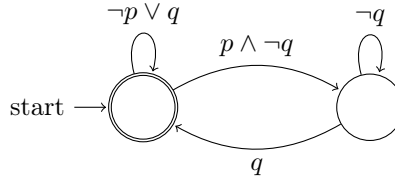
*Example 3.* We consider a V-SUM  $(M_{BS}, CR_{BS})$  for the brake system where  $M_{BS} = M_{mech} \times M_{hydr} \times M_{thml} \times M_{ctrl} \times M_{soft}$ . We use this example to highlight the construction of the V-SUM consistency relation by Klare et al. [20]. Thus,  $CR_{BS}$  is defined based on binary relations between pairs of meta-models. For instance, the mechanical model  $M_{mech}$  and the hydraulic model  $M_{hydr}$  describe the geometric properties of some components, e.g., the cylinder, the calipers, and the brake pads. Thus, a relationship between  $M_{mech}$  and  $M_{hydr}$  could be that the shape of these components is identical in both models. Let this relation be denoted by  $R_{mech,hydr} \subseteq M_{mech} \times M_{hydr}$ . Then  $R_{mech,hydr}$  can be extended to  $CR_{mech,hydr} \subseteq M_{BS}$  by considering tuples  $(m_{mech}, m_{hydr}, m_{thml}, m_{ctrl}, m_{soft})$  such that  $(m_{mech}, m_{hydr}) \in R_{mech,hydr}$ . Similarly, the pressure increase within the brake components leads to heat generation. In other words, the laws of thermodynamics yield a relation  $R_{hydr,thml}$  between  $M_{hydr}$  and  $M_{thml}$ , which can again be extended to  $CR_{hydr,thml} \subseteq M_{BS}$ . The sensors in the control model retrieve values associated with the mechanical, hydraulic, and thermal models, meaning that binary relations  $R_{ctrl,mech}$ ,  $R_{ctrl,hydr}$ , and  $R_{ctrl,thml}$  exist to ensure that the computations performed by each model correctly result in the value measured by the sensors. Each of these relations can again be extended to relations on  $M_{BS}$ . We can combine these relations into a consistency relation

$$CR_{BS} = CR_{mech,hydr} \cap CR_{hydr,thml} \cap CR_{ctrl,mech} \cap CR_{ctrl,hydr} \cap CR_{ctrl,thml}$$

allowing to define  $\mathcal{M}_{BS} = (M_{BS}, CR_{BS})$ .

**Definition 3 (V-SUM model).** A V-SUM model  $m$  of a V-SUM meta-model  $\mathcal{M} = (M, CR)$  is an element of the Cartesian product  $M = M_1 \times \dots \times M_n$ , hence a tuple  $m = (m_1, \dots, m_n)$  of models  $m_i \in M_i$ . A V-SUM model  $m$  is consistent with respect to  $CR$  if  $m \in CR$  (we will use the notation  $CR(m)$ ).

*Example 4.* Let us consider the product of meta-models  $M = \mathcal{F}_{LTL} \times \mathcal{B}$ , the LTL formulae  $\phi = \Box(p \rightarrow \Diamond q)$  and  $\psi = (p \rightarrow \mathbf{X}q)$  and the following Büchi automaton  $\mathcal{A}$ :





where all three models  $\mathcal{A}$ ,  $\phi$ , and  $\psi$  have  $\{p, q\}$  as the declarative part. Then,  $(\phi, \mathcal{A})$  and  $(\psi, \mathcal{A})$  are V-SUM models.

Whether they are consistent depends on which consistency relation  $CR$  we use. If we consider  $CR$  as the equivalence with respect to trace semantics, then  $(\phi, \mathcal{A})$  is consistent. However,  $(\psi, \mathcal{A})$  is not consistent: for example, the trace whose first state has the valuation  $\{p \mapsto \top, q \mapsto \top\}$  and whose following states all have the valuation  $\{p \mapsto \perp, q \mapsto \perp\}$  is accepted by  $\mathcal{A}$ , but it does not satisfy  $\psi$ .

If we consider  $CR$  to be co-satisfiability, then both  $(\phi, \mathcal{A})$  and  $(\psi, \mathcal{A})$  are consistent, since the trace where  $p, q$  are false in all states is accepted by  $\mathcal{A}$  and satisfies both  $\phi$  and  $\psi$ .

Also, if  $CR$  is the simple syntactical criterion that the declarative parts are identical, then both  $(\phi, \mathcal{A})$  and  $(\psi, \mathcal{A})$  are trivially consistent.

While the V-SUM meta-model encompasses all abstractions needed to describe the system, some specific analyses may only require some of the models  $m_i$  of the V-SUM model. More precisely, we are interested in whether distinct parts of the model influence each other. Thus, one may select a subset  $J$  of the integers between 1 and  $n$  and build the Cartesian product of meta-models with indices in  $J$ . Without loss of generality, we can consider  $J$  to consist of the first  $k$  indices and reinterpret the product  $M_1 \times \dots \times M_n$  as a binary Cartesian product  $M = N_0 \times N_1 =$

$$\underbrace{(M_1 \times \dots \times M_k)}_{=:N_0} \times \underbrace{(M_{k+1} \times \dots \times M_n)}_{=:N_1} \quad (1)$$

of two factors  $N_0$  and  $N_1$ . To consider  $N_0$  and  $N_1$  as (smaller) V-SUM meta-models, we need to equip them with a consistency relation, which naturally is the projection of the consistency relation of  $\mathcal{M}$  onto the subset of meta-models.

**Definition 4 (Projected consistency relation and internal consistency).**

Given a V-SUM meta-model  $\mathcal{M} = (M, CR)$  and  $N_i$  (for  $i \in \{0, 1\}$ ) defined as in eq. (1), we consider the V-SUM meta-model  $\mathcal{N}_i = (N_i, CR_i)$  where

$$CR_i = \{n_i \mid \exists n_{1-i} \in CR_{1-i}, CR(n_0, n_1)\} \subseteq N_i$$

is the projected consistency relation from  $\mathcal{M}$  to  $\mathcal{N}_i$ . A model  $n_i \in N_i$  is called internally consistent with respect to  $CR$  if it belongs to the projected consistency relation from  $M$  to  $N_i$ , i.e.,  $CR_i(n_i)$ .

The notions of projected consistency relation and internal consistency allow for splitting a V-SUM meta-model into smaller V-SUM meta-models such that one can reason about the consistency of a part of the system. Essentially, internal consistency states that a model  $n_i$  is consistent in the sense that it does not contain an inconsistency in itself and can be extended to a full V-SUM model.

*Example 5.* It is well known that Büchi automata are more expressive than LTL formulae. Therefore, when considering the consistency relation to be the equivalence of the semantics, the projected consistency relation  $CR_{\mathcal{B}}$  on  $\mathcal{B}$  gives exactly

the set of those automata that admit an LTL counterpart. On the contrary, the projected consistency relation  $CR_{LTL}$  on  $\mathcal{F}_{LTL}$  simply yields  $\mathcal{F}_{LTL}$ , meaning that any LTL formula is internally consistent. Interestingly, if a Büchi automaton is not equivalent to any LTL formula, then it is internally inconsistent, which can be surprising since it could still be implemented. Here, the consistency relation prevents such implementation by discarding any automata that cannot be represented by an LTL formula.

Similarly, the non-equivalence in expressiveness between Büchi automata and LTL formulae means that the projected consistency relations when considering co-satisfiability differ between the two meta-models. On  $\mathcal{F}_{LTL}$ , the internally consistent models are the satisfiable LTL formulae, but on  $\mathcal{B}$ , a Büchi automaton is internally consistent if it accepts at least one trace that can be represented by an LTL formula.

Note that the projected consistency relations from the V-SUM defined with the equality on the declarative part do not restrict the meta-models as one can always build an LTL formula or a Büchi automaton using a specific set of propositional variables. In that sense, any automaton and any formula are internally consistent.

**Definition 5 (Independent meta-models).** *The V-SUM meta-model  $\mathcal{M}$  is independent with respect to the partition into  $\mathcal{N}_0$  and  $\mathcal{N}_1$  if  $CR = CR_0 \times CR_1$ .*

The above definition implies that the V-SUM meta-model  $\mathcal{M}$  is independent with respect to the partition into  $\mathcal{N}_0$  and  $\mathcal{N}_1$  if for every internally consistent sub-models  $n_0 \in \mathcal{N}_0$  and  $n_1 \in \mathcal{N}_1$  the V-SUM model  $(n_0, n_1) \in \mathcal{M}$  is consistent, i.e.,  $CR(n_0, n_1)$ . Intuitively, the modeling choices made in  $n_0$  do not take away possibilities in  $n_1$  or vice versa. Thus, when modifying models in  $n_0$ , internal consistency within  $\mathcal{N}_0$  must be checked, but (external) consistency with  $n_1$  cannot be affected.

**Definition 6 (Overlap).** *Two components  $M_x$  and  $M_y$  of the V-SUM meta-model  $\mathcal{M}$  are without overlap if there exists a partition of  $\mathcal{M}$  into  $\mathcal{N}_0$  and  $\mathcal{N}_1$  such that  $M_x \in \mathcal{N}_0$  and  $M_y \in \mathcal{N}_1$  and  $\mathcal{M}$  is independent with respect to the partition into  $\mathcal{N}_0$  and  $\mathcal{N}_1$ . The two components  $M_x$  and  $M_y$  have an overlap if no such partition exists.*

*Example 6.* While  $\mathcal{F}_{LTL} \times \mathcal{B}$  only contain two meta-models, making the search for a partition trivial, one can still ask whether  $\mathcal{F}_{LTL}$  and  $\mathcal{B}$  have an overlap for the three proposed consistency relations from Example 2. Unsurprisingly, since each criterion explicitly relates the formula with the automaton, the two meta-models indeed have an overlap for the three consistency relations. Using equivalence as consistency relation,  $CR_{LTL} \times CR_{\mathcal{B}}$  contains all pairs consisting of an LTL formula and an automaton that admits an LTL counterpart. Similarly, with the co-satisfiability criterion, we obtain pairs with a satisfiable formula and an automaton that admits some trace that can be described by an LTL formula, while with the same-signature criterion, we obtain any pair with a formula and an automaton.

*Example 7.* Consider  $\mathcal{F}_{LTL} \times \mathcal{B} \times \mathcal{F}_{LTL} \times \mathcal{B}$  containing four meta-models. A tuple of models then contains two formulae and two automata, which we consider as two pairs (formula, automaton). For each pair, we require that the automaton implies the formula, i.e., each trace accepted by the automaton satisfies the formula, intuitively encoding a security specification. If we consider that the pairs describe separate traces without further consistency requirements, then there is no overlap. However, there is an overlap as soon as we introduce an additional constraint, e.g., the first trace must be a refinement of the second.

In this section, we investigated overlaps between meta-models composing the V-SUM meta-model only leveraging a supposedly provided consistency relation. In the next two sections, we discuss how these consistency relations can be built, either from syntactic rules performing edits on the models (Sect. 6) or a mapping into some semantic space (Sect. 7).

## 6 Defining V-SUM Consistency from Rules

In model-driven development, standard approaches rely on model transformations that encode the evolution of models by means of rules. In this section, we recall the standard model transformation approaches, with an emphasis on bidirectional transformations (Bx) in the context of the VITRUVIUS approach. The goal is to explain how our formalization allows formal reasoning about consistency within this approach.

### 6.1 Model Transformation

Model transformation is a core part of MDD and refers to the process of converting one model into another within the same or a different modeling language, i.e., within the same or a different meta-model, thus automating the manipulation of models. Model transformation enables model synchronization [15], i.e., propagating changes from one model to another to preserve consistency between them. Additionally, model transformation allows for model refinement [32], i.e., transforming an abstract model into a more detailed one, model integration [10], i.e., ‘joining’ models in the sense of schema integration for databases, and model migration [27], i.e., updating models as a result of an update on the meta-model.

*Example 8.* Model checking deals with the verification of the system behavior against a specification. When the behavior is given by a Büchi automaton and the specification by an LTL formula, the standard approach involves transforming the formula into an automaton. Algorithms for LTL to Büchi Automata Translation [3, 13] can be seen as examples of model transformations.

*Example 9.* Given an LTL formula and a Büchi automaton, a transformation of the V-SUM model may modify either the declarative parts, i.e., the signatures, or the content part, i.e., the formulae. Modifications of the signature might correspond to restrictions, extensions, or even mapping from one signature to another, i.e., signature morphisms in the sense of institutions [24].

Model transformations are typically defined via a set of rules that define how elements in the source model should be transformed into elements of the target model. These rules can explicitly give input-output pairs, e.g., using graph transformation, triple graph grammars, or bidirectional transformations, or describe how to obtain the output from the input, e.g., using programming constructs that compute the changes to be made. Model transformation is supported by tools and frameworks, such as ATL (Atlas Transformation Language) and QVT (Query/View/Transformation). QVT is an OMG (Object Modeling Group) standard containing several model transformation languages [21], while ATL builds on top of the *Eclipse Modeling Framework* [30] and provides a language as well as an execution environment [19].

Model transformations are used for a purpose, such as to produce user-understandable representations or to make a model accessible to a particular approach. It may, however, be that this purpose is not compatible with a consistency relation  $CR$ . If there are two models  $m_1, m_2 \in M$  of a meta-model  $M$  such that  $m_1$  is consistent wrt.  $CR$  within a V-SUM while  $m_2$  is inconsistent wrt.  $CR$  that are transformed to the same model in the target meta-model (e.g. by omitting details), then the transformation is incompatible with  $CR$ . Otherwise, it is possible to define a consistency relation  $CR'$ , operating on the image of the transformation instead of the original meta-model.

## 6.2 Rule-Based Consistency

The VITRUVIUS tool [20] allows the specification of consistency for structural models that are composed of model elements. Then, a consistency rule relates model elements that can occur in a model to model elements that have to occur in another model. This relation ensures the consistency of the pair of models. Based on the consistency rules, VITRUVIUS defines *delta*-based consistency preservation rules [9]. Delta-based approaches explicitly describe the modifications between two states of the models, called a *change*. A change consists of the creation, deletion, or update of some elements and can be formally described as a function  $\delta_M: M \rightarrow M$  from a meta-model  $M$  to itself, such that the set of all changes for  $M$  can be written  $\Delta_M$ .

*Example 10.* We can consider the models used for the description of a car's brake system as structural models where the model elements correspond to the components of the system, e.g., the brake pedal, the master cylinder, or the calipers. Since these components occur in several models, e.g., the mechanical and hydraulic models, the various models should agree on elements that describe the same component. Thus, a possible consistency rule could state that the model elements describing the same component have the same name.

A consistency preservation rule  $CPR$  for a product of meta-models  $M = M_1 \times \dots \times M_n$  is a function  $CPR: M \times \Delta_M \rightarrow \Delta_M$ , where  $\Delta_M = \Delta_{M_1} \times \dots \times \Delta_{M_n}$ . Given a tuple of models and a tuple of changes (one per model), the consistency preservation rule outputs a tuple of changes. Intuitively,  $CPR$

transforms a tuple of changes over a tuple of consistent models into a tuple of modified changes, which, if applied to the original models, yields a tuple of consistent models. The set of consistent tuples of models can then be derived inductively from a tuple of models axiomatically considered consistent, e.g., the tuple of all empty models, by applying changes returned by the consistency preservation rule. Formally the consistency relation  $CR_{CPR}$  associated with the consistency preservation rule  $CPR$  and the initial tuple of models  $(m_1^0, \dots, m_n^0)$  is the set of tuple of models  $(m_1, \dots, m_n)$  for which there exist  $\delta_M^1, \dots, \delta_M^k$  in  $\Delta_M$  such that for all  $i$  in  $\{1, \dots, k\}$ , there exists a tuple of models  $(m_1^i, \dots, m_n^i)$  satisfying  $(m_1^i, \dots, m_n^i) = CPR((m_1^{i-1}, \dots, m_n^{i-1}), \delta_M^i)((m_1^{i-1}, \dots, m_n^{i-1}))$  and  $(m_1^k, \dots, m_n^k) = (m_1, \dots, m_n)$ .

Therefore, the VITRUVIUS tool defines the consistency relation  $CR$  of a V-SUM meta-model by a consistency preservation rule  $CPR$ . For practical purposes,  $CPR$  is usually split into several consistency preservation rules, each acting on a pair of meta-models. Each consistency preservation rule then defines a consistency relation, and  $CR$  is retrieved as the intersection of all consistency relations, allowing modularity and reusability of the consistency preservation rules. As a result, their execution needs to be orchestrated to restore consistency. Therefore, the VITRUVIUS tool considers an application function that executes the consistency preservation rules  $CPR_1, \dots, CPR_k$  in an order such that applying the resulting changes to the initial models yields models that are consistent according to the consistency relations of all consistency preservation rules.

## 7 Defining V-SUM Consistency from Semantics

While consistency relations can be defined directly at the meta-model level, it only seems natural that the relation might be defined based on some semantic information about the models. Actually, such a construction corresponds to the consistency relation already defined for the equivalence and co-satisfiability criterion between LTL formulae and Büchi automata. After formalizing a notion of abstract semantics as a mapping of each model to some value, we show that such semantics form a lattice, enabling comparison, combination, and refinement of abstract semantics. This lattice is the key to relating consistency and semantics, more precisely, to lift semantic consistency relations to the V-SUM and to derive semantics from consistency.

The main result is that for any consistency relation and any V-SUM meta-model, there always exists a semantics (called the *natural* semantics) that captures exactly the right information from a model needed to decide if it is consistent within the V-SUM. Any other semantics either abstracts too much or contains more details than necessary. This result is quite astonishing in this general setting where we did not make any assumptions about the nature or inner structure of models (we did not need to introduce a notion of model elements, e.g.).

## 7.1 An Abstract Semantics

There can be very different ways of defining semantics, e.g., the Tarskian approach in logic, denotational and operational semantics for programming languages or implicit semantics for most engineering models, which are given as the output result of a tool running some computations. As to not restrict ourselves on how semantics is defines, we propose to consider semantics abstractly.

**Definition 7 (Abstract semantics).** *An abstract semantics is a mapping  $\llbracket \cdot \rrbracket : M \rightarrow S$  for a meta-model  $M$ , where  $S$  is called the semantic space.*

This definition naturally raises two questions: (a) what is the codomain  $S$  of the function  $\llbracket \cdot \rrbracket$ ? and (b) what is the meaning of  $\llbracket \cdot \rrbracket$ ? There is no definite answer to these questions since the semantic space  $S$  is necessarily purpose-dependent, meaning that the mapping is expected to vary according to user intent. In particular, that opens the way for tool-defined or user-defined semantics.

*Example 11.* Following Tarskian principles, the mapping  $\llbracket \cdot \rrbracket$  can provide the set of satisfying structures for a model.<sup>2</sup> This is the point of view used in Sect. 5 when dealing with LTL formulae and Büchi automata, considered through the lens of their satisfying traces. Formally, we considered  $\llbracket \cdot \rrbracket_{LTL} : \mathcal{F}_{LTL} \rightarrow \mathcal{P}((2^{\mathcal{P}})^{\omega})$  and  $\llbracket \cdot \rrbracket_{\mathcal{B}} : \mathcal{B} \rightarrow \mathcal{P}((2^{\mathcal{P}})^{\omega})$ , where  $\mathcal{P}((2^{\mathcal{P}})^{\omega})$  is the power set of  $\omega$ -words on propositional valuations.

*Example 12.* One could also consider different (meta-)models by fixing a given trace  $\tau$  and asking whether the trace satisfies LTL formulae. Then, the semantic space  $S$  becomes the standard two-element Boolean algebra  $\mathbb{B}$  of truth values and the function  $\llbracket \cdot \rrbracket_{\mathbb{B}}^{\tau} : \mathcal{F}_{LTL} \rightarrow \mathbb{B}$  maps formulae to their truth value for the fixed trace  $\tau$ .

*Example 13.* Additionally, the mapping  $\mathcal{F}_{LTL} \rightarrow \mathcal{P}$  corresponding to the declarative part of the LTL models can also be considered as a semantic mapping, even though it is rather trivial and syntactic in nature. Similarly, if we consider  $M$  to be a set of Java classes,  $\llbracket \cdot \rrbracket$  could yield a syntactic property, such as the number of methods or attributes in the class. One could even consider the identity function as a semantic mapping.

Similar to the definition of models, the notion of semantic mapping is inherently driven by its intended purpose, meaning it depends on which properties of the model we aim to address or discuss. We will see in Sect. 7.3 that semantics induces consistency relations, and in Sect. 7.4 that all consistency relations come with natural semantics for the concerned meta-models. Before that, we investigate the structure of the set of all possible semantics of a single meta-model.

<sup>2</sup> Note that we refrain from using the word ‘model’ from logic, and instead call a model in logic a ‘satisfying structure’.

## 7.2 The Lattice of Semantics

At first glance, one could consider various abstract semantics for a meta-model, even mappings to different semantic spaces, seemingly hindering the possibility of comparing the values that a model  $m$  takes for each semantics. We will see that the semantics for a meta-model up to isomorphism actually form a complete lattice. To obtain this lattice of semantics, we shift our attention from the semantic space to the quotients of the meta-model by equivalence relations and then consider the equivalence lattice, i.e., the lattice of equivalence relations. This allows for restricting the analysis of all abstract semantics to all possible quotients of the meta-model.

We recall that the *equivalence kernel* of a function  $f$  is the equivalence relation  $\equiv$  defined on the domain of  $f$  by  $x \equiv y$  if and only if  $f(x) = f(y)$ . Interestingly, a kernel allows shifting the focus from the codomain of the function to its domain: we can use the kernel of an abstract semantics to reason within the meta-model and not in the semantic space.

*Example 14.* Consider the case of the Boolean semantics  $\llbracket \cdot \rrbracket_{\mathbb{B}}^{\tau}$  from Example 12 which states whether a model (formula) is either semantically true or false. The semantic function is entirely characterized by which models get the same truth value. Whether  $\mathbb{B}$  is the set  $\{0, 1\}$ , the set  $\{\text{false}, \text{true}\}$ , or the set  $\{\perp, \top\}$  does not matter. The important part is that  $\llbracket \cdot \rrbracket_{\mathbb{B}}$  partitions  $M$  into the equivalence classes for true and false. Note that for practical purposes, i.e., when actually working with semantic values, it *does* matter whether the binary semantics space  $\mathbb{B}$  is encoded as  $\{0, 1\}$ ,  $\{\text{false}, \text{true}\}$ , or  $\{\perp, \top\}$ . However, the concrete choice of representatives (or names) for the equivalence classes is irrelevant for comparing the degree of abstraction or the amount of information that is kept by the abstract semantics.

Any function  $f: X \rightarrow Y$  can be uniquely factorized (up to isomorphism) into  $f = i \circ s$  where  $i$  is an injection and  $s$  a surjection. Indeed,  $s$  is the canonical surjection  $X \rightarrow X/\equiv$  to the quotient by the equivalence kernel. It maps any element  $x$  of  $X$  to its equivalent class  $[x]_{\equiv} = \{x' \in X \mid x' \equiv x\}$ . Formally restricting  $S$  to the image of  $\llbracket \cdot \rrbracket_x$  ensures that for an abstract semantics  $\llbracket \cdot \rrbracket_x: M \rightarrow S$ ,  $M/\equiv_x$  and  $S$  are isomorphic (where  $\equiv_x$  is the kernel of  $\llbracket \cdot \rrbracket_x$ ). This isomorphism ensures that the study can be shifted from the semantic spaces to the quotient structures  $M/R$  for the equivalence relations  $R \subseteq M \times M$ , with the benefit that these quotient structures are now comparable. Indeed, the set of all equivalence relations on a set form a complete lattice called the *equivalence lattice* (for more details, see [7, Chap. 12] or [16, Chap. IV, Sect. 4]). The equivalence lattice has set-inclusion as the partial order,  $\bigwedge_{\equiv} X = \bigcap X$  as the meet (infimum, or greatest lower bound), and  $\bigvee_{\equiv} X = (\bigcup X)^*$  as the join (supremum, or least upper bound). Note that the transitive closure needs to be added in case of  $\bigvee$  since the union alone does not guarantee transitivity.

The isomorphism between abstract semantics and equivalence relations allows for transferring the equivalence lattice on  $M$  into a complete lattice on the semantics, i.e., by mapping the equivalence relation  $R$  to  $M/R$ . Note that the

order of the lattice is inverted since  $R$  moves to the denominator of the quotient structure, i.e.,  $M/R_1 \sqsubseteq M/R_2$  if and only if  $R_2 \subseteq R_1$ . In particular, given two abstract semantics  $S_1$  and  $S_2$ , we have that  $S_1 \sqsubseteq S_2$  if and only if  $S_2$  allows distinguishing between the same model elements as  $S_1$  and possibly more. In the sequel, we write  $\mathcal{L}_{\text{sem}}^M$  for the lattice of semantics on  $M$ .

*Example 15.* The bottom element  $\llbracket \cdot \rrbracket_{\perp} : M \rightarrow M/M^2 \simeq \{\star\}$  in the lattice of semantics corresponds to the top element in the lattice of equivalence relations, which is the trivial relation  $M^2$  that relates any two elements. This corner-case semantic space contains a single element  $\star$ , meaning that all models have the same semantics  $\llbracket m \rrbracket_{\perp} = \star$ . All information is lost in the semantic evaluation: it is maximally abstract.

*Example 16.* On the other side, the top element  $\llbracket \cdot \rrbracket_{\top} : M \rightarrow M/\text{id}_M \simeq M$  corresponds to the smallest possible equivalence relation, the identity (also called diagonal) relation  $\text{id}_M = \{(m, m) \mid m \in M\}$  which relates every element only to itself. In this extreme semantics, every model  $m \in M$  is its own semantic value<sup>3</sup>  $\llbracket m \rrbracket_{\top} = m$ . No information is lost by the semantic evaluation: there is no abstraction.

*Example 17.* Let us consider again the family of abstract semantics from Example 12,  $\llbracket \cdot \rrbracket_{\mathbb{B}}^{\tau} : \mathcal{F}_{LTL} \rightarrow \mathbb{B}$ , where  $\tau$  is a fixed trace. Each one maps an LTL formula to the truth value indicating whether the trace  $\tau$  satisfies the formula. Two different traces  $\tau_1$  and  $\tau_2$  give rise to two different semantics  $\llbracket \cdot \rrbracket_{\mathbb{B}}^{\tau_1}$  and  $\llbracket \cdot \rrbracket_{\mathbb{B}}^{\tau_2}$  that have the same semantic space  $\mathbb{B}$ . However, they are not related in the lattice ( $\llbracket \cdot \rrbracket_{\mathbb{B}}^{\tau_1} \not\sqsubseteq \llbracket \cdot \rrbracket_{\mathbb{B}}^{\tau_2}$  and  $\llbracket \cdot \rrbracket_{\mathbb{B}}^{\tau_2} \not\sqsubseteq \llbracket \cdot \rrbracket_{\mathbb{B}}^{\tau_1}$ ) if there is no refinement relation between the equivalence relations on the formulas.

The next step is to relate our notions of abstract semantics and consistency.

### 7.3 Semantics-Induced Consistency

If we equip two meta-models  $M_i$  with dedicated semantic mappings  $\llbracket \cdot \rrbracket_i : M_i \rightarrow S_i$  for  $i \in \{1, \dots, n\}$ , we can impose conditions on the models within the semantic spaces by considering a relation on  $S_1 \times \dots \times S_n$ , i.e., we can define a relation  $SCR \subseteq S_1 \times \dots \times S_n$  and then define models  $m_i \in M_i$  to be consistent if and only if  $SCR(\llbracket m_1 \rrbracket_1, \dots, \llbracket m_n \rrbracket_n)$ .

*Example 18.* In Sect. 5, we instantiated  $SCR$  to be the equality relation to obtain the equivalence criterion for consistency, and we used ‘have non-empty intersection’ to obtain co-satisfiability. Formally, with  $\llbracket \cdot \rrbracket_{LTL} : \mathcal{F}_{LTL} \rightarrow \mathcal{P}((2^{\mathcal{P}})^{\omega})$  and  $\llbracket \cdot \rrbracket_{\mathcal{B}} : \mathcal{B} \rightarrow \mathcal{P}((2^{\mathcal{P}})^{\omega})$ , we can consider  $SCR_{eq}$  and  $SCR_{cosat}$  such that for two sets of traces  $T$  and  $T'$ ,  $SCR_{eq}(T, T')$  if and only if  $T = T'$  and  $SCR_{cosat}(T, T')$  if and only if  $T \cap T' \neq \emptyset$ . Note that other relations might have a meaningful

<sup>3</sup> This is a generalization of the notion of the *Herbrand semantics* of first-order logic, where every ground term is its own interpretation (which does also not lose any information during the semantic mapping).



use, e.g.,  $SCR$  such that  $SCR(T, T')$  if and only if  $T \subseteq T'$  yields a refinement relation, where an automaton and a formula are consistent if each trace accepted by the automaton also satisfies the formula.

We call  $SCR \subseteq S_1 \times \dots \times S_n$  a *semantic consistency relation*, from which we can define a *semantics-induced consistency relation*  $CR_{SCR}$  by

$$CR_{SCR}(m_1, \dots, m_n) :\iff SCR(\llbracket m_1 \rrbracket_1, \dots, \llbracket m_n \rrbracket_n).$$

This semantics-induced consistency relation only considers the interpretation of the models and not their syntactic form or identity. In particular, given models  $m_i, m'_i \in M_i$  for  $i \in \{1, \dots, n\}$ , the semantic consistency relation  $CR_{SCR}$  ensures that, if  $m_i, m'_i$  have the same semantics (i.e.,  $\llbracket m_i \rrbracket_i = \llbracket m'_i \rrbracket_i$ ), then the V-SUM  $(m_1, \dots, m_n)$  is consistent if and only if  $(m'_1, \dots, m'_n)$  is consistent (i.e.,  $CR_{SCR}(m_1, \dots, m_n) \iff CR_{SCR}(m'_1, \dots, m'_n)$ ).

*Example 19.* The two first V-SUM meta-models of Example 2 were built using the semantics-induced consistency relations from  $SCR_{eq}$  and  $SCR_{cosat}$  of Example 18.

We showed how consistency relations can be induced by given semantics. Next, we will look at the converse question of defining a semantics from a consistency relation.

#### 7.4 Consistency-Induced Semantics

We introduced the notion of abstract semantics as a way to add some meaning to the models of a given meta-model. Since consistency is a relation on models, it encodes some information on each model. Therefore, it is natural that a somewhat canonical or natural semantics can be derived from a consistency relation. In fact, we will show that this semantics is canonical in the sense that it corresponds to a meet in the semantic lattice introduced in Sect. 7.2. This semantics encodes precisely the information needed to determine if two models are consistent but abstracts away everything else. Moreover, we will establish that all abstract semantics that are compatible with a given consistency relation  $CR$  form a lattice with the natural semantics as its bottom element.

**Definition 8 (Compatible semantics).** *Given a V-SUM meta-model  $\mathcal{M} = (M_1 \times \dots \times M_n, CR)$ , a family of abstract semantics  $\llbracket \cdot \rrbracket_i: M_i \rightarrow S_i$  is called compatible with  $CR$  if and only if there is a semantic consistency relation  $SCR \subseteq S_1 \times \dots \times S_n$ , such that  $CR$  coincides with the semantics-induced consistency relation  $CR_{SCR}$  (according to Sect. 7.3), i.e.,  $CR = CR_{SCR}$ .*

As a first step, we can consider the two corner cases of Sect. 7.2. The semantics  $\llbracket \cdot \rrbracket_{\perp}$  from Example 15 maps all models to the single value  $S_{\perp} = \{\star\}$ , meaning that a consistency relation defined via  $\llbracket \cdot \rrbracket_{\perp}$  cannot distinguish between any two different models and either all models are consistent, or none are. On

the opposite side of the spectrum, the semantics  $\llbracket \cdot \rrbracket_{\top}$  from Example 16 maps each model to itself and, while it would allow for specifying consistency, it does not abstract away anything, meaning that all needed information still needs to be added. The question is then whether there is always a semantics that can be used to define a consistency relation, and, if there are several, which fits best.

In the following, we define for each and every component of the V-SUM meta-model the largest possible equivalence relation that induces a well-suited semantic function that can be used as part of a family of abstract semantics compatible with  $CR$ . First, given a V-SUM meta-model  $\mathcal{M} = (M, CR)$  with  $M = M_1 \times \dots \times M_n$ , we write  $CR^{\nabla^i}(\nu)$  for the set

$$\{(m_1, \dots, m_{i-1}, m_{i+1}, \dots, m_n) \in M_1 \times \dots \times M_{i-1} \times M_{i+1} \times \dots \times M_n \mid \\ CR(m_1, \dots, m_{i-1}, \nu, m_{i+1}, \dots, m_n)\}$$

where  $\nu$  is a model for some component  $M_i$  of the V-SUM meta-model. Then, we define, for each component  $M_i$  of the V-SUM meta-model, the equivalence relation  $\sim_i \subseteq M_i \times M_i$  by

$$m_a \sim_i m_b \iff CR^{\nabla^i}(m_a) = CR^{\nabla^i}(m_b) .$$

Two models  $m_a, m_b \in M_i$  are related if and only if the sets of tuples that extend them to consistent V-SUM models are the same. Therefore, any such two models are indistinguishable from the perspective of the other components of the V-SUM meta-model, meaning that we can semantically identify them.

**Definition 9 (Natural semantics).** *The semantics  $\llbracket \cdot \rrbracket_i^{\text{nat}} : M_i \rightarrow M_i / \sim_i$  induced for the components  $M_i$  of the V-SUM meta-model by the equivalence relations  $\sim_i$  are called the natural semantics for  $CR$ .*

Note that the natural semantics depend on the investigated consistency relation.

**Proposition 1.** *The natural semantics are compatible with  $CR$  (according to Def. 8).*

*Proof.* Let us consider the semantic consistency relation

$$SCR^{\text{nat}} = \{(\llbracket m_1 \rrbracket_1^{\text{nat}}, \dots, \llbracket m_n \rrbracket_n^{\text{nat}}) \mid CR(m_1, \dots, m_n)\}$$

induced by the family of natural semantics. First, let us remark that for any two models  $m_a$  and  $m_b$  of some component  $M_i$ , the construction of the natural semantics for  $CR$  ensures that  $m_a \sim_i m_b$  if and only if  $\llbracket m_a \rrbracket_i^{\text{nat}} = \llbracket m_b \rrbracket_i^{\text{nat}}$ . Therefore,  $SCR^{\text{nat}}$  is a well-defined semantic consistency relation. Additionally, the constructions of  $SCR^{\text{nat}}$  and of the natural semantics  $\llbracket \cdot \rrbracket_i^{\text{nat}} : M_i \rightarrow M_i / \sim_i$  from  $CR^{\nabla^i}$  directly yield that  $CR = CR_{SCR^{\text{nat}}}$ , i.e., that family of natural semantics is compatible with  $CR$ .

The family of natural semantics of the components of a V-SUM meta-model is not the only compatible family of semantics that can be used to represent  $CR$  semantically, but it is the lower bound of all compatible semantics. In fact, we already saw in Sect. 7.2 that the set of semantics of a meta-model  $M$  forms a lattice  $\mathcal{L}_{\text{sem}}^M$  for the partial order  $\sqsubseteq$ . Given any semantics  $\llbracket \cdot \rrbracket : M \rightarrow S$ , we can therefore consider the quotient sublattice  $\mathcal{L}_{\text{sem}}^M / \llbracket \cdot \rrbracket$  defined as [7, Chap. 2]

$$\mathcal{L}_{\text{sem}}^M / \llbracket \cdot \rrbracket = \{ \llbracket \cdot \rrbracket_x \in \mathcal{L}_{\text{sem}}^M \mid \llbracket \cdot \rrbracket_x \sqsubseteq \llbracket \cdot \rrbracket \} .$$

The quotient sublattice inherits the join and meet operations from  $\mathcal{L}_{\text{sem}}^M$  and has  $\llbracket \cdot \rrbracket$  as minimal element.

**Proposition 2.** *Let  $\mathcal{M} = (M_1 \times \dots \times M_n, CR)$  be a V-SUM meta-model, and let  $\mathcal{L}_i^{\text{nat}}$  be the quotient sublattice  $\mathcal{L}_{\text{sem}}^{M_i} / \llbracket \cdot \rrbracket_i^{\text{nat}}$  of the lattice of semantics of the component  $M_i$  by the natural semantics  $\llbracket \cdot \rrbracket_i^{\text{nat}}$ . Any family of semantics  $\llbracket \cdot \rrbracket_i$  where each abstract semantics  $\llbracket \cdot \rrbracket_i$  is an element of  $\mathcal{L}_i^{\text{nat}}$  is compatible with  $CR$ .*

*Proof.* By definition, an abstract semantics  $\llbracket \cdot \rrbracket_i$  in  $\mathcal{L}_i^{\text{nat}}$  refines the partition induced by  $\llbracket \cdot \rrbracket_i^{\text{nat}}$  on  $M_i$ . In particular, any equivalence class  $[x]_{\sim_i}$  is the disjoint union of some equivalence classes  $[m]_{\equiv_i}$  where  $\equiv_i$  is the kernel of  $\llbracket \cdot \rrbracket_i$ . In other words, there is a function  $\pi_i : M_i / \equiv_i \rightarrow M_i / \sim_i$  that maps  $[m]_{\equiv_i}$  to  $[m]_{\sim_i}$ .

The same arguments as the proof of Prop. 1 hold, since

$$SCR^{\text{nat}} = \{ (\pi_1(\llbracket m_1 \rrbracket_1^{\text{nat}}), \dots, \pi_n(\llbracket m_n \rrbracket_n^{\text{nat}})) \mid CR(m_1, \dots, m_n) \} .$$

In particular, any family of semantics  $\llbracket \cdot \rrbracket_i$  where each abstract semantics  $\llbracket \cdot \rrbracket_i$  is an element of  $\mathcal{L}_i^{\text{nat}}$  is compatible with  $CR$ .

The *natural semantics* is hence the canonical function  $\llbracket \cdot \rrbracket_i : M_i \rightarrow M_i / \sim_i$  mapping each model in  $M_i$  to its equivalence class modulo  $\sim_i$ . Since  $\sim_i$  unifies all models indistinguishable by  $CR$ , the natural semantics retains exactly the information needed to compute consistency with respect to  $CR$  without redundancy and is, in this sense, minimal.

*Example 20.* If the consistency relation between  $(m_1, m_2) \in M_1 \times M_2$  requires that the model  $m_2$  must contain at least as many elements as the model  $m_1$ , then the equivalence relation  $\sim_i$  for both  $M_1$  and  $M_2$  relates any two models with the same number of elements. Therefore, we have  $M_i / \sim_i \simeq \mathbb{N}$ .

*Example 21.* If we consider a V-SUM meta-model on  $M_1 \times M_2$  which has no overlap, then models need not be distinguishable. Thus, the equivalence is universal and the natural semantics for each component is the bottom element  $\llbracket \cdot \rrbracket_{\perp} : M_i \rightarrow M_i / M_i^2 \simeq \{ \star \}$  in the lattice of semantics from Example 15.

*Example 22.* If we consider models to be equivalent if and only if they are absolutely identical, then the equivalences are the top elements  $\llbracket \cdot \rrbracket_{\top} : M_i \rightarrow M_i / \text{id}_{M_i} \simeq M_i$  from Example 16, i.e., the reflexivity relations.

## 8 Conclusion

The realizability of a system may be approximated by conditions on the models used to describe the system. When several models are involved, these conditions typically amount to relations between the models. Aggregating these relations yields a global relation – the consistency relation – over all the models involved. After investigating the notion of overlap that states whether the meta-model can be modularized, we reviewed the standard approaches in model-driven development, i.e., based on model transformations, and showed how they fit our framework. We clarified the relation between consistency and semantics, and explored the structure of semantics when considered abstractly. While these abstract notions of consistency and semantics enable engineers to provide their own consistency and semantics, they also allow for changing the usual notion of consistency given by artifacts from normative consistency, i.e., consistency defined by the artifacts themselves, to descriptive consistency, i.e., consistency that has to be shown to conform to an already (semantically) defined consistency. In other words, the abstract notions of consistency and semantics and their highlighted relation allow for checking the correctness of a descriptive notion of consistency, e.g., given by model transformations, with respect to an already existing normative consistency notion.

For future work, some simplifying hypotheses within our work should be addressed. In particular, we plan to investigate the structure of models, e.g., given by model elements, model components, or submodels, and their relation with consistency. Additionally, our abstract notions should allow encoding semantics within the meta-model, in order to make consistency a first-class citizen of the V-SUM approach. On a more practical level, we plan to apply our formal foundations to complex system designs, particularly in CPS engineering. Our formalization is inherently agnostic to both the meta-models involved and the way of specifying consistency. Thus, it enables consistency between heterogeneous models, i.e., continuous and discrete models. This practical application may also uncover unforeseen challenges in our abstract framework, thereby deepening the understanding of model consistency.

**Acknowledgments.** This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – CRC 1608 – 501798263, from the topic Engineering Secure Systems of the Helmholtz Association (HGF), and by KASTEL Security Research Labs.

**Disclosure of Interests.** The authors have no competing interests to declare that are relevant to the content of this article.

## References

1. Ambler, S.: The Object Primer: Agile Model-Driven Development with UML 2.0. Cambridge University Press (2004). <https://doi.org/10.1017/CB09780511584077>

2. Atkinson, C., Stoll, D., Bostan, P.: Orthographic software modeling: A practical approach to view-based development. In: Maciaszek, L., González-Pérez, C., Jablonski, S. (eds.) *Evaluation of Novel Approaches to Software Engineering*. pp. 206–219. Springer (2010). [https://doi.org/10.1007/978-3-642-14819-4\\_15](https://doi.org/10.1007/978-3-642-14819-4_15)
3. Babiak, T., Křetínský, M., Řehák, V., Strejček, J.: LTL to Büchi automata translation: Fast and more deterministic. In: Flanagan, C., König, B. (eds.) *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2012)*. pp. 95–109. Springer (2012). [https://doi.org/10.1007/978-3-642-28756-5\\_8](https://doi.org/10.1007/978-3-642-28756-5_8)
4. Bertossi, L.: *Database Repairs and Consistent Query Answering, Synthesis Lectures on Data Management*, vol. 20. Morgan & Claypool Publishers (M & C) (2011), <https://dl.acm.org/doi/10.5555/2371212>
5. Bohannon, A., Foster, J., Pierce, B., Pilkiewicz, A., Schmitt, A.: Boomerang: resourceful lenses for string data. In: *Proceedings of the 35th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. pp. 407–419. POPL ’08, Association for Computing Machinery (2008). <https://doi.org/10.1145/1328438.1328487>
6. Bowman, H., Steen, M., Boiten, E., Derrick, J.: A formal framework for viewpoint consistency. *Formal Methods in System Design* **21**(2), 111–166 (2002). <https://doi.org/10.1023/A:1016000201864>
7. Crawley, P., Dilworth, R.: *Algebraic theory of lattices*. Prentice-Hall (1973)
8. Demri, S., Gastin, P.: Specification and verification using temporal logics. In: *Modern Applications of Automata Theory, IISc Research Monographs Series*, vol. Volume 2, pp. 457–493. World Scientific (2011). [https://doi.org/10.1142/9789814271059\\_0015](https://doi.org/10.1142/9789814271059_0015)
9. Diskin, Z., Xiong, Y., Czarnecki, K.: From state- to delta-based bidirectional model transformations. In: Tratt, L., Gogolla, M. (eds.) *3rd International Conference on Theory and Practice of Model Transformations*. pp. 61–76. *Lecture Notes in Computer Science*, Springer (2010). [https://doi.org/10.1007/978-3-642-13688-7\\_5](https://doi.org/10.1007/978-3-642-13688-7_5)
10. Dolk, D., Kottemann, J.: Model integration and a theory of models. *Decision Support Systems* **9**(1), 51–63 (1993). [https://doi.org/10.1016/0167-9236\(93\)90022-U](https://doi.org/10.1016/0167-9236(93)90022-U)
11. Elmagarmid, A., Ipeirotis, P., Verykios, V.: Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering* **19**(1), 1–16 (2007). <https://doi.org/10.1109/TKDE.2007.250581>
12. Foster, J., Greenwald, M., Moore, J., Pierce, B., Schmitt, A.: Combinators for bidirectional tree transformations: A linguistic approach to the view-update problem. *ACM Transactions on Programming Languages and Systems* **29**(3), 17–es (2007). <https://doi.org/10.1145/1232420.1232424>
13. Gastin, P., Oddoux, D.: Fast LTL to Büchi automata translation. In: Berry, G., Comon, H., Finkel, A. (eds.) *Computer Aided Verification*. pp. 53–65. Springer (2001). [https://doi.org/10.1007/3-540-44585-4\\_6](https://doi.org/10.1007/3-540-44585-4_6)
14. Giese, H., Hildebrandt, S., Neumann, S.: Model synchronization at work: Keeping SysML and AUTOSAR models consistent. In: Engels, G., Lewerentz, C., Schäfer, W., Schür, A., Westfechtel, B. (eds.) *Graph Transformations and Model-Driven Engineering: Essays Dedicated to Manfred Nagl on the Occasion of his 65th Birthday*, pp. 555–579. Springer (2010). [https://doi.org/10.1007/978-3-642-17322-6\\_24](https://doi.org/10.1007/978-3-642-17322-6_24)
15. Giese, H., Wagner, R.: Incremental model synchronization with triple graph grammars. In: Nierstrasz, O., Whittle, J., Harel, D., Reggio, G. (eds.) *Model Driven Engineering Languages and Systems (MoDELS 2006)*. pp. 543–557. Springer (2006). [https://doi.org/10.1007/11880240\\_38](https://doi.org/10.1007/11880240_38)

16. Grätzer, G.: General Lattice Theory. Birkhäuser Verlag, second edition edn. (2003)
17. Hailpern, B., Tarr, P.: Model-driven development: The good, the bad, and the ugly. *IBM Systems Journal* **45**(3), 451–461 (2006). <https://doi.org/10.1147/sj.453.0451>
18. Ilyas, I., Chu, X.: Trends in cleaning relational data: Consistency and deduplication. *Foundations and Trends® in Databases* **5**(4), 281–393 (2015). <https://doi.org/10.1561/19000000045>
19. Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I.: ATL: A model transformation tool. *Science of Computer Programming* **72**(1), 31–39 (2008). <https://doi.org/10.1016/j.scico.2007.08.002>
20. Klare, H., Kramer, M., Langhammer, M., Werle, D., Burger, E., Reussner, R.: Enabling consistency in view-based system development — The Vitruvius approach. *Journal of Systems and Software* **171** (2021). <https://doi.org/10.1016/j.jss.2020.110815>
21. Kurtev, I.: State of the art of QVT: A model transformation language standard. In: Schürr, A., Nagl, M., Zündorf, A. (eds.) *Applications of Graph Transformations with Industrial Relevance*. pp. 377–393. Springer (2008). [https://doi.org/10.1007/978-3-540-89020-1\\_26](https://doi.org/10.1007/978-3-540-89020-1_26)
22. Lee, E.: CPS foundations. In: *Proceedings of the 47th Design Automation Conference*. pp. 737–742. DAC '10, Association for Computing Machinery (2010). <https://doi.org/10.1145/1837274.1837462>
23. Lucas, F.J., Molina, F., Toval, A.: A systematic review of UML model consistency management. *Information and Software Technology* **51**(12), 1631–1645 (2009). <https://doi.org/10.1016/j.infsof.2009.04.009>
24. Mossakowski, T., Krumnack, U., Maibaum, T.: What is a derived signature morphism? In: Codescu, M., Diaconescu, R., Țuțu, I. (eds.) *Recent Trends in Algebraic Development Techniques*. pp. 90–109. Springer (2015). [https://doi.org/10.1007/978-3-319-28114-8\\_6](https://doi.org/10.1007/978-3-319-28114-8_6)
25. Nickel, U., Niere, J., Zündorf, A.: The FUJABA environment. In: *Proceedings of the 22nd international conference on Software engineering*. pp. 742–745. ICSE '00, Association for Computing Machinery (2000). <https://doi.org/10.1145/337180.337620>
26. Pnueli, A.: The temporal logic of programs. In: *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*. pp. 46–57 (1977). <https://doi.org/10.1109/SFCS.1977.32>
27. Rose, L., Herrmannsdoerfer, M., Williams, J., Kolovos, D., Garcés, K., Paige, R., Polack, F.: A comparison of model migration tools. In: Petriu, D., Rouquette, N., Haugen, O. (eds.) *13th International Conference on Model Driven Engineering Languages and Systems (MODELS 2010)*. pp. 61–75. Springer (2010). [https://doi.org/10.1007/978-3-642-16145-2\\_5](https://doi.org/10.1007/978-3-642-16145-2_5)
28. Spanoudakis, G., Zisman, A.: Inconsistency management in software engineering: Survey and open research issues. In: *Handbook of Software Engineering and Knowledge Engineering*, pp. 329–380. World Scientific Publishing Company (2001). [https://doi.org/10.1142/9789812389718\\_0015](https://doi.org/10.1142/9789812389718_0015)
29. Stachowiak, H.: *Allgemeine Modelltheorie*. Springer (1973)
30. Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.: *Eclipse Modeling Framework*. Addison-Wesley Professional. (2008)
31. Stevens, P.: Bidirectional model transformations in QVT: semantic issues and open questions. *Software & Systems Modeling* **9**(1), 7–20 (2010). <https://doi.org/10.1007/s10270-008-0109-9>

32. Van Der Straeten, R., Jonckers, V., Mens, T.: A formal approach to model refactoring and model refinement. *Software & Systems Modeling* **6**(2), 139–162 (2007). <https://doi.org/10.1007/s10270-006-0025-9>
33. Xiong, Y., Liu, D., Hu, Z., Zhao, H., Takeichi, M., Mei, H.: Towards automatic model synchronization from model transformations. In: *Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering*. pp. 164–173. ASE '07, Association for Computing Machinery (2007). <https://doi.org/10.1145/1321631.1321657>