



HAL
open science

Formal representation of SysML/KAOS domain model

Steve Jeffrey Tueno Fotso, Amel Mammar, Régine Laleau, Marc Frappier

► **To cite this version:**

Steve Jeffrey Tueno Fotso, Amel Mammar, Régine Laleau, Marc Frappier. Formal representation of SysML/KAOS domain model. [Research Report] LACL, Université Paris-Est/Créteil. 2019. hal-02874826

HAL Id: hal-02874826

<https://hal.science/hal-02874826>

Submitted on 19 Jun 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

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

Steve Jeffrey Tueno Fotso^{1,3}, Amel Mammam², Régine Laleau¹, and Marc Frappier³

¹ Université Paris-Est Créteil, LACL, Créteil, France,
steve.tuenofotso@univ-paris-est.fr, lalaeu@u-pec.fr

² Télécom SudParis, SAMOVAR-CNRS, Evry, France,
amel.mammam@telecom-sudparis.eu

³ Université de Sherbrooke, GRIL, Québec, Canada,
Marc.Frappier@usherbrooke.ca

07.03.2019

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 formal⁴, 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 chosen 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 [20] 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.

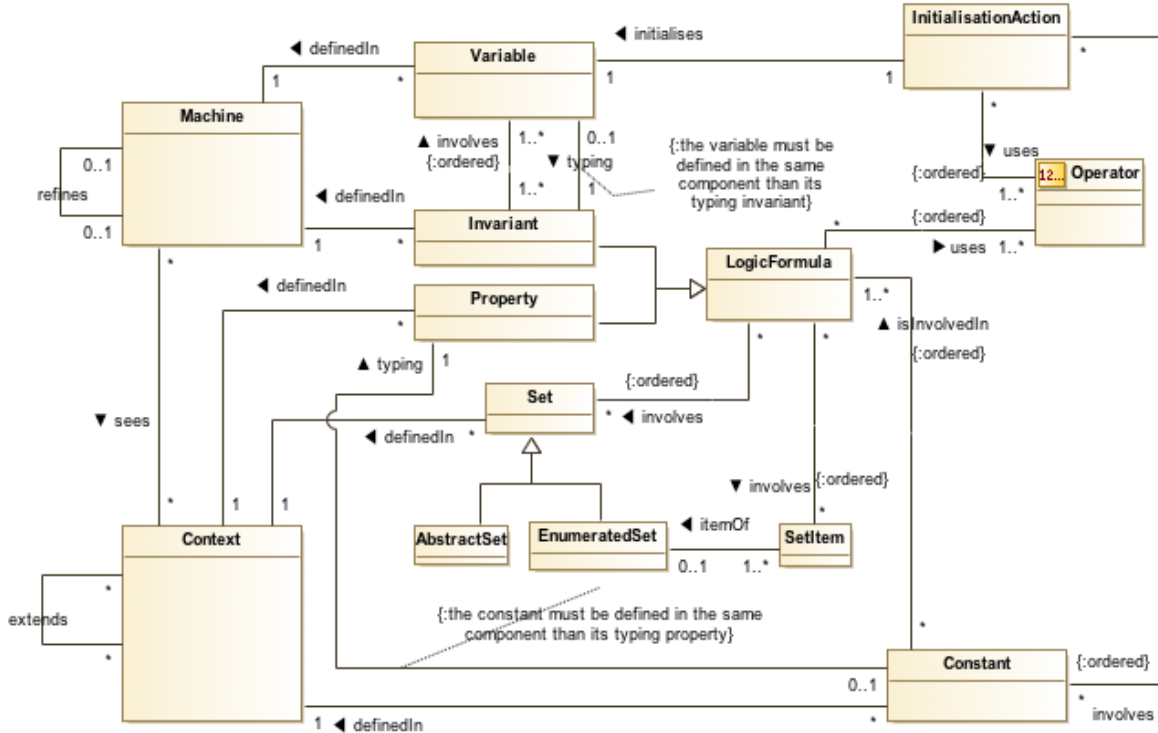


Fig. 1. Our abstraction of the Event-B specification language

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 :
 $(FunctionSet_OP, [op_1, op_2, op_3]) \Leftrightarrow op_1 = op_2 \rightarrow op_3$.
- **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 :
 $(Equal2SetOf_OP, [op_1, op_2, \dots, op_n]) \Leftrightarrow op_1 = \{op_2, \dots, op_n\}$.
- **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 :
 $(BecomeEqual2SetOf_OP, [va, op_2, \dots, op_n]) \Leftrightarrow va := \{op_2, \dots, op_n\}$.
- **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.

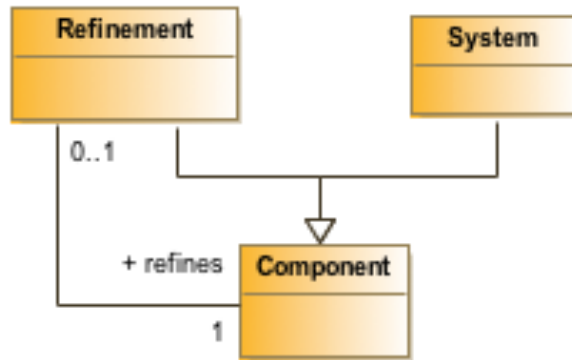


Fig. 2. B System Components

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.

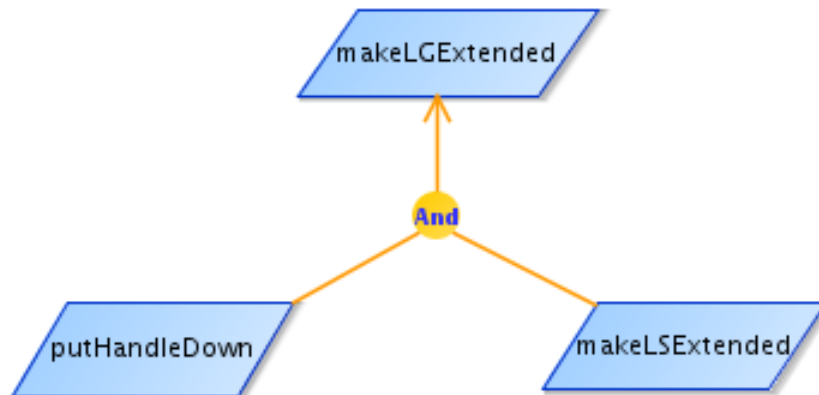


Fig. 3. Excerpt from the landing gear system goal diagram

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 [20] 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,

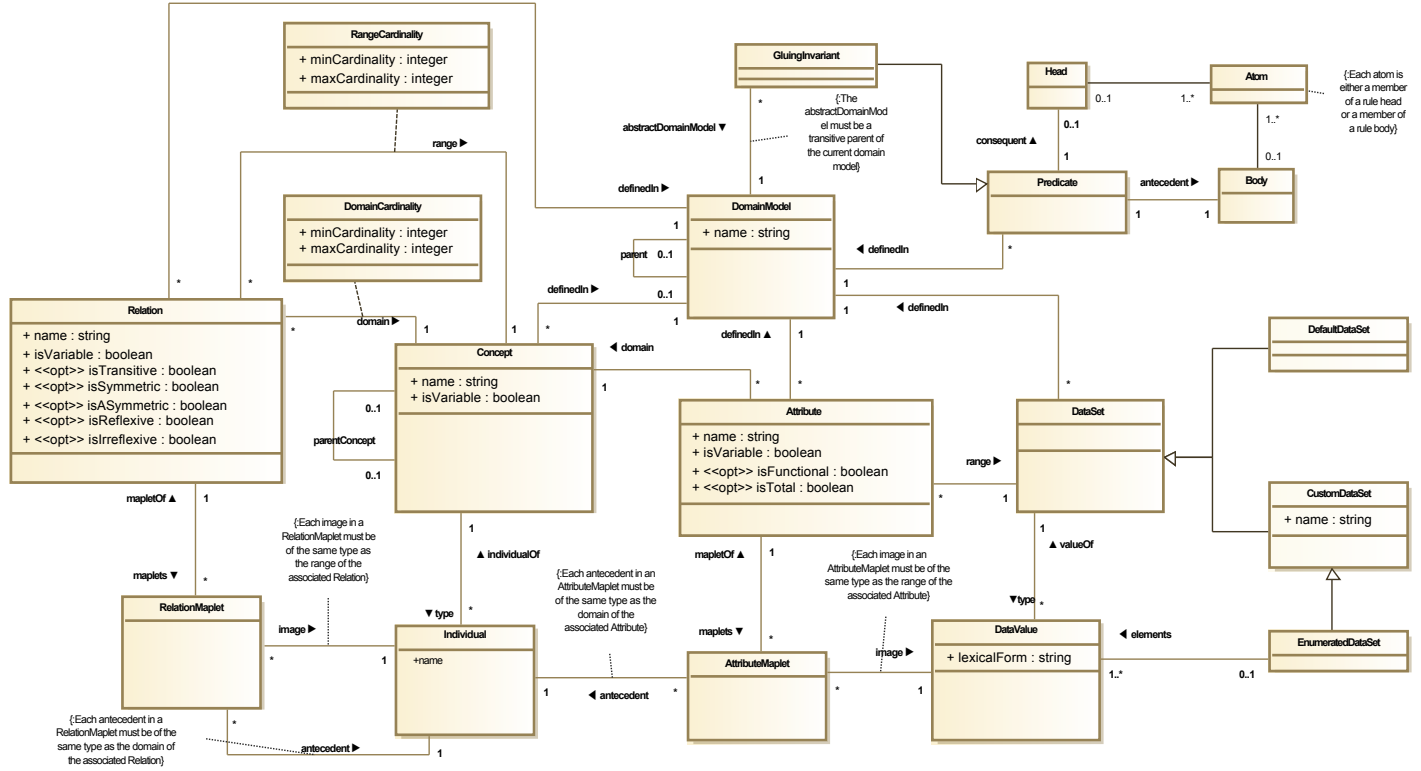


Fig. 5. Metamodel associated with SysML/KAOS domain modeling

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 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

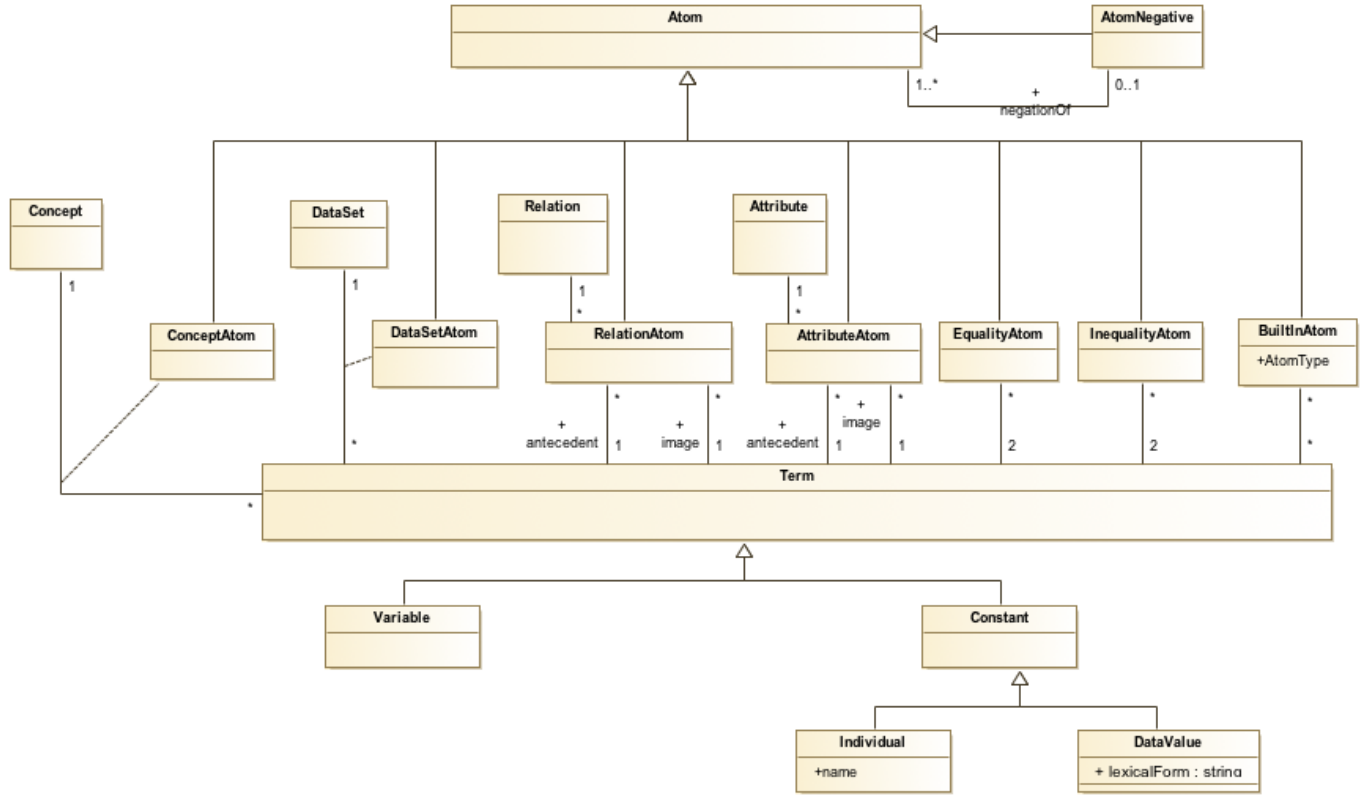


Fig. 6. Extension of the metamodel associated with SysML/KAOS domain modeling for atom specification

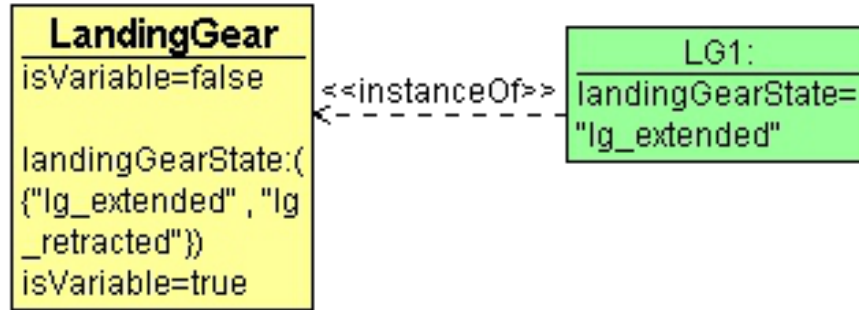


Fig. 7. *lg_system_ref_0*: ontology associated to the root level of the landing gear goal model

representation of particular constraints between several terms. For example, an arithmetic constraint between several integer data values.

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

$$landingGearState(LG1, "lg_retracted") \leftarrow LandingSet(?ls) \wedge landingSetState(?ls, "ls_retracted") \quad (inv1)$$

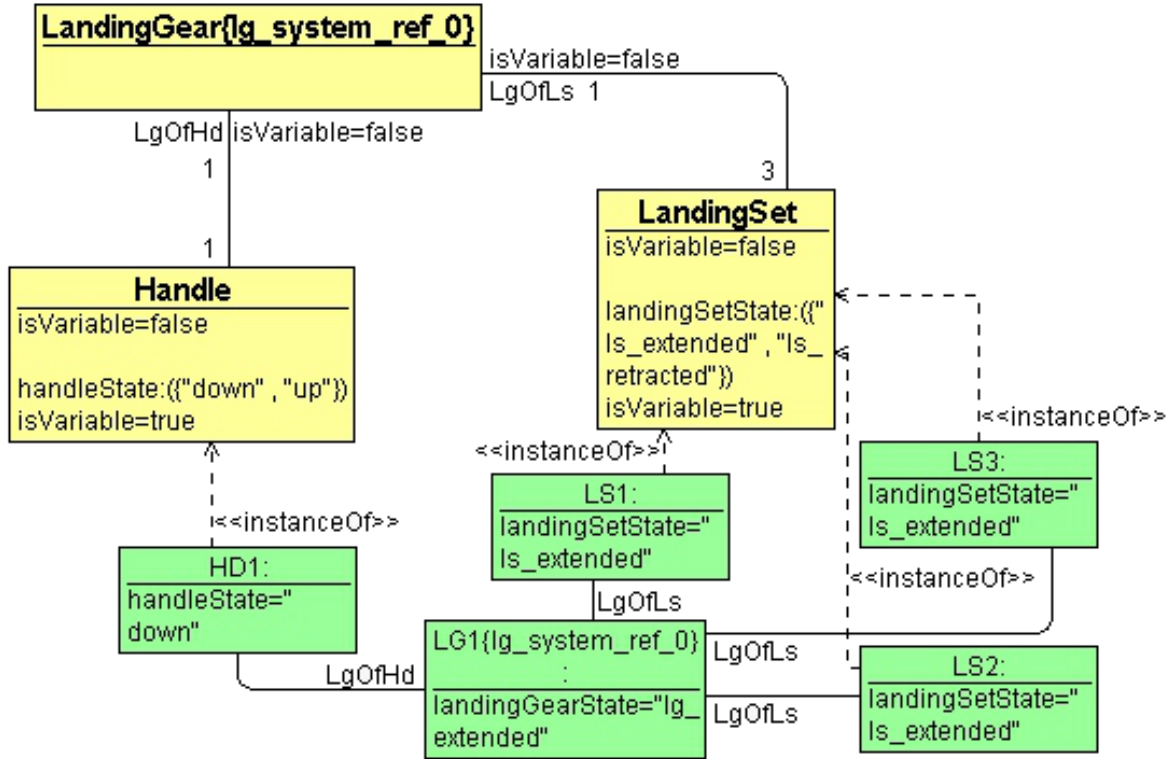


Fig. 8. *lg_system_ref_1*: ontology associated to the first level of refinement of the landing gear goal model

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

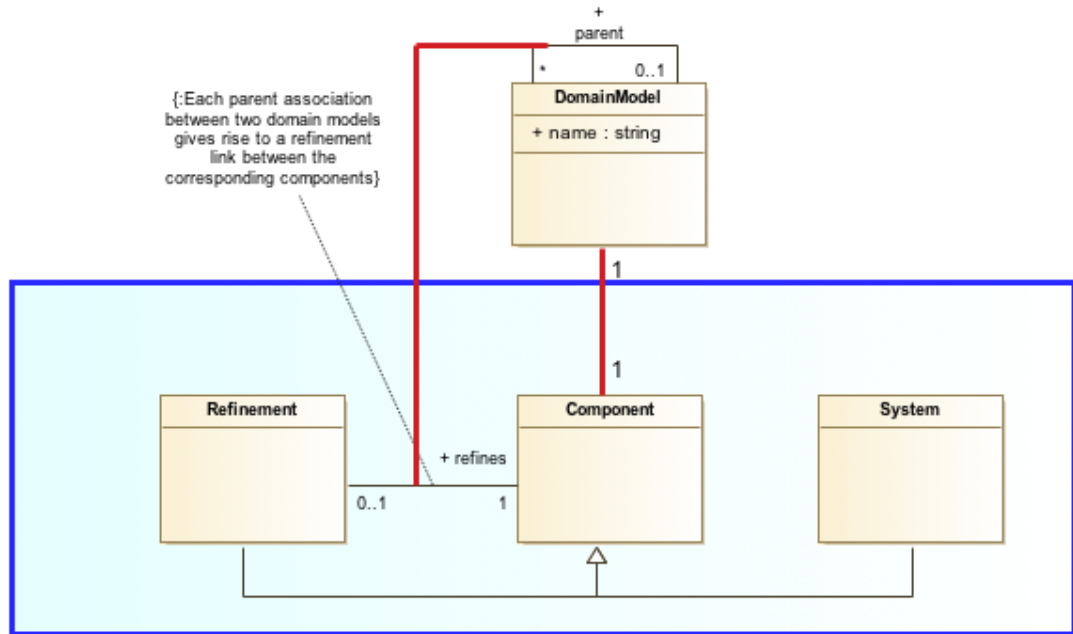


Fig. 9. Correspondence to B System Components

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 suffixed 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* : $Concept_corresp_AbstractSet \in Concept \mapsto AbstractSet$ (*event_b_specs_from_ontologies_ref_1*).
- Each rule is represented as an event by following the correspondence links.

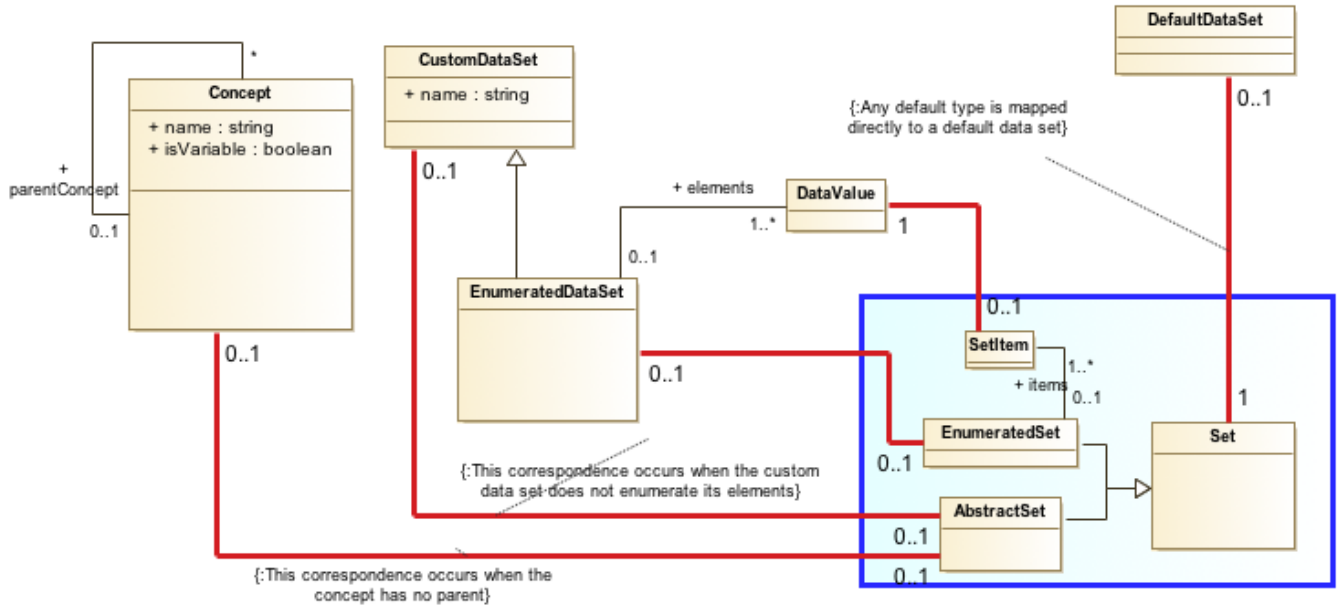


Fig. 10. Correspondence to Sets

- 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.

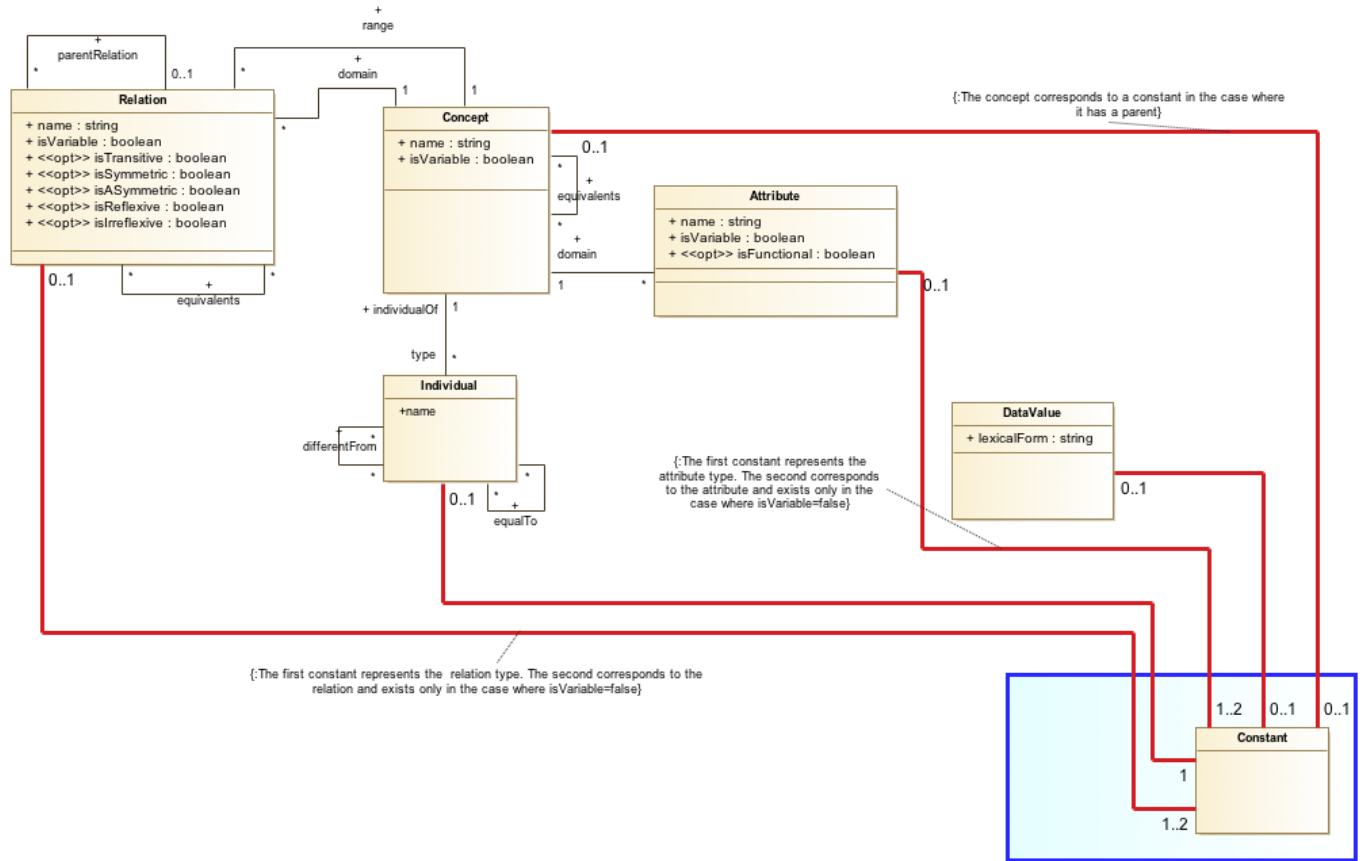


Fig. 11. Correspondence to Constants

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.

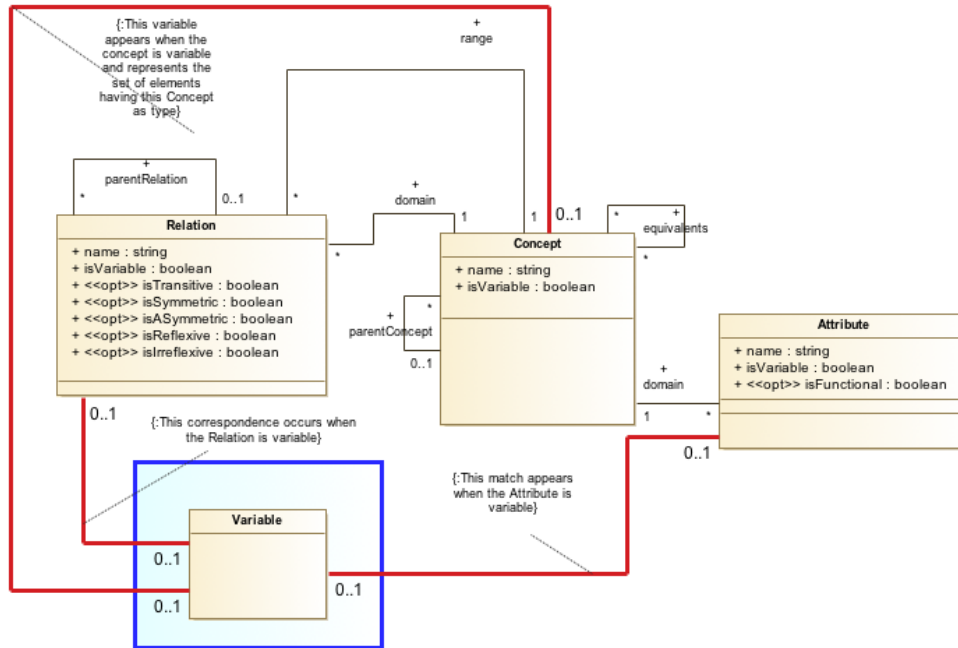


Fig. 12. Correspondence to Variables

```

SYSTEM lg_system_ref_0
SETS LandingGear; DataSet_1= {lg_extended, lg_retracted}
CONSTANTS T_landingGearState, LG1
PROPERTIES
(0.1) LG1 ∈ LandingGear
(0.2) ∧ LandingGear={LG1}
(0.3) ∧ T_landingGearState = LandingGear → DataSet_1
VARIABLES landingGearState
INVARIANT
(0.4) landingGearState ∈ T_landingGearState
INITIALISATION
(0.5) landingGearState := {LG1 ↦ 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: $DomainModel_parent_DomainModel \in DomainModel \leftrightarrow DomainModel$
inv0_5: $DomainModel_corresp_Component \in DomainModel \leftrightarrow Component$
inv0_6: $Refinement_refines_Component \in Refinement \rightarrow Component$
inv0_7:
 $\forall xx \cdot ($
 $\forall px \cdot ($
 $($
 $xx \in dom(DomainModel_parent_DomainModel)$
 $\wedge px = DomainModel_parent_DomainModel(xx)$
 $\wedge px \in dom(DomainModel_corresp_Component)$
 $\wedge xx \notin dom(DomainModel_corresp_Component)$
 $)$
 $\Rightarrow DomainModel_corresp_Component(px) \notin ran(Refinement_refines_Component)$
 $)$
 $)$
 $)$

•••
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: $SetItem_itemOf_EnumeratedSet \in SetItem \rightarrow EnumeratedSet$

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 \rightarrow 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 \rightarrow EnumeratedDataSet
inv1_23: Concept_corresp_AbstractSet \in Concept \rightarrow AbstractSet
inv1_24: EnumeratedDataSet_corresp_EnumeratedSet \in EnumeratedDataSet \rightarrow EnumeratedSet
inv1_25: DataValue_corresp_SetItem \in DataValue \rightarrow SetItem
*inv1_26: $\forall xx.(xx \in EnumeratedDataSet \wedge xx \notin \text{dom}(EnumeratedDataSet_corresp_EnumeratedSet)) \Rightarrow$
 $DataValue_elements_EnumeratedDataSet^{-1}[\{xx\}] \cap \text{dom}(DataValue_corresp_SetItem) = \emptyset$*
inv1_27: CustomDataSet_corresp_AbstractSet \in CustomDataSet \rightarrow AbstractSet
inv1_28: $\{_NATURAL, _INTEGER, _FLOAT, _BOOL, _STRING\} \cap CustomDataSet = \emptyset$
inv1_29: DefaultDataSet_corresp_AbstractSet \in DefaultDataSet \rightarrow AbstractSet
inv1_30: $\{B_NATURAL, B_INTEGER, B_FLOAT, B_BOOL, B_STRING\} \cap EnumeratedSet = \emptyset$
inv1_31: Concept_corresp_Constant \in Concept \rightarrow 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 $(\mathbb{N}_1 \rightarrow Variable)$
 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: $\text{ran}(\text{union}(\text{ran}(\text{Invariant_involves_Variables}))) = Variable$
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: $\forall cons.(cons \in Constant \Rightarrow \text{ran}(\text{Constant_isInvolvedIn_LogicFormulas}(cons)) \cap Property \neq \emptyset)$
inv1_41: LogicFormula_involves_Sets \in LogicFormula \rightarrow $(\mathbb{N}_1 \rightarrow Set)$
inv1_42: LogicFormula_uses_Operators \in LogicFormula \rightarrow $(\mathbb{N}_1 \rightarrow Operator)$
inv1_44: Individual_corresp_Constant \in Individual \rightarrow Constant
inv1_45: DataValue_corresp_Constant \in DataValue \rightarrow Constant
inv1_46: Concept_corresp_Variable \in Concept \rightarrow Variable
inv1_47: InitialisationAction \subseteq InitialisationAction_Set
inv1_49: InitialisationAction_uses_Operators \in InitialisationAction \rightarrow $(\mathbb{N}_1 \rightarrow Operator)$
inv1_50: Variable_init_InitialisationAction \in Variable \rightarrow InitialisationAction
 for initialisation actions, the assigned operand is the involved variable.
inv1_52: InitialisationAction_involves_Constants \in InitialisationAction \rightarrow $(\mathbb{N}_1 \rightarrow Constant)$
 *****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 \rightarrow BOOL
inv1_61: Relation_isSymmetric \in Relation \rightarrow BOOL
inv1_62: relation_isASymmetric \in Relation \rightarrow BOOL
inv1_63: Relation_isReflexive \in Relation \rightarrow BOOL
inv1_64: Relation_isIrreflexive \in Relation \rightarrow BOOL
inv1_65: Relation_DomainCardinality_minCardinality \in Relation \rightarrow \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: $Attribute_isFunctionnal \in Attribute \rightarrow BOOL$
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: $\forall rm. (rm \in RelationMaplet \Rightarrow Individual_individualOf_Concept(RelationMaplet_antecedent_Individual(rm)) = Relation_domain_Concept(RelationMaplet_mapletOf_Relation(rm))) =$
inv1_86: $\forall rm. (rm \in RelationMaplet \Rightarrow Individual_individualOf_Concept(RelationMaplet_image_Individual(rm)) = Relation_range_Concept(RelationMaplet_mapletOf_Relation(rm))) =$
inv1_87: $\forall am. (am \in AttributeMaplet \Rightarrow Individual_individualOf_Concept(AttributeMaplet_antecedent_Individual(am)) = Attribute_domain_Concept(AttributeMaplet_mapletOf_Attribute(am))) =$
inv1_88: $\forall am. (am \in AttributeMaplet \Rightarrow DataValue_valueOf_DataSet(AttributeMaplet_image_DataValue(am)) = Attribute_range_DataSet(AttributeMaplet_mapletOf_Attribute(am))) =$
inv1_89: $Relation_Type \in Relation \rightsquigarrow Constant$
inv1_90: $Relation_corresp_Constant \in Relation \rightsquigarrow Constant$
inv1_91: $Relation_corresp_Variable \in Relation \rightsquigarrow Variable$
inv1_92: $\forall re. (re \in dom(Relation_Type) \Leftrightarrow (re \in dom(Relation_corresp_Constant) \vee (re \in dom(Relation_corresp_Variable))))$

inv1_93: $Attribute_Type \in Attribute \rightsquigarrow Constant$
inv1_94: $Attribute_corresp_Constant \in Attribute \rightsquigarrow Constant$
inv1_95: $Attribute_corresp_Variable \in Attribute \rightsquigarrow Variable$
inv1_96: $\forall re. (re \in dom(Attribute_Type) \Leftrightarrow (re \in dom(Attribute_corresp_Constant) \vee (re \in dom(Attribute_corresp_Variable))))$

inv1_97: $Variable_typing_Invariant \in Variable \rightsquigarrow Invariant$
inv1_98: $Constant_typing_Property \in Constant \rightsquigarrow Property$
inv1_99: $RelationCharacteristic_corresp_LogicFormula \in (Relation \rightarrow RelationCharacteristics_Set) \rightsquigarrow LogicFormula$
inv1_100: $RelationMaplet_corresp_Constant \in RelationMaplet \rightsquigarrow Constant$
inv1_101: $DataSet_corresp_Set \in DataSet \rightsquigarrow Set$
inv1_102: $AttributeMaplet_corresp_Constant \in AttributeMaplet \rightsquigarrow Constant$
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 $\langle \text{ordinary} \rangle \hat{=}$

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:  $o\_DM = \text{DomainModel\_corresp\_Component}(DM)$ 
  then
    act1:  $\text{DefaultDataSet} := \{\_NATURAL, \_INTEGER, \_FLOAT, \_BOOL, \_STRING\}$ 
    act2:  $\text{DataSet} := \text{DataSet} \cup \{\_NATURAL, \_INTEGER, \_FLOAT, \_BOOL, \_STRING\}$ 
    act3:  $\text{DataSet\_definedIn\_DomainModel} := \text{DataSet\_definedIn\_DomainModel} \cup \{(xx \mapsto yy) \mid xx \in \{\_NATURAL, \_INTEGER\}, yy = DM\}$ 
    act4:  $\text{AbstractSet} := \text{AbstractSet} \cup \{B\_NATURAL, B\_INTEGER, B\_FLOAT, B\_BOOL, B\_STRING\}$ 
    act5:  $\text{Set} := \text{Set} \cup \{B\_NATURAL, B\_INTEGER, B\_FLOAT, B\_BOOL, B\_STRING\}$ 
    act6:  $\text{DefaultDataSet\_corresp\_AbstractSet} := \{\_NATURAL \mapsto B\_NATURAL, \_INTEGER \mapsto B\_INTEGER, \_FLOAT \mapsto B\_FLOAT, \_BOOL \mapsto B\_BOOL, \_STRING \mapsto B\_STRING\}$ 
    act7:  $\text{Set\_definedIn\_Component} := \text{Set\_definedIn\_Component} \cup \{(xx \mapsto yy) \mid xx \in \{B\_NATURAL, B\_INTEGER, B\_FLOAT, B\_BOOL, B\_STRING\} \wedge yy = o\_DM\}$ 
    act8:  $\text{DataSet\_corresp\_Set} := \text{DataSet\_corresp\_Set} \leftarrow \{\_NATURAL \mapsto B\_NATURAL, \_INTEGER \mapsto B\_INTEGER, \_FLOAT \mapsto B\_FLOAT, \_BOOL \mapsto B\_BOOL, \_STRING \mapsto B\_STRING\}$ 
  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) $\hat{=}$ correspondence of a domain model not associated to a parent domain model

```

  any
    DM
    o_DM
  where
    grd0:  $\text{DomainModel} \setminus (\text{dom}(\text{DomainModel\_corresp\_Component}) \cup \text{dom}(\text{DomainModel\_parent\_DomainModel})) \neq \emptyset$ 
    grd1:  $DM \in \text{DomainModel}$ 
    grd2:  $DM \notin \text{dom}(\text{DomainModel\_corresp\_Component})$ 
    grd3:  $DM \notin \text{dom}(\text{DomainModel\_parent\_DomainModel})$ 
    grd4:  $\text{Component\_Set} \setminus \text{Component} \neq \emptyset$ 
    grd5:  $o\_DM \in \text{Component\_Set} \setminus \text{Component}$ 
  then
    act1:  $\text{System} := \text{System} \cup \{o\_DM\}$ 
    act2:  $\text{Component} := \text{Component} \cup \{o\_DM\}$ 
    act3:  $\text{DomainModel\_corresp\_Component}(DM) := o\_DM$ 
  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) $\hat{=}$

correspondence of a domain model associated to a parent domain model

```

  any
    DM
    PDM
    o_DM

```

```

where
  grd0:  $dom(\text{DomainModel\_parent\_DomainModel}) \setminus dom(\text{DomainModel\_corresp\_Component}) \neq \emptyset$ 
  grd1:  $DM \in dom(\text{DomainModel\_parent\_DomainModel})$ 
  grd2:  $DM \notin dom(\text{DomainModel\_corresp\_Component})$ 
  grd3:  $dom(\text{DomainModel\_corresp\_Component}) \neq \emptyset$ 
  grd4:  $PDM \in dom(\text{DomainModel\_corresp\_Component})$ 
  grd5:  $\text{DomainModel\_parent\_DomainModel}(DM) = PDM$ 
  grd6:  $\text{Component\_Set} \setminus \text{Component} \neq \emptyset$ 
  grd7:  $o\_DM \in \text{Component\_Set} \setminus \text{Component}$ 
then
  act1:  $\text{Refinement} := \text{Refinement} \cup \{o\_DM\}$ 
  act2:  $\text{Component} := \text{Component} \cup \{o\_DM\}$ 
  act3:  $\text{Refinement\_refines\_Component}(o\_DM) := \text{DomainModel\_corresp\_Component}(PDM)$ 
  act4:  $\text{DomainModel\_corresp\_Component}(DM) := o\_DM$ 
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)  $\hat{=}$ 
  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)  $\hat{=}$ 
  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:  $mapping\_elements\_o\_elements \in elements \mapsto o\_elements$ 

```



```

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  $\leftarrow$  {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)  $\hat{=}$ 
  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  $\leftarrow$  {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)  $\hat{=}$ 
  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: $\text{dom}(\text{Concept_parentConcept_Concept}) \setminus \text{dom}(\text{Concept_corresp_Constant}) \neq \emptyset$
grd1: $CO \in \text{dom}(\text{Concept_parentConcept_Concept}) \setminus \text{dom}(\text{Concept_corresp_Constant})$
grd2: $\text{dom}(\text{Concept_corresp_AbstractSet}) \neq \emptyset$
grd3: $PCO \in \text{dom}(\text{Concept_corresp_AbstractSet})$
grd4: $\text{Concept_parentConcept_Concept}(CO) = PCO$
grd5: $\text{Concept_definedIn_DomainModel}(CO) \in \text{dom}(\text{DomainModel_corresp_Component})$
grd6: $\text{Constant_Set} \setminus \text{Constant} \neq \emptyset$
grd7: $o_CO \in \text{Constant_Set} \setminus \text{Constant}$
grd8: $\text{LogicFormula_Set} \setminus \text{LogicFormula} \neq \emptyset$
grd9: $o_lg \in \text{LogicFormula_Set} \setminus \text{LogicFormula}$
grd10: $o_PCO \in \text{AbstractSet}$
grd11: $o_PCO = \text{Concept_corresp_AbstractSet}(PCO)$

then
act1: $\text{Constant} := \text{Constant} \cup \{o_CO\}$
act2: $\text{Concept_corresp_Constant}(CO) := o_CO$
act3: $\text{Constant_definedIn_Component}(o_CO) := \text{DomainModel_corresp_Component}(\text{Concept_definedIn_DomainModel}(CO))$
act4: $\text{Property} := \text{Property} \cup \{o_lg\}$
act5: $\text{LogicFormula} := \text{LogicFormula} \cup \{o_lg\}$
act6: $\text{LogicFormula_uses_Operators}(o_lg) := \{1 \mapsto \text{Inclusion_OP}\}$
act7: $\text{Constant_isInvolvedIn_LogicFormulas}(o_CO) := \{1 \mapsto o_lg\}$
act8: $\text{LogicFormula_involves_Sets}(o_lg) := \{2 \mapsto o_PCO\}$
act9: $\text{LogicFormula_definedIn_Component}(o_lg) := \text{DomainModel_corresp_Component}(\text{Concept_definedIn_DomainModel}(CO))$
act10: $\text{Constant_typing_Property}(o_CO) := o_lg$

end
Event rule_6_2 (ordinary) $\hat{=}$
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: $\text{dom}(\text{Concept_parentConcept_Concept}) \setminus \text{dom}(\text{Concept_corresp_Constant}) \neq \emptyset$
grd1: $CO \in \text{dom}(\text{Concept_parentConcept_Concept}) \setminus \text{dom}(\text{Concept_corresp_Constant})$
grd2: $\text{dom}(\text{Concept_corresp_Constant}) \neq \emptyset$
grd3: $PCO \in \text{dom}(\text{Concept_corresp_Constant})$
grd4: $\text{Concept_parentConcept_Concept}(CO) = PCO$
grd5: $\text{Concept_definedIn_DomainModel}(CO) \in \text{dom}(\text{DomainModel_corresp_Component})$
grd6: $\text{Constant_Set} \setminus \text{Constant} \neq \emptyset$
grd7: $o_CO \in \text{Constant_Set} \setminus \text{Constant}$
grd8: $\text{LogicFormula_Set} \setminus \text{LogicFormula} \neq \emptyset$
grd9: $o_lg \in \text{LogicFormula_Set} \setminus \text{LogicFormula}$
grd10: $o_PCO \in \text{Constant}$
grd11: $o_PCO = \text{Concept_corresp_Constant}(PCO)$

then
act1: $\text{Constant} := \text{Constant} \cup \{o_CO\}$

```

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)* $\hat{=}$

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) \ dom(Individual_corresp_Constant)  $\neq$   $\emptyset$ 
grd1: ind  $\in$  dom(Individual_individualOf_Concept) \ 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 \ Constant  $\neq$   $\emptyset$ 
grd7: o_ind  $\in$  Constant_Set \ Constant
grd8: LogicFormula_Set \ LogicFormula  $\neq$   $\emptyset$ 
grd9: o_lg  $\in$  LogicFormula_Set \ 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 ↦ 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⟩ ≐
  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) \ dom(Individual_corresp_Constant) ≠ ∅
  grd1: ind ∈ dom(Individual_individualOf_Concept) \ dom(Individual_corresp_Constant)
  grd2: dom(Concept_corresp_Constant) ≠ ∅
  grd3: CO ∈ dom(Concept_corresp_Constant)
  grd4: Individual_individualOf_Concept(ind) = CO
  grd5: Concept_definedIn_DomainModel(CO) ∈ dom(DomainModel_corresp_Component)
  grd6: Constant_Set \ Constant ≠ ∅
  grd7: o_ind ∈ Constant_Set \ Constant
  grd8: LogicFormula_Set \ LogicFormula ≠ ∅
  grd9: o_lg ∈ LogicFormula_Set \ LogicFormula
  grd10: o_CO ∈ Constant
  grd11: o_CO = Concept_corresp_Constant(CO)
then
  act1: Constant := Constant ∪ {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 ∪ {o_lg}
  act5: LogicFormula := LogicFormula ∪ {o_lg}
  act6: LogicFormula_uses_Operators(o_lg) := {1 ↦ Belonging_OP}
  act7: Constant_isInvolvedIn_LogicFormulas := Constant_isInvolvedIn_LogicFormulas ← { (o_ind ↦
    {1 ↦ o_lg}), o_CO ↦ Constant_isInvolvedIn_LogicFormulas(o_CO) ∪ {2 ↦ o_lg} }
  act8: LogicFormula_involves_Sets(o_lg) := ∅
  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⟩ ≐

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)* $\hat{=}$

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)* $\hat{=}$

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:  $o\_CO2 = \text{Concept\_corresp\_Constant}(CO2)$ 
grd15:  $DM = \text{Relation\_definedIn\_DomainModel}(RE)$ 
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:  $Constant\_definedIn\_Component := Constant\_definedIn\_Component \cup$ 
       $\{o\_RE \mapsto \text{DomainModel\_corresp\_Component}(DM), T\_RE \mapsto \text{DomainModel\_corresp\_Component}(DM)\}$ 

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:  $Constant\_isInvolvedIn\_LogicFormulas := Constant\_isInvolvedIn\_LogicFormulas \leftarrow \{T\_RE \mapsto$ 
       $\{1 \mapsto o\_lg1, 2 \mapsto o\_lg2\}, o\_RE \mapsto \{1 \mapsto o\_lg2\}, o\_CO2 \mapsto \{3 \mapsto o\_lg1\} \cup \text{Constant\_isInvolvedIn\_LogicFormulas}(o\_CO2)\}$ 

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

```

end

Event rule_10_3 (ordinary) $\hat{=}$

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:  $Relation\_isVariable^{-1}[\{FALSE\}] \setminus \text{dom}(Relation\_Type) \neq \emptyset$ 
grd1:  $RE \in Relation\_isVariable^{-1}[\{FALSE\}] \setminus \text{dom}(Relation\_Type)$ 
grd2:  $\text{dom}(\text{Concept\_corresp\_Constant}) \neq \emptyset$ 
grd3:  $CO1 = \text{Relation\_domain\_Concept}(RE)$ 
grd4:  $CO1 \in \text{dom}(\text{Concept\_corresp\_Constant})$ 
grd5:  $\text{dom}(\text{Concept\_corresp\_AbstractSet}) \neq \emptyset$ 
grd6:  $CO2 = \text{Relation\_range\_Concept}(RE)$ 
grd7:  $CO2 \in \text{dom}(\text{Concept\_corresp\_AbstractSet})$ 
grd8:  $Relation\_definedIn\_DomainModel(RE) \in \text{dom}(\text{DomainModel\_corresp\_Component})$ 
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:  $o\_CO2 = \text{Concept\_corresp\_AbstractSet}(CO2)$ 
grd14:  $o\_CO1 = \text{Concept\_corresp\_Constant}(CO1)$ 
grd15:  $DM = \text{Relation\_definedIn\_DomainModel}(RE)$ 
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:  $Constant\_isInvolvedIn\_LogicFormulas := Constant\_isInvolvedIn\_LogicFormulas \leftarrow \{T\_RE \mapsto$ 
       $\{1 \mapsto o\_lg1, 2 \mapsto o\_lg2\}, o\_RE \mapsto \{1 \mapsto o\_lg2\}, o\_CO1 \mapsto \{2 \mapsto o\_lg1\} \cup Constant\_isInvolvedIn\_LogicFormulas(o\_CO1)\}$ 

    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:  $LogicFormula\_definedIn\_Component := LogicFormula\_definedIn\_Component \cup \{o\_lg1 \mapsto DomainModel\_corresp\_Component(DM)\}$ 
  end
Event rule_10_4 (ordinary)  $\hat{=}$ 
  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: $Constant_isInvolvedIn_LogicFormulas := Constant_isInvolvedIn_LogicFormulas \leftarrow \{T_RE \mapsto \{1 \mapsto o_lg1, 2 \mapsto o_lg2\}, o_RE \mapsto \{1 \mapsto o_lg2\}, o_CO1 \mapsto \{2 \mapsto o_lg1\} \cup Constant_isInvolvedIn_LogicFormulas(o_CO1), o_CO2 \mapsto \{3 \mapsto o_lg1\} \cup Constant_isInvolvedIn_LogicFormulas(o_CO2)\}$
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: $LogicFormula_definedIn_Component := LogicFormula_definedIn_Component \cup \{o_lg1 \mapsto DomainModel_corresp_Component(DM)\}$
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>* $\hat{=}$

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: $Constant_isInvolvedIn_LogicFormulas := Constant_isInvolvedIn_LogicFormulas \leftarrow \{o_remap \mapsto \{1 \mapsto o_lg\}, o_antecedent \mapsto \{2 \mapsto o_lg\} \cup Constant_isInvolvedIn_LogicFormulas(o_antecedent), o_image \mapsto \{3 \mapsto o_lg\} \cup Constant_isInvolvedIn_LogicFormulas(o_image)\}$
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) $\hat{=}$

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: $Constant_isInvolvedIn_LogicFormulas := Constant_isInvolvedIn_LogicFormulas \leftarrow \{o_atmap \mapsto \{1 \mapsto o_lg\}, o_antecedent \mapsto \{2 \mapsto o_lg\} \cup Constant_isInvolvedIn_LogicFormulas(o_antecedent), o_image \mapsto \{3 \mapsto o_lg\} \cup Constant_isInvolvedIn_LogicFormulas(o_image)\}$
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⟩ ≐
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 \ dom(AttributeMaplet_corresp_Constant) ≠ ∅
grd1: atmap ∈ AttributeMaplet \ dom(AttributeMaplet_corresp_Constant)
grd2: dom(Attribute_corresp_Constant) ∪ dom(Attribute_corresp_Variable) ≠ ∅
grd3: AttributeMaplet_mapletOf_Attribute(atmap) = AT
grd4: AT ∈ dom(Attribute_corresp_Constant) ∪ dom(Attribute_corresp_Variable)
grd5: Attribute_definedIn_DomainModel(AT) ∈ dom(DomainModel_corresp_Component)
grd6: Constant_Set \ Constant ≠ ∅
grd7: o_atmap ∈ Constant_Set \ Constant
grd8: LogicFormula_Set \ LogicFormula ≠ ∅
grd9: o_lg ∈ LogicFormula_Set \ LogicFormula
grd10: antecedent = AttributeMaplet_antecedent_Individual(atmap)
grd11: image = AttributeMaplet_image_DataValue(atmap)
grd12: antecedent ∈ dom(Individual_corresp_Constant)
grd13: image ∈ dom(DataValue_corresp_SetItem)
grd14: o_antecedent = Individual_corresp_Constant(antecedent)
grd15: o_image = DataValue_corresp_SetItem(image)
then
act1: Constant := Constant ∪ {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 ∪ {o_lg}
act5: LogicFormula := LogicFormula ∪ {o_lg}
act6: LogicFormula_uses_Operators(o_lg) := {1 ↦ Maplet_OP}
act7: Constant_isInvolvedIn_LogicFormulas := Constant_isInvolvedIn_LogicFormulas <- {o_atmap ↦
    {1 ↦ o_lg}, o_antecedent ↦ {2 ↦ o_lg} ∪ Constant_isInvolvedIn_LogicFormulas(o_antecedent)}
act8: LogicFormula_involves_Sets(o_lg) := ∅
act9: LogicFormula_involves_SetItems(o_lg) := {3 ↦ 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 *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) $\hat{=}$

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: $LogicFormula_definedIn_Component := LogicFormula_definedIn_Component \cup \{o_lg1 \mapsto DomainModel_corresp_Component(DM)\}$

end

Event rule_14_2 (ordinary) $\hat{=}$

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: $Constant_isInvolvedIn_LogicFormulas := Constant_isInvolvedIn_LogicFormulas \leftarrow \{T_AT \mapsto \{1 \mapsto o_lg1, 2 \mapsto o_lg2\}, o_AT \mapsto \{1 \mapsto o_lg2\}, o_CO \mapsto \{2 \mapsto o_lg1\}\} \cup Constant_isInvolvedIn_LogicFormulas(o_CO)$

act9: $LogicFormula_uses_Operators := LogicFormula_uses_Operators \cup \{o_lg1 \mapsto \{1 \mapsto RelationSet_OP\}, o_lg2 \mapsto \{1 \mapsto Belonging_OP\}\}$

act10: $LogicFormula_involves_Sets := LogicFormula_involves_Sets \cup \{o_lg1 \mapsto \{3 \mapsto o_DS\}, o_lg2 \mapsto \emptyset\}$

act11: $LogicFormula_definedIn_Component := LogicFormula_definedIn_Component \cup \{o_lg1 \mapsto DomainModel_corresp_Component(DM)\}$

end

Event rule_14.3 (ordinary) $\hat{=}$

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>* $\hat{=}$

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 `isFunction` 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) $\hat{=}$

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: $(\text{dom}(\text{Concept_corresp_AbstractSet}) \cap \text{Concept_isVariable}^{-1}[\{\text{TRUE}\}]) \setminus \text{dom}(\text{Concept_corresp_Variable}) \neq \emptyset$
 grd1: $CO \in (\text{dom}(\text{Concept_corresp_AbstractSet}) \cap \text{Concept_isVariable}^{-1}[\{\text{TRUE}\}]) \setminus \text{dom}(\text{Concept_corresp_Variable})$
 grd2: $\text{Concept_definedIn_DomainModel}(CO) \in \text{dom}(\text{DomainModel_corresp_Component})$
 grd3: $\text{Individual_individualOf_Concept}^{-1}[\{CO\}] \subseteq \text{dom}(\text{Individual_corresp_Constant})$
 grd4: $\text{Variable_Set} \setminus \text{Variable} \neq \emptyset$
 grd5: $x_CO \in \text{Variable_Set} \setminus \text{Variable}$
 grd6: $\text{LogicFormula_Set} \setminus \text{LogicFormula} \neq \emptyset$
 grd7: $o_lg \in \text{LogicFormula_Set} \setminus \text{LogicFormula}$
 grd8: $o_CO \in \text{AbstractSet}$
 grd9: $o_CO = \text{Concept_corresp_AbstractSet}(CO)$
 grd10: $\text{InitialisationAction_Set} \setminus \text{InitialisationAction} \neq \emptyset$
 grd11: $o_ia \in \text{InitialisationAction_Set} \setminus \text{InitialisationAction}$
 grd12: $o_inds = \text{Individual_corresp_Constant}[\text{Individual_individualOf_Concept}^{-1}[\{CO\}]]$
 grd13: $\text{finite}(o_inds)$
 grd14: $\text{bij_o_inds} \in 1.. \text{card}(o_inds) \twoheadrightarrow o_inds$

then

act1: $\text{Variable} := \text{Variable} \cup \{x_CO\}$
 act2: $\text{Concept_corresp_Variable}(CO) := x_CO$
 act3: $\text{Variable_definedIn_Component}(x_CO) := \text{DomainModel_corresp_Component}(\text{Concept_definedIn_DomainModel}(CO))$
 act4: $\text{Invariant} := \text{Invariant} \cup \{o_lg\}$
 act5: $\text{LogicFormula} := \text{LogicFormula} \cup \{o_lg\}$
 act6: $\text{LogicFormula_uses_Operators}(o_lg) := \{1 \mapsto \text{Inclusion_OP}\}$
 act7: $\text{Invariant_involves_Variables}(o_lg) := \{1 \mapsto x_CO\}$
 act8: $\text{LogicFormula_involves_Sets}(o_lg) := \{2 \mapsto o_CO\}$
 act9: $\text{LogicFormula_definedIn_Component}(o_lg) := \text{DomainModel_corresp_Component}(\text{Concept_definedIn_DomainModel}(CO))$


```

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)  $\hat{=}$ 
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)  $\mapsto$  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) $\hat{=}$

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 explicitly 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>  $\hat{=}$ 

```

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  $\leftarrow$  ({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)})
appearance 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) $\hat{=}$
 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: $bij_o_maplets \in 1 \dots card(o_maplets) \mapsto o_maplets$
then
 act1: $InitialisationAction := (InitialisationAction \setminus \{ex_o_ia\}) \cup \{o_ia\}$
 act2: $InitialisationAction_uses_Operators := (InitialisationAction_uses_Operators \setminus \{ex_o_ia \mapsto InitialisationAction_uses_Operators(ex_o_ia)\}) \leftarrow \{o_ia \mapsto \{1 \mapsto BecomeEqual2SetOf_OP\}\}$
 act3: $Variable_init_InitialisationAction(o_RE) := o_ia$
 act4: $InitialisationAction_involves_Constants := (InitialisationAction_involves_Constants \setminus \{ex_o_ia \mapsto InitialisationAction_involves_Constants(ex_o_ia)\}) \leftarrow \{o_ia \mapsto bij_o_maplets\}$
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) $\hat{=}$

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: $maplets \subseteq dom(AttributeMaplet_corresp_Constant)$
grd6: $o_maplets = AttributeMaplet_corresp_Constant[maplets]$
grd7: $Attribute_definedIn_DomainModel(AT) \in dom(DomainModel_corresp_Component)$
grd8: $LogicFormula_Set \setminus LogicFormula \neq \emptyset$
grd9: $o_lg \in LogicFormula_Set \setminus LogicFormula$
grd10: $o_AT \notin o_maplets$

then

act1: $Property := Property \cup \{o_lg\}$
act2: $LogicFormula := LogicFormula \cup \{o_lg\}$
act3: $LogicFormula_uses_Operators(o_lg) := \{1 \mapsto Equal2SetOf_OP\}$
act4: $Constant_isInvolvedIn_LogicFormulas := Constant_isInvolvedIn_LogicFormulas \leftarrow (\{o_AT \mapsto (\{1 \mapsto o_lg\} \cup Constant_isInvolvedIn_LogicFormulas(o_AT))\} \cup \{co \mapsto lgs | co \in o_maplets \wedge lgs = \{2 \mapsto o_lg\} \cup Constant_isInvolvedIn_LogicFormulas(co)\})$
 appearance order does not matter
act5: $LogicFormula_involves_Sets(o_lg) := \emptyset$
act6: $LogicFormula_definedIn_Component(o_lg) := DomainModel_corresp_Component(Attribute_definedIn_DomainModel(AT))$

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)* $\hat{=}$

handling the transitivity of a constant relation

any

RE
 o_lg1
 o_lg2
 o_RE
 composition

where

grd0: $(dom(Relation_corresp_Constant) \cap Relation_isTransitive^{-1}[\{TRUE\}]) \neq \emptyset$
grd1: $RE \in (dom(Relation_corresp_Constant) \cap Relation_isTransitive^{-1}[\{TRUE\}])$
grd2: $(\{RE \mapsto isTransitive\}) \notin dom(RelationCharacteristic_corresp_LogicFormula)$
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: $composition \in Constant_Set \setminus Constant$

then

act0: $Constant := Constant \cup \{composition\}$
act1: $Property := Property \cup \{o_lg1, o_lg2\}$
act2: $LogicFormula := LogicFormula \cup \{o_lg1, o_lg2\}$
act3: $Constant_typing_Property(composition) := o_lg1$
act4: $RelationCharacteristic_corresp_LogicFormula(\{RE \mapsto isTransitive\}) := o_lg2$
act5: $Constant_isInvolvedIn_LogicFormulas := Constant_isInvolvedIn_LogicFormulas \leftarrow \{composition \mapsto \{1 \mapsto o_lg1, 1 \mapsto o_lg2\}, o_RE \mapsto \{2 \mapsto o_lg1, 3 \mapsto o_lg1, 2 \mapsto o_lg2\} \cup Constant_isInvolvedIn_LogicFormulas(o_RE)\}$
act6: $LogicFormula_uses_Operators := LogicFormula_uses_Operators \cup \{o_lg1 \mapsto \{1 \mapsto RelationComposition_OP\}, o_lg2 \mapsto \{1 \mapsto Inclusion_OP\}\}$
act7: $LogicFormula_involves_Sets := LogicFormula_involves_Sets \cup \{o_lg1 \mapsto \emptyset, o_lg2 \mapsto \emptyset\}$
act8: $LogicFormula_definedIn_Component := LogicFormula_definedIn_Component \cup \{o_lg1 \mapsto DomainModel_corresp_Component(o_lg1 \mapsto DomainModel_corresp_Component(Relation_definedIn_DomainModel(RE)))\}$

```

    act9: Constant_definedIn_Component(composition) := DomainModel_corresp_Component(
      Relation_definedIn_DomainModel(RE))
  end
Event rule_16_2 ⟨ordinary⟩ ≐
  handling the symmetrie of a constant relation
  any
    RE
    o_lg1
    o_lg2
    o_RE
    inverse
  where
    grd0: (dom(Relation_corresp_Constant) ∩ Relation_isSymmetric-1[{TRUE}]) ≠ ∅
    grd1: RE ∈ (dom(Relation_corresp_Constant) ∩ Relation_isSymmetric-1[{TRUE}])
    grd2: ({RE ↦ isSymmetric}) ∉ dom(RelationCharacteristic_corresp_LogicFormula)
    grd3: o_RE = Relation_corresp_Constant(RE)
    grd4: Relation_definedIn_DomainModel(RE) ∈ dom(DomainModel_corresp_Component)
    grd5: LogicFormula_Set \ LogicFormula ≠ ∅
    grd6: {o_lg1, o_lg2} ⊆ LogicFormula_Set \ LogicFormula
    grd7: partition({o_lg1, o_lg2}, {o_lg1}, {o_lg2})
    grd8: Constant_Set \ Constant ≠ ∅
    grd9: inverse ∈ Constant_Set \ Constant
  then
    act0: Constant := Constant ∪ {inverse}
    act1: Property := Property ∪ {o_lg1, o_lg2}
    act2: LogicFormula := LogicFormula ∪ {o_lg1, o_lg2}
    act3: Constant_typing_Property(inverse) := o_lg1
    act4: RelationCharacteristic_corresp_LogicFormula({RE ↦ isSymmetric}) := o_lg2
    act5: Constant_isInvolvedIn_LogicFormulas := Constant_isInvolvedIn_LogicFormulas ← {inverse ↦
      {1 ↦ o_lg1, 1 ↦ o_lg2}, o_RE ↦ {2 ↦ o_lg1, 2 ↦ o_lg2}} ∪ Constant_isInvolvedIn_LogicFormulas(o_RE)}
    act6: LogicFormula_uses_Operators := LogicFormula_uses_Operators ∪ {o_lg1 ↦ {1 ↦ Inversion_OP}, o_lg2 ↦
      {1 ↦ Equality_OP}}
    act7: LogicFormula_involves_Sets := LogicFormula_involves_Sets ∪ {o_lg1 ↦ ∅, o_lg2 ↦ ∅}
    act8: LogicFormula_definedIn_Component := LogicFormula_definedIn_Component ∪ {o_lg1 ↦ DomainModel_corresp_Comp_
      o_lg2 ↦ 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) $\hat{=}$
 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) $\hat{=}$
 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) $\hat{=}$

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:  $EDS \in \text{DataSet\_Set} \setminus \text{DataSet}$ 
grd5:  $\text{DataValue\_Set} \setminus \text{DataValue} \neq \emptyset$ 
grd6:  $\text{elements} \subseteq \text{DataValue\_Set} \setminus \text{DataValue}$ 
grd7:  $\text{o\_elements} = \text{SetItem\_itemOf\_EnumeratedSet}^{-1}[\{\text{o\_EDS}\}]$ 
grd8:  $\text{card}(\text{o\_elements}) = \text{card}(\text{elements})$ 
grd9:  $\text{mapping\_elements\_o\_elements} \in \text{elements} \mapsto \text{o\_elements}$ 
grd10:  $\text{ran}(\text{DataValue\_corresp\_SetItem}) \cap \text{o\_elements} = \emptyset$ 
grd11:  $EDS \notin \{\_NATURAL, \_INTEGER, \_FLOAT, \_BOOL, \_STRING\}$ 
then
  act1:  $\text{EnumeratedDataSet} := \text{EnumeratedDataSet} \cup \{EDS\}$ 
  act2:  $\text{DataSet} := \text{DataSet} \cup \{EDS\}$ 
  act3:  $\text{EnumeratedDataSet\_corresp\_EnumeratedSet}(EDS) := \text{o\_EDS}$ 
  act4:  $\text{DataSet\_definedIn\_DomainModel}(EDS) := \text{DomainModel\_corresp\_Component}^{-1}(\text{Set\_definedIn\_Component}(\text{o\_EDS}))$ 
  act5:  $\text{DataValue} := \text{DataValue} \cup \text{elements}$ 
  act6:  $\text{DataValue\_elements\_EnumeratedDataSet} := \text{DataValue\_elements\_EnumeratedDataSet} \cup \{(xx \mapsto yy) \mid xx \in \text{elements} \wedge yy = EDS\}$ 
  act7:  $\text{DataValue\_corresp\_SetItem} := \text{DataValue\_corresp\_SetItem} \cup \text{mapping\_elements\_o\_elements}$ 
  act8:  $\text{DataSet\_corresp\_Set} := \text{DataSet\_corresp\_Set} \leftarrow \{EDS \mapsto \text{o\_EDS}\}$ 
  act9:  $\text{DataValue\_valueOf\_DataSet} := \text{DataValue\_valueOf\_DataSet} \cup \{(xx \mapsto yy) \mid xx \in \text{elements} \wedge yy = EDS\}$ 
  act10:  $\text{CustomDataSet} := \text{CustomDataSet} \cup \{EDS\}$ 
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) $\hat{=}$

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

```

any
  EDS
  o_EDS
  element
  o_element
where
  grd0:  $\text{dom}(\text{SetItem\_itemOf\_EnumeratedSet}) \setminus \text{ran}(\text{DataValue\_corresp\_SetItem}) \neq \emptyset$ 
  grd1:  $\text{o\_element} \in \text{dom}(\text{SetItem\_itemOf\_EnumeratedSet}) \setminus \text{ran}(\text{DataValue\_corresp\_SetItem})$ 
  grd2:  $\text{o\_EDS} = \text{SetItem\_itemOf\_EnumeratedSet}(\text{o\_element})$ 
  grd3:  $\text{o\_EDS} \in \text{ran}(\text{EnumeratedDataSet\_corresp\_EnumeratedSet})$ 
  grd4:  $EDS = \text{EnumeratedDataSet\_corresp\_EnumeratedSet}^{-1}(\text{o\_EDS})$ 
  grd5:  $\text{DataValue\_Set} \setminus \text{DataValue} \neq \emptyset$ 
  grd6:  $\text{element} \in \text{DataValue\_Set} \setminus \text{DataValue}$ 
then
  act1:  $\text{DataValue} := \text{DataValue} \cup \{\text{element}\}$ 
  act2:  $\text{DataValue\_elements\_EnumeratedDataSet}(\text{element}) := EDS$ 
  act3:  $\text{DataValue\_corresp\_SetItem}(\text{element}) := \text{o\_element}$ 
  act4:  $\text{DataValue\_valueOf\_DataSet}(\text{element}) := EDS$ 
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 \langle ordinary $\rangle \hat{=}$

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 \langle ordinary $\rangle \hat{=}$

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 \langle ordinary $\rangle \hat{=}$

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 \langle ordinary $\rangle \hat{=}$

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)  $\hat{=}$ 
  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)  $\hat{=}$ 
  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:  $o\_CO \in \text{ran}(\text{Concept\_corresp\_AbstractSet})$ 
    grd7:  $CO = \text{Concept\_corresp\_AbstractSet}^{-1}(o\_CO)$ 
  then
    act1:  $\text{Concept\_isVariable}(CO) := \text{TRUE}$ 
    act2:  $\text{Concept\_corresp\_Variable}(CO) := x\_CO$ 
  end
Event rule_108_2  $\langle \text{ordinary} \rangle \hat{=}$ 
  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:  $\text{dom}(\text{Variable\_typing\_Invariant}) \setminus \text{ran}(\text{Concept\_corresp\_Variable}) \neq \emptyset$ 
    grd1:  $x\_CO \in \text{dom}(\text{Variable\_typing\_Invariant}) \setminus \text{ran}(\text{Concept\_corresp\_Variable})$ 
    grd2:  $o\_lg = \text{Variable\_typing\_Invariant}(x\_CO)$ 
    grd3:  $\text{LogicFormula\_uses\_Operators}(o\_lg) = \{1 \mapsto \text{Inclusion\_OP}\}$ 
    grd4:  $\text{LogicFormula\_involves\_Sets}(o\_lg) = \emptyset$ 
    grd5:  $o\_CO \in \text{dom}(\text{Constant\_isInvolvedIn\_LogicFormulas})$ 
    grd6:  $(2 \mapsto o\_lg) \in \text{Constant\_isInvolvedIn\_LogicFormulas}(o\_CO)$ 
    grd7:  $o\_CO \in \text{ran}(\text{Concept\_corresp\_Constant})$ 
    grd8:  $CO = \text{Concept\_corresp\_Constant}^{-1}(o\_CO)$ 
  then
    act1:  $\text{Concept\_isVariable}(CO) := \text{TRUE}$ 
    act2:  $\text{Concept\_corresp\_Variable}(CO) := x\_CO$ 
  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* [21]. 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)

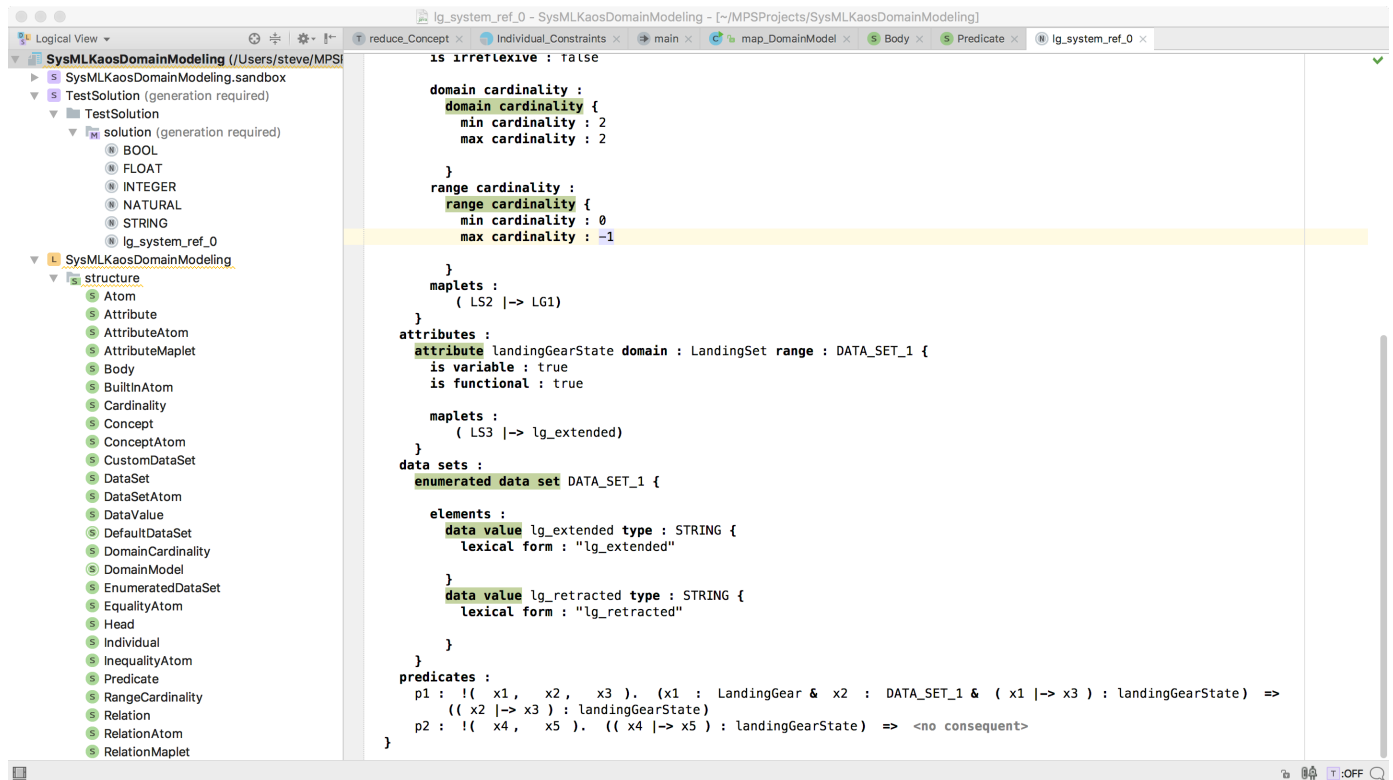


Fig. 15. Preview of the SysML/KAOS Domain Model Parser Tool

2. Alkhamash, E., Butler, M.J., Fathabadi, A.S., Cirstea, C.: Building traceable Event-B models from requirements. *Sci. Comput. Program.* 111, 318–338 (2015)
3. Alkhamash, 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)
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. *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
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
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)

```

1  /* lg_system_ref_0
2  * Author: SysML/KAOS Domain Model Parser
3  * Creation date: 21/08/2017
4  */
5
6  SYSTEM
7  lg_system_ref_0
8
9
10 SETS
11 LandingGear; LandingSet; DATA_SET_1={lg_extended, lg_retracted}
12
13 CONSTANTS
14 LG1, LS1, LS2, LS3, T_re, re, T_landingGearState
15
16 VARIABLES
17 LandingGearState
18
19 PROPERTIES
20 LG1 : LandingGear &
21 LandingGear = {LG1} &
22 LS1 : LandingSet &
23 LS2 : LandingSet &
24 LS3 : LandingSet &
25 LandingSet = {LS1, LS2, LS3} &
26 T_re = LandingSet <-> LandingGear &
27 re : T_re &
28 !xx. (xx : ran(re) => card(re~{xx}) = 2) &
29 !xx. (xx : dom(re) => card(re~{xx}) >= 0) &
30 re = {LS1|->LG1, LS1|->LG1, LS2|->LG1} &
31 T_landingGearState = LandingSet --> DATA_SET_1
32
33 INVARIANT
34 LandingGearState : T_landingGearState &
35 //predicate p1

```

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., 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*
21. Tueno, S., Laleau, R., Mammar, A., Frappier, M.: SysML/KAOS Domain Modeling Tool (2017), https://github.com/stuenofotso/SysML_KAOS_discretionary-{}-{}Domain_Model_Parser
22. UL, I.: Owlged home (2017), <http://owlged.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)