Formal Representation of SysML/KAOS Domain Model (Complete Version)

Steve Jeffrey Tueno Fotso[1,3], Amel Mammar[2], Régine Laleau[1], and Marc Frappier[3]

[1] Université Paris-Est Créteil, LACL, Créteil, France,
steve.tuenofotso@univ-paris-est.fr, laleau@u-pec.fr
[2] Télécom SudParis, SAMOVAR-CNRS, Evry, France,
amel.mammar@telecom-sudparis.eu
[3] Université de Sherbrooke, GRIL, Québec, Canada,
Marc.Frappier@usherbrooke.ca

28.11.2017

Abstract. Nowadays, the usefulness of a formal language for ensuring the consistency of requirements is well established. The work presented here is part of the definition of a formally-grounded, model-based requirements engineering method for critical and complex systems. Requirements are captured through the *SysML/KAOS* method and the targeted formal specification is written using the *Event-B* method. Firstly, an *Event-B* skeleton is produced from the goal hierarchy provided by the SysML/KAOS goal model. This skeleton is then completed in a second step by the Event-B specification obtained from system application domain properties that gives rise to the system structure. Considering that the domain is represented using ontologies through the *SysML/KAOS Domain Model* method, is it possible to automatically produce the structural part of system Event-B models ? This paper proposes a set of generic rules that translate SysML/KAOS domain ontologies into an Event-B specification. They are illustrated through a case study dealing with a landing gear system. Our proposition makes it possible to automatically obtain, from a representation of the system application domain in the form of ontologies, the structural part of the Event-B specification which will be used to formally validate the consistency of system requirements.

Keywords: *Event-B*, Domain Modeling, Ontologies, Requirements Engineering, *SysML/KAOS*, Formal Validation

1 Introduction

This article focuses on the development of systems in critical areas such as railway or aeronautics. The implementation of such systems, in view of their complexity, requires several validation steps, more or less formal[4], with regard to the current regulations. Our work is part of the *FORMOSE* project [4] which integrates industrial partners involved in the implementation of critical systems for which the regulation imposes formal validations. The contribution presented in this paper represents a straight continuation of our research work on the formal specification of systems whose requirements are captured with *SysML/KAOS* goal models. The *Event-B* method [1] has been choosen for the formal validation steps because it involves simple mathematical concepts and has a powerful refinement logic facilitating the separation of concerns. Furthermore, it is supported by many industrial tools. In [15], we have defined translation rules to produce an Event-B specification from *SysML/KAOS* goal models. Nevertheless, the generated Event-B specification does not contain the system state. This is why in [14], we have presented the use of ontologies and *UML* class and object diagrams for domain properties representation and have also introduced a first attempt to complete the Event-B model with specifications obtained from the translation of these domain representations. Unfortunately, the proposed approach raised several concerns such as the use of several modeling formalisms for the representation of domain knowledge or the disregard of variable entities. In addition, the proposed translation rules did not take into account several elements of the domain model such as data sets or predicates. We have therefore proposed in [21] a formalism for domain knowledge representation through ontologies. This paper is specifically concerned with establishing correspondence links between this new formalism called *SysML/KAOS Domain Modeling* and Event-B. The proposed approach allows a high-level modeling of domain properties

[4] through formal methods

by encapsulating the difficulties inherent in the manipulation of formal specifications. This facilitates system constraining and enables the expression of more precise and complete properties. The approach also allows further reuse and separation of concerns.

The remainder of this paper is structured as follows: Section 2 briefly describes our abstraction of the *Event-B* specification language, the SysML/KAOS requirements engineering method, the formalization in Event-B of SysML/KAOS goal models and the SysML/KAOS domain modeling formalism. Follows a presentation, in Section 3, of the relevant state of the art on the formalization of domain knowledge representations. In Section 4, we describe and illustrate our matching rules between domain models and Event-B specifications. Finally, Section 5 reports our conclusions and discusses our future work.

2 Formalism Overviews

2.1 Event-B

Event-B is an industrial-strength formal method defined by *J. R. Abrial* in 2010 for *system modeling* [1]. It is used to prove the preservation of safety invariants about a system. *Event-B* is mostly used for the modeling of closed systems: the modeling of the system is accompanied by that of its environment and of all interactions likely to occur between them.

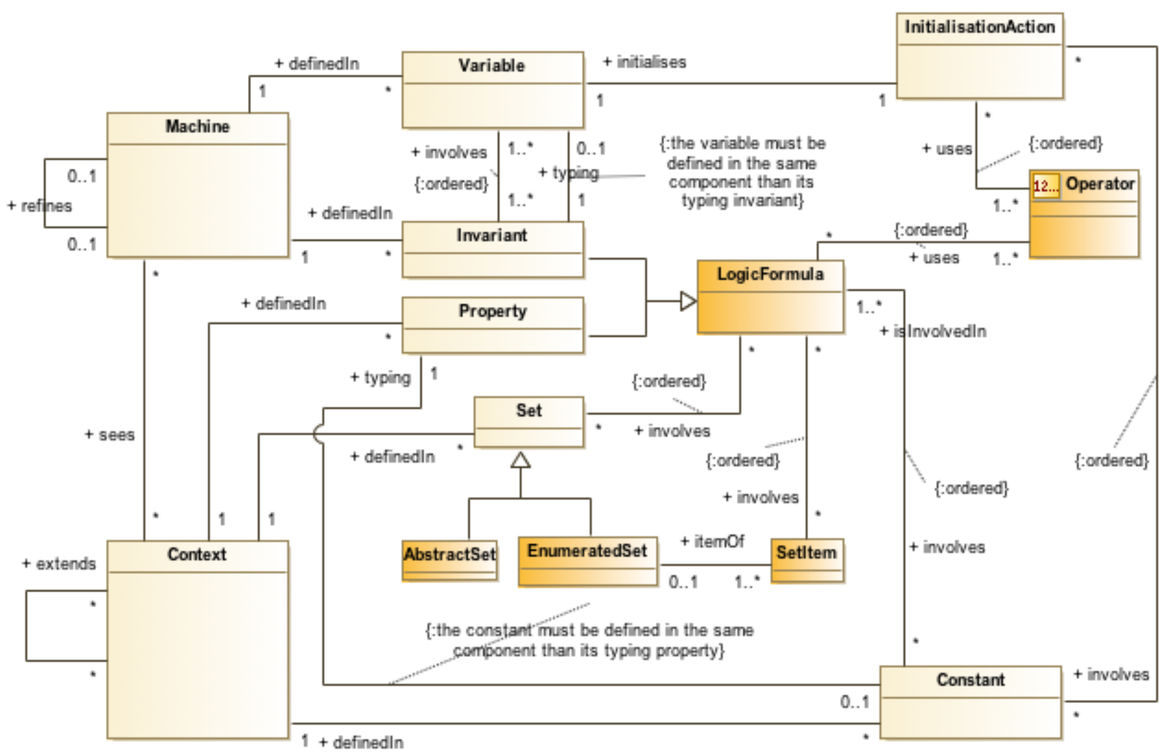

Fig. 1. Our abstraction of the Event-B specification language

Figure 1 is an excerpt from our abstraction of the *Event-B* specification language restricted and adjusted to fulfill the expression of our formalization rules. We have represented in orange some categories that do not appear explicitly in Event-B specifications, but which will be useful to better describe our formalization rules. An *Event-B* model includes a static part called ***Context*** and a dynamic part called ***Machine***. The ***context*** contains the definitions of abstract and enumerated sets, constants and properties. An enumerated set is constructed by specifying its items which are instances of SetItem. The system state is represented in the ***machine*** using variables constrained through invariants and initialised through initialisation actions. Moreover, a machine can see contexts. Properties and invariants can be categorised as instances of LogicFormula. An instance of LogicFormula consists of a certain number of operators applied, according to their order of appearance, on the operands that may be variables, constants, sets or set items, following their associated order of appearance. An instance of InitialisationAction references the operator and the operands of the assignment. We describe here some operators and their actions :

- ***Inclusion_ OP*** is used to assert that the first operand is a subset of the second operand : $(Inclusion\_OP, [op_1, op_2]) \Leftrightarrow op_1 \subset op_2$.
- ***Belonging_ OP*** is used to assert that the first operand is an element of the second operand : $(Belonging\_OP, [op_1, op_2]) \Leftrightarrow op_1 \in op_2$.
- ***RelationSet_ OP*** is used to construct the set of relations between two operands : $(RelationSet\_OP, [op_1, op_2, op_3]) \Leftrightarrow op_1 = op_2 \leftrightarrow op_3$.
- ***FunctionSet_ OP*** is used to construct the set of functional relations between two operands : $(FunctionSet\_OP, [op_1, op_2, op_3]) \Leftrightarrow op_1 = op_2 \longrightarrow op_3$.
- ***Maplet_ OP*** is used to construct a maplet having the operands as antecedent and image : $(Maplet\_OP, [op_1, op_2, op_3]) \Leftrightarrow op_1 = op_2 \mapsto op_3$.
- ***RelationComposition_ OP*** is used to assert that the first operand is the result of the composition of the second operand by the third operand : $(RelationComposition\_OP, [op_1, op_2, op_3]) \Leftrightarrow op_1 = op_2 \circ op_3$.
- ***Equal2SetOf_ OP*** is used to define the elements constituting a set : $(Equal2SetOf\_OP, [op_1, op_2, ..., op_n]) \Leftrightarrow op_1 = \{op_2, ..., op_n\}$.
- ***Inversion_ OP*** is used to assert that the first operand is the inverse of the second operand : $(Inversion\_OP, [op_1, op_2]) \Leftrightarrow op_1 = op_2^{-1}$.
- ***Equality_ OP*** is used to assert that the first operand is equal to the second operand : $(Equality\_OP, [op_1, op_2]) \Leftrightarrow op_1 = op_2$.
- ***BecomeEqual2SetOf_ OP*** is used to initialize a variable as a set of elements : $(BecomeEqual2SetOf\_OP, [va, op_2, ..., op_n]) \Leftrightarrow va := \{op_2, ..., op_n\}$.
- ***BecomeEqual2EmptySet_ OP*** is used to initialize a variable as an empty set : $(BecomeEqual2EmptySet\_OP, [va]) \Leftrightarrow va := \emptyset$.

The system specification can be constructed using stepwise refinement. A machine can refine another one, adding new events, reducing nondeterminacy of existing events, introducing new state variables, or replacing abstract variables by more concrete variables. Furthermore, a context can extend another one in order to access the elements defined in it and to reuse them for new constructions.

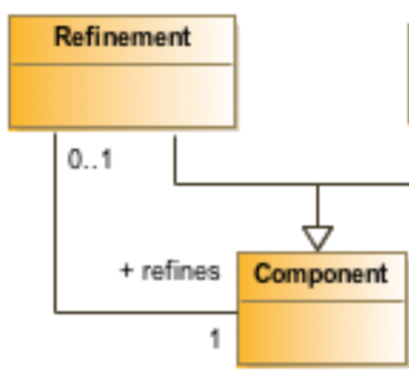

Fig. 2. B System Components

In the rest of this paper, we will illustrate our formal models using *B System*, an *Event-B* variant proposed by *ClearSy*, an industrial partner in the *FORMOSE* project, in its integrated development environment *Atelier B* [7]. A *B System* specification considers the notion of Component to specify machines and contexts, knowing that a component can be a system or a refinement (figure 2). Although it is advisable to always isolate the static and dynamic parts of the *B System* formal model, it is possible to define the two parts within the same component, for simplification purposes. In the following sections, our *B System* models will be presented using this facility.

2.2 SysML/KAOS Requirements Engineering Method

Requirements engineering focuses on defining and handling requirements. These and all related activities, in order to be carried out, require the choice of an adequate means for requirements representation. The *KAOS* method [13,14], proposes to represent the requirements in the form of goals, which can be *functional* or *non-functional*, through five sub-models of which the two main ones are : **the object model** which uses the *UML* class diagram for the representation of domain vocabulary and **the goal model** for the determination of requirements to be satisfied by the system and of expectations with regard to the environment through a goals hierarchy. *KAOS* proposes a structured approach to obtaining the requirements based on expectations formulated by stakeholders. Unfortunately, it offers no mechanism to maintain a strong traceability between those requirements and deliverables associated with system design and implementation, making it difficult to validate them against the needs formulated.

The *SysML UML profile* has been specially designed by the Object Management Group (OMG) for the analysis and specification of complex systems and allows for the capturing of requirements and the maintaining of traceability links between those requirements and design diagrams resulting from the system design phase. Unfortunately, OMG has not defined a formal semantics and an unambiguous syntax for requirements specification. *SysML/KAOS* [10] therefore proposes to extend the *SysML* metamodel with a set of concepts allowing to represent requirements in *SysML* models as *KAOS* goals.

Figure 3 is an excerpt from the landing gear system [6] goal diagram focused on the purpose of landing gear expansion. We assume that each aircraft has one landing gear system which is equipped with three landing sets which can be each extended or retracted. We also assume that in the initial state, there is one landing gear named *LG1* which is extended and is associated to one handle named *HD1* which is down and to landing sets *LS1*, *LS2* and *LS3* which are all extended.

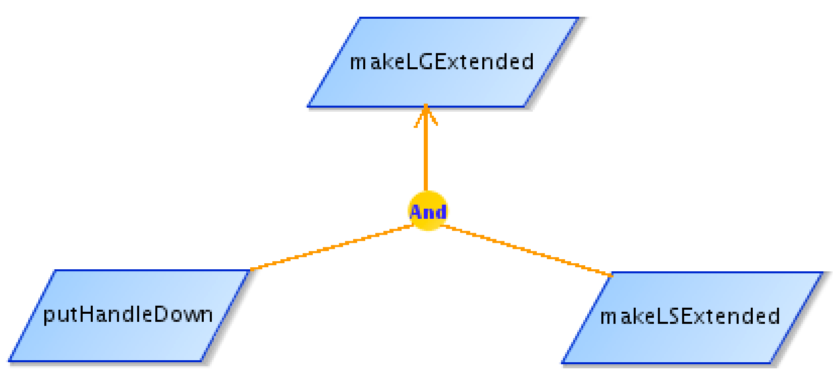

Fig. 3. Excerpt from the landing gear system goal diagram

To achieve the root goal, which is the extension of the landing gear (**makeLGExtended**), the handle must be put down (**putHandleDown**) and landing gear sets must be extended (**makeLSExtended**).

2.3 From SysML/KAOS Goal Model to Event-B

The matching between *SysML/KAOS* modeling and *Event-B* specifications is the focus of the work done by [15]. Each layer of abstraction of the goal diagram gives rise to an *Event-B* machine, each goal of the layer giving rise to an event. The refinement links are materialized within the *Event-B* specification through a set of proof obligations and refinement links between machines and between events. Figure 4 represents the

B System specifications associated with the most abstract layer of the *SysML/KAOS* goal diagram of the Landing Gear System illustrated through Figure 3.

SYSTEM
    LandingGearSystem
SETS
CONSTANTS
PROPERTIES
VARIABLES
INVARIANT
INITIALISATION
EVENTS
    makeLGExtended=
    BEGIN /* extension of the landing gear */
    END
END

Fig. 4. Formalization of the root level of the Landing Gear System goal model

As we can see, the state of the system and the body of events must be manually completed. The state of a system is composed of variables, constrained by an invariant, and constants, constrained by properties. The objective of our study is to automatically derive this state in the *Event-B* model starting from *SysML/KAOS* domain models.

2.4 SysML/KAOS Domain Modeling

We present, through Figures 5 and 6 the metamodel associated with the *SysML/KAOS* domain modeling approach [21] which is an ontology modeling formalism for the modeling of domain knowledge in the framework of the *SysML/KAOS* requirements engineering method.

Figure 7 represents the *SysML/KAOS* domain model associated to the root level of the landing gear system goal model of Figure 3, and Figure 8 represents the first refinement level. They are illustrated using the syntax proposed by *OWLGred* [22] and, for readability purposes, we have decided to remove optional characteristics representation. It should be noted that the *individualOf* association is illustrated by *OWLGred* within the figures as a stereotyped link with the tag *«instanceOf»*. The domain model associated to the goal diagram root level is named ***lg_system_ref_0*** and the one associated to the first refinement level is named ***lg_system_ref_1***.

Each domain model is associated with a level of refinement of the *SysML/KAOS* goal diagram and is likely to have as its parent, through the *parent* association, another domain model. This allows the child domain model to access and extend some elements defined in the parent domain model. For example, in *lg_system_ref_1* (Fig. 8), elements defined in *lg_system_ref_0* (Fig. 7) are imported and reused.

A *concept* (instance of metaclass *Concept* of Figure 5) represents a group of individuals sharing common characteristics. It can be declared *variable* (isVariable=true) when the set of its individuals is likely to be updated through addition or deletion of individuals. Otherwise, it is considered to be *constant* (isVariable=false). A concept may be associated with another, known as its parent concept, through the parentConcept association, from which it inherits properties. For example, in *lg_system_ref_0* (Fig. 7), a *landing gear* is modeled as an instance of Concept named *"LandingGear"*. Since it is impossible to dynamically add or remove a landing gear, the attribute isVariable of ***LandingGear*** is set to *false*. *LG1* is modeled as an instance of Individual (Fig. 5) named *"LG1"* individual of ***LandingGear***.

Instances of Relation are used to capture links between concepts, and instances of Attribute capture links between concepts and data sets, knowing that data sets (instances of DataSet) are used to group data values (instances of DataValue) having the same type. The most basic way to build an instance of DataSet is by listing its elements. This can be done through the DataSet specialization called EnumeratedDataSet. A relation or an attribute can be declared *variable* if the list of maplets related to it is likely to change over time. Otherwise, it is considered to be *constant*. Each instance of DomainCardinality (respectively RangeCardinality) makes it possible to define, for an instance of Relation *re*, the minimum and maximum limits of the number of instances

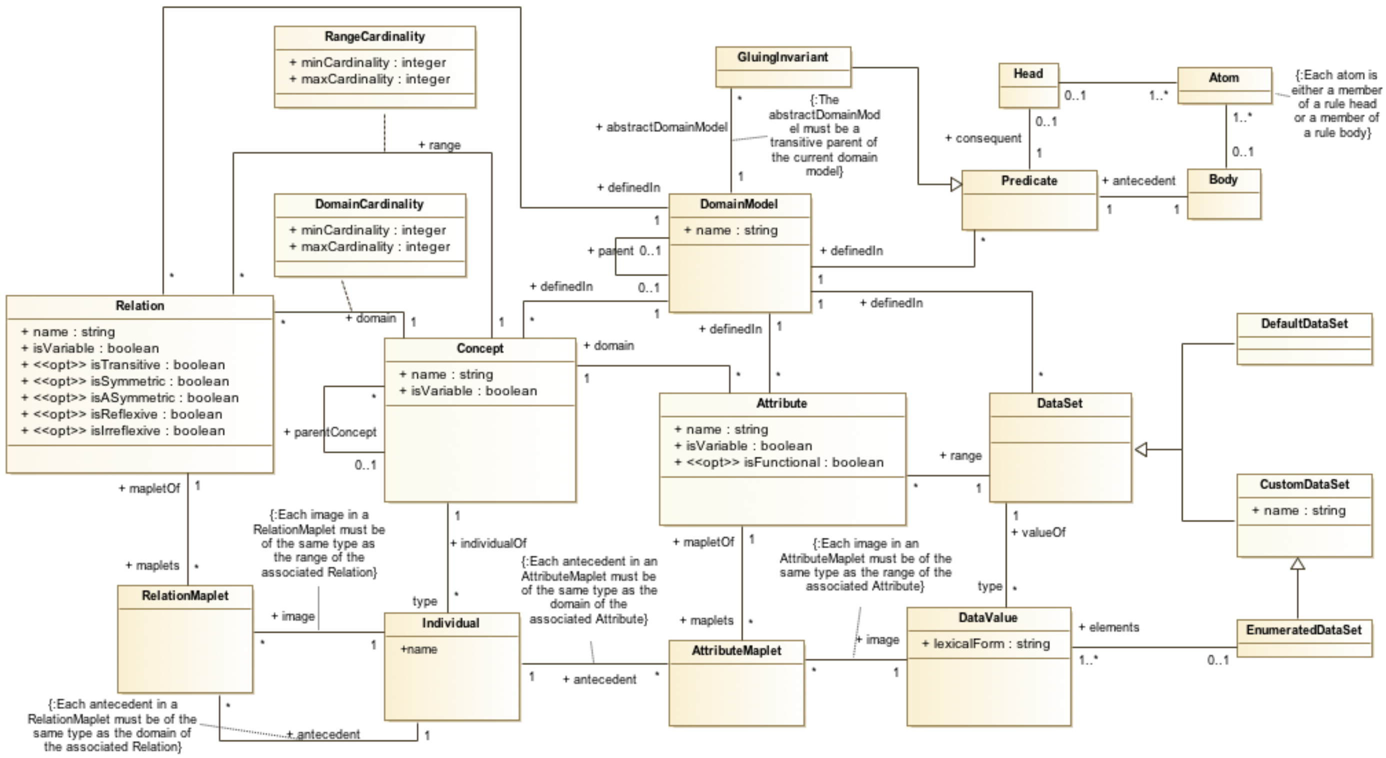

Fig. 5. Metamodel associated with SysML/KAOS domain modeling

of Individual, having the domain (respectively range) of *re* as *type*, that can be put in relation with one instance of Individual, having the range (respectively domain) of *re* as *type*. The following constraint is associated with these limits : $(minCardinality \geq 0) \wedge (maxCardinality = * \vee maxCardinality \geq minCardinality)$, knowing that if $maxCardinality = *$, then the maximum limit is *infinity*. Instances of RelationMaplet are used to define associations between instances of Individual through instances of Relation. In an identical manner, instances of AttributeMaplet are used to define associations between instances of Individual and instances of DataValue through instances of Attribute. Optional characteristics can be specified for a relation : *transitive* (isTransitive, default *false*), *symmetrical* (isSymmetric, default *false*), *asymmetrical* (isASymmetric, default *false*), *reflexive* (isReflexive, default *false*) or *irreflexive* (isIrreflexive, default *false*). Moreover, an attribute can be *functional* (isFunctional, default *true*). For example, in *lg_system_ref_0* (Fig. 7), the possible states of a landing gear is modeled as an instance of Attribute named *"landingGearState"*, having ***LandingGear*** as domain and as range an instance of EnumeratedDataSet containing two instances of DataValue of type STRING: ***"lg_extended"*** for the extended state and ***"lg_retracted"*** for the retracted state. Since it is possible to dynamically change a landing gear state, its isVariable attribute is set to *true*.

The notion of Predicate is used to represent constraints between different elements of the domain model in the form of *Horn clauses*: each predicate has a body which represents its *antecedent* and a head which represents its *consequent*, body and head designating conjunctions of atoms (Fig. 6). A *typing atom* is used to define the type of a term : ConceptAtom for individuals and DataSetAtom for data values. An *association atom* is used to define associations between terms : RelationshipAtom for the connection of two terms through a *relation*, AttributeAtom for the connection of two terms through an *attribute* and DataFunctionAtom for the connection of terms through a *data function*. A *comparison atom* is used to define comparison relationships between terms : EqualityAtom for equality and InequalityAtom for difference. Built in atoms are some specialized atoms, characterized by identifiers captured through the AtomType enumeration, and used for the representation of particular constraints between several terms. For example, an arithmetic constraint between several integer data values.

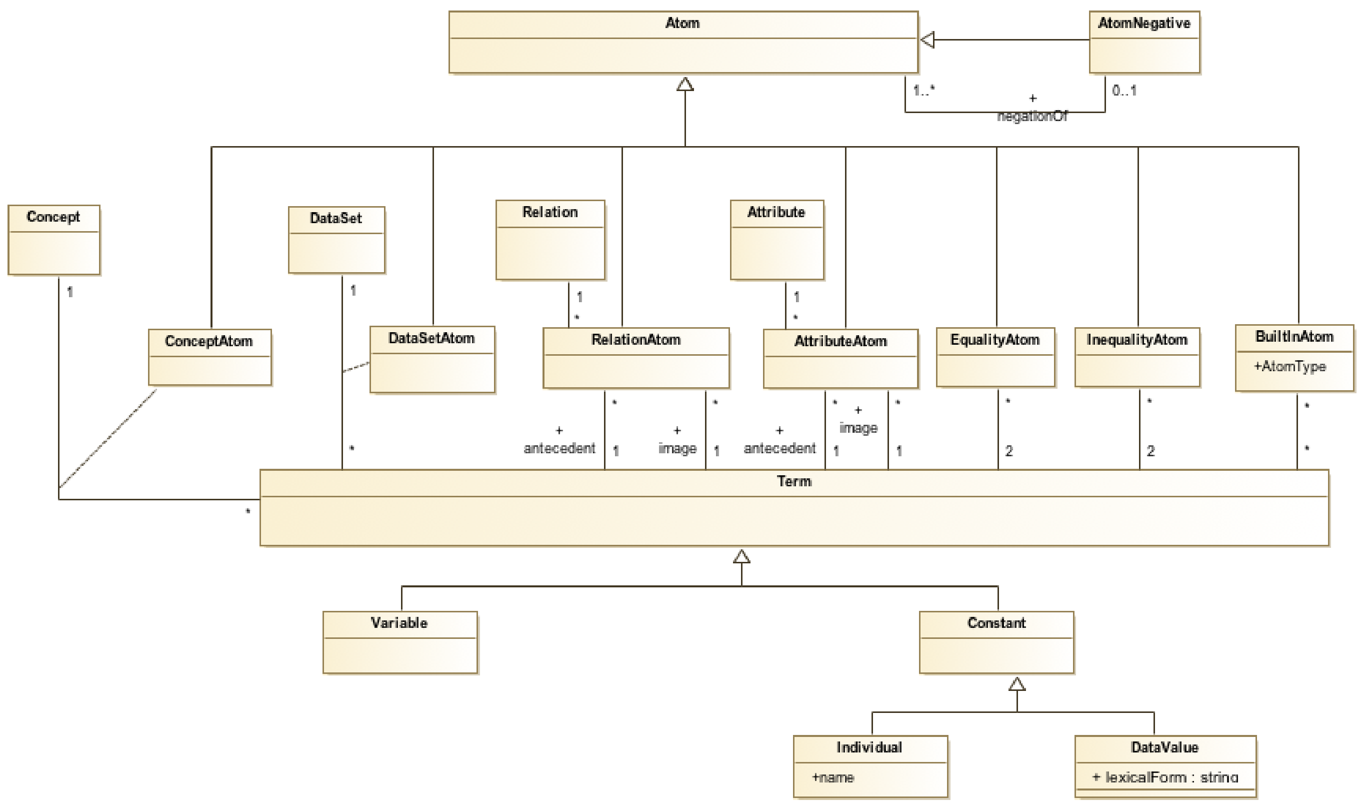

Fig. 6. Extension of the metamodel associated with SysML/KAOS domain modeling for atom specification

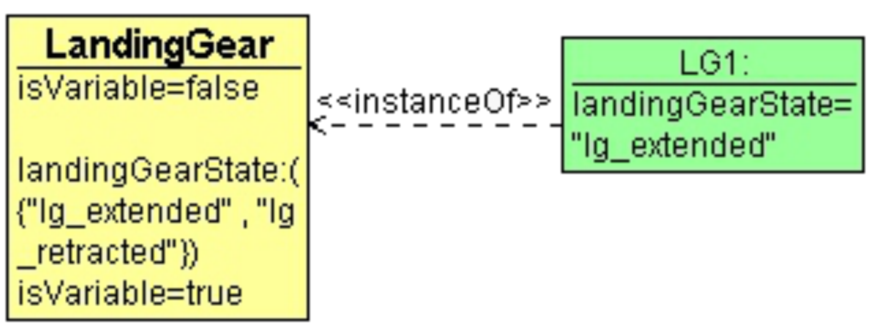

Fig. 7. ***lg_system_ref_0***: ontology associated to the root level of the landing gear goal model

GluingInvariant, specialization of Predicate, is used to represent links between variables and constants defined within a domain model and those appearing in more abstract domain models, transitively linked to it through the *parent* association. Gluing invariants are extremely important because they capture relationships between abstract and concrete data during refinement which are used to discharge proof obligations. The following gluing invariant is associated with our case study: if there is at least one landing set having the retracted state, then the state of LG1 is retracted

$$landingGearState(LG1, "lg\_retracted") \leftarrow LandingSet(?ls) \wedge landingSetState(?ls, "ls\_retracted") \quad \text{(inv1)}$$

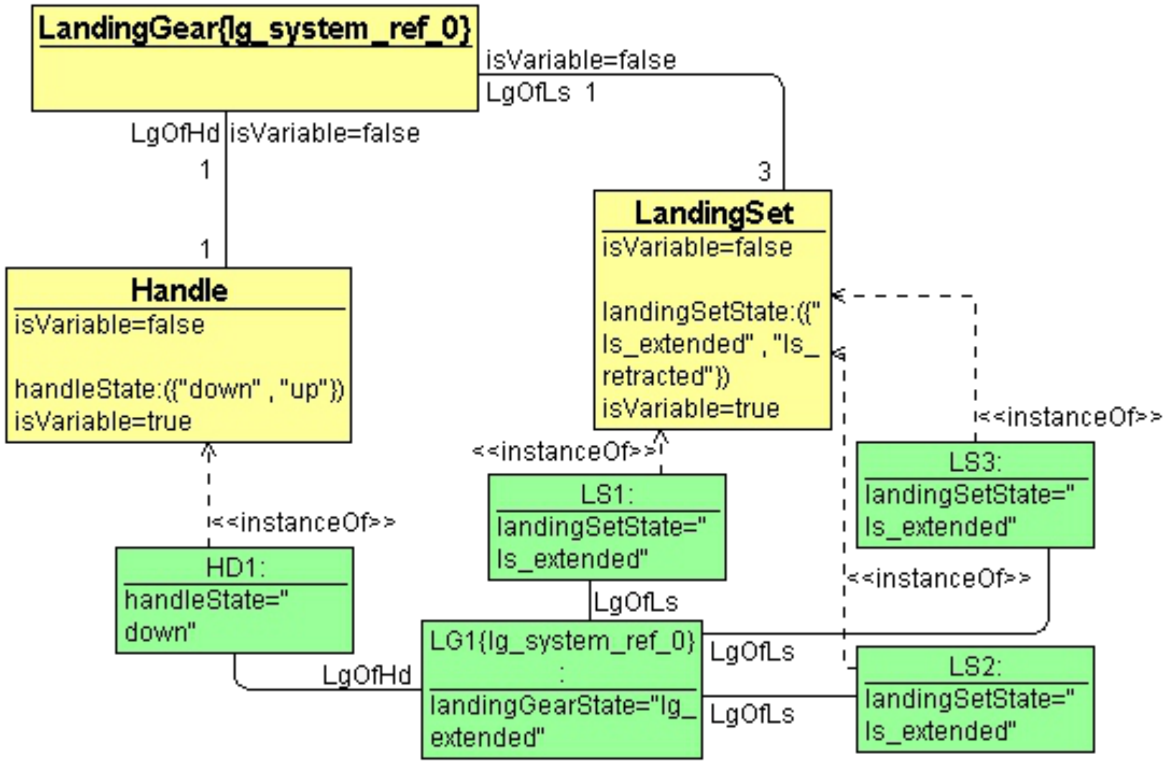

Fig. 8. ***lg_system_ref_1*** : ontology associated to the first level of refinement of the landing gear goal model

3 Existing Approaches for the Formalization of Domain Models

In [5], domain models consist of entities and operations which can be atomic or composite. Atomic entities correspond to states of the formal model. Composite entities correspond to sets, groups, lists or associations of entities. Furthermore, operations are translated into state-changing actions, composite operations corresponding to composition of actions. In [23], an approach is proposed for the automatic extraction of domain knowledge, as *OWL* ontologies, from *Z/Object-Z (OZ)* models [8] : OZ types and classes are transformed into OWL classes. Relations and functions are transformed into OWL properties, with the *cardinality* restricted to *1* for total functions and the *maxCardinality* restricted to *1* for partial functions. OZ constants are translated into OWL individuals. Rules are also proposed for subsets and state schemas. Unfortunately, the approach is only interested in static domain knowledge and it does not propose any rule regarding predicates. Furthermore, refinement links between models are not handled. A similar approach is proposed in [9], for the extraction of *DAML* ontologies [11] from *Z* models.

An approach for generating an *Event-B* specification from an *OWL* ontology [18] is provided in [3]. The proposed mapping requires the generation of an *ACE (Attempto Controlled English)* version of the *OWL* ontology which serves as the basis for the development of the *Event-B* specification. This is done through a step called *OWL verbalization*. The verbalization method, proposed by [3], transforms *OWL* instances into capitalized proper names, classes into common names, and properties into active and passive verbs. Once the verbalization process has been completed, [3] proposes a set of rules for obtaining the Event-B specification: classes are translated as *Event-B* sets, properties are translated as relations, etc. In addition, [3] proposes rules for the *Event-B* representation of property characteristics and associations between classes or properties. Unfortunately, the proposal makes no distinction between constant and variable : It does not specify when it is necessary to use constants or variables, when it is necessary to express an ontology rule as an invariant or as an axiom. Moreover, the proposal imposes a two-step sequence for the transition from an OWL ontology to an Event-B model, the first step requiring the ontology to be constructed in English. Finally, the approach does not propose anything regarding the referencing from an ontology into another one.

In [17], domain is modeled by defining agents, business entities and relations between them. The paper proposes rules for mapping domain models so designed in *Event-B* specifications : agents are transformed into machines, business entities are transformed into sets, and relations are transformed into *Event-B* variable relations. These rules are certainly sufficient for domain models of interest for [17], but they are very far from covering the extent of *SysML/KAOS* domain modeling formalism.

In [2], domain properties are described through data-oriented requirements for concepts, attributes and associations and through constraint-oriented requirements for axioms. Possible states of a *variable* element are represented using UML state machines. Concepts, attributes and associations arising from data-oriented requirements are modeled as UML class diagrams and translated to Event-B using *UML-B* [19] : nouns and attributes are represented as UML classes and relationships between nouns are represented as UML associations. *UML-B* is also used for the translation of state machines to Event-B variables, invariants and events. Unfortunately, constraints arising from constraint-oriented requirements are modeled using a semi-formal language called *Structured English*, following a method similar to the *Verbalization* approach described in [3] and manually translated to Event-B. Moreover, it is impossible to rely solely on the representation of an element of the class diagram to know if its state is likely to change dynamically. The consequence being that in an Event-B model, the same element can appear as a set, a constant or a variable and its properties are likely to appear both in the *PROPERTIES* and in the *INVARIANT* clauses.

Some rules for passing from an *OWL* ontology representing a domain model to *Event-B* specifications are proposed through a case study in [14]. This case study reveals that each ontology class, having no instance, is modeled as an *Event-B* abstract set. The others are modeled as an enumerated set. Finally, each object property between two classes is modeled as a constant defines as a relation. These rules allow the generation of a first version of an *Event-B* specification from a domain model ontology. Unfortunately, the case study does not address several concerns. For example, object properties are always modeled as constants, despite the fact that they may be variable. Moreover, the case study does not provide any rule for some domain model elements such as datasets or predicates. In the remainder of this paper, we propose to enrich this proposal for a complete mapping of *SysML/KAOS* domain models with *Event-B* specifications.

4 SysML/KAOS Domain Model Formalization

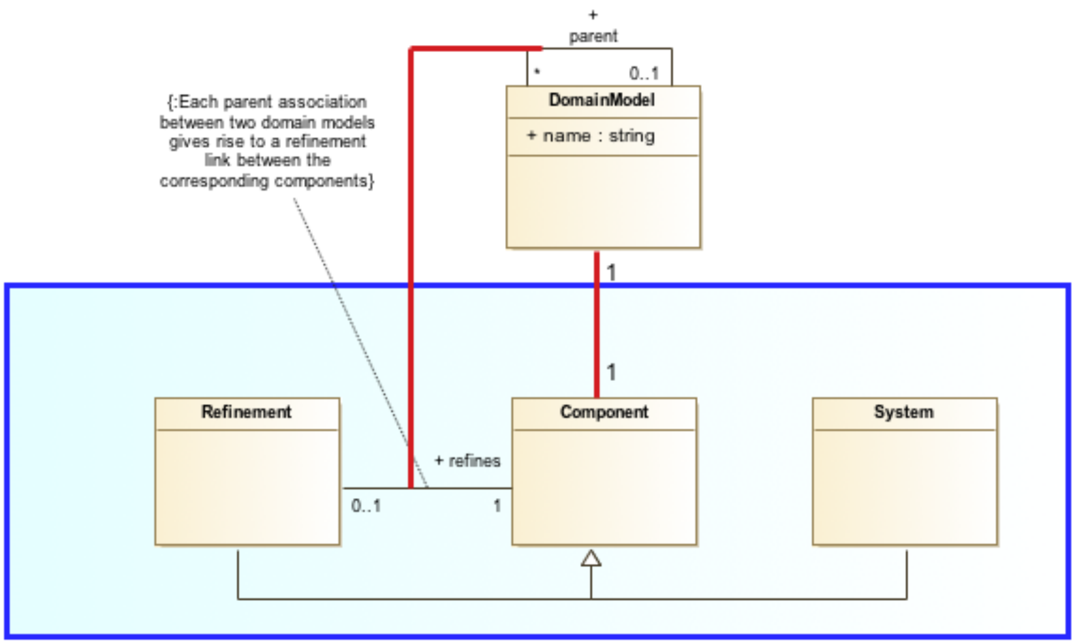

Fig. 9. Correspondence to B System Components

Figures 9, 10 , 11 and 12 are schematizations of correspondence links between domain models and Event-B formal models. Red links represent correspondence links, the part inside the blue rectangle representing the portion of the Event-B metamodel under consideration.

In the following, we describe a set of rules that allow to obtain an *Event-B* specification from domain models associated with refinement levels of a *SysML/KAOS* goal model. They are illustrated using the **B syntax** :

- Regarding the representation of metamodels, we have followed the rules proposed by [19] for the translation of *UML* class diagrams to *B* specifications: for example, classes which are not subclasses give rise to abstract sets, each class gives rise to a variable typed as a subset and containing its instances and each association or property gives rise to a variable typed as a relation. For example, DomainModel, Concept, Relation, Attribute and DataSet of the SysML/KAOS domain metamodel (***Domain_ Metamodel_ Context***) and Component, Set, LogicFormula and Variable of the Event-B metamodel (***Event_ B_ Metamodel_ Context***) give rise to abstract sets representing all their possible instances. Variables appear to capture, for each class, all the currently defined instances. Variables are also used to represent attributes and associations such as ParentConcept, Relation_isVariable, Attribute_isFunctional of the SysML/KAOS domain metamodel and Refines of the Event-B metamodel (***event_b_specs_from_ontologies*** and ***event_b_specs_from_ontologies_ref_1***). In case of ambiguity as to the nomenclature of an element, its name is prefixed and sufixed by that of the class to which it is attached.
- Correspondence links between classes are represented through variables typed as partial injections having the ***B*** representation of the first class as domain and the ***B*** representation of the second class as range. For example, correspondence links between instances of Concept and instances of AbstractSet illustrated through figure 10, are captured through a variable typed as a partial injective function between Concept and AbstractSet : $Concept\_corresp\_AbstractSet \in Concept \rightarrowtail\!\!\!\!\!\!\!+\; AbstractSet$ (***event_b_specs_from_ontologies_ref_1***).
- Each rule is represented as an event by following the correspondence links.

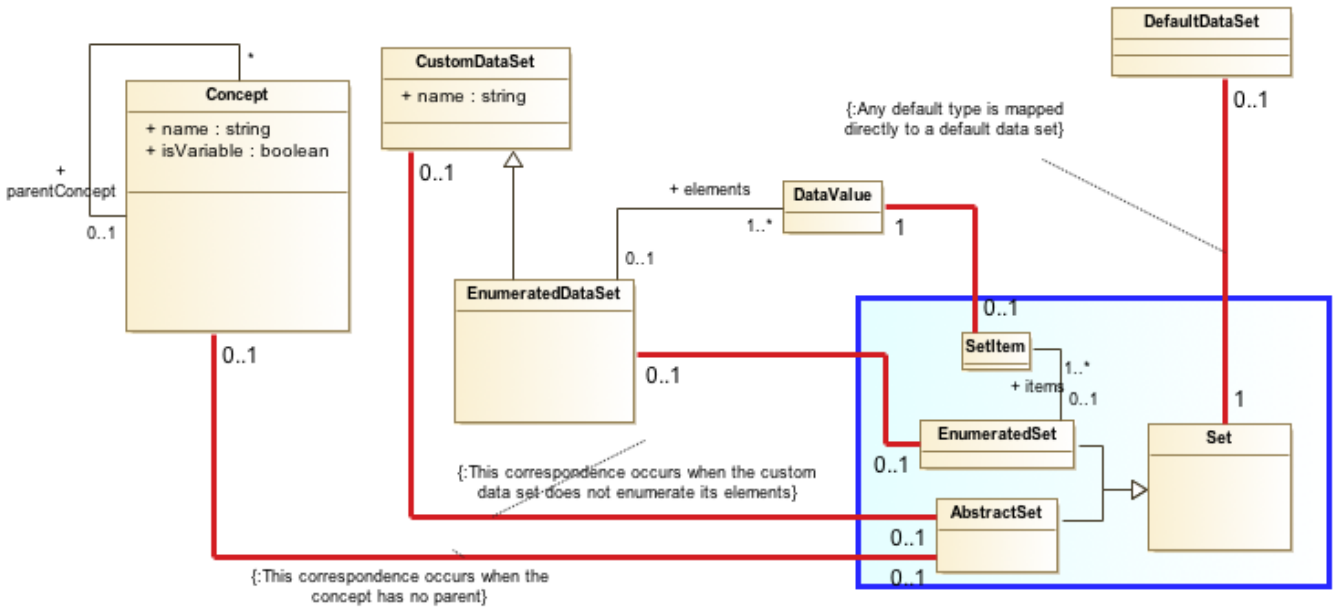

Fig. 10. Correspondence to Sets

– Whereas no additional precision is given, we consider that all *Event-B* content associated with a refinement level is defined within a single component (SYSTEM/REFINEMENT) : it is always possible to separate it into two parts: the context for the static part (SETS, CONSTANTS and PROPERTIES) and the machine for the dynamic part (VARIABLES, INVARIANT, INITIALIZATION and EVENTS).

Figures 13 and 14 represents respectively the *B System* specifications associated with the root level of the landing gear system domain model illustrated through Figure 7 and that associated with the first refinement level domain model illustrated through Figure 8.

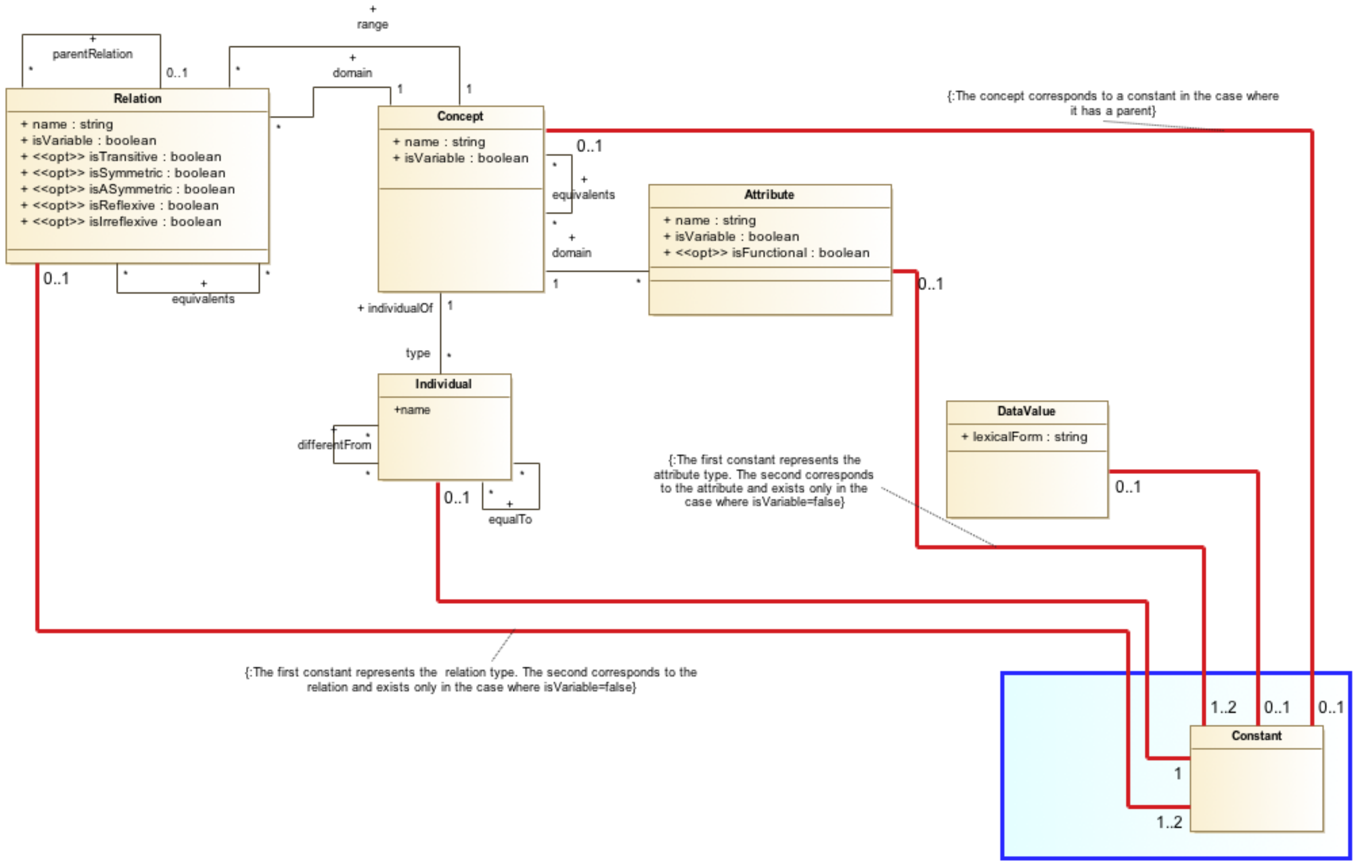

Fig. 11. Correspondence to Constants

4.1 Formalization of SysML/KAOS Domain Modeling and Event-B Formalisms

Event_B_Metamodel_Context and ***Domain_Metamodel_Context*** represent respectively the context associated to our abstraction of the *Event-B* specification language and that associated to the SysML/KAOS Domain Metamodel. ***event_b_specs_from_ontologies*** and ***event_b_specs_from_ontologies_ref_1*** represent the corresponding variables and the associated invariants.

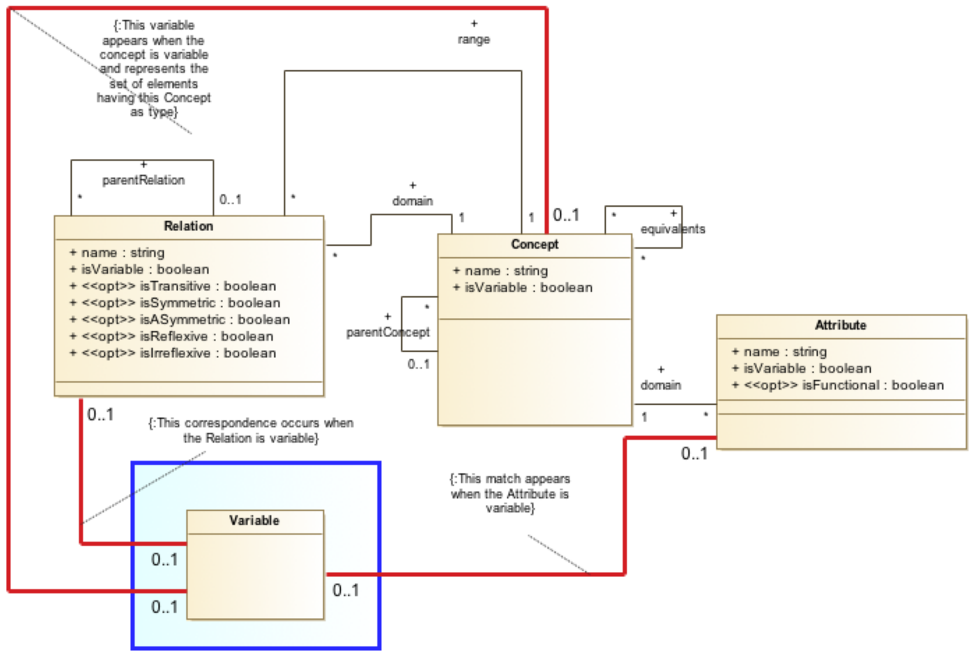

Fig. 12. Correspondence to Variables

SYSTEM   *lg_system_ref_0*
SETS   *LandingGear; DataSet_1= {lg_extended, lg_retracted}*
CONSTANTS   *T_landingGearState, LG1*
PROPERTIES
(0.1) $LG1 \in LandingGear$
(0.2) $\wedge\ LandingGear = \{LG1\}$
(0.3) $\wedge\ T\_landingGearState = LandingGear \longrightarrow DataSet\_1$
VARIABLES   *landingGearState*
INVARIANT
(0.4) $landingGearState \in T\_landingGearState$
INITIALISATION
(0.5) $landingGearState := \{LG1 \mapsto lg\_extended\ \}$
EVENTS
•••
END

Fig. 13. *Formalization of the Root Level of the Landing Gear System Domain Model*

CONTEXT Domain_Metamodel_Context
SETS
DomainModel_Set
Relation_Set
Concept_Set
Relation_Maplet_Set
Individual_Set
Attribute_Maplet_Set
Attribute_Set
DataValue_Set
DataSet_Set
RelationCharacteristics_Set
CONSTANTS
_NATURAL

REFINEMENT   *lg_system_ref_1*
REFINES   *lg_system_ref_0*
SETS   *Handle; LandingSet; DataSet_2={ls_extended, ls_retracted}; DataSet_3={down, up}*
CONSTANTS   *T_LgOfHd, LgOfHd, T_LgOfLs, LgOfLs, T_landingSetState, T_handleState, HD1, LS1, LS2, LS3*
PROPERTIES
(1.1) $HD1 \in Handle$
(1.2) $\wedge$ $Handle=\{HD1\}$
(1.3) $\wedge$ $LS1 \in LandingSet$
(1.4) $\wedge$ $LS2 \in LandingSet$
(1.5) $\wedge$ $LS3 \in LandingSet$
(1.6) $\wedge$ $LandingSet=\{LS1, LS2, LS3\}$
(1.7) $\wedge$ $T\_LgOfHd = Handle \leftrightarrow LandingGear$
(1.8) $\wedge$ $LgOfHd \in T\_LgOfHd$
(1.9) $\wedge$ $\forall\ xx.(xx \in Handle \Rightarrow card(LgOfHd[\{xx\}])=1)$
(1.10) $\wedge$ $\forall xx.(xx \in LandingGear \Rightarrow card(LgOfHd^{-1}[\{xx\}])=1)$
(1.11) $\wedge$ $LgOfHd = \{HD1 \mapsto LG1\}$
(1.12) $\wedge$ $T\_LgOfLs = LandingSet \leftrightarrow LandingGear$
(1.13) $\wedge$ $LgOfLs \in T\_LgOfLs$
(1.14) $\wedge$ $\forall xx.(xx \in LandingSet \Rightarrow card(LgOfLs[\{xx\}])=1)$
(1.15) $\wedge$ $\forall xx.(xx \in LandingGear \Rightarrow card(LgOfLs^{-1}[\{xx\}])=3)$
(1.16) $\wedge$ $LgOfLs = \{LS1 \mapsto LG1, LS2 \mapsto LG1, LS3 \mapsto LG1\}$
(1.17) $\wedge$ $T\_landingSetState = LandingSet \longrightarrow DataSet\_2$
(1.18) $\wedge$ $T\_handleState = Handle \longrightarrow DataSet\_3$
VARIABLES   *landingSetState, handleState*
INVARIANT
(1.19) $landingSetState \in T\_landingSetState$
(1.20) $\wedge$ $handleState \in T\_handleState$
(1.21) $\wedge$ $\forall ls.(ls \in LandingSet \wedge landingSetState(ls, ls\_extended) \Rightarrow$
$landingGearState(LG1, lg\_extended))$
INITIALISATION
(1.22) $landingSetState := \{LS1 \mapsto ls\_extended, LS2 \mapsto ls\_extended, LS3 \mapsto ls\_extended\}$
(1.23) $||$ $handleState := \{HD1 \mapsto down\}$
EVENTS
•••
END

Fig. 14. *Formalization of the First Refinement Level of the Landing Gear System Domain Model*

_INTEGER
_FLOAT
_BOOL
_STRING
isTransitive
isSymmetric

AXIOMS

axiom1: $finite(DataValue\_Set)$
axiom2: $\{\_NATURAL, \_INTEGER, \_FLOAT, \_BOOL, \_STRING\} \subseteq DataSet\_Set$
axiom3: $partition(\{\_NATURAL, \_INTEGER, \_FLOAT, \_BOOL, \_STRING\}, \{\_NATURAL\}, \{\_INTEGER\}, \{\_FLOAT\}, \{\_BOOL\}, \{\_STRING\})$
axiom4: $partition(RelationCharacteristics\_Set, \{isTransitive\}, \{isSymmetric\})$

END

CONTEXT EventB_Metamodel_Context

SETS

Component_Set
Variable_Set
Constant_Set
Set_Set
SetItem_Set
LogicFormula_Set
the subset of logical formulas that can directly be expressed within the specification, without the need for an explicit constructor, will not be contained in this set. This is for example the case of equality between elements.

Operator
InitialisationAction_Set

CONSTANTS

B_NATURAL
B_INTEGER
B_FLOAT
B_BOOL
B_STRING
Inclusion_OP
Belonging_OP
BecomeEqual2SetOf_OP
RelationSet_OP
FunctionSet_OP
Maplet_OP
Equal2SetOf_OP
BecomeEqual2EmptySet_OP
RelationComposition_OP
Inversion_OP
Equality_OP

AXIOMS

axiom1: $finite(SetItem\_Set)$
axiom2: $\{B\_NATURAL, B\_INTEGER, B\_FLOAT, B\_BOOL, B\_STRING\} \subseteq Set\_Set$
axiom3: $partition(\{B\_NATURAL, B\_INTEGER, B\_FLOAT, B\_BOOL, B\_STRING\}, \{B\_NATURAL\}, \{B\_INTEGER\}, \{B\_FLOAT\}, \{B\_BOOL\}, \{B\_STRING\})$
axiom4: $partition(Operator, \{Inclusion\_OP\}, \{Belonging\_OP\}, \{BecomeEqual2SetOf\_OP\}, \{RelationSet\_OP\}, \{Maplet\_OP\}, \{Equal2SetOf\_OP\}, \{BecomeEqual2EmptySet\_OP\}, \{FunctionSet\_OP\}, \{RelationComposition\_OP\}, \{Inversion\_OP\}, \{Equality\_OP\})$

END

MACHINE event_b_specs_from_ontologies
SEES EventB_Metamodel_Context,Domain_Metamodel_Context
VARIABLES

Component
System
Refinement Event-B associations
Refinement_refines_Component Domain Model sets
DomainModel Domain Model associations
DomainModel_parent_DomainModel correspondences
DomainModel_corresp_Component

INVARIANTS

inv0_1: $Component \subseteq Component\_Set$
inv0_2: $partition(Component, System, Refinement)$
Domain Model
inv0_3: $DomainModel \subseteq DomainModel\_Set$
inv0_4: $DomainModel\_parent\_DomainModel \in DomainModel \twoheadrightarrow\!\!\!\!\!\rightarrowtail DomainModel$
inv0_5: $DomainModel\_corresp\_Component \in DomainModel \twoheadrightarrow\!\!\!\!\!\rightarrowtail Component$
inv0_6: $Refinement\_refines\_Component \in Refinement \rightarrowtail Component$
inv0_7:

$\forall xx \cdot ($
$\forall px \cdot ($
$($
$xx \in dom(DomainModel\_parent\_DomainModel)$
$\land px = DomainModel\_parent\_DomainModel(xx)$
$\land px \in dom(DomainModel\_corresp\_Component)$
$\land xx \notin dom(DomainModel\_corresp\_Component)$
$)$
$\Rightarrow DomainModel\_corresp\_Component(px) \notin ran(Refinement\_refines\_Component)$
$)$
$)$

•••

END

MACHINE event_b_specs_from_ontologies_ref_1
REFINES event_b_specs_from_ontologies
SEES EventB_Metamodel_Context,Domain_Metamodel_Context
VARIABLES

DomainModel
DomainModel_parent_DomainModel
Variable
Constant
Set
SetItem
AbstractSet
EnumeratedSet
Invariant
Property
LogicFormula
InitialisationAction
  Event-B associations
Variable_definedIn_Component
Constant_definedIn_Component
Set_definedIn_Component
LogicFormula_definedIn_Component
Invariant_involves_Variables
Constant_isInvolvedIn_LogicFormulas
LogicFormula_involves_Sets
LogicFormula_involves_SetItems
LogicFormula_uses_Operators
Variable_typing_Invariant
Constant_typing_Property
SetItem_itemOf_EnumeratedSet
InitialisationAction_uses_Operators
Variable_init_InitialisationAction
InitialisationAction_involves_Constants
  Domain Model sets
Concept
Individual
DataValue
DataSet
DefaultDataSet
CustomDataSet
EnumeratedDataSet
  ************relations/attributes**************
Relation
RelationMaplet
AttributeMaplet
Attribute Domain Model attributes
Concept_isVariable
  ************relations/attributes****************
Relation_isVariable
Relation_isTransitive
Relation_isSymmetric
relation_isASymmetric
Relation_isReflexive
Relation_isIrreflexive
Attribute_isVariable
Attribute_isFunctional
  Domain Model associations

Concept_definedIn_DomainModel
DataSet_definedIn_DomainModel
Concept_parentConcept_Concept
Individual_individualOf_Concept
DataValue_valueOf_DataSet
DataValue_elements_EnumeratedDataSet
Relation_definedIn_DomainModel
Attribute_definedIn_DomainModel
************relations/attributes**********************
Relation_domain_Concept
Relation_range_Concept
Relation_DomainCardinality_minCardinality
Relation_DomainCardinality_maxCardinality
Relation_RangeCardinality_minCardinality
Relation_RangeCardinality_maxCardinality
RelationMaplet_mapletOf_Relation
RelationMaplet_antecedent_Individual
RelationMaplet_image_Individual
Attribute_domain_Concept
Attribute_range_DataSet
AttributeMaplet_mapletOf_Attribute
AttributeMaplet_antecedent_Individual
AttributeMaplet_image_DataValue
correspondences
Concept_corresp_AbstractSet
DomainModel_corresp_Component
EnumeratedDataSet_corresp_EnumeratedSet
DataValue_corresp_SetItem
CustomDataSet_corresp_AbstractSet
DefaultDataSet_corresp_AbstractSet
Concept_corresp_Constant
Individual_corresp_Constant
DataValue_corresp_Constant
Concept_corresp_Variable
************relations/attributes***************************
Relation_Type
Relation_corresp_Constant
Relation_corresp_Variable
Attribute_Type
Attribute_corresp_Constant
Attribute_corresp_Variable
RelationCharacteristic_corresp_LogicFormula
RelationMaplet_corresp_Constant
DataSet_corresp_Set
AttributeMaplet_corresp_Constant

INVARIANTS

inv1_1: $Variable \subseteq Variable\_Set$
inv1_2: $Constant \subseteq Constant\_Set$
inv1_3: $Set \subseteq Set\_Set$
inv1_4: $partition(Set, AbstractSet, EnumeratedSet)$
inv1_5: $SetItem \subseteq SetItem\_Set$
inv1_6: $Variable\_definedIn\_Component \in Variable \rightarrow Component$
inv1_7: $Constant\_definedIn\_Component \in Constant \rightarrow Component$
inv1_8: $Set\_definedIn\_Component \in Set \rightarrow Component$
inv1_9: $SetItem\_itemOf\_EnumeratedSet \in SetItem \twoheadrightarrow EnumeratedSet$
Domain Model

`inv1_10`: $Concept \subseteq Concept\_Set$
`inv1_11`: $Individual \subseteq Individual\_Set$
`inv1_12`: $DataValue \subseteq DataValue\_Set$
`inv1_13`: $DataSet \subseteq DataSet\_Set$
`inv1_14`: $partition(DataSet, DefaultDataSet, CustomDataSet)$
`inv1_15`: $EnumeratedDataSet \subseteq CustomDataSet$
`inv1_16`: $Concept\_isVariable \in Concept \rightarrow BOOL$
`inv1_17`: $Concept\_definedIn\_DomainModel \in Concept \rightarrow DomainModel$
`inv1_18`: $DataSet\_definedIn\_DomainModel \in DataSet \rightarrow DomainModel$
`inv1_19`: $Concept\_parentConcept\_Concept \in Concept ⇸ Concept$
`inv1_20`: $Individual\_individualOf\_Concept \in Individual \rightarrow Concept$
`inv1_21`: $DataValue\_valueOf\_DataSet \in DataValue \rightarrow DataSet$
`inv1_22`: $DataValue\_elements\_EnumeratedDataSet \in DataValue ⤀ EnumeratedDataSet$
`inv1_23`: $Concept\_corresp\_AbstractSet \in Concept ⤔ AbstractSet$
`inv1_24`: $EnumeratedDataSet\_corresp\_EnumeratedSet \in EnumeratedDataSet ⤔ EnumeratedSet$
`inv1_25`: $DataValue\_corresp\_SetItem \in DataValue ⤔ SetItem$
`inv1_26`: $\forall xx \cdot (xx \in EnumeratedDataSet \land xx \notin dom(EnumeratedDataSet\_corresp\_EnumeratedSet) \Rightarrow DataValue\_elements\_EnumeratedDataSet^{-1}[\{xx\}] \cap dom(DataValue\_corresp\_SetItem) = \emptyset)$
`inv1_27`: $CustomDataSet\_corresp\_AbstractSet \in CustomDataSet ⤔ AbstractSet$
`inv1_28`: $\{\_NATURAL, \_INTEGER, \_FLOAT, \_BOOL, \_STRING\} \cap CustomDataSet = \emptyset$
`inv1_29`: $DefaultDataSet\_corresp\_AbstractSet \in DefaultDataSet ⤔ AbstractSet$
`inv1_30`: $\{B\_NATURAL, B\_INTEGER, B\_FLOAT, B\_BOOL, B\_STRING\} \cap EnumeratedSet = \emptyset$
`inv1_31`: $Concept\_corresp\_Constant \in Concept ⤔ Constant$
`inv1_33`: $LogicFormula \subseteq LogicFormula\_Set$
`inv1_34`: $Property \subseteq LogicFormula$
`inv1_35`: $Invariant \subseteq LogicFormula$
`inv1_36`: $LogicFormula\_definedIn\_Component \in LogicFormula \rightarrow Component$
`inv1_37`: $Invariant\_involves\_Variables \in Invariant \rightarrow (\mathbb{N}_1 ⇸ Variable)$
logic formula operands can be variables, constants, sets or set items, indexed by their appearance order number. The first operand is indexed by 1, no matter it's type.
`inv1_38`: $ran(union(ran(Invariant\_involves\_Variables))) = Variable$
`inv1_39`: $Constant\_isInvolvedIn\_LogicFormulas \in Constant \rightarrow \mathbb{P}_1(\mathbb{N}_1 \times LogicFormula)$
When appearance order does not matter, we may index all constants using the same number.
`inv1_40`: $\forall cons \cdot (cons \in Constant \Rightarrow ran(Constant\_isInvolvedIn\_LogicFormulas(cons)) \cap Property \neq \emptyset)$
`inv1_41`: $LogicFormula\_involves\_Sets \in LogicFormula \rightarrow (\mathbb{N}_1 ⇸ Set)$
`inv1_42`: $LogicFormula\_uses\_Operators \in LogicFormula \rightarrow (\mathbb{N}_1 ⇸ Operator)$
`inv1_44`: $Individual\_corresp\_Constant \in Individual ⤔ Constant$
`inv1_45`: $DataValue\_corresp\_Constant \in DataValue ⤔ Constant$
`inv1_46`: $Concept\_corresp\_Variable \in Concept ⤔ Variable$
`inv1_47`: $InitialisationAction \subseteq InitialisationAction\_Set$
`inv1_49`: $InitialisationAction\_uses\_Operators \in InitialisationAction \rightarrow (\mathbb{N}_1 ⇸ Operator)$
`inv1_50`: $Variable\_init\_InitialisationAction \in Variable ⤖ InitialisationAction$
for initialisation actions, the assigned operand is the involved variable.
`inv1_52`: $InitialisationAction\_involves\_Constants \in InitialisationAction \rightarrow (\mathbb{N}_1 ⇸ Constant)$
***********relations/attributes***
`inv1_53`: $Relation \subseteq Relation\_Set$
`inv1_56`: $RelationMaplet \subseteq Relation\_Maplet\_Set$
`inv1_57`: $AttributeMaplet \subseteq Attribute\_Maplet\_Set$
`inv1_58`: $Attribute \subseteq Attribute\_Set$
`inv1_59`: $Relation\_isVariable \in Relation \rightarrow BOOL$
`inv1_60`: $Relation\_isTransitive \in Relation ⇸ BOOL$
`inv1_61`: $Relation\_isSymmetric \in Relation ⇸ BOOL$
`inv1_62`: $relation\_isASymmetric \in Relation ⇸ BOOL$
`inv1_63`: $Relation\_isReflexive \in Relation ⇸ BOOL$
`inv1_64`: $Relation\_isIrreflexive \in Relation ⇸ BOOL$
`inv1_65`: $Relation\_DomainCardinality\_minCardinality \in Relation ⇸ \mathbb{N}$

inv1_66: $Relation\_DomainCardinality\_maxCardinality \in Relation \rightarrow\!\!\!\!\!+ (\mathbb{N} \cup \{-1\})$
inv1_67: $Relation\_RangeCardinality\_minCardinality \in Relation \rightarrow\!\!\!\!\!+ \mathbb{N}$
inv1_68: $Relation\_RangeCardinality\_maxCardinality \in Relation \rightarrow\!\!\!\!\!+ (\mathbb{N} \cup \{-1\})$
inv1_69: $Attribute\_isVariable \in Attribute \rightarrow BOOL$
inv1_70: $Attribute\_isFunctional \in Attribute \rightarrow\!\!\!\!\!+ BOOL$
inv1_71: $Relation\_definedIn\_DomainModel \in Relation \rightarrow DomainModel$
inv1_72: $Attribute\_definedIn\_DomainModel \in Attribute \rightarrow DomainModel$
inv1_73: $Relation\_domain\_Concept \in Relation \rightarrow Concept$
inv1_74: $Relation\_range\_Concept \in Relation \rightarrow Concept$
inv1_77: $RelationMaplet\_mapletOf\_Relation \in RelationMaplet \rightarrow Relation$
inv1_78: $RelationMaplet\_antecedent\_Individual \in RelationMaplet \rightarrow Individual$
inv1_79: $RelationMaplet\_image\_Individual \in RelationMaplet \rightarrow Individual$
inv1_80: $Attribute\_domain\_Concept \in Attribute \rightarrow Concept$
inv1_81: $Attribute\_range\_DataSet \in Attribute \rightarrow DataSet$
inv1_82: $AttributeMaplet\_mapletOf\_Attribute \in AttributeMaplet \rightarrow Attribute$
inv1_83: $AttributeMaplet\_antecedent\_Individual \in AttributeMaplet \rightarrow Individual$
inv1_84: $AttributeMaplet\_image\_DataValue \in AttributeMaplet \rightarrow DataValue$
inv1_85: $\forall rm\cdot(rm \in RelationMaplet \Rightarrow Individual\_individualOf\_Concept(RelationMaplet\_antecedent\_Individual(rm)) = Relation\_domain\_Concept(RelationMaplet\_mapletOf\_Relation(rm)))$
inv1_86: $\forall rm\cdot(rm \in RelationMaplet \Rightarrow Individual\_individualOf\_Concept(RelationMaplet\_image\_Individual(rm)) = Relation\_range\_Concept(RelationMaplet\_mapletOf\_Relation(rm)))$
inv1_87: $\forall am\cdot(am \in AttributeMaplet \Rightarrow Individual\_individualOf\_Concept(AttributeMaplet\_antecedent\_Individual(am)) = Attribute\_domain\_Concept(AttributeMaplet\_mapletOf\_Attribute(am)))$
inv1_88: $\forall am\cdot(am \in AttributeMaplet \Rightarrow DataValue\_valueOf\_DataSet(AttributeMaplet\_image\_DataValue(am)) = Attribute\_range\_DataSet(AttributeMaplet\_mapletOf\_Attribute(am)))$
inv1_89: $Relation\_Type \in Relation \rightarrowtail\!\!\!\!\!+ Constant$
inv1_90: $Relation\_corresp\_Constant \in Relation \rightarrowtail\!\!\!\!\!+ Constant$
inv1_91: $Relation\_corresp\_Variable \in Relation \rightarrowtail\!\!\!\!\!+ Variable$
inv1_92: $\forall re\cdot(re \in dom(Relation\_Type) \Leftrightarrow (re \in dom(Relation\_corresp\_Constant) \lor (re \in dom(Relation\_corresp\_Variable))))$

inv1_93: $Attribute\_Type \in Attribute \rightarrowtail\!\!\!\!\!+ Constant$
inv1_94: $Attribute\_corresp\_Constant \in Attribute \rightarrowtail\!\!\!\!\!+ Constant$
inv1_95: $Attribute\_corresp\_Variable \in Attribute \rightarrowtail\!\!\!\!\!+ Variable$
inv1_96: $\forall re\cdot(re \in dom(Attribute\_Type) \Leftrightarrow (re \in dom(Attribute\_corresp\_Constant) \lor (re \in dom(Attribute\_corresp\_Variable))))$

inv1_97: $Variable\_typing\_Invariant \in Variable \rightarrowtail Invariant$
inv1_98: $Constant\_typing\_Property \in Constant \rightarrowtail Property$
inv1_99: $RelationCharacteristic\_corresp\_LogicFormula \in (Relation \rightarrow\!\!\!\!\!+ RelationCharacteristics\_Set) \rightarrowtail\!\!\!\!\!+ LogicFormula$
inv1_100: $RelationMaplet\_corresp\_Constant \in RelationMaplet \rightarrowtail\!\!\!\!\!+ Constant$
inv1_101: $DataSet\_corresp\_Set \in DataSet \rightarrowtail\!\!\!\!\!+ Set$
inv1_102: $AttributeMaplet\_corresp\_Constant \in AttributeMaplet \rightarrowtail\!\!\!\!\!+ Constant$
inv1_103: $LogicFormula\_involves\_SetItems \in LogicFormula \rightarrow\!\!\!\!\!+ (\mathbb{N}_1 \rightarrow\!\!\!\!\!+ SetItem)$
inv1_104: $EnumeratedDataSet\_corresp\_EnumeratedSet \subseteq DataSet\_corresp\_Set$
inv1_105: $CustomDataSet\_corresp\_AbstractSet \subseteq DataSet\_corresp\_Set$

EVENTS

Event initialize_default_datasets ⟨ordinary⟩ $\widehat{=}$

any

DM
o_DM

where

grd0: $dom(DomainModel\_corresp\_Component) \setminus dom(DomainModel\_parent\_DomainModel) \neq \emptyset$
grd1: $DefaultDataSet = \emptyset$
grd2: $DM \in dom(DomainModel\_corresp\_Component)$
grd3: $DM \notin dom(DomainModel\_parent\_DomainModel)$
grd4: $AbstractSet \cap \{B\_NATURAL, B\_INTEGER, B\_FLOAT, B\_BOOL, B\_STRING\} = \emptyset$

grd5: $o\_DM = DomainModel\_corresp\_Component(DM)$

then

act1: $DefaultDataSet := \{\_NATURAL, \_INTEGER, \_FLOAT, \_BOOL, \_STRING\}$

act2: $DataSet := DataSet \cup \{\_NATURAL, \_INTEGER, \_FLOAT, \_BOOL, \_STRING\}$

act3: $DataSet\_definedIn\_DomainModel := DataSet\_definedIn\_DomainModel \cup \{(xx \mapsto yy) | xx \in \{\_NATURAL, \_INTEG$ $yy = DM\}$

act4: $AbstractSet := AbstractSet \cup \{B\_NATURAL, B\_INTEGER, B\_FLOAT, B\_BOOL, B\_STRING\}$

act5: $Set := Set \cup \{B\_NATURAL, B\_INTEGER, B\_FLOAT, B\_BOOL, B\_STRING\}$

act6: $DefaultDataSet\_corresp\_AbstractSet := \{\_NATURAL \mapsto B\_NATURAL, \_INTEGER \mapsto B\_INTEGER, \_FLOAT \mapsto B\_FLOAT, \_BOOL \mapsto B\_BOOL, \_STRING \mapsto B\_STRING\}$

act7: $Set\_definedIn\_Component := Set\_definedIn\_Component \cup \{(xx \mapsto yy) | xx \in \{B\_NATURAL, B\_INTEGER, B\_FLOAT, B\_BOOL, B\_STRING\} \land yy = o\_DM\}$

act8: $DataSet\_corresp\_Set := DataSet\_corresp\_Set \lhd\!\!- \{\_NATURAL \mapsto B\_NATURAL, \_INTEGER \mapsto B\_INTEGER, \_FLOAT \mapsto B\_FLOAT, \_BOOL \mapsto B\_BOOL, \_STRING \mapsto B\_STRING\}$

end

•••

END

4.2 From Domain Models to Event-B Specifications

Event-B Machines and Contexts

Rule 1: Domain model without parent

MACHINE event_b_specs_from_ontologies
SEES EventB_Metamodel_Context,Domain_Metamodel_Context
Event rule_1 ⟨ordinary⟩ $\widehat{=}$ correspondence of a domain model not associated to a parent domain model

any

DM

o_DM

where

grd0: $DomainModel \setminus (dom(DomainModel\_corresp\_Component) \cup dom(DomainModel\_parent\_DomainModel)) \neq \emptyset$

grd1: $DM \in DomainModel$

grd2: $DM \notin dom(DomainModel\_corresp\_Component)$

grd3: $DM \notin dom(DomainModel\_parent\_DomainModel)$

grd4: $Component\_Set \setminus Component \neq \emptyset$

grd5: $o\_DM \in Component\_Set \setminus Component$

then

act1: $System := System \cup \{o\_DM\}$

act2: $Component := Component \cup \{o\_DM\}$

act3: $DomainModel\_corresp\_Component(DM) := o\_DM$

end

END

Any domain model that is not associated with another domain model (Fig. 9), through the *parent* association, gives rise to a system component. **Example :** in Figure 13, the root level domain model is translated into a system component named ***lg_system_ref_0***.

Rule 2: Domain model with parent

MACHINE event_b_specs_from_ontologies
SEES EventB_Metamodel_Context,Domain_Metamodel_Context
Event rule_2 ⟨ordinary⟩ $\widehat{=}$

correspondence of a domain model associated to a parent domain model

any

DM

PDM

o_DM

where

grd0: $dom(DomainModel\_parent\_DomainModel) \setminus dom(DomainModel\_corresp\_Component) \neq \emptyset$

grd1: $DM \in dom(DomainModel\_parent\_DomainModel)$

grd2: $DM \notin dom(DomainModel\_corresp\_Component)$

grd3: $dom(DomainModel\_corresp\_Component) \neq \emptyset$

grd4: $PDM \in dom(DomainModel\_corresp\_Component)$

grd5: $DomainModel\_parent\_DomainModel(DM) = PDM$

grd6: $Component\_Set \setminus Component \neq \emptyset$

grd7: $o\_DM \in Component\_Set \setminus Component$

then

act1: $Refinement := Refinement \cup \{o\_DM\}$

act2: $Component := Component \cup \{o\_DM\}$

act3: $Refinement\_refines\_Component(o\_DM) := DomainModel\_corresp\_Component(PDM)$

act4: $DomainModel\_corresp\_Component(DM) := o\_DM$

end

END

A domain model associated with another one representing its parent (Fig. 9) gives rise to a refinement component. The refinement component must refine the component corresponding to the parent domain model.

Example : in Figure 14, the first refinement level domain model is translated into a refinement component named ***lg_system_ref_1*** refining ***lg_system_ref_0***.

Event-B Sets

Rule 3: Concept without parent

MACHINE event_b_specs_from_ontologies_ref_1
REFINES event_b_specs_from_ontologies
SEES EventB_Metamodel_Context,Domain_Metamodel_Context
Event rule_3 ⟨ordinary⟩ $\widehat{=}$

correspondence of a concept not associated to a parent concept

any

CO
o_CO

where

grd0: $Concept \setminus (dom(Concept\_parentConcept\_Concept) \cup dom(Concept\_corresp\_AbstractSet)) \neq \emptyset$
grd1: $CO \in Concept$
grd2: $CO \notin dom(Concept\_parentConcept\_Concept)$
grd3: $CO \notin dom(Concept\_corresp\_AbstractSet)$
grd4: $Concept\_definedIn\_DomainModel(CO) \in dom(DomainModel\_corresp\_Component)$
grd5: $Set\_Set \setminus Set \neq \emptyset$
grd6: $o\_CO \in Set\_Set \setminus Set$

then

act1: $AbstractSet := AbstractSet \cup \{o\_CO\}$
act2: $Set := Set \cup \{o\_CO\}$
act3: $Concept\_corresp\_AbstractSet(CO) := o\_CO$
act4: $Set\_definedIn\_Component(o\_CO) := DomainModel\_corresp\_Component(Concept\_definedIn\_DomainModel(CO))$

end
END

Any concept that is not associated with another one known as its parent concept (Fig. 10), through the parentConcept association, gives rise to an *Event-B* abstract set. **Example :** in Figure 13, the abstract set *LandingGear* appears because of Concept instance ***LandingGear***.

Rule 4: Enumerated data set

MACHINE event_b_specs_from_ontologies_ref_1
REFINES event_b_specs_from_ontologies
SEES EventB_Metamodel_Context,Domain_Metamodel_Context
Event rule_4 ⟨ordinary⟩ $\widehat{=}$

correspondence of an instance of EnumeratedDataSet

any

EDS
o_EDS
elements
o_elements
mapping_elements_o_elements

where

grd0: $EnumeratedDataSet \setminus dom(DataSet\_corresp\_Set) \neq \emptyset$
grd1: $EDS \in EnumeratedDataSet$
grd2: $EDS \notin dom(DataSet\_corresp\_Set)$
grd4: $DataSet\_definedIn\_DomainModel(EDS) \in dom(DomainModel\_corresp\_Component)$
grd5: $Set\_Set \setminus Set \neq \emptyset$
grd6: $o\_EDS \in Set\_Set \setminus Set$
grd8: $o\_EDS \notin \{B\_NATURAL, B\_INTEGER, B\_FLOAT, B\_BOOL, B\_STRING\}$
elements
grd9: $o\_elements \subseteq SetItem\_Set \setminus SetItem$
grd11: $elements = DataValue\_elements\_EnumeratedDataSet^{-1}[\{EDS\}]$
grd12: $card(o\_elements) = card(elements)$
grd13: $mapping\_elements\_o\_elements \in elements \rightarrowtail\!\!\!\!\twoheadrightarrow o\_elements$

then
act1: $EnumeratedSet := EnumeratedSet \cup \{o\_EDS\}$
act2: $Set := Set \cup \{o\_EDS\}$
act3: $EnumeratedDataSet\_corresp\_EnumeratedSet(EDS) := o\_EDS$
act4: $Set\_definedIn\_Component(o\_EDS) := DomainModel\_corresp\_Component($
$DataSet\_definedIn\_DomainModel(EDS))$
elements
act5: $SetItem := SetItem \cup o\_elements$
act6: $SetItem\_itemOf\_EnumeratedSet := SetItem\_itemOf\_EnumeratedSet \cup (o\_elements \times \{o\_EDS\})$
act7: $DataValue\_corresp\_SetItem := DataValue\_corresp\_SetItem \cup mapping\_elements\_o\_elements$
act8: $DataSet\_corresp\_Set := DataSet\_corresp\_Set \lhd\!\!- \{EDS \mapsto o\_EDS\}$
end
END

Rule 5 : Custom data set not defined through an enumeration

MACHINE event_b_specs_from_ontologies_ref_1
REFINES event_b_specs_from_ontologies
SEES EventB_Metamodel_Context,Domain_Metamodel_Context
Event rule_5 ⟨ordinary⟩ $\widehat{=}$
correspondence of an instance of CustomDataSet which is not an instance of EnumeratedDataSet
any
CS
o_CS
where
grd0: $CustomDataSet \setminus (EnumeratedDataSet \cup dom(DataSet\_corresp\_Set)) \neq \emptyset$
grd1: $CS \in CustomDataSet$
grd2: $CS \notin EnumeratedDataSet$
grd3: $CS \notin dom(DataSet\_corresp\_Set)$
grd4: $DataSet\_definedIn\_DomainModel(CS) \in dom(DomainModel\_corresp\_Component)$
grd5: $Set\_Set \setminus Set \neq \emptyset$
grd6: $o\_CS \in Set\_Set \setminus Set$
then
act1: $AbstractSet := AbstractSet \cup \{o\_CS\}$
act2: $Set := Set \cup \{o\_CS\}$
act3: $CustomDataSet\_corresp\_AbstractSet(CS) := o\_CS$
act4: $Set\_definedIn\_Component(o\_CS) := DomainModel\_corresp\_Component($
$DataSet\_definedIn\_DomainModel(CS))$
act5: $DataSet\_corresp\_Set := DataSet\_corresp\_Set \lhd\!\!- \{CS \mapsto o\_CS\}$
end
END

Any instance of CustomDataSet, defined through an enumeration, gives rise to an *Event-B* enumerated set. **Example :** in Figure 13, the data set ***{"lg_extended", "lg_retracted"}***, defined in domain model represented in Figure (Fig. 7), gives rise to the enumerated set *DataSet_1={lg_extended, lg_retracted}*.

Any instance of DefaultDataSet is mapped directly to an *Event-B* default data set (NATURAL, INTEGER, FLOAT, STRING or BOOL) following the ***initialize_default_datasets*** event.

Event-B Constants

Rule 6 : Concept with parent

MACHINE event_b_specs_from_ontologies_ref_1
REFINES event_b_specs_from_ontologies
SEES EventB_Metamodel_Context,Domain_Metamodel_Context
Event rule_6_1 ⟨ordinary⟩ $\widehat{=}$
correspondence of a concept associated to a parent concept (where the parent concept corresponds to an abstract set)

any

CO

o_CO

PCO

o_lg

o_PCO

where

grd0: $dom(Concept\_parentConcept\_Concept) \setminus dom(Concept\_corresp\_Constant) \neq \emptyset$

grd1: $CO \in dom(Concept\_parentConcept\_Concept) \setminus dom(Concept\_corresp\_Constant)$

grd2: $dom(Concept\_corresp\_AbstractSet) \neq \emptyset$

grd3: $PCO \in dom(Concept\_corresp\_AbstractSet)$

grd4: $Concept\_parentConcept\_Concept(CO) = PCO$

grd5: $Concept\_definedIn\_DomainModel(CO) \in dom(DomainModel\_corresp\_Component)$

grd6: $Constant\_Set \setminus Constant \neq \emptyset$

grd7: $o\_CO \in Constant\_Set \setminus Constant$

grd8: $LogicFormula\_Set \setminus LogicFormula \neq \emptyset$

grd9: $o\_lg \in LogicFormula\_Set \setminus LogicFormula$

grd10: $o\_PCO \in AbstractSet$

grd11: $o\_PCO = Concept\_corresp\_AbstractSet(PCO)$

then

act1: $Constant := Constant \cup \{o\_CO\}$

act2: $Concept\_corresp\_Constant(CO) := o\_CO$

act3: $Constant\_definedIn\_Component(o\_CO) := DomainModel\_corresp\_Component(Concept\_definedIn\_DomainModel(CO))$

act4: $Property := Property \cup \{o\_lg\}$

act5: $LogicFormula := LogicFormula \cup \{o\_lg\}$

act6: $LogicFormula\_uses\_Operators(o\_lg) := \{1 \mapsto Inclusion\_OP\}$

act7: $Constant\_isInvolvedIn\_LogicFormulas(o\_CO) := \{1 \mapsto o\_lg\}$

act8: $LogicFormula\_involves\_Sets(o\_lg) := \{2 \mapsto o\_PCO\}$

act9: $LogicFormula\_definedIn\_Component(o\_lg) := DomainModel\_corresp\_Component(Concept\_definedIn\_DomainModel(CO))$

act10: $Constant\_typing\_Property(o\_CO) := o\_lg$

end

Event rule_6_2 ⟨ordinary⟩ $\widehat{=}$

correspondence of a concept associated to a parent concept (where the parent concept corresponds to a constant)

any

CO

o_CO

PCO

o_lg

o_PCO

where

grd0: $dom(Concept\_parentConcept\_Concept) \setminus dom(Concept\_corresp\_Constant) \neq \emptyset$

grd1: $CO \in dom(Concept\_parentConcept\_Concept) \setminus dom(Concept\_corresp\_Constant)$

grd2: $dom(Concept\_corresp\_Constant) \neq \emptyset$

grd3: $PCO \in dom(Concept\_corresp\_Constant)$

grd4: $Concept\_parentConcept\_Concept(CO) = PCO$

grd5: $Concept\_definedIn\_DomainModel(CO) \in dom(DomainModel\_corresp\_Component)$

grd6: $Constant\_Set \setminus Constant \neq \emptyset$

grd7: $o\_CO \in Constant\_Set \setminus Constant$

grd8: $LogicFormula\_Set \setminus LogicFormula \neq \emptyset$

grd9: $o\_lg \in LogicFormula\_Set \setminus LogicFormula$

grd10: $o\_PCO \in Constant$

grd11: $o\_PCO = Concept\_corresp\_Constant(PCO)$

then

act1: $Constant := Constant \cup \{o\_CO\}$

act2: $Concept\_corresp\_Constant(CO) := o\_CO$
act3: $Constant\_definedIn\_Component(o\_CO) := DomainModel\_corresp\_Component($
$Concept\_definedIn\_DomainModel(CO))$
act4: $Property := Property \cup \{o\_lg\}$
act5: $LogicFormula := LogicFormula \cup \{o\_lg\}$
act6: $LogicFormula\_uses\_Operators(o\_lg) := \{1 \mapsto Inclusion\_OP\}$
act7: $Constant\_isInvolvedIn\_LogicFormulas := Constant\_isInvolvedIn\_LogicFormulas \Leftarrow \{(o\_CO \mapsto \{1 \mapsto o\_lg\}), o\_PCO \mapsto Constant\_isInvolvedIn\_LogicFormulas(o\_PCO) \cup \{2 \mapsto o\_lg\}\}$
act8: $LogicFormula\_involves\_Sets(o\_lg) := \emptyset$
act9: $LogicFormula\_definedIn\_Component(o\_lg) := DomainModel\_corresp\_Component($
$Concept\_definedIn\_DomainModel(CO))$
act10: $Constant\_typing\_Property(o\_CO) := o\_lg$

end

END

Any concept associated with another one known as its parent concept (Fig. 9), through the parentConcept association, gives rise to a constant typed as a subset of the *Event-B* element corresponding to the parent concept.

Each individual (or data value) gives rise to a constant having its name (or with his *lexicalForm* typed as value) and each instance of CustomDataSet, not defined through an enumeration of its elements, unlike ***DataSet_1*** of Figure 13, gives rise to a constant having its name. **Example :** in Figure 14, the constant named *HD1* is the correspondent of the individual ***HD1***.

Rule 7 : Individual

MACHINE event_b_specs_from_ontologies_ref_1
REFINES event_b_specs_from_ontologies
SEES EventB_Metamodel_Context,Domain_Metamodel_Context
Event rule_7_1 ⟨ordinary⟩ $\widehat{=}$

correspondence of an instance of Individual (where the concept corresponds to an abstract set)

any

ind
o_ind
CO
o_lg
o_CO

where

grd0: $dom(Individual\_individualOf\_Concept) \setminus dom(Individual\_corresp\_Constant) \neq \emptyset$
grd1: $ind \in dom(Individual\_individualOf\_Concept) \setminus dom(Individual\_corresp\_Constant)$
grd2: $dom(Concept\_corresp\_AbstractSet) \neq \emptyset$
grd3: $CO \in dom(Concept\_corresp\_AbstractSet)$
grd4: $Individual\_individualOf\_Concept(ind) = CO$
grd5: $Concept\_definedIn\_DomainModel(CO) \in dom(DomainModel\_corresp\_Component)$
grd6: $Constant\_Set \setminus Constant \neq \emptyset$
grd7: $o\_ind \in Constant\_Set \setminus Constant$
grd8: $LogicFormula\_Set \setminus LogicFormula \neq \emptyset$
grd9: $o\_lg \in LogicFormula\_Set \setminus LogicFormula$
grd10: $o\_CO \in AbstractSet$
grd11: $o\_CO = Concept\_corresp\_AbstractSet(CO)$

then

act1: $Constant := Constant \cup \{o\_ind\}$
act2: $Individual\_corresp\_Constant(ind) := o\_ind$
act3: $Constant\_definedIn\_Component(o\_ind) := DomainModel\_corresp\_Component($
$Concept\_definedIn\_DomainModel(CO))$
act4: $Property := Property \cup \{o\_lg\}$
act5: $LogicFormula := LogicFormula \cup \{o\_lg\}$
act6: $LogicFormula\_uses\_Operators(o\_lg) := \{1 \mapsto Belonging\_OP\}$
act7: $Constant\_isInvolvedIn\_LogicFormulas(o\_ind) := \{1 \mapsto o\_lg\}$

act8: $LogicFormula\_involves\_Sets(o\_lg) := \{2 \mapsto o\_CO\}$
act9: $LogicFormula\_definedIn\_Component(o\_lg) := DomainModel\_corresp\_Component($
$Concept\_definedIn\_DomainModel(CO))$
act10: $Constant\_typing\_Property(o\_ind) := o\_lg$
end
Event rule_7_2 ⟨ordinary⟩ $\widehat{=}$
correspondence of an instance of Individual (where the concept corresponds to a constant)
any
ind
o_ind
CO
o_lg
o_CO
where
grd0: $dom(Individual\_individualOf\_Concept) \setminus dom(Individual\_corresp\_Constant) \neq \emptyset$
grd1: $ind \in dom(Individual\_individualOf\_Concept) \setminus dom(Individual\_corresp\_Constant)$
grd2: $dom(Concept\_corresp\_Constant) \neq \emptyset$
grd3: $CO \in dom(Concept\_corresp\_Constant)$
grd4: $Individual\_individualOf\_Concept(ind) = CO$
grd5: $Concept\_definedIn\_DomainModel(CO) \in dom(DomainModel\_corresp\_Component)$
grd6: $Constant\_Set \setminus Constant \neq \emptyset$
grd7: $o\_ind \in Constant\_Set \setminus Constant$
grd8: $LogicFormula\_Set \setminus LogicFormula \neq \emptyset$
grd9: $o\_lg \in LogicFormula\_Set \setminus LogicFormula$
grd10: $o\_CO \in Constant$
grd11: $o\_CO = Concept\_corresp\_Constant(CO)$
then
act1: $Constant := Constant \cup \{o\_ind\}$
act2: $Individual\_corresp\_Constant(ind) := o\_ind$
act3: $Constant\_definedIn\_Component(o\_ind) := DomainModel\_corresp\_Component($
$Concept\_definedIn\_DomainModel(CO))$
act4: $Property := Property \cup \{o\_lg\}$
act5: $LogicFormula := LogicFormula \cup \{o\_lg\}$
act6: $LogicFormula\_uses\_Operators(o\_lg) := \{1 \mapsto Belonging\_OP\}$
act7: $Constant\_isInvolvedIn\_LogicFormulas := Constant\_isInvolvedIn\_LogicFormulas \lhd\!\!- \{(o\_ind \mapsto$
$\{1 \mapsto o\_lg\}), o\_CO \mapsto Constant\_isInvolvedIn\_LogicFormulas(o\_CO) \cup \{2 \mapsto o\_lg\}\}$
act8: $LogicFormula\_involves\_Sets(o\_lg) := \emptyset$
act9: $LogicFormula\_definedIn\_Component(o\_lg) := DomainModel\_corresp\_Component($
$Concept\_definedIn\_DomainModel(CO))$
act10: $Constant\_typing\_Property(o\_ind) := o\_lg$
end
END

Rule 8 : Data value

MACHINE event_b_specs_from_ontologies_ref_1
REFINES event_b_specs_from_ontologies
SEES EventB_Metamodel_Context,Domain_Metamodel_Context
Event rule_8 ⟨ordinary⟩ $\widehat{=}$
correspondence of an instance of DataValue (When the data set is an instance of CustomDataSet not instance of EnumeratedDataSet
(for this last case, the rule for instances of EnumeratedDataSet also handles data values))
any
dva
o_dva
DS
o_lg
o_DS

where

grd0: $dom(DataValue\_valueOf\_DataSet) \setminus dom(DataValue\_corresp\_Constant) \neq \emptyset$

grd1: $dva \in dom(DataValue\_valueOf\_DataSet) \setminus dom(DataValue\_corresp\_Constant)$

grd2: $dom(CustomDataSet\_corresp\_AbstractSet) \neq \emptyset$

grd3: $DS \in dom(CustomDataSet\_corresp\_AbstractSet)$

grd4: $DataValue\_valueOf\_DataSet(dva) = DS$

grd5: $DataSet\_definedIn\_DomainModel(DS) \in dom(DomainModel\_corresp\_Component)$

grd6: $Constant\_Set \setminus Constant \neq \emptyset$

grd7: $o\_dva \in Constant\_Set \setminus Constant$

grd8: $LogicFormula\_Set \setminus LogicFormula \neq \emptyset$

grd9: $o\_lg \in LogicFormula\_Set \setminus LogicFormula$

grd10: $o\_DS \in AbstractSet$

grd11: $o\_DS = CustomDataSet\_corresp\_AbstractSet(DS)$

then

act1: $Constant := Constant \cup \{o\_dva\}$

act2: $DataValue\_corresp\_Constant(dva) := o\_dva$

act3: $Constant\_definedIn\_Component(o\_dva) := DomainModel\_corresp\_Component($
$DataSet\_definedIn\_DomainModel(DS))$

act4: $Property := Property \cup \{o\_lg\}$

act5: $LogicFormula := LogicFormula \cup \{o\_lg\}$

act6: $LogicFormula\_uses\_Operators(o\_lg) := \{1 \mapsto Belonging\_OP\}$

act7: $Constant\_isInvolvedIn\_LogicFormulas(o\_dva) := \{1 \mapsto o\_lg\}$

act8: $LogicFormula\_involves\_Sets(o\_lg) := \{2 \mapsto o\_DS\}$

act9: $LogicFormula\_definedIn\_Component(o\_lg) := DomainModel\_corresp\_Component($
$DataSet\_definedIn\_DomainModel(DS))$

act10: $Constant\_typing\_Property(o\_dva) := o\_lg$

end

END

Rule 10 : Constant relation

MACHINE event_b_specs_from_ontologies_ref_1
REFINES event_b_specs_from_ontologies
SEES EventB_Metamodel_Context,Domain_Metamodel_Context
Event rule_10_1 ⟨ordinary⟩ $\widehat{=}$

correspondence of an instance of Relation having its isVariable property set to false (case where domain and range correspond to abstract sets)

any

RE
T_RE
o_RE
CO1
o_CO1
CO2
o_CO2
o_lg1
o_lg2
DM

where

grd0: $Relation\_isVariable^{-1}[\{FALSE\}] \setminus dom(Relation\_Type) \neq \emptyset$

grd1: $RE \in Relation\_isVariable^{-1}[\{FALSE\}] \setminus dom(Relation\_Type)$

grd2: $dom(Concept\_corresp\_AbstractSet) \neq \emptyset$

grd3: $CO1 = Relation\_domain\_Concept(RE)$

grd4: $CO2 = Relation\_range\_Concept(RE)$

grd5: $\{CO1, CO2\} \subseteq dom(Concept\_corresp\_AbstractSet)$

grd6: $Relation\_definedIn\_DomainModel(RE) \in dom(DomainModel\_corresp\_Component)$

grd7: $Constant\_Set \setminus Constant \neq \emptyset$

grd8: $\{T\_RE, o\_RE\} \subseteq Constant\_Set \setminus Constant$
grd9: $LogicFormula\_Set \setminus LogicFormula \neq \emptyset$
grd10: $\{o\_lg1, o\_lg2\} \subseteq LogicFormula\_Set \setminus LogicFormula$
grd11: $o\_CO1 = Concept\_corresp\_AbstractSet(CO1)$
grd12: $o\_CO2 = Concept\_corresp\_AbstractSet(CO2)$
grd13: $DM = Relation\_definedIn\_DomainModel(RE)$
grd14: $T\_RE \neq o\_RE$
grd15: $o\_lg1 \neq o\_lg2$

then

act1: $Constant := Constant \cup \{T\_RE, o\_RE\}$
act2: $Relation\_Type(RE) := T\_RE$
act3: $Relation\_corresp\_Constant(RE) := o\_RE$
act4: $Constant\_definedIn\_Component := Constant\_definedIn\_Component \cup \{o\_RE \mapsto DomainModel\_corresp\_Component(DM), T\_RE \mapsto DomainModel\_corresp\_Component(DM)\}$

act5: $Property := Property \cup \{o\_lg1, o\_lg2\}$
act6: $LogicFormula := LogicFormula \cup \{o\_lg1, o\_lg2\}$
act7: $Constant\_typing\_Property := Constant\_typing\_Property \cup \{T\_RE \mapsto o\_lg1, o\_RE \mapsto o\_lg2\}$
act8: $Constant\_isInvolvedIn\_LogicFormulas := Constant\_isInvolvedIn\_LogicFormulas \cup \{T\_RE \mapsto \{1 \mapsto o\_lg1, 2 \mapsto o\_lg2\}, o\_RE \mapsto \{1 \mapsto o\_lg2\}\}$
act9: $LogicFormula\_uses\_Operators := LogicFormula\_uses\_Operators \cup \{o\_lg1 \mapsto \{1 \mapsto RelationSet\_OP\}, o\_lg2 \mapsto \{1 \mapsto Belonging\_OP\}\}$
act10: $LogicFormula\_involves\_Sets := LogicFormula\_involves\_Sets \cup \{o\_lg1 \mapsto \{2 \mapsto o\_CO1, 3 \mapsto o\_CO2\}, o\_lg2 \mapsto \emptyset\}$
act11: $LogicFormula\_definedIn\_Component := LogicFormula\_definedIn\_Component \cup \{o\_lg1 \mapsto DomainModel\_corresp\_$ $DomainModel\_corresp\_Component(DM)\}$

end

Event rule_10_2 ⟨ordinary⟩ $\widehat{=}$

correspondence of an instance of Relation having its isVariable property set to false (case where domain corresponds to an abstract set and range corresponds to a constant)

any

RE
T_RE
o_RE
CO1
o_CO1
CO2
o_CO2
o_lg1
o_lg2
DM

where

grd0: $Relation\_isVariable^{-1}[\{FALSE\}] \setminus dom(Relation\_Type) \neq \emptyset$
grd1: $RE \in Relation\_isVariable^{-1}[\{FALSE\}] \setminus dom(Relation\_Type)$
grd2: $dom(Concept\_corresp\_AbstractSet) \neq \emptyset$
grd3: $CO1 = Relation\_domain\_Concept(RE)$
grd4: $CO1 \in dom(Concept\_corresp\_AbstractSet)$
grd5: $dom(Concept\_corresp\_Constant) \neq \emptyset$
grd6: $CO2 = Relation\_range\_Concept(RE)$
grd7: $CO2 \in dom(Concept\_corresp\_Constant)$
grd8: $Relation\_definedIn\_DomainModel(RE) \in dom(DomainModel\_corresp\_Component)$
grd9: $Constant\_Set \setminus Constant \neq \emptyset$
grd10: $\{T\_RE, o\_RE\} \subseteq Constant\_Set \setminus Constant$
grd11: $LogicFormula\_Set \setminus LogicFormula \neq \emptyset$
grd12: $\{o\_lg1, o\_lg2\} \subseteq LogicFormula\_Set \setminus LogicFormula$
grd13: $o\_CO1 = Concept\_corresp\_AbstractSet(CO1)$

grd14: $o\_CO2 = Concept\_corresp\_Constant(CO2)$
grd15: $DM = Relation\_definedIn\_DomainModel(RE)$
grd16: $T\_RE \neq o\_RE$
grd17: $o\_lg1 \neq o\_lg2$

then

act1: $Constant := Constant \cup \{T\_RE, o\_RE\}$
act2: $Relation\_Type(RE) := T\_RE$
act3: $Relation\_corresp\_Constant(RE) := o\_RE$
act4: $Constant\_definedIn\_Component := Constant\_definedIn\_Component \cup \{o\_RE \mapsto DomainModel\_corresp\_Component(DM), T\_RE \mapsto DomainModel\_corresp\_Component(DM)\}$

act5: $Property := Property \cup \{o\_lg1, o\_lg2\}$
act6: $LogicFormula := LogicFormula \cup \{o\_lg1, o\_lg2\}$
act7: $Constant\_typing\_Property := Constant\_typing\_Property \cup \{T\_RE \mapsto o\_lg1, o\_RE \mapsto o\_lg2\}$
act8: $Constant\_isInvolvedIn\_LogicFormulas := Constant\_isInvolvedIn\_LogicFormulas \lhd\!\!- \{T\_RE \mapsto \{1 \mapsto o\_lg1, 2 \mapsto o\_lg2\}, o\_RE \mapsto \{1 \mapsto o\_lg2\}, o\_CO2 \mapsto \{3 \mapsto o\_lg1\} \cup Constant\_isInvolvedIn\_LogicFormulas(o\_CO2)\}$

act9: $LogicFormula\_uses\_Operators := LogicFormula\_uses\_Operators \cup \{o\_lg1 \mapsto \{1 \mapsto RelationSet\_OP\}, o\_lg2 \mapsto \{2 \mapsto Belonging\_OP\}\}$
act10: $LogicFormula\_involves\_Sets := LogicFormula\_involves\_Sets \cup \{o\_lg1 \mapsto \{2 \mapsto o\_CO1\}, o\_lg2 \mapsto \emptyset\}$
act11: $LogicFormula\_definedIn\_Component := LogicFormula\_definedIn\_Component \cup \{o\_lg1 \mapsto DomainModel\_corresp\_$
$DomainModel\_corresp\_Component(DM)\}$

end

Event rule_10_3 ⟨ordinary⟩ $\widehat{=}$

correspondence of an instance of Relation having its isVariable property set to false (case where range corresponds to an abstract set and domain corresponds to a constant)

any

RE
T_RE
o_RE
CO1
o_CO1
CO2
o_CO2
o_lg1
o_lg2
DM

where

grd0: $Relation\_isVariable^{-1}[\{FALSE\}] \setminus dom(Relation\_Type) \neq \emptyset$
grd1: $RE \in Relation\_isVariable^{-1}[\{FALSE\}] \setminus dom(Relation\_Type)$
grd2: $dom(Concept\_corresp\_Constant) \neq \emptyset$
grd3: $CO1 = Relation\_domain\_Concept(RE)$
grd4: $CO1 \in dom(Concept\_corresp\_Constant)$
grd5: $dom(Concept\_corresp\_AbstractSet) \neq \emptyset$
grd6: $CO2 = Relation\_range\_Concept(RE)$
grd7: $CO2 \in dom(Concept\_corresp\_AbstractSet)$
grd8: $Relation\_definedIn\_DomainModel(RE) \in dom(DomainModel\_corresp\_Component)$
grd9: $Constant\_Set \setminus Constant \neq \emptyset$
grd10: $\{T\_RE, o\_RE\} \subseteq Constant\_Set \setminus Constant$
grd11: $LogicFormula\_Set \setminus LogicFormula \neq \emptyset$
grd12: $\{o\_lg1, o\_lg2\} \subseteq LogicFormula\_Set \setminus LogicFormula$
grd13: $o\_CO2 = Concept\_corresp\_AbstractSet(CO2)$
grd14: $o\_CO1 = Concept\_corresp\_Constant(CO1)$
grd15: $DM = Relation\_definedIn\_DomainModel(RE)$
grd16: $T\_RE \neq o\_RE$

grd17: $o\_lg1 \neq o\_lg2$

then

act1: $Constant := Constant \cup \{T\_RE, o\_RE\}$

act2: $Relation\_Type(RE) := T\_RE$

act3: $Relation\_corresp\_Constant(RE) := o\_RE$

act4: $Constant\_definedIn\_Component := Constant\_definedIn\_Component \cup \{o\_RE \mapsto DomainModel\_corresp\_Component(DM), T\_RE \mapsto DomainModel\_corresp\_Component(DM)\}$

act5: $Property := Property \cup \{o\_lg1, o\_lg2\}$

act6: $LogicFormula := LogicFormula \cup \{o\_lg1, o\_lg2\}$

act7: $Constant\_typing\_Property := Constant\_typing\_Property \cup \{T\_RE \mapsto o\_lg1, o\_RE \mapsto o\_lg2\}$

act8: $Constant\_isInvolvedIn\_LogicFormulas := Constant\_isInvolvedIn\_LogicFormulas \lhd\!\!- \{T\_RE \mapsto \{1 \mapsto o\_lg1, 2 \mapsto o\_lg2\}, o\_RE \mapsto \{1 \mapsto o\_lg2\}, o\_CO1 \mapsto \{2 \mapsto o\_lg1\} \cup Constant\_isInvolvedIn\_LogicFormulas(o\_CO1)\}$

act9: $LogicFormula\_uses\_Operators := LogicFormula\_uses\_Operators \cup \{o\_lg1 \mapsto \{1 \mapsto RelationSet\_OP\}, o\_lg2 \mapsto \{1 \mapsto Belonging\_OP\}\}$

act10: $LogicFormula\_involves\_Sets := LogicFormula\_involves\_Sets \cup \{o\_lg1 \mapsto \{3 \mapsto o\_CO2\}, o\_lg2 \mapsto \emptyset\}$

act11: $LogicFormula\_definedIn\_Component := LogicFormula\_definedIn\_Component \cup \{o\_lg1 \mapsto DomainModel\_corresp\_$
$DomainModel\_corresp\_Component(DM)\}$

end

Event rule_10_4 ⟨ordinary⟩ $\widehat{=}$

correspondence of an instance of Relation having its isVariable property set to false (case where domain and range correspond to constants)

any

RE
T_RE
o_RE
CO1
o_CO1
CO2
o_CO2
o_lg1
o_lg2
DM

where

grd0: $Relation\_isVariable^{-1}[\{FALSE\}] \setminus dom(Relation\_Type) \neq \emptyset$

grd1: $RE \in Relation\_isVariable^{-1}[\{FALSE\}] \setminus dom(Relation\_Type)$

grd2: $dom(Concept\_corresp\_Constant) \neq \emptyset$

grd3: $CO1 = Relation\_domain\_Concept(RE)$

grd4: $CO2 = Relation\_range\_Concept(RE)$

grd5: $\{CO1, CO2\} \subseteq dom(Concept\_corresp\_Constant)$

grd6: $Relation\_definedIn\_DomainModel(RE) \in dom(DomainModel\_corresp\_Component)$

grd7: $Constant\_Set \setminus Constant \neq \emptyset$

grd8: $\{T\_RE, o\_RE\} \subseteq Constant\_Set \setminus Constant$

grd9: $LogicFormula\_Set \setminus LogicFormula \neq \emptyset$

grd10: $\{o\_lg1, o\_lg2\} \subseteq LogicFormula\_Set \setminus LogicFormula$

grd11: $o\_CO1 = Concept\_corresp\_Constant(CO1)$

grd12: $o\_CO2 = Concept\_corresp\_Constant(CO2)$

grd13: $DM = Relation\_definedIn\_DomainModel(RE)$

grd14: $T\_RE \neq o\_RE$

grd15: $o\_lg1 \neq o\_lg2$

grd16: $o\_CO1 \neq o\_CO2$

then

act1: $Constant := Constant \cup \{T\_RE, o\_RE\}$

act2: $Relation\_Type(RE) := T\_RE$

act3: $Relation\_corresp\_Constant(RE) := o\_RE$
act4: $Constant\_definedIn\_Component := Constant\_definedIn\_Component \cup$
$\{o\_RE \mapsto DomainModel\_corresp\_Component(DM), T\_RE \mapsto DomainModel\_corresp\_Component(DM)\}$

act5: $Property := Property \cup \{o\_lg1, o\_lg2\}$
act6: $LogicFormula := LogicFormula \cup \{o\_lg1, o\_lg2\}$
act7: $Constant\_typing\_Property := Constant\_typing\_Property \cup \{T\_RE \mapsto o\_lg1, o\_RE \mapsto o\_lg2\}$
act8: $Constant\_isInvolvedIn\_LogicFormulas := Constant\_isInvolvedIn\_LogicFormulas \lhd\!\!- \{T\_RE \mapsto$
$\{1 \mapsto o\_lg1, 2 \mapsto o\_lg2\}, o\_RE \mapsto \{1 \mapsto o\_lg2\}, o\_CO1 \mapsto \{2 \mapsto o\_lg1\} \cup Constant\_isInvolvedIn\_LogicFormulas(o\_CO1),$
$o\_CO2 \mapsto \{3 \mapsto o\_lg1\} \cup Constant\_isInvolvedIn\_LogicFormulas(o\_CO2)\}$
act9: $LogicFormula\_uses\_Operators := LogicFormula\_uses\_Operators \cup \{o\_lg1 \mapsto \{1 \mapsto RelationSet\_OP\}, o\_lg2 \mapsto$
$\{1 \mapsto Belonging\_OP\}\}$
act10: $LogicFormula\_involves\_Sets := LogicFormula\_involves\_Sets \cup \{o\_lg1 \mapsto \emptyset, o\_lg2 \mapsto \emptyset\}$
act11: $LogicFormula\_definedIn\_Component := LogicFormula\_definedIn\_Component \cup \{o\_lg1 \mapsto DomainModel\_corresp\_$
$DomainModel\_corresp\_Component(DM)\}$

end

END

Rule 11 : relation and attribute maplet

MACHINE event_b_specs_from_ontologies_ref_1
REFINES event_b_specs_from_ontologies
SEES EventB_Metamodel_Context,Domain_Metamodel_Context
Event rule_11_1 ⟨ordinary⟩ $\widehat{=}$

correspondence of an instance of RelationMaplet

any

remap
o_remap
RE
antecedent
image
o_lg
o_antecedent
o_image

where

grd0: $RelationMaplet \setminus dom(RelationMaplet\_corresp\_Constant) \neq \emptyset$
grd1: $remap \in RelationMaplet \setminus dom(RelationMaplet\_corresp\_Constant)$
grd2: $dom(Relation\_corresp\_Constant) \cup dom(Relation\_corresp\_Variable) \neq \emptyset$
grd3: $RelationMaplet\_mapletOf\_Relation(remap) = RE$
grd4: $RE \in dom(Relation\_corresp\_Constant) \cup dom(Relation\_corresp\_Variable)$
grd5: $Relation\_definedIn\_DomainModel(RE) \in dom(DomainModel\_corresp\_Component)$
grd6: $Constant\_Set \setminus Constant \neq \emptyset$
grd7: $o\_remap \in Constant\_Set \setminus Constant$
grd8: $LogicFormula\_Set \setminus LogicFormula \neq \emptyset$
grd9: $o\_lg \in LogicFormula\_Set \setminus LogicFormula$
grd10: $antecedent = RelationMaplet\_antecedent\_Individual(remap)$
grd11: $image = RelationMaplet\_image\_Individual(remap)$
grd12: $\{antecedent, image\} \subseteq dom(Individual\_corresp\_Constant)$
grd13: $o\_antecedent = Individual\_corresp\_Constant(antecedent)$
grd14: $o\_image = Individual\_corresp\_Constant(image)$
grd15: $o\_antecedent \neq o\_image$

then, for each relation already treated for which all the maplets have been processed,
if it is variable, we generate the initialization, otherwise, we generate the closure property,
knowing that the maplets give rise to variables in case of variable relation and constants
in case of constant relationship

then

act1: $Constant := Constant \cup \{o\_remap\}$

act2: $RelationMaplet\_corresp\_Constant(remap) := o\_remap$
act3: $Constant\_definedIn\_Component(o\_remap) := DomainModel\_corresp\_Component($
$Relation\_definedIn\_DomainModel(RE))$
act4: $Property := Property \cup \{o\_lg\}$
act5: $LogicFormula := LogicFormula \cup \{o\_lg\}$
act6: $LogicFormula\_uses\_Operators(o\_lg) := \{1 \mapsto Maplet\_OP\}$
act7: $Constant\_isInvolvedIn\_LogicFormulas := Constant\_isInvolvedIn\_LogicFormulas \lhd\!\!- \{o\_remap \mapsto \{1 \mapsto o\_lg\}, o\_antecedent \mapsto \{2 \mapsto o\_lg\} \cup Constant\_isInvolvedIn\_LogicFormulas(o\_antecedent), o\_image \mapsto \{3 \mapsto o\_lg\} \cup Constant\_isInvolvedIn\_LogicFormulas(o\_image)\}$
act8: $LogicFormula\_involves\_Sets(o\_lg) := \emptyset$
act9: $LogicFormula\_definedIn\_Component(o\_lg) := DomainModel\_corresp\_Component($
$Relation\_definedIn\_DomainModel(RE))$
act10: $Constant\_typing\_Property(o\_remap) := o\_lg$

end

Event rule_11_2_1 ⟨ordinary⟩ $\widehat{=}$

correspondence of an instance of AttributeMaplet (case where the image (of type DataValue) corresponds to a constant (it can also corresponds to a set item)

any

atmap
o_atmap
AT
antecedent
image
o_lg
o_antecedent
o_image

where

grd0: $AttributeMaplet \setminus dom(AttributeMaplet\_corresp\_Constant) \neq \emptyset$
grd1: $atmap \in AttributeMaplet \setminus dom(AttributeMaplet\_corresp\_Constant)$
grd2: $dom(Attribute\_corresp\_Constant) \cup dom(Attribute\_corresp\_Variable) \neq \emptyset$
grd3: $AttributeMaplet\_mapletOf\_Attribute(atmap) = AT$
grd4: $AT \in dom(Attribute\_corresp\_Constant) \cup dom(Attribute\_corresp\_Variable)$
grd5: $Attribute\_definedIn\_DomainModel(AT) \in dom(DomainModel\_corresp\_Component)$
grd6: $Constant\_Set \setminus Constant \neq \emptyset$
grd7: $o\_atmap \in Constant\_Set \setminus Constant$
grd8: $LogicFormula\_Set \setminus LogicFormula \neq \emptyset$
grd9: $o\_lg \in LogicFormula\_Set \setminus LogicFormula$
grd10: $antecedent = AttributeMaplet\_antecedent\_Individual(atmap)$
grd11: $image = AttributeMaplet\_image\_DataValue(atmap)$
grd12: $antecedent \in dom(Individual\_corresp\_Constant)$
grd13: $image \in dom(DataValue\_corresp\_Constant)$
grd14: $o\_antecedent = Individual\_corresp\_Constant(antecedent)$
grd15: $o\_image = DataValue\_corresp\_Constant(image)$
grd16: $o\_antecedent \neq o\_image$

then

act1: $Constant := Constant \cup \{o\_atmap\}$
act2: $AttributeMaplet\_corresp\_Constant(atmap) := o\_atmap$
act3: $Constant\_definedIn\_Component(o\_atmap) := DomainModel\_corresp\_Component($
$Attribute\_definedIn\_DomainModel(AT))$
act4: $Property := Property \cup \{o\_lg\}$
act5: $LogicFormula := LogicFormula \cup \{o\_lg\}$
act6: $LogicFormula\_uses\_Operators(o\_lg) := \{1 \mapsto Maplet\_OP\}$
act7: $Constant\_isInvolvedIn\_LogicFormulas := Constant\_isInvolvedIn\_LogicFormulas \lhd\!\!- \{o\_atmap \mapsto \{1 \mapsto o\_lg\}, o\_antecedent \mapsto \{2 \mapsto o\_lg\} \cup Constant\_isInvolvedIn\_LogicFormulas(o\_antecedent), o\_image \mapsto \{3 \mapsto o\_lg\} \cup Constant\_isInvolvedIn\_LogicFormulas(o\_image)\}$
act8: $LogicFormula\_involves\_Sets(o\_lg) := \emptyset$

act9: $LogicFormula\_definedIn\_Component(o\_lg) := DomainModel\_corresp\_Component($
$Attribute\_definedIn\_DomainModel(AT))$

act10: $Constant\_typing\_Property(o\_atmap) := o\_lg$

end

Event rule_11_2_2 ⟨ordinary⟩ $\widehat{=}$

correspondence of an instance of AttributeMaplet (case where the image (of type DataValue) corresponds to a set item

any

atmap

o_atmap

AT

antecedent

image

o_lg

o_antecedent

o_image

where

grd0: $AttributeMaplet \setminus dom(AttributeMaplet\_corresp\_Constant) \neq \emptyset$

grd1: $atmap \in AttributeMaplet \setminus dom(AttributeMaplet\_corresp\_Constant)$

grd2: $dom(Attribute\_corresp\_Constant) \cup dom(Attribute\_corresp\_Variable) \neq \emptyset$

grd3: $AttributeMaplet\_mapletOf\_Attribute(atmap) = AT$

grd4: $AT \in dom(Attribute\_corresp\_Constant) \cup dom(Attribute\_corresp\_Variable)$

grd5: $Attribute\_definedIn\_DomainModel(AT) \in dom(DomainModel\_corresp\_Component)$

grd6: $Constant\_Set \setminus Constant \neq \emptyset$

grd7: $o\_atmap \in Constant\_Set \setminus Constant$

grd8: $LogicFormula\_Set \setminus LogicFormula \neq \emptyset$

grd9: $o\_lg \in LogicFormula\_Set \setminus LogicFormula$

grd10: $antecedent = AttributeMaplet\_antecedent\_Individual(atmap)$

grd11: $image = AttributeMaplet\_image\_DataValue(atmap)$

grd12: $antecedent \in dom(Individual\_corresp\_Constant)$

grd13: $image \in dom(DataValue\_corresp\_SetItem)$

grd14: $o\_antecedent = Individual\_corresp\_Constant(antecedent)$

grd15: $o\_image = DataValue\_corresp\_SetItem(image)$

then

act1: $Constant := Constant \cup \{o\_atmap\}$

act2: $AttributeMaplet\_corresp\_Constant(atmap) := o\_atmap$

act3: $Constant\_definedIn\_Component(o\_atmap) := DomainModel\_corresp\_Component($
$Attribute\_definedIn\_DomainModel(AT))$

act4: $Property := Property \cup \{o\_lg\}$

act5: $LogicFormula := LogicFormula \cup \{o\_lg\}$

act6: $LogicFormula\_uses\_Operators(o\_lg) := \{1 \mapsto Maplet\_OP\}$

act7: $Constant\_isInvolvedIn\_LogicFormulas := Constant\_isInvolvedIn\_LogicFormulas \lhd\!\!- \{o\_atmap \mapsto$
$\{1 \mapsto o\_lg\}, o\_antecedent \mapsto \{2 \mapsto o\_lg\} \cup Constant\_isInvolvedIn\_LogicFormulas(o\_antecedent)\}$

act8: $LogicFormula\_involves\_Sets(o\_lg) := \emptyset$

act9: $LogicFormula\_involves\_SetItems(o\_lg) := \{3 \mapsto o\_image\}$

act10: $LogicFormula\_definedIn\_Component(o\_lg) := DomainModel\_corresp\_Component($
$Attribute\_definedIn\_DomainModel(AT))$

act11: $Constant\_typing\_Property(o\_atmap) := o\_lg$

end

END

Each relation gives rise to a constant representing the type of its associated *Event-B* element and defined as the set of relations between the *Event-B* element corresponding to the relation domain and the one corresponding to the relation range. Moreover, if the relation has its isVariable attribute set to *false*, it is translated through a second constant. **Example :** in Figure 14, ***LgOfHd***, for which isVariable is set to *false*, is translated into a constant named $\overline{LgOfHd}$ and having as type ***T_LgOfHd*** defined as the set of relations between ***Handle*** and ***LandingGear*** (assertions *1.7* and *1.8*).

Rule 14 : Constant attribute

MACHINE event_b_specs_from_ontologies_ref_1
REFINES event_b_specs_from_ontologies
SEES EventB_Metamodel_Context,Domain_Metamodel_Context
Event rule_14_1 ⟨ordinary⟩ $\widehat{=}$

correspondence of an instance of Attribute having its isVariable property set to false and its isFunctional property set to false (case where the domain corresponds to an abstract set, knowing that the range always corresponds to a set)

any

AT
T_AT
o_AT
CO
o_CO
DS
o_DS
o_lg1
o_lg2
DM

where

grd0: $Attribute\_isVariable^{-1}[\{FALSE\}] \setminus dom(Attribute\_Type) \neq \emptyset$
grd1: $AT \in Attribute\_isVariable^{-1}[\{FALSE\}] \setminus dom(Attribute\_Type)$
grd2: $dom(Concept\_corresp\_AbstractSet) \neq \emptyset$
grd3: $CO = Attribute\_domain\_Concept(AT)$
grd4: $CO \in dom(Concept\_corresp\_AbstractSet)$
grd5: $dom(DataSet\_corresp\_Set) \neq \emptyset$
grd6: $DS = Attribute\_range\_DataSet(AT)$
grd7: $DS \in dom(DataSet\_corresp\_Set)$
grd8: $Attribute\_definedIn\_DomainModel(AT) \in dom(DomainModel\_corresp\_Component)$
grd9: $Constant\_Set \setminus Constant \neq \emptyset$
grd10: $\{T\_AT, o\_AT\} \subseteq Constant\_Set \setminus Constant$
grd11: $LogicFormula\_Set \setminus LogicFormula \neq \emptyset$
grd12: $\{o\_lg1, o\_lg2\} \subseteq LogicFormula\_Set \setminus LogicFormula$
grd13: $o\_CO = Concept\_corresp\_AbstractSet(CO)$
grd14: $o\_DS = DataSet\_corresp\_Set(DS)$
grd15: $DM = Attribute\_definedIn\_DomainModel(AT)$
grd16: $T\_AT \neq o\_AT$
grd17: $o\_lg1 \neq o\_lg2$
grd18: $AT \in Attribute\_isFunctional^{-1}[\{FALSE\}]$

then

act1: $Constant := Constant \cup \{T\_AT, o\_AT\}$
act2: $Attribute\_Type(AT) := T\_AT$
act3: $Attribute\_corresp\_Constant(AT) := o\_AT$
act4: $Constant\_definedIn\_Component := Constant\_definedIn\_Component \cup \{o\_AT \mapsto DomainModel\_corresp\_Component(DM), T\_AT \mapsto DomainModel\_corresp\_Component(DM)\}$

act5: $Property := Property \cup \{o\_lg1, o\_lg2\}$
act6: $LogicFormula := LogicFormula \cup \{o\_lg1, o\_lg2\}$
act7: $Constant\_typing\_Property := Constant\_typing\_Property \cup \{T\_AT \mapsto o\_lg1, o\_AT \mapsto o\_lg2\}$
act8: $Constant\_isInvolvedIn\_LogicFormulas := Constant\_isInvolvedIn\_LogicFormulas \cup \{T\_AT \mapsto \{1 \mapsto o\_lg1, 2 \mapsto o\_lg2\}, o\_AT \mapsto \{1 \mapsto o\_lg2\}\}$
act9: $LogicFormula\_uses\_Operators := LogicFormula\_uses\_Operators \cup \{o\_lg1 \mapsto \{1 \mapsto RelationSet\_OP\}, o\_lg2 \mapsto \{1 \mapsto Belonging\_OP\}\}$
act10: $LogicFormula\_involves\_Sets := LogicFormula\_involves\_Sets \cup \{o\_lg1 \mapsto \{2 \mapsto o\_CO, 3 \mapsto o\_DS\}, o\_lg2 \mapsto \emptyset\}$

act11: $LogicFormula\_definedIn\_Component := LogicFormula\_definedIn\_Component \cup \{o\_lg1 \mapsto DomainModel\_corresp\_$
$DomainModel\_corresp\_Component(DM)\}$

end

Event rule_14_2 ⟨ordinary⟩ $\widehat{=}$

correspondence of an instance of Attribute having its isVariable property set to false and its isFunctional property set to false (case where the domain corresponds to a constant, knowing that the range always corresponds to a set)

any

AT
T_AT
o_AT
CO
o_CO
DS
o_DS
o_lg1
o_lg2
DM

where

grd0: $Attribute\_isVariable^{-1}[\{FALSE\}] \setminus dom(Attribute\_Type) \neq \emptyset$
grd1: $AT \in Attribute\_isVariable^{-1}[\{FALSE\}] \setminus dom(Attribute\_Type)$
grd2: $dom(Concept\_corresp\_Constant) \neq \emptyset$
grd3: $CO = Attribute\_domain\_Concept(AT)$
grd4: $CO \in dom(Concept\_corresp\_Constant)$
grd5: $dom(DataSet\_corresp\_Set) \neq \emptyset$
grd6: $DS = Attribute\_range\_DataSet(AT)$
grd7: $DS \in dom(DataSet\_corresp\_Set)$
grd8: $Attribute\_definedIn\_DomainModel(AT) \in dom(DomainModel\_corresp\_Component)$
grd9: $Constant\_Set \setminus Constant \neq \emptyset$
grd10: $\{T\_AT, o\_AT\} \subseteq Constant\_Set \setminus Constant$
grd11: $LogicFormula\_Set \setminus LogicFormula \neq \emptyset$
grd12: $\{o\_lg1, o\_lg2\} \subseteq LogicFormula\_Set \setminus LogicFormula$
grd13: $o\_CO = Concept\_corresp\_Constant(CO)$
grd14: $o\_DS = DataSet\_corresp\_Set(DS)$
grd15: $DM = Attribute\_definedIn\_DomainModel(AT)$
grd16: $T\_AT \neq o\_AT$
grd17: $o\_lg1 \neq o\_lg2$
grd18: $AT \in Attribute\_isFunctional^{-1}[\{FALSE\}]$

then

act1: $Constant := Constant \cup \{T\_AT, o\_AT\}$
act2: $Attribute\_Type(AT) := T\_AT$
act3: $Attribute\_corresp\_Constant(AT) := o\_AT$
act4: $Constant\_definedIn\_Component := Constant\_definedIn\_Component \cup$
$\{o\_AT \mapsto DomainModel\_corresp\_Component(DM), T\_AT \mapsto DomainModel\_corresp\_Component(DM)\}$

act5: $Property := Property \cup \{o\_lg1, o\_lg2\}$
act6: $LogicFormula := LogicFormula \cup \{o\_lg1, o\_lg2\}$
act7: $Constant\_typing\_Property := Constant\_typing\_Property \cup \{T\_AT \mapsto o\_lg1, o\_AT \mapsto o\_lg2\}$
act8: $Constant\_isInvolvedIn\_LogicFormulas := Constant\_isInvolvedIn\_LogicFormulas \mathbin{\lhd\!\!-} \{T\_AT \mapsto$
$\{1 \mapsto o\_lg1, 2 \mapsto o\_lg2\}, o\_AT \mapsto \{1 \mapsto o\_lg2\}, o\_CO \mapsto \{2 \mapsto o\_lg1\} \cup Constant\_isInvolvedIn\_LogicFormulas(o\_CO)\}$

act9: $LogicFormula\_uses\_Operators := LogicFormula\_uses\_Operators \cup \{o\_lg1 \mapsto \{1 \mapsto RelationSet\_OP\}, o\_lg2 \mapsto$
$\{1 \mapsto Belonging\_OP\}\}$
act10: $LogicFormula\_involves\_Sets := LogicFormula\_involves\_Sets \cup \{o\_lg1 \mapsto \{3 \mapsto o\_DS\}, o\_lg2 \mapsto \emptyset\}$
act11: $LogicFormula\_definedIn\_Component := LogicFormula\_definedIn\_Component \cup \{o\_lg1 \mapsto DomainModel\_corresp\_$
$DomainModel\_corresp\_Component(DM)\}$

end

Event rule_14_3 ⟨ordinary⟩ $\widehat{=}$

correspondence of an instance of Attribute having its isVariable property set to false and its isFunctional property set to true (case where the domain corresponds to an abstract set, knowing that the range always corresponds to a set)

any

AT
T_AT
o_AT
CO
o_CO
DS
o_DS
o_lg1
o_lg2
DM

where

grd0: $Attribute\_isVariable^{-1}[\{FALSE\}] \setminus dom(Attribute\_Type) \neq \emptyset$
grd1: $AT \in Attribute\_isVariable^{-1}[\{FALSE\}] \setminus dom(Attribute\_Type)$
grd2: $dom(Concept\_corresp\_AbstractSet) \neq \emptyset$
grd3: $CO = Attribute\_domain\_Concept(AT)$
grd4: $CO \in dom(Concept\_corresp\_AbstractSet)$
grd5: $dom(DataSet\_corresp\_Set) \neq \emptyset$
grd6: $DS = Attribute\_range\_DataSet(AT)$
grd7: $DS \in dom(DataSet\_corresp\_Set)$
grd8: $Attribute\_definedIn\_DomainModel(AT) \in dom(DomainModel\_corresp\_Component)$
grd9: $Constant\_Set \setminus Constant \neq \emptyset$
grd10: $\{T\_AT, o\_AT\} \subseteq Constant\_Set \setminus Constant$
grd11: $LogicFormula\_Set \setminus LogicFormula \neq \emptyset$
grd12: $\{o\_lg1, o\_lg2\} \subseteq LogicFormula\_Set \setminus LogicFormula$
grd13: $o\_CO = Concept\_corresp\_AbstractSet(CO)$
grd14: $o\_DS = DataSet\_corresp\_Set(DS)$
grd15: $DM = Attribute\_definedIn\_DomainModel(AT)$
grd16: $T\_AT \neq o\_AT$
grd17: $o\_lg1 \neq o\_lg2$
grd18: $AT \in Attribute\_isFunctional^{-1}[\{TRUE\}]$

then

act1: $Constant := Constant \cup \{T\_AT, o\_AT\}$
act2: $Attribute\_Type(AT) := T\_AT$
act3: $Attribute\_corresp\_Constant(AT) := o\_AT$
act4: $Constant\_definedIn\_Component := Constant\_definedIn\_Component \cup \{o\_AT \mapsto DomainModel\_corresp\_Component(DM), T\_AT \mapsto DomainModel\_corresp\_Component(DM)\}$

act5: $Property := Property \cup \{o\_lg1, o\_lg2\}$
act6: $LogicFormula := LogicFormula \cup \{o\_lg1, o\_lg2\}$
act7: $Constant\_typing\_Property := Constant\_typing\_Property \cup \{T\_AT \mapsto o\_lg1, o\_AT \mapsto o\_lg2\}$
act8: $Constant\_isInvolvedIn\_LogicFormulas := Constant\_isInvolvedIn\_LogicFormulas \cup \{T\_AT \mapsto \{1 \mapsto o\_lg1, 2 \mapsto o\_lg2\}, o\_AT \mapsto \{1 \mapsto o\_lg2\}\}$
act9: $LogicFormula\_uses\_Operators := LogicFormula\_uses\_Operators \cup \{o\_lg1 \mapsto \{1 \mapsto FunctionSet\_OP\}, o\_lg2 \mapsto \{1 \mapsto Belonging\_OP\}\}$
act10: $LogicFormula\_involves\_Sets := LogicFormula\_involves\_Sets \cup \{o\_lg1 \mapsto \{2 \mapsto o\_CO, 3 \mapsto o\_DS\}, o\_lg2 \mapsto \emptyset\}$
act11: $LogicFormula\_definedIn\_Component := LogicFormula\_definedIn\_Component \cup \{o\_lg1 \mapsto DomainModel\_corresp\_$ $DomainModel\_corresp\_Component(DM)\}$

end

Event rule_14_4 ⟨ordinary⟩ $\widehat{=}$

correspondence of an instance of Attribute having its isVariable property set to false and its isFunctional property set to true (case where the domain corresponds to a constant, knowing that the range always corresponds to a set)

any

AT
T_AT
o_AT
CO
o_CO
DS
o_DS
o_lg1
o_lg2
DM

where

grd0: $Attribute\_isVariable^{-1}[\{FALSE\}] \setminus dom(Attribute\_Type) \neq \emptyset$
grd1: $AT \in Attribute\_isVariable^{-1}[\{FALSE\}] \setminus dom(Attribute\_Type)$
grd2: $dom(Concept\_corresp\_Constant) \neq \emptyset$
grd3: $CO = Attribute\_domain\_Concept(AT)$
grd4: $CO \in dom(Concept\_corresp\_Constant)$
grd5: $dom(DataSet\_corresp\_Set) \neq \emptyset$
grd6: $DS = Attribute\_range\_DataSet(AT)$
grd7: $DS \in dom(DataSet\_corresp\_Set)$
grd8: $Attribute\_definedIn\_DomainModel(AT) \in dom(DomainModel\_corresp\_Component)$
grd9: $Constant\_Set \setminus Constant \neq \emptyset$
grd10: $\{T\_AT, o\_AT\} \subseteq Constant\_Set \setminus Constant$
grd11: $LogicFormula\_Set \setminus LogicFormula \neq \emptyset$
grd12: $\{o\_lg1, o\_lg2\} \subseteq LogicFormula\_Set \setminus LogicFormula$
grd13: $o\_CO = Concept\_corresp\_Constant(CO)$
grd14: $o\_DS = DataSet\_corresp\_Set(DS)$
grd15: $DM = Attribute\_definedIn\_DomainModel(AT)$
grd16: $T\_AT \neq o\_AT$
grd17: $o\_lg1 \neq o\_lg2$
grd18: $AT \in Attribute\_isFunctional^{-1}[\{TRUE\}]$

then

act1: $Constant := Constant \cup \{T\_AT, o\_AT\}$
act2: $Attribute\_Type(AT) := T\_AT$
act3: $Attribute\_corresp\_Constant(AT) := o\_AT$
act4: $Constant\_definedIn\_Component := Constant\_definedIn\_Component \cup \{o\_AT \mapsto DomainModel\_corresp\_Component(DM), T\_AT \mapsto DomainModel\_corresp\_Component(DM)\}$

act5: $Property := Property \cup \{o\_lg1, o\_lg2\}$
act6: $LogicFormula := LogicFormula \cup \{o\_lg1, o\_lg2\}$
act7: $Constant\_typing\_Property := Constant\_typing\_Property \cup \{T\_AT \mapsto o\_lg1, o\_AT \mapsto o\_lg2\}$
act8: $Constant\_isInvolvedIn\_LogicFormulas := Constant\_isInvolvedIn\_LogicFormulas \Leftarrow \{T\_AT \mapsto \{1 \mapsto o\_lg1, 2 \mapsto o\_lg2\}, o\_AT \mapsto \{1 \mapsto o\_lg2\}, o\_CO \mapsto \{2 \mapsto o\_lg1\} \cup Constant\_isInvolvedIn\_LogicFormulas(o\_CO)\}$

act9: $LogicFormula\_uses\_Operators := LogicFormula\_uses\_Operators \cup \{o\_lg1 \mapsto \{1 \mapsto FunctionSet\_OP\}, o\_lg2 \mapsto \{1 \mapsto Belonging\_OP\}\}$
act10: $LogicFormula\_involves\_Sets := LogicFormula\_involves\_Sets \cup \{o\_lg1 \mapsto \{3 \mapsto o\_DS\}, o\_lg2 \mapsto \emptyset\}$
act11: $LogicFormula\_definedIn\_Component := LogicFormula\_definedIn\_Component \cup \{o\_lg1 \mapsto DomainModel\_corresp\_$
$DomainModel\_corresp\_Component(DM)\}$

end

END

Similarly to relations, each attribute gives rise to a constant representing the type of its associated *Event-B* element and, in the case when isVariable is set to *false*, to another constant having its name. However, when the isFunctional attribute is set to *true*, the constant representing the type is defined as the set of functions between the *Event-B* element corresponding to the attribute domain and the one corresponding to the attribute range. The *Event-B* element corresponding to the attribute is then typed as a function. **Example :** in Figure 13, *landingGearState* is typed as a function (assertions *0.3* and *0.4*) since its type is the set of functions between ***LandingGear*** and ***DataSet_1*** (***DataSet_1***={lg_extended, lg_retracted}).

Event-B Variables

Rule 9 : Variable concept

MACHINE event_b_specs_from_ontologies_ref_1
REFINES event_b_specs_from_ontologies
SEES EventB_Metamodel_Context,Domain_Metamodel_Context
Event rule_9_1 ⟨ordinary⟩ $\widehat{=}$

handling the variability of a concept and initializing the associated variable (when the concept corresponds to an abstract set)

any

CO
x_CO
o_lg
o_CO
o_ia
o_inds
bij_o_inds

where

grd0: $(dom(Concept\_corresp\_AbstractSet) \cap Concept\_isVariable^{-1}[\{TRUE\}]) \setminus dom(Concept\_corresp\_Variable) \neq \emptyset$

grd1: $CO \in (dom(Concept\_corresp\_AbstractSet) \cap Concept\_isVariable^{-1}[\{TRUE\}]) \setminus dom(Concept\_corresp\_Variable)$

grd2: $Concept\_definedIn\_DomainModel(CO) \in dom(DomainModel\_corresp\_Component)$

grd3: $Individual\_individualOf\_Concept^{-1}[\{CO\}] \subseteq dom(Individual\_corresp\_Constant)$

grd4: $Variable\_Set \setminus Variable \neq \emptyset$

grd5: $x\_CO \in Variable\_Set \setminus Variable$

grd6: $LogicFormula\_Set \setminus LogicFormula \neq \emptyset$

grd7: $o\_lg \in LogicFormula\_Set \setminus LogicFormula$

grd8: $o\_CO \in AbstractSet$

grd9: $o\_CO = Concept\_corresp\_AbstractSet(CO)$

grd10: $InitialisationAction\_Set \setminus InitialisationAction \neq \emptyset$

grd11: $o\_ia \in InitialisationAction\_Set \setminus InitialisationAction$

grd12: $o\_inds = Individual\_corresp\_Constant[Individual\_individualOf\_Concept^{-1}[\{CO\}]]$

grd13: $finite(o\_inds)$

grd14: $bij\_o\_inds \in 1\,..\,card(o\_inds) \twoheadrightarrowtail o\_inds$

then

act1: $Variable := Variable \cup \{x\_CO\}$

act2: $Concept\_corresp\_Variable(CO) := x\_CO$

act3: $Variable\_definedIn\_Component(x\_CO) := DomainModel\_corresp\_Component(Concept\_definedIn\_DomainModel(CO))$

act4: $Invariant := Invariant \cup \{o\_lg\}$

act5: $LogicFormula := LogicFormula \cup \{o\_lg\}$

act6: $LogicFormula\_uses\_Operators(o\_lg) := \{1 \mapsto Inclusion\_OP\}$

act7: $Invariant\_involves\_Variables(o\_lg) := \{1 \mapsto x\_CO\}$

act8: $LogicFormula\_involves\_Sets(o\_lg) := \{2 \mapsto o\_CO\}$

act9: $LogicFormula\_definedIn\_Component(o\_lg) := DomainModel\_corresp\_Component(Concept\_definedIn\_DomainModel(CO))$

act10: $InitialisationAction := InitialisationAction \cup \{o\_ia\}$
act11: $InitialisationAction\_uses\_Operators(o\_ia) := \{1 \mapsto BecomeEqual2SetOf\_OP\}$
act12: $Variable\_init\_InitialisationAction(x\_CO) := o\_ia$
act13: $InitialisationAction\_involves\_Constants(o\_ia) := bij\_o\_inds$
act14: $Variable\_typing\_Invariant(x\_CO) := o\_lg$

end

Event rule_9_2 ⟨ordinary⟩ $\widehat{=}$

handling the variability of a concept and initializing the associated variable (when the concept corresponds to a constant)

any

CO
x_CO
o_lg
o_CO
o_ia
o_inds
bij_o_inds

where

grd0: $(dom(Concept\_corresp\_Constant) \cap Concept\_isVariable^{-1}[\{TRUE\}]) \setminus dom(Concept\_corresp\_Variable) \neq \emptyset$
grd1: $CO \in (dom(Concept\_corresp\_Constant) \cap Concept\_isVariable^{-1}[\{TRUE\}]) \setminus dom(Concept\_corresp\_Variable)$
grd2: $Concept\_definedIn\_DomainModel(CO) \in dom(DomainModel\_corresp\_Component)$
grd3: $Individual\_individualOf\_Concept^{-1}[\{CO\}] \subseteq dom(Individual\_corresp\_Constant)$
grd4: $Variable\_Set \setminus Variable \neq \emptyset$
grd5: $x\_CO \in Variable\_Set \setminus Variable$
grd6: $LogicFormula\_Set \setminus LogicFormula \neq \emptyset$
grd7: $o\_lg \in LogicFormula\_Set \setminus LogicFormula$
grd8: $o\_CO \in Constant$
grd9: $o\_CO = Concept\_corresp\_Constant(CO)$
grd10: $InitialisationAction\_Set \setminus InitialisationAction \neq \emptyset$
grd11: $o\_ia \in InitialisationAction\_Set \setminus InitialisationAction$
grd12: $o\_inds = Individual\_corresp\_Constant[Individual\_individualOf\_Concept^{-1}[\{CO\}]]$
grd13: $finite(o\_inds)$
grd14: $bij\_o\_inds \in 1\,..\,card(o\_inds) \twoheadrightarrowtail o\_inds$

then

act1: $Variable := Variable \cup \{x\_CO\}$
act2: $Concept\_corresp\_Variable(CO) := x\_CO$
act3: $Variable\_definedIn\_Component(x\_CO) := DomainModel\_corresp\_Component(Concept\_definedIn\_DomainModel(CO))$
act4: $Invariant := Invariant \cup \{o\_lg\}$
act5: $LogicFormula := LogicFormula \cup \{o\_lg\}$
act6: $LogicFormula\_uses\_Operators(o\_lg) := \{1 \mapsto Inclusion\_OP\}$
act7: $Invariant\_involves\_Variables(o\_lg) := \{1 \mapsto x\_CO\}$
act8: $Constant\_isInvolvedIn\_LogicFormulas(o\_CO) := Constant\_isInvolvedIn\_LogicFormulas(o\_CO) \cup \{2 \mapsto o\_lg\}$
act9: $LogicFormula\_involves\_Sets(o\_lg) := \emptyset$
act10: $LogicFormula\_definedIn\_Component(o\_lg) := DomainModel\_corresp\_Component(Concept\_definedIn\_DomainModel(CO))$
act11: $InitialisationAction := InitialisationAction \cup \{o\_ia\}$
act12: $InitialisationAction\_uses\_Operators(o\_ia) := \{1 \mapsto BecomeEqual2SetOf\_OP\}$
act13: $Variable\_init\_InitialisationAction(x\_CO) := o\_ia$
act14: $InitialisationAction\_involves\_Constants(o\_ia) := bij\_o\_inds$
act15: $Variable\_typing\_Invariant(x\_CO) := o\_lg$

end

END

Rule 13 : variable relation

MACHINE event_b_specs_from_ontologies_ref_1
REFINES event_b_specs_from_ontologies
SEES EventB_Metamodel_Context,Domain_Metamodel_Context
Event rule_13_1 ⟨ordinary⟩ $\widehat{=}$

correspondence of an instance of Relation having its isVariable property set to true (case where domain and range correspond to abstract sets. The others cases will not explicitely included here, since they can easily be obtained based on rules 10_2, 10_3 and 10_4)

any

RE
T_RE
o_RE
CO1
o_CO1
CO2
o_CO2
o_lg1
o_lg2
DM
o_ia

where

grd0: $Relation\_isVariable^{-1}[\{TRUE\}] \setminus dom(Relation\_Type) \neq \emptyset$
grd1: $RE \in Relation\_isVariable^{-1}[\{TRUE\}] \setminus dom(Relation\_Type)$
grd2: $dom(Concept\_corresp\_AbstractSet) \neq \emptyset$
grd3: $CO1 = Relation\_domain\_Concept(RE)$
grd4: $CO2 = Relation\_range\_Concept(RE)$
grd5: $\{CO1, CO2\} \subseteq dom(Concept\_corresp\_AbstractSet)$
grd6: $Relation\_definedIn\_DomainModel(RE) \in dom(DomainModel\_corresp\_Component)$
grd7: $Constant\_Set \setminus Constant \neq \emptyset$
grd8: $T\_RE \in Constant\_Set \setminus Constant$
grd9: $Variable\_Set \setminus Variable \neq \emptyset$
grd10: $o\_RE \in Variable\_Set \setminus Variable$
grd11: $LogicFormula\_Set \setminus LogicFormula \neq \emptyset$
grd12: $\{o\_lg1, o\_lg2\} \subseteq LogicFormula\_Set \setminus LogicFormula$
grd13: $o\_CO1 = Concept\_corresp\_AbstractSet(CO1)$
grd14: $o\_CO2 = Concept\_corresp\_AbstractSet(CO2)$
grd15: $DM = Relation\_definedIn\_DomainModel(RE)$
grd16: $o\_lg1 \neq o\_lg2$
grd17: $InitialisationAction\_Set \setminus InitialisationAction \neq \emptyset$
grd18: $o\_ia \in InitialisationAction\_Set \setminus InitialisationAction$

then

act1: $Constant := Constant \cup \{T\_RE\}$
act2: $Variable := Variable \cup \{o\_RE\}$
act3: $Relation\_Type(RE) := T\_RE$
act4: $Relation\_corresp\_Variable(RE) := o\_RE$
act5: $Constant\_definedIn\_Component(T\_RE) := DomainModel\_corresp\_Component(DM)$
act6: $Variable\_definedIn\_Component(o\_RE) := DomainModel\_corresp\_Component(DM)$
act7: $Property := Property \cup \{o\_lg1\}$
act8: $Invariant := Invariant \cup \{o\_lg2\}$
act9: $LogicFormula := LogicFormula \cup \{o\_lg1, o\_lg2\}$
act10: $Constant\_typing\_Property(T\_RE) := o\_lg1$
act11: $Variable\_typing\_Invariant(o\_RE) := o\_lg2$
act12: $Constant\_isInvolvedIn\_LogicFormulas(T\_RE) := \{1 \mapsto o\_lg1, 2 \mapsto o\_lg2\}$
act13: $Invariant\_involves\_Variables(o\_lg2) := \{1 \mapsto o\_RE\}$
act14: $LogicFormula\_uses\_Operators := LogicFormula\_uses\_Operators \cup \{o\_lg1 \mapsto \{1 \mapsto RelationSet\_OP\}, o\_lg2 \mapsto \{1 \mapsto Belonging\_OP\}\}$

act15: $LogicFormula\_involves\_Sets := LogicFormula\_involves\_Sets \cup \{o\_lg1 \mapsto \{2 \mapsto o\_CO1, 3 \mapsto o\_CO2\}, o\_lg2 \mapsto \emptyset\}$

act16: $LogicFormula\_definedIn\_Component := LogicFormula\_definedIn\_Component \cup \{o\_lg1 \mapsto DomainModel\_corresp\_$ $DomainModel\_corresp\_Component(DM)\}$

act17: $InitialisationAction := InitialisationAction \cup \{o\_ia\}$

act18: $InitialisationAction\_uses\_Operators(o\_ia) := \{1 \mapsto BecomeEqual2EmptySet\_OP\}$

act19: $Variable\_init\_InitialisationAction(o\_RE) := o\_ia$

act20: $InitialisationAction\_involves\_Constants(o\_ia) := \emptyset$

end

END

An instance of Relation, of Concept or of Attribute, having its isVariable property set to *true* gives rise to a variable (Fig. 12). For a concept, the variable represents the set of *Event-B* elements having this concept as type. For a relation or an attribute, it represents the set of links between individuals (in case of relation) or between individuals and data values (in case of attribute) defined through it.**Example :** in Figure 14, variables named *landingSetState* and *handleState* appear because of Attribute instances ***landingSetState*** and ***handleState*** for which the isVariable property is set to *true* (Fig. 8).

Invariants and Properties In this section, we are interested in the correspondences between the domain model and the *Event-B* model that are likely to give rise to *invariants*, *properties* or *initialization* clauses.

Rule 12 : closure property or action raised by relation maplets

MACHINE event_b_specs_from_ontologies_ref_1
REFINES event_b_specs_from_ontologies
SEES EventB_Metamodel_Context,Domain_Metamodel_Context
Event rule_12_1 ⟨ordinary⟩ $\widehat{=}$

closure property for constant relations

any

RE
o_lg
o_RE
maplets
o_maplets

where

grd0: $dom(Relation\_corresp\_Constant) \neq \emptyset$

grd1: $RE \in dom(Relation\_corresp\_Constant)$

grd2: $o\_RE = Relation\_corresp\_Constant(RE)$

grd3: $LogicFormula\_uses\_Operators^{-1}[\{\{1 \mapsto Equal2SetOf\_OP\}\}] \cap ran(Constant\_isInvolvedIn\_LogicFormulas(o\_RE)) = \emptyset$

grd4: $RelationMaplet\_mapletOf\_Relation^{-1}[\{RE\}] = maplets$

grd5: $maplets \subseteq dom(RelationMaplet\_corresp\_Constant)$

grd6: $o\_maplets = RelationMaplet\_corresp\_Constant[maplets]$

grd7: $Relation\_definedIn\_DomainModel(RE) \in dom(DomainModel\_corresp\_Component)$

grd8: $LogicFormula\_Set \setminus LogicFormula \neq \emptyset$

grd9: $o\_lg \in LogicFormula\_Set \setminus LogicFormula$

grd10: $o\_RE \notin o\_maplets$

then

act1: $Property := Property \cup \{o\_lg\}$

act2: $LogicFormula := LogicFormula \cup \{o\_lg\}$

act3: $LogicFormula\_uses\_Operators(o\_lg) := \{1 \mapsto Equal2SetOf\_OP\}$

act4: $Constant\_isInvolvedIn\_LogicFormulas := Constant\_isInvolvedIn\_LogicFormulas \lhd\!\!- (\{o\_RE \mapsto \{1 \mapsto o\_lg\} \cup Constant\_isInvolvedIn\_LogicFormulas(o\_RE)\} \cup \{co \mapsto lgs | co \in o\_maplets \wedge lgs = \{2 \mapsto o\_lg\} \cup Constant\_isInvolvedIn\_LogicFormulas(co)\})$

appearence order does not matter

act5: $LogicFormula\_involves\_Sets(o\_lg) := \emptyset$

act6: $LogicFormula\_definedIn\_Component(o\_lg) := DomainModel\_corresp\_Component(Relation\_definedIn\_DomainModel(RE))$

end

Event rule_12_2 ⟨ordinary⟩ $\widehat{=}$

closure action for variable relations

any

RE

o_ia

o_RE

maplets

o_maplets

ex_o_ia

bij_o_maplets

where

grd0: $dom(Relation\_corresp\_Variable) \neq \emptyset$

grd1: $RE \in dom(Relation\_corresp\_Variable)$

grd2: $o\_RE = Relation\_corresp\_Variable(RE)$

grd3: $Variable\_init\_InitialisationAction(o\_RE) \notin InitialisationAction\_uses\_Operators^{-1}[\{\{1 \mapsto BecomeEqual2SetOf\_OP\}\}]$

grd4: $RelationMaplet\_mapletOf\_Relation^{-1}[\{RE\}] = maplets$

grd5: $maplets \subseteq dom(RelationMaplet\_corresp\_Constant)$

grd6: $o\_maplets = RelationMaplet\_corresp\_Constant[maplets]$

grd7: $Relation\_definedIn\_DomainModel(RE) \in dom(DomainModel\_corresp\_Component)$

grd8: $InitialisationAction\_Set \setminus InitialisationAction \neq \emptyset$

grd9: $o\_ia \in InitialisationAction\_Set \setminus InitialisationAction$

grd10: $ex\_o\_ia = Variable\_init\_InitialisationAction(o\_RE)$

grd11: $Variable\_init\_InitialisationAction^{-1}[\{ex\_o\_ia\}] = \{o\_RE\}$

grd12: $finite(o\_maplets)$

grd13: $bij\_o\_maplets \in 1 \,..\, card(o\_maplets) \rightarrowtail\!\!\!\!\rightarrow o\_maplets$

then

act1: $InitialisationAction := (InitialisationAction \setminus \{ex\_o\_ia\}) \cup \{o\_ia\}$

act2: $InitialisationAction\_uses\_Operators := (InitialisationAction\_uses\_Operators \setminus \{ex\_o\_ia \mapsto InitialisationAction\_uses\_Operators(ex\_o\_ia)\}) \lhd\!\!- \{o\_ia \mapsto \{1 \mapsto BecomeEqual2SetOf\_OP\}\}$

act3: $Variable\_init\_InitialisationAction(o\_RE) := o\_ia$

act4: $InitialisationAction\_involves\_Constants := (InitialisationAction\_involves\_Constants \setminus \{ex\_o\_ia \mapsto InitialisationAction\_involves\_Constants(ex\_o\_ia)\}) \lhd\!\!- \{o\_ia \mapsto bij\_o\_maplets\}$

end

END

Rule 15 : closure property or action raised by relation maplets

MACHINE event_b_specs_from_ontologies_ref_1

REFINES event_b_specs_from_ontologies

SEES EventB_Metamodel_Context,Domain_Metamodel_Context

Event rule_15_1 ⟨ordinary⟩ $\widehat{=}$

closure property for constant attribute

any

AT

o_lg

o_AT

maplets

o_maplets

where

grd0: $dom(Attribute\_corresp\_Constant) \neq \emptyset$

grd1: $AT \in dom(Attribute\_corresp\_Constant)$

grd2: $o\_AT = Attribute\_corresp\_Constant(AT)$

grd3: $LogicFormula\_uses\_Operators^{-1}[\{\{1 \mapsto Equal2SetOf\_OP\}\}] \cap ran(Constant\_isInvolvedIn\_LogicFormulas(o\_AT)) = \emptyset$

grd4: $AttributeMaplet\_mapletOf\_Attribute^{-1}[\{AT\}] = maplets$

grd5: $\mathit{maplets} \subseteq \mathit{dom}(\mathit{AttributeMaplet\_corresp\_Constant})$
grd6: $\mathit{o\_maplets} = \mathit{AttributeMaplet\_corresp\_Constant}[\mathit{maplets}]$
grd7: $\mathit{Attribute\_definedIn\_DomainModel}(\mathit{AT}) \in \mathit{dom}(\mathit{DomainModel\_corresp\_Component})$
grd8: $\mathit{LogicFormula\_Set} \setminus \mathit{LogicFormula} \neq \emptyset$
grd9: $\mathit{o\_lg} \in \mathit{LogicFormula\_Set} \setminus \mathit{LogicFormula}$
grd10: $\mathit{o\_AT} \notin \mathit{o\_maplets}$

then

act1: $\mathit{Property} := \mathit{Property} \cup \{\mathit{o\_lg}\}$
act2: $\mathit{LogicFormula} := \mathit{LogicFormula} \cup \{\mathit{o\_lg}\}$
act3: $\mathit{LogicFormula\_uses\_Operators}(\mathit{o\_lg}) := \{1 \mapsto \mathit{Equal2SetOf\_OP}\}$
act4: $\mathit{Constant\_isInvolvedIn\_LogicFormulas} := \mathit{Constant\_isInvolvedIn\_LogicFormulas} \mathbin{<\!\!+} (\{\mathit{o\_AT} \mapsto (\{1 \mapsto \mathit{o\_lg}\} \cup \mathit{Constant\_isInvolvedIn\_LogicFormulas}(\mathit{o\_AT}))\} \cup \{\mathit{co} \mapsto \mathit{lgs} | \mathit{co} \in \mathit{o\_maplets} \land \mathit{lgs} = \{2 \mapsto \mathit{o\_lg}\} \cup \mathit{Constant\_isInvolvedIn\_LogicFormulas}(\mathit{co})\})$
appearence order does not matter
act5: $\mathit{LogicFormula\_involves\_Sets}(\mathit{o\_lg}) := \emptyset$
act6: $\mathit{LogicFormula\_definedIn\_Component}(\mathit{o\_lg}) := \mathit{DomainModel\_corresp\_Component}(\mathit{Attribute\_definedIn\_DomainModel}(\mathit{AT}))$

end

END

Rule 16 : optional characteristics of relations

MACHINE event_b_specs_from_ontologies_ref_1
REFINES event_b_specs_from_ontologies
SEES EventB_Metamodel_Context,Domain_Metamodel_Context
Event rule_16_1 ⟨ordinary⟩ $\widehat{=}$

handling the transitivity of a constant relation

any

RE
o_lg1
o_lg2
o_RE
composition

where

grd0: $(\mathit{dom}(\mathit{Relation\_corresp\_Constant}) \cap \mathit{Relation\_isTransitive}^{-1}[\{\mathit{TRUE}\}]) \neq \emptyset$
grd1: $\mathit{RE} \in (\mathit{dom}(\mathit{Relation\_corresp\_Constant}) \cap \mathit{Relation\_isTransitive}^{-1}[\{\mathit{TRUE}\}])$
grd2: $(\{\mathit{RE} \mapsto \mathit{isTransitive}\}) \notin \mathit{dom}(\mathit{RelationCharacteristic\_corresp\_LogicFormula})$
grd3: $\mathit{o\_RE} = \mathit{Relation\_corresp\_Constant}(\mathit{RE})$
grd4: $\mathit{Relation\_definedIn\_DomainModel}(\mathit{RE}) \in \mathit{dom}(\mathit{DomainModel\_corresp\_Component})$
grd5: $\mathit{LogicFormula\_Set} \setminus \mathit{LogicFormula} \neq \emptyset$
grd6: $\{\mathit{o\_lg1}, \mathit{o\_lg2}\} \subseteq \mathit{LogicFormula\_Set} \setminus \mathit{LogicFormula}$
grd7: $\mathit{partition}(\{\mathit{o\_lg1}, \mathit{o\_lg2}\}, \{\mathit{o\_lg1}\}, \{\mathit{o\_lg2}\})$
grd8: $\mathit{Constant\_Set} \setminus \mathit{Constant} \neq \emptyset$
grd9: $\mathit{composition} \in \mathit{Constant\_Set} \setminus \mathit{Constant}$

then

act0: $\mathit{Constant} := \mathit{Constant} \cup \{\mathit{composition}\}$
act1: $\mathit{Property} := \mathit{Property} \cup \{\mathit{o\_lg1}, \mathit{o\_lg2}\}$
act2: $\mathit{LogicFormula} := \mathit{LogicFormula} \cup \{\mathit{o\_lg1}, \mathit{o\_lg2}\}$
act3: $\mathit{Constant\_typing\_Property}(\mathit{composition}) := \mathit{o\_lg1}$
act4: $\mathit{RelationCharacteristic\_corresp\_LogicFormula}(\{\mathit{RE} \mapsto \mathit{isTransitive}\}) := \mathit{o\_lg2}$
act5: $\mathit{Constant\_isInvolvedIn\_LogicFormulas} := \mathit{Constant\_isInvolvedIn\_LogicFormulas} \mathbin{<\!\!+} \{\mathit{composition} \mapsto \{1 \mapsto \mathit{o\_lg1}, 1 \mapsto \mathit{o\_lg2}\}, \mathit{o\_RE} \mapsto \{2 \mapsto \mathit{o\_lg1}, 3 \mapsto \mathit{o\_lg1}, 2 \mapsto \mathit{o\_lg2}\} \cup \mathit{Constant\_isInvolvedIn\_LogicFormulas}(\mathit{o\_RE})\}$

act6: $\mathit{LogicFormula\_uses\_Operators} := \mathit{LogicFormula\_uses\_Operators} \cup \{\mathit{o\_lg1} \mapsto \{1 \mapsto \mathit{RelationComposition\_OP}\}, \mathit{o\_lg2} \mapsto \{1 \mapsto \mathit{Inclusion\_OP}\}\}$
act7: $\mathit{LogicFormula\_involves\_Sets} := \mathit{LogicFormula\_involves\_Sets} \cup \{\mathit{o\_lg1} \mapsto \emptyset, \mathit{o\_lg2} \mapsto \emptyset\}$
act8: $\mathit{LogicFormula\_definedIn\_Component} := \mathit{LogicFormula\_definedIn\_Component} \cup \{\mathit{o\_lg1} \mapsto \mathit{DomainModel\_corresp\_C}$
$\mathit{o\_lg2} \mapsto \mathit{DomainModel\_corresp\_Component}(\mathit{Relation\_definedIn\_DomainModel}(\mathit{RE}))\}$

act9: $Constant\_definedIn\_Component(composition) := DomainModel\_corresp\_Component($
$Relation\_definedIn\_DomainModel(RE))$

end

Event rule_16_2 ⟨ordinary⟩ $\widehat{=}$

handling the symmetrie of a constant relation

any

RE

o_lg1

o_lg2

o_RE

inverse

where

grd0: $(dom(Relation\_corresp\_Constant) \cap Relation\_isSymmetric^{-1}[\{TRUE\}]) \neq \emptyset$

grd1: $RE \in (dom(Relation\_corresp\_Constant) \cap Relation\_isSymmetric^{-1}[\{TRUE\}])$

grd2: $(\{RE \mapsto isSymmetric\}) \notin dom(RelationCharacteristic\_corresp\_LogicFormula)$

grd3: $o\_RE = Relation\_corresp\_Constant(RE)$

grd4: $Relation\_definedIn\_DomainModel(RE) \in dom(DomainModel\_corresp\_Component)$

grd5: $LogicFormula\_Set \setminus LogicFormula \neq \emptyset$

grd6: $\{o\_lg1, o\_lg2\} \subseteq LogicFormula\_Set \setminus LogicFormula$

grd7: $partition(\{o\_lg1, o\_lg2\}, \{o\_lg1\}, \{o\_lg2\})$

grd8: $Constant\_Set \setminus Constant \neq \emptyset$

grd9: $inverse \in Constant\_Set \setminus Constant$

then

act0: $Constant := Constant \cup \{inverse\}$

act1: $Property := Property \cup \{o\_lg1, o\_lg2\}$

act2: $LogicFormula := LogicFormula \cup \{o\_lg1, o\_lg2\}$

act3: $Constant\_typing\_Property(inverse) := o\_lg1$

act4: $RelationCharacteristic\_corresp\_LogicFormula(\{RE \mapsto isSymmetric\}) := o\_lg2$

act5: $Constant\_isInvolvedIn\_LogicFormulas := Constant\_isInvolvedIn\_LogicFormulas \lhd\!\!- \{inverse \mapsto$
$\{1 \mapsto o\_lg1, 1 \mapsto o\_lg2\}, o\_RE \mapsto \{2 \mapsto o\_lg1, 2 \mapsto o\_lg2\} \cup Constant\_isInvolvedIn\_LogicFormulas(o\_RE)\}$

act6: $LogicFormula\_uses\_Operators := LogicFormula\_uses\_Operators \cup \{o\_lg1 \mapsto \{1 \mapsto Inversion\_OP\}, o\_lg2 \mapsto$
$\{1 \mapsto Equality\_OP\}\}$

act7: $LogicFormula\_involves\_Sets := LogicFormula\_involves\_Sets \cup \{o\_lg1 \mapsto \emptyset, o\_lg2 \mapsto \emptyset\}$

act8: $LogicFormula\_definedIn\_Component := LogicFormula\_definedIn\_Component \cup \{o\_lg1 \mapsto DomainModel\_corresp\_C$
$o\_lg2 \mapsto DomainModel\_corresp\_Component(Relation\_definedIn\_DomainModel(RE))\}$

act9: $Constant\_definedIn\_Component(inverse) := DomainModel\_corresp\_Component($
$Relation\_definedIn\_DomainModel(RE))$

end

END

4.3 Handling Updates on Event-B Specifications within SysML/KAOS Domain Models

Here, we are interested in handling modifications on Event-B specifications within *SysML/KAOS* domain models. We choose to support only the most repetitive operations that can be performed within the formal specification, the domain model remaining the one to be updated in case of any major changes. Currently supported operations include : addition of sets and of items in existing sets, addition of subsets of existing sets, addition of individuals and of data values, addition of relations and of attributes and finally addition of relation and of attribute maplets.

Addition of Non-Existing Sets

Rules 101-102 : addition of a new abstract set

MACHINE event_b_specs_from_ontologies_ref_1
REFINES event_b_specs_from_ontologies
SEES EventB_Metamodel_Context,Domain_Metamodel_Context

Event rule_101 ⟨ordinary⟩ $\widehat{=}$

handling the addition of a new abstract set (correspondence to a concept)

any

CO

o_CO

where

grd0: $AbstractSet \setminus (ran(Concept\_corresp\_AbstractSet) \cup ran(DataSet\_corresp\_Set)) \neq \emptyset$

grd1: $o\_CO \in AbstractSet \setminus (ran(Concept\_corresp\_AbstractSet) \cup ran(DataSet\_corresp\_Set))$

grd2: $Set\_definedIn\_Component(o\_CO) \in ran(DomainModel\_corresp\_Component)$

grd3: $Concept\_Set \setminus Concept \neq \emptyset$

grd4: $CO \in Concept\_Set \setminus Concept$

then

act1: $Concept := Concept \cup \{CO\}$

act2: $Concept\_corresp\_AbstractSet(CO) := o\_CO$

act3: $Concept\_definedIn\_DomainModel(CO) := DomainModel\_corresp\_Component^{-1}($
$Set\_definedIn\_Component(o\_CO))$

act4: $Concept\_isVariable(CO) := FALSE$

end

Event rule_102 ⟨ordinary⟩ $\widehat{=}$

handling the addition of a new abstract set (correspondence to a custom data set)

any

DS

o_DS

where

grd0: $AbstractSet \setminus (ran(Concept\_corresp\_AbstractSet) \cup ran(DataSet\_corresp\_Set)) \neq \emptyset$

grd1: $o\_DS \in AbstractSet \setminus (ran(Concept\_corresp\_AbstractSet) \cup ran(DataSet\_corresp\_Set))$

grd2: $Set\_definedIn\_Component(o\_DS) \in ran(DomainModel\_corresp\_Component)$

grd3: $DataSet\_Set \setminus DataSet \neq \emptyset$

grd4: $DS \in DataSet\_Set \setminus DataSet$

grd5: $DS \notin \{\_NATURAL, \_INTEGER, \_FLOAT, \_BOOL, \_STRING\}$

then

act1: $CustomDataSet := CustomDataSet \cup \{DS\}$

act2: $DataSet := DataSet \cup \{DS\}$

act3: $CustomDataSet\_corresp\_AbstractSet(DS) := o\_DS$

act4: $DataSet\_definedIn\_DomainModel(DS) := DomainModel\_corresp\_Component^{-1}($
$Set\_definedIn\_Component(o\_DS))$

act5: $DataSet\_corresp\_Set(DS) := o\_DS$

end

END

Rule 103 : addition of an enumerated set

MACHINE event_b_specs_from_ontologies_ref_1
REFINES event_b_specs_from_ontologies
SEES EventB_Metamodel_Context,Domain_Metamodel_Context
Event rule_103 ⟨ordinary⟩ $\widehat{=}$

handling the addition of an enumerated set

any

EDS

o_EDS

elements

o_elements

mapping_elements_o_elements

where

grd0: $EnumeratedSet \setminus ran(DataSet\_corresp\_Set) \neq \emptyset$

grd1: $o\_EDS \in EnumeratedSet \setminus ran(DataSet\_corresp\_Set)$

grd2: $Set\_definedIn\_Component(o\_EDS) \in ran(DomainModel\_corresp\_Component)$

grd3: $DataSet\_Set \setminus DataSet \neq \emptyset$

grd4: $EDS \in DataSet\_Set \setminus DataSet$
grd5: $DataValue\_Set \setminus DataValue \neq \emptyset$
grd6: $elements \subseteq DataValue\_Set \setminus DataValue$
grd7: $o\_elements = SetItem\_itemOf\_EnumeratedSet^{-1}[\{o\_EDS\}]$
grd8: $card(o\_elements) = card(elements)$
grd9: $mapping\_elements\_o\_elements \in elements \rightarrowtail\!\!\!\!\twoheadrightarrow o\_elements$
grd10: $ran(DataValue\_corresp\_SetItem) \cap o\_elements = \emptyset$
grd11: $EDS \notin \{\_NATURAL, \_INTEGER, \_FLOAT, \_BOOL, \_STRING\}$

then

act1: $EnumeratedDataSet := EnumeratedDataSet \cup \{EDS\}$
act2: $DataSet := DataSet \cup \{EDS\}$
act3: $EnumeratedDataSet\_corresp\_EnumeratedSet(EDS) := o\_EDS$
act4: $DataSet\_definedIn\_DomainModel(EDS) := DomainModel\_corresp\_Component^{-1}(Set\_definedIn\_Component(o\_EDS))$
act5: $DataValue := DataValue \cup elements$
act6: $DataValue\_elements\_EnumeratedDataSet := DataValue\_elements\_EnumeratedDataSet \cup \{(xx \mapsto yy) | xx \in elements \wedge yy = EDS\}$
act7: $DataValue\_corresp\_SetItem := DataValue\_corresp\_SetItem \cup mapping\_elements\_o\_elements$
act8: $DataSet\_corresp\_Set := DataSet\_corresp\_Set \Leftarrow \{EDS \mapsto o\_EDS\}$
act9: $DataValue\_valueOf\_DataSet := DataValue\_valueOf\_DataSet \cup \{(xx \mapsto yy) | xx \in elements \wedge yy = EDS\}$
act10: $CustomDataSet := CustomDataSet \cup \{EDS\}$

end

END

Addition of Non-Existing Set Items or Constants

Rule 104 : addition of a set item

MACHINE event_b_specs_from_ontologies_ref_1
REFINES event_b_specs_from_ontologies
SEES EventB_Metamodel_Context,Domain_Metamodel_Context
Event rule_104 ⟨ordinary⟩ $\widehat{=}$

handling the addition of a new element in an existing enumerated set

any

EDS
o_EDS
element
o_element

where

grd0: $dom(SetItem\_itemOf\_EnumeratedSet) \setminus ran(DataValue\_corresp\_SetItem) \neq \emptyset$
grd1: $o\_element \in dom(SetItem\_itemOf\_EnumeratedSet) \setminus ran(DataValue\_corresp\_SetItem)$
grd2: $o\_EDS = SetItem\_itemOf\_EnumeratedSet(o\_element)$
grd3: $o\_EDS \in ran(EnumeratedDataSet\_corresp\_EnumeratedSet)$
grd4: $EDS = EnumeratedDataSet\_corresp\_EnumeratedSet^{-1}(o\_EDS)$
grd5: $DataValue\_Set \setminus DataValue \neq \emptyset$
grd6: $element \in DataValue\_Set \setminus DataValue$

then

act1: $DataValue := DataValue \cup \{element\}$
act2: $DataValue\_elements\_EnumeratedDataSet(element) := EDS$
act3: $DataValue\_corresp\_SetItem(element) := o\_element$
act4: $DataValue\_valueOf\_DataSet(element) := EDS$

end

END

Rule 105 : addition of a constant, sub set of an instance of Concept

MACHINE event_b_specs_from_ontologies_ref_1

REFINES event_b_specs_from_ontologies
SEES EventB_Metamodel_Context,Domain_Metamodel_Context
Event rule_105_1 ⟨ordinary⟩ $\widehat{=}$
handling the addition of a constant, sub set of an instance of Concept (case where the concept corresponds to an abstract set)
any
CO
o_CO
PCO
o_lg
o_PCO
where
grd0: $dom(Constant\_typing\_Property) \setminus ran(Concept\_corresp\_Constant) \neq \emptyset$
grd1: $o\_CO \in dom(Constant\_typing\_Property) \setminus ran(Concept\_corresp\_Constant)$
grd2: $o\_lg = Constant\_typing\_Property(o\_CO)$
grd3: $LogicFormula\_uses\_Operators(o\_lg) = \{1 \mapsto Inclusion\_OP\}$
grd4: $LogicFormula\_involves\_Sets(o\_lg) \neq \emptyset$
grd5: $(2 \mapsto o\_PCO) \in LogicFormula\_involves\_Sets(o\_lg)$
grd6: $o\_PCO \in ran(Concept\_corresp\_AbstractSet)$
grd7: $PCO = Concept\_corresp\_AbstractSet^{-1}(o\_PCO)$
grd8: $Concept\_Set \setminus Concept \neq \emptyset$
grd9: $CO \in Concept\_Set \setminus Concept$
grd10: $Constant\_definedIn\_Component(o\_CO) \in ran(DomainModel\_corresp\_Component)$
then
act1: $Concept := Concept \cup \{CO\}$
act2: $Concept\_corresp\_Constant(CO) := o\_CO$
act3: $Concept\_definedIn\_DomainModel(CO) := DomainModel\_corresp\_Component^{-1}($
$Constant\_definedIn\_Component(o\_CO))$
act4: $Concept\_parentConcept\_Concept(CO) := PCO$
act5: $Concept\_isVariable(CO) := FALSE$
end
Event rule_105_2 ⟨ordinary⟩ $\widehat{=}$
handling the addition of a constant, sub set of an instance of Concept (case where the concept corresponds to a constant)
any
CO
o_CO
PCO
o_lg
o_PCO
where
grd0: $dom(Constant\_typing\_Property) \setminus ran(Concept\_corresp\_Constant) \neq \emptyset$
grd1: $o\_CO \in dom(Constant\_typing\_Property) \setminus ran(Concept\_corresp\_Constant)$
grd2: $o\_lg = Constant\_typing\_Property(o\_CO)$
grd3: $LogicFormula\_uses\_Operators(o\_lg) = \{1 \mapsto Inclusion\_OP\}$
grd4: $LogicFormula\_involves\_Sets(o\_lg) = \emptyset$
grd5: $o\_PCO \in dom(Constant\_isInvolvedIn\_LogicFormulas)$
grd6: $(2 \mapsto o\_lg) \in Constant\_isInvolvedIn\_LogicFormulas(o\_PCO)$
grd7: $o\_PCO \in ran(Concept\_corresp\_Constant)$
grd8: $PCO = Concept\_corresp\_Constant^{-1}(o\_PCO)$
grd9: $Concept\_Set \setminus Concept \neq \emptyset$
grd10: $CO \in Concept\_Set \setminus Concept$
grd11: $Constant\_definedIn\_Component(o\_CO) \in ran(DomainModel\_corresp\_Component)$
then
act1: $Concept := Concept \cup \{CO\}$
act2: $Concept\_corresp\_Constant(CO) := o\_CO$

act3: $Concept\_definedIn\_DomainModel(CO) := DomainModel\_corresp\_Component^{-1}($
$Constant\_definedIn\_Component(o\_CO))$
act4: $Concept\_parentConcept\_Concept(CO) := PCO$
act5: $Concept\_isVariable(CO) := FALSE$

end

END

Rule 106 : addition of an individual

MACHINE event_b_specs_from_ontologies_ref_1
REFINES event_b_specs_from_ontologies
SEES EventB_Metamodel_Context,Domain_Metamodel_Context
Event rule_106_1 ⟨ordinary⟩ $\widehat{=}$

handling the addition of an individual (case where the concept corresponds to an abstract set)

any

ind
o_ind
CO
o_lg
o_CO

where

grd0: $dom(Constant\_typing\_Property) \setminus ran(Individual\_corresp\_Constant) \neq \emptyset$
grd1: $o\_ind \in dom(Constant\_typing\_Property) \setminus ran(Individual\_corresp\_Constant)$
grd2: $o\_lg = Constant\_typing\_Property(o\_ind)$
grd3: $LogicFormula\_uses\_Operators(o\_lg) = \{1 \mapsto Belonging\_OP\}$
grd4: $LogicFormula\_involves\_Sets(o\_lg) \neq \emptyset$
grd5: $(2 \mapsto o\_CO) \in LogicFormula\_involves\_Sets(o\_lg)$
grd6: $o\_CO \in ran(Concept\_corresp\_AbstractSet)$
grd7: $CO = Concept\_corresp\_AbstractSet^{-1}(o\_CO)$
grd8: $Individual\_Set \setminus Individual \neq \emptyset$
grd9: $ind \in Individual\_Set \setminus Individual$

then

act1: $Individual := Individual \cup \{ind\}$
act2: $Individual\_individualOf\_Concept(ind) := CO$
act3: $Individual\_corresp\_Constant(ind) := o\_ind$

end

Event rule_106_2 ⟨ordinary⟩ $\widehat{=}$

handling the addition of an individual (case where the concept corresponds to a constant)

any

ind
o_ind
CO
o_lg
o_CO

where

grd0: $dom(Constant\_typing\_Property) \setminus ran(Individual\_corresp\_Constant) \neq \emptyset$
grd1: $o\_ind \in dom(Constant\_typing\_Property) \setminus ran(Individual\_corresp\_Constant)$
grd2: $o\_lg = Constant\_typing\_Property(o\_ind)$
grd3: $LogicFormula\_uses\_Operators(o\_lg) = \{1 \mapsto Belonging\_OP\}$
grd4: $LogicFormula\_involves\_Sets(o\_lg) = \emptyset$
grd5: $o\_CO \in dom(Constant\_isInvolvedIn\_LogicFormulas)$
grd6: $(2 \mapsto o\_lg) \in Constant\_isInvolvedIn\_LogicFormulas(o\_CO)$
grd7: $o\_CO \in ran(Concept\_corresp\_Constant)$
grd8: $CO = Concept\_corresp\_Constant^{-1}(o\_CO)$
grd9: $Individual\_Set \setminus Individual \neq \emptyset$
grd10: $ind \in Individual\_Set \setminus Individual$

then

act1: $Individual := Individual \cup \{ind\}$

act2: $Individual\_individualOf\_Concept(ind) := CO$

act3: $Individual\_corresp\_Constant(ind) := o\_ind$

end

END

Rule 107 : addition of a data value

MACHINE event_b_specs_from_ontologies_ref_1
REFINES event_b_specs_from_ontologies
SEES EventB_Metamodel_Context,Domain_Metamodel_Context
Event rule_107 ⟨ordinary⟩ $\widehat{=}$

handling the addition of a data value

any

dva
o_dva
DS
o_lg
o_DS

where

grd0: $dom(Constant\_typing\_Property) \setminus ran(DataValue\_corresp\_Constant) \neq \emptyset$

grd1: $o\_dva \in dom(Constant\_typing\_Property) \setminus ran(DataValue\_corresp\_Constant)$

grd2: $o\_lg = Constant\_typing\_Property(o\_dva)$

grd3: $LogicFormula\_uses\_Operators(o\_lg) = \{1 \mapsto Belonging\_OP\}$

grd4: $LogicFormula\_involves\_Sets(o\_lg) \neq \emptyset$

grd5: $(2 \mapsto o\_DS) \in LogicFormula\_involves\_Sets(o\_lg)$

grd6: $o\_DS \in ran(DataSet\_corresp\_Set)$

grd7: $DS = DataSet\_corresp\_Set^{-1}(o\_DS)$

grd8: $DataValue\_Set \setminus DataValue \neq \emptyset$

grd9: $dva \in DataValue\_Set \setminus DataValue$

then

act1: $DataValue := DataValue \cup \{dva\}$

act2: $DataValue\_valueOf\_DataSet(dva) := DS$

act3: $DataValue\_corresp\_Constant(dva) := o\_dva$

end

END

Addition of Non-Existing Variables

Rule 108 : addition of a variable, sub set of an instance of Concept

MACHINE event_b_specs_from_ontologies_ref_1
REFINES event_b_specs_from_ontologies
SEES EventB_Metamodel_Context,Domain_Metamodel_Context
Event rule_108_1 ⟨ordinary⟩ $\widehat{=}$

handling the addition of a variable, sub set of an instance of Concept (case where the concept corresponds to an abstract set)

any

x_CO
CO
o_lg
o_CO

where

grd0: $dom(Variable\_typing\_Invariant) \setminus ran(Concept\_corresp\_Variable) \neq \emptyset$

grd1: $x\_CO \in dom(Variable\_typing\_Invariant) \setminus ran(Concept\_corresp\_Variable)$

grd2: $o\_lg = Variable\_typing\_Invariant(x\_CO)$

grd3: $LogicFormula\_uses\_Operators(o\_lg) = \{1 \mapsto Inclusion\_OP\}$

grd4: $LogicFormula\_involves\_Sets(o\_lg) \neq \emptyset$

grd5: $(2 \mapsto o\_CO) \in LogicFormula\_involves\_Sets(o\_lg)$

grd6: $o\_CO \in ran(Concept\_corresp\_AbstractSet)$
grd7: $CO = Concept\_corresp\_AbstractSet^{-1}(o\_CO)$
then
act1: $Concept\_isVariable(CO) := TRUE$
act2: $Concept\_corresp\_Variable(CO) := x\_CO$
end
Event rule_108_2 ⟨ordinary⟩ $\widehat{=}$
handling the addition of a variable, sub set of an instance of Concept (case where the concept corresponds to a constant)
any
x_CO
CO
o_lg
o_CO
where
grd0: $dom(Variable\_typing\_Invariant) \setminus ran(Concept\_corresp\_Variable) \neq \emptyset$
grd1: $x\_CO \in dom(Variable\_typing\_Invariant) \setminus ran(Concept\_corresp\_Variable)$
grd2: $o\_lg = Variable\_typing\_Invariant(x\_CO)$
grd3: $LogicFormula\_uses\_Operators(o\_lg) = \{1 \mapsto Inclusion\_OP\}$
grd4: $LogicFormula\_involves\_Sets(o\_lg) = \emptyset$
grd5: $o\_CO \in dom(Constant\_isInvolvedIn\_LogicFormulas)$
grd6: $(2 \mapsto o\_lg) \in Constant\_isInvolvedIn\_LogicFormulas(o\_CO)$
grd7: $o\_CO \in ran(Concept\_corresp\_Constant)$
grd8: $CO = Concept\_corresp\_Constant^{-1}(o\_CO)$
then
act1: $Concept\_isVariable(CO) := TRUE$
act2: $Concept\_corresp\_Variable(CO) := x\_CO$
end
END

4.4 The SysML/KAOS Domain Model Parser Tool

The correspondence rules outlined here have been implemented within an open source tool called *SysML/KAOS Domain Model Parser* [20]. It allows the construction of domain models *(Fig. 15)* and generates the corresponding *Event-B* specifications *(Fig. 16)*. It is build through *Jetbrains Meta Programming System* [12], a tool to design domain specific languages using language-oriented programming.

5 Conclusion and Future Works

This paper was focused on a presentation of mapping rules between SysML/KAOS domain models and Event-B specifications illustrated through a case study dealing with a landing gear system. The specifications thus obtained can also be seen as a formal semantics for SysML/KAOS domain models. They complement the formalization of the SysML/KAOS goal model by providing a description of the state of the system.

Work in progress is aimed at integrating our approach, implemented through the *SysML/KAOS Domain Model Parser* tool, within the open-source platform *Openflexo* [16].

Acknowledgment

This work is carried out within the framework of the *FORMOSE* project [4] funded by the French National Research Agency (ANR).

References

1. Abrial, J.: Modeling in Event-B - System and Software Engineering. Cambridge University Press (2010)

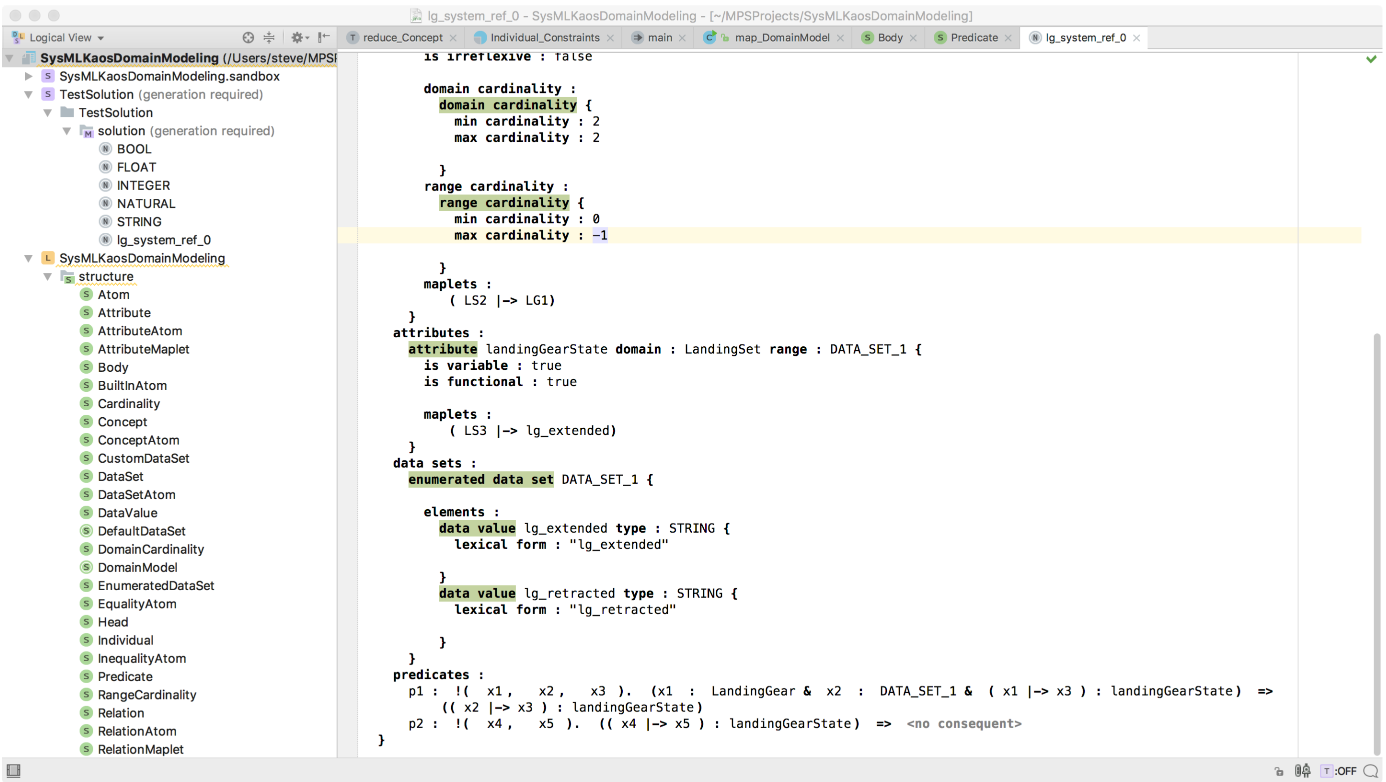

Fig. 15. Preview of the SysML/KAOS Domain Model Parser Tool

2. Alkhammash, E., Butler, M.J., Fathabadi, A.S., Cîrstea, C.: Building traceable Event-B models from requirements. Sci. Comput. Program. 111, 318–338 (2015)
3. Alkhammash, Eman H.: Derivation of Event-B Models from OWL Ontologies. MATEC Web Conf. 76, 04008 (2016)
4. ANR-14-CE28-0009: Formose ANR project (2017), `http://formose.lacl.fr/`
5. Bjørner, D., Eir, A.: Compositionality: Ontology and mereology of domains. Essays in Honor of Willem-Paul de Roever, LNCS, vol. 5930, pp. 22–59. Springer (2010)
6. Boniol, F., Wiels, V.: The landing gear system case study. pp. 1–18. ABZ, Springer (2014)
7. ClearSy: Atelier B: B System (2014), `http://clearsy.com/`
8. Doberkat, E.: The Object-Z specification language. Softwaretechnik-Trends 21(1) (2001)
9. Dong, J.S., Sun, J., Wang, H.H.: Z approach to semantic web. In: Formal Methods and Software Engineering - ICFEM, LNCS. vol. 2495, pp. 156–167. Springer (2002)
10. Gnaho, C., Semmak, F.: Une extension SysML pour l'ingénierie des exigences dirigée par les buts. In: 28e Congrès INFORSID, France. pp. 277–292 (2010)
11. van Harmelen, F., Patel-Schneider, P.F., Horrocks, I.: Reference description of the DAML+ OIL ontology markup language (2001)
12. Jetbrains: Jetbrains mps (2017), `https://www.jetbrains.com/mps/`
13. van Lamsweerde, A.: Requirements Engineering - From System Goals to UML Models to Software Specifications. Wiley (2009)
14. Mammar, A., Laleau, R.: On the use of domain and system knowledge modeling in goal-based Event-B specifications. In: ISoLA 2016, LNCS. pp. 325–339. Springer (2016)
15. Matoussi, A., Gervais, F., Laleau, R.: A goal-based approach to guide the design of an abstract Event-B specification. In: ICECCS 2011. pp. 139–148. IEEE Computer Society (2011)
16. Openflexo: Openflexo project (2015), `http://www.openflexo.org`
17. Poernomo, I., Umarov, T.: A mapping from normative requirements to Event-B to facilitate verified data-centric business process management. CEE-SET LNCS, vol. 7054, pp. 136–149. Springer (2009)
18. Sengupta, K., Hitzler, P.: Web ontology language (OWL). In: Encyclopedia of Social Network Analysis and Mining, pp. 2374–2378 (2014)

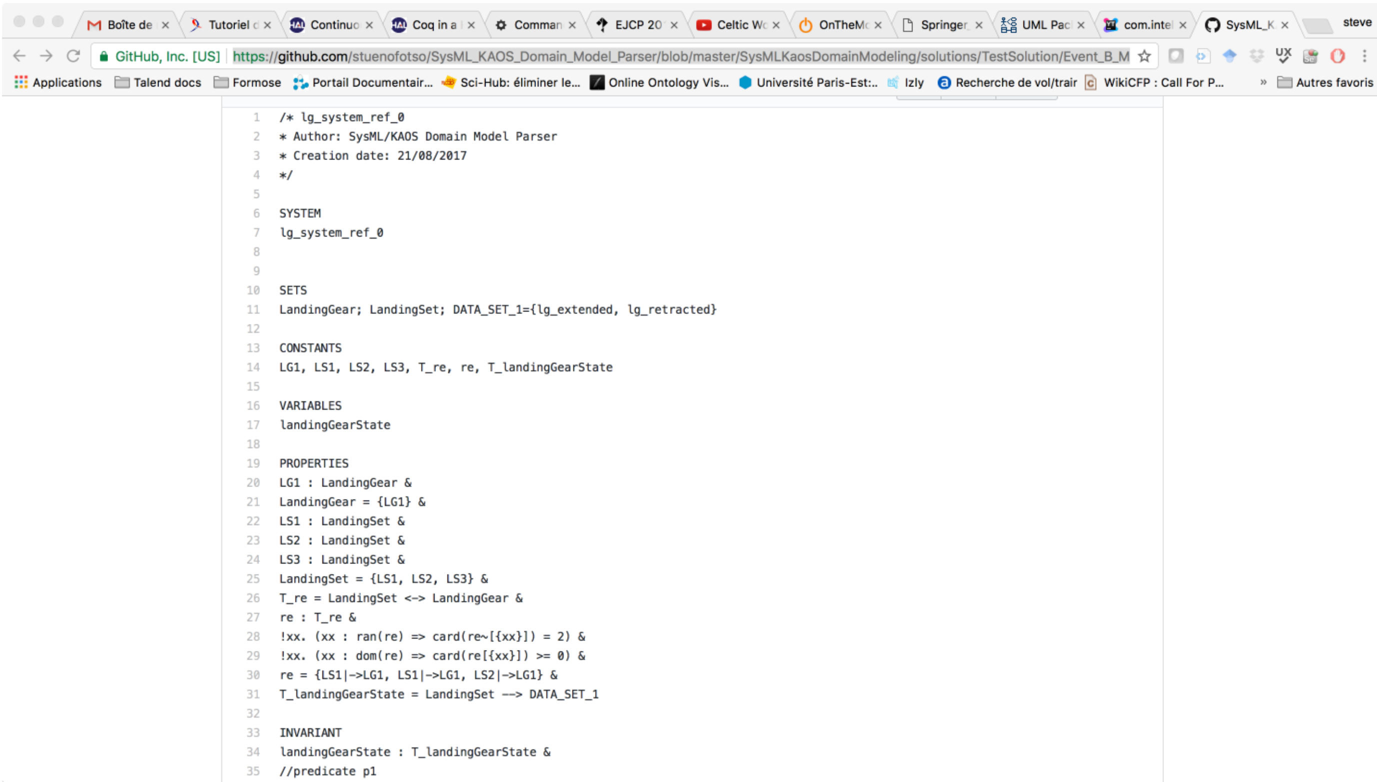

Fig. 16. Preview of *B System* Specifications Generated by the SysML/KAOS Domain Model Parser Tool for the Landing Gear System Case Study

19. Snook, C., Butler, M.: UML-B: Formal Modeling and Design Aided by UML. ACM Trans. Softw. Eng. Methodol. 15(1), 92–122 (Jan 2006)
20. Tueno, S.: SysML/KAOS Domain Model Parser (2017), `https://github.com/stuenofotso/SysML_KAOS_Domain_Model_Parser`
21. Tueno, S., Laleau, R., Mammar, A., Frappier, M.: Towards Using Ontologies for Domain Modeling within the SysML/KAOS Approach. IEEE proceedings of MoDRE workshop, 25th IEEE International Requirements Engineering Conference (2017)
22. UL, I.: Owlgred home (2017), `http://owlgred.lumii.lv/`
23. Wang, H.H., Damljanovic, D., Sun, J.: Enhanced semantic access to formal software models. In: Formal Methods and Software Engineering - ICFEM, LNCS. vol. 6447, pp. 237–252. Springer (2010)