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

Steve Jeffrey Tueno Fotso¹,³, Amel Mammar², Régine Laleau¹, and Marc Frappier³

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

15.07.2018

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

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

1 Introduction

This article focuses on the development of systems in critical areas such as railway or aeronautics. The implementation of such systems, in view of their complexity, requires several validation steps, more or less formal 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 [10], 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 [15], 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 [22] 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.

![Diagram of Event-B formalism](image)

Figure 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 **Logic-Formula**. 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:
  \((\text{Inclusion}_\text{OP}, [\text{op}_1, \text{op}_2]) \iff \text{op}_1 \subset \text{op}_2.\)

- **Belonging OP** is used to assert that the first operand is an element of the second operand:
  \((\text{Belonging}_\text{OP}, [\text{op}_1, \text{op}_2]) \iff \text{op}_1 \in \text{op}_2.\)

- **RelationSet OP** is used to construct the set of relations between two operands:
  \((\text{RelationSet}_\text{OP}, [\text{op}_1, \text{op}_2, \text{op}_3]) \iff \text{op}_1 \leftrightarrow \text{op}_3.\)

- **FunctionSet OP** is used to construct the set of functional relations between two operands:
  \((\text{FunctionSet}_\text{OP}, [\text{op}_1, \text{op}_2, \text{op}_3]) \iff \text{op}_1 \rightarrow \text{op}_3.\)

- **Maplet OP** is used to construct a maplet having the operands as antecedent and image:
  \((\text{Maplet}_\text{OP}, [\text{op}_1, \text{op}_2, \text{op}_3]) \iff \text{op}_1 \mapsto \text{op}_3.\)

- **RelationComposition OP** is used to assert that the first operand is the result of the composition of the second operand by the third operand:
  \((\text{RelationComposition}_\text{OP}, [\text{op}_1, \text{op}_2, \text{op}_3]) \iff \text{op}_1 = \text{op}_2 \circ \text{op}_3.\)

- **Equal2SetOf OP** is used to define the elements constituting a set:
  \((\text{Equal}_\text{2SetOf}_\text{OP}, [\text{op}_1, \text{op}_2, ..., \text{op}_n]) \iff \text{op}_1 = \{\text{op}_2, ..., \text{op}_n\}.\)

- **Inversion OP** is used to assert that the first operand is the inverse of the second operand:
  \((\text{Inversion}_\text{OP}, [\text{op}_1, \text{op}_2]) \iff \text{op}_1 = \text{op}_2^{-1}.\)

- **Equality OP** is used to assert that the first operand is equal to the second operand:
  \((\text{Equality}_\text{OP}, [\text{op}_1, \text{op}_2]) \iff \text{op}_1 = \text{op}_2.\)

- **BecomeEqual2SetOf OP** is used to initialize a variable as a set of elements:
  \((\text{BecomeEqual}_\text{2SetOf}_\text{OP}, [\text{va}, \text{op}_2, ..., \text{op}_n]) \iff \text{va} := \{\text{op}_2, ..., \text{op}_n\}.\)

- **BecomeEqual2EmptySet OP** is used to initialize a variable as an empty set:
  \((\text{BecomeEqual}_\text{2EmptySet}_\text{OP}, [\text{va}]) \iff \text{va} := \emptyset.\)

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](image-url)

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 [14,15], 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](image)

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 (*makeLSEExtended*).

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 [16]. 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
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 [22] 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 OWL Gred [23] and, for readability purposes, we have decided to remove optional characteristics representation. It should be noted that the individualOf association is illustrated by OWL Gred 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 $lg_{system\_ref\_0}$ (Fig. 7), elements defined in $lg_{system\_ref\_0}$ (Fig. 7) are imported and reused.

A concept (instance of metaclass Concept of Figure 4) represents a group of individuals sharing common characteristics. It can be declared variable (isVariable=true) when the set of its individuals is likely to be updated through addition or deletion of individuals. Otherwise, it is considered to be constant (isVariable=false). A concept may be associated with another, known as its parent concept, through the parentConcept association, from which it inherits properties. For example, in $lg_{system\_ref\_0}$ (Fig. 7), a landing gear is modeled as an instance of Concept named "LandingGear". Since it is impossible to dynamically add or remove a landing gear, the attribute isVariable of LandingGear is set to false. LG1 is modeled as an instance of Individual (Fig. 5) named "LG1" individual of LandingGear.

Instances of Relation are used to capture links between concepts, and instances of Attribute capture links between concepts and data sets, knowing that data sets (instances of DataSet) are used to group data values (instances of DataValue) having the same type. The most basic way to build an instance of DataSet is by listing its elements. This can be done through the DataSet specialization called EnumeratedDataSet. A relation or an attribute can be declared variable if the list of maplets related to it is likely to change over time. Otherwise, it is considered to be constant. Each instance of DomainCardinality (respectively RangeCardinality) makes it possible to define, for an instance of Relation re, the minimum and maximum limits of the number of instances
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: \( \text{minCardinality} \geq 0 \land (\text{maxCardinality} = \ast \lor \text{maxCardinality} \geq \text{minCardinality}) \), knowing that if maxCardinality = \ast, 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.

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.
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 [24], 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 [19] 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 [18], 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 [18], 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 [20]: 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 [15]. 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.
Fig. 9. Correspondence to B System Components

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

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

- Regarding the representation of metamodels, we have followed the rules proposed in [13,20] 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`, `Attribute` and `DataSet` of the SysML/KAOS domain metamodel (`Domain_Metamodel_Context`) and `Component`, `Set`, `LogicFormula` and `Variable` of the B System metamodel (`BSystem_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 B System metamodel (`Ontologies_BSystem specs_translation` and `Ontologies_BSystem specs_translation_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 Event-B representation of the first class as domain and the Event-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 ∈ Concept ↦ AbstractSet` (`Ontologies_BSystem specs_translation_ref_1`).
Fig. 10. Correspondence to Sets

- Each rule is represented as an event by following the correspondence links.
- 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, INVARİANT, INITIALİZATİON and EVENTS).

Figures 13 and 14 represent 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.
4.1 Formalization of SysML/KAOS Domain Modeling and BSystem Formalisms in Event-B

The metamodels `BSystem_Metamodel_Context` and `Domain_Metamodel_Context` represent respectively the context associated to our abstraction of the BSystem specification language and that associated to the SysML/KAOS Domain Metamodel. The correspondences to constants are indicated in the metamodels with `Ontologies_BSystem_specs_translation` and `Ontologies_BSystem_specs_translation_ref_1` representing the corresponding variables and the associated invariants.
Fig. 12. Correspondence to Variables

\[0.1\]
\[0.2\]
\[0.3\]
\[0.4\]
\[0.5\]

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

**SYSTEM**  \(lg\_\text{system\_ref}\_0\)

**SETS**  \(\text{LandingGear};\ \text{DataSet}\_1=\{lg\_\text{extended}, lg\_\text{retracted}\}\)

**CONSTANTS**  \(T\text{\_landingGearState}, LG1\)

**PROPERTIES**

\[0.1\]  \(LG1 \in \text{LandingGear}\)

\[0.2\]  \(\land \text{LandingGear} = \{LG1\}\)

\[0.3\]  \(\land T\text{\_landingGearState} = \text{LandingGear} \rightarrow \text{DataSet}\_1\)

**VARIABLES**  \(\text{landingGearState}\)

**INVARIANT**

\[0.4\]  \(\text{landingGearState} \in T\text{\_landingGearState}\)

**INITIALISATION**

\[0.5\]  \(\text{landingGearState} := \{LG1 \mapsto lg\_\text{extended}\}\)

**EVENTS**

***

END

**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_
Fig. 14. Formalization of the First Refinement Level of the Landing Gear System Domain Model
CONTEXT  BSystem_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
InititalisationAction_Set

CONSTANTS
  B_NATURAL
  B_INTEGER
  B_FLOAT
  B_BOOL
  B_STRING
  Inclusion_OP
  Belonging_OP
  BecomeEqual2SetOP
  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 Ontologies_BSystem_specs_translation
SEES BSystem_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

INVARINTS
  inv0_1: Component ⊆ Component_Set
  inv0_2: partition(Component, System, Refinement) Domain Model
  inv0_3: DomainModel ⊆ DomainModel_Set
  inv0_4: DomainModel_parent_DomainModel ∈ DomainModel ⇔ DomainModel
  inv0_5: DomainModel_corresp_Component ∈ DomainModel ⇔ Component
  inv0_6: Refinement_refines_Component ∈ Refinement ⇔ Component
  inv0_7:
    ∀xx·
    ∀px·
      (xx ∈ dom(DomainModel_parent_DomainModel)
       ∧ px = DomainModel_parent_DomainModel(xx)
       ∧ px ∈ dom(DomainModel_corresp_Component)
       ∧ xx /∈ dom(DomainModel_corresp_Component))
    ⇒ DomainModel_corresp_Component(px) /∈ ran(Refinement_refines_Component)
  )
  )
  )

END
MACHINE Ontologies_BSystem_specs_translation_ref_1
REFINES Ontologies_BSystem_specs_translation
SEES BSystem_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
  LogicFormulaUses_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_elementsEnumeratedDataSet
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

INVARINTS
inv1_1: Variable ⊆ Variable_Set
inv1_2: Constant ⊆ Constant_Set
inv1_3: Set ⊆ Set_Set
inv1_4: partition(set, AbstractSet, EnumeratedSet)
inv1_5: SetItem ⊆ SetItem_Set
inv1_6: Variable_definedIn_Component ∈ Variable → Component
inv1_7: Constant_definedIn_Component ∈ Constant → Component
inv1_8: Set_definedIn_Component ∈ Set → Component
inv1_9: SetItem_itemOf_EnumeratedSet ∈ SetItem → EnumeratedSet

Domain Model
When appearance order does not matter, we may index all constants using the same number. The first operand is indexed by 1, no matter it’s type.

When appearance order does not matter, we may index all constants using the same number.
\text{inv1}_{66}: \text{Relation\_DomainCardinality\_maxCardinality} \in \text{Relation} \rightarrow (\mathbb{N} \cup \{-1\})
\text{inv1}_{67}: \text{Relation\_RangeCardinality\_minCardinality} \in \text{Relation} \rightarrow \mathbb{N}
\text{inv1}_{68}: \text{Relation\_RangeCardinality\_maxCardinality} \in \text{Relation} \rightarrow (\mathbb{N} \cup \{-1\})
\text{inv1}_{69}: \text{Attribute\_isVariable} \in \text{Attribute} \rightarrow \text{BOOL}
\text{inv1}_{70}: \text{Attribute\_isFunction} \in \text{Attribute} \rightarrow \text{BOOL}
\text{inv1}_{71}: \text{Relation\_definedIn\_DomainModel} \in \text{Relation} \rightarrow \text{DomainModel}
\text{inv1}_{72}: \text{Attribute\_definedIn\_DomainModel} \in \text{Attribute} \rightarrow \text{DomainModel}
\text{inv1}_{73}: \text{Relation\_domain\_Concept} \in \text{Relation} \rightarrow \text{Concept}
\text{inv1}_{74}: \text{Relation\_range\_Concept} \in \text{Relation} \rightarrow \text{Concept}
\text{inv1}_{77}: \text{RelationMaplet\_mapletOf\_Relation} \in \text{RelationMaplet} \rightarrow \text{Relation}
\text{inv1}_{78}: \text{RelationMaplet\_antecedent\_Individual} \in \text{RelationMaplet} \rightarrow \text{Individual}
\text{inv1}_{79}: \text{RelationMaplet\_image\_Individual} \in \text{RelationMaplet} \rightarrow \text{Individual}
\text{inv1}_{80}: \text{Attribute\_domain\_Concept} \in \text{Attribute} \rightarrow \text{Concept}
\text{inv1}_{81}: \text{Attribute\_range\_DataSet} \in \text{Attribute} \rightarrow \text{DataSet}
\text{inv1}_{82}: \text{Attribute\_Maplet\_mapletOf\_Attribute} \in \text{AttributeMaplet} \rightarrow \text{Attribute}
\text{inv1}_{83}: \text{Attribute\_Maplet\_antecedent\_Individual} \in \text{AttributeMaplet} \rightarrow \text{Individual}
\text{inv1}_{84}: \text{Attribute\_Maplet\_image\_DataValue} \in \text{AttributeMaplet} \rightarrow \text{DataValue}
\text{inv1}_{85}: \forall \text{rm} \cdot (\text{rm} \in \text{RelationMaplet} \Rightarrow \text{Individual\_individualOf\_Concept}(\text{RelationMaplet\_antecedent\_Individual}(\text{rm})) = \text{Relation\_domain\_Concept}(\text{RelationMaplet\_mapletOf\_Relation}(\text{rm})))
\text{inv1}_{86}: \forall \text{rm} \cdot (\text{rm} \in \text{RelationMaplet} \Rightarrow \text{Individual\_individualOf\_Concept}(\text{RelationMaplet\_image\_Individual}(\text{rm})) = \text{Relation\_range\_Concept}(\text{RelationMaplet\_mapletOf\_Relation}(\text{rm})))
\text{inv1}_{87}: \forall \text{am} \cdot (\text{am} \in \text{AttributeMaplet} \Rightarrow \text{Individual\_individualOf\_Concept}(\text{AttributeMaplet\_antecedent\_Individual}(\text{am})) = \text{Attribute\_domain\_Concept}(\text{AttributeMaplet\_mapletOf\_Attribute}(\text{am})))
\text{inv1}_{88}: \forall \text{am} \cdot (\text{am} \in \text{AttributeMaplet} \Rightarrow \text{DataValue\_valueOf\_DataValue}(\text{AttributeMaplet\_image\_DataValue}(\text{am})) = \text{Attribute\_range\_DataSet}(\text{AttributeMaplet\_mapletOf\_Attribute}(\text{am})))
\text{inv1}_{89}: \text{Relation\_Type} \in \text{Relation} \rightarrow \text{Constant}
\text{inv1}_{90}: \text{Relation\_corresp\_Constant} \in \text{Relation} \rightarrow \text{Constant}
\text{inv1}_{91}: \text{Relation\_corresp\_Variable} \in \text{Relation} \rightarrow \text{Variable}
\text{inv1}_{92}: \forall \text{re} \cdot (\text{re} \in \text{dom}(\text{Relation\_Type}) \Rightarrow (\text{re} \in \text{dom}(\text{Relation\_corresp\_Constant}) \lor (\text{re} \in \text{dom}(\text{Relation\_corresp\_Variable})))
\text{inv1}_{93}: \text{Attribute\_Type} \in \text{Attribute} \rightarrow \text{Constant}
\text{inv1}_{94}: \text{Attribute\_corresp\_Constant} \in \text{Attribute} \rightarrow \text{Constant}
\text{inv1}_{95}: \text{Attribute\_corresp\_Variable} \in \text{Attribute} \rightarrow \text{Variable}
\text{inv1}_{96}: \forall \text{re} \cdot (\text{re} \in \text{dom}(\text{Attribute\_Type}) \Rightarrow (\text{re} \in \text{dom}(\text{Attribute\_corresp\_Constant}) \lor (\text{re} \in \text{dom}(\text{Attribute\_corresp\_Variable})))
\text{inv1}_{97}: \text{Variable\_typing\_Invariant} \in \text{Variable} \rightarrow \text{Invariant}
\text{inv1}_{98}: \text{Constant\_typing\_Property} \in \text{Constant} \rightarrow \text{Property}
\text{inv1}_{99}: \text{RelationCharacteristic\_corresp\_LogicFormula} \in (\text{Relation} \rightarrow \text{RelationCharacteristics\_Set}) \rightarrow \text{LogicFormula}
\text{inv1}_{100}: \text{RelationMaplet\_corresp\_Constant} \in \text{RelationMaplet} \rightarrow \text{Constant}
\text{inv1}_{101}: \text{DataSet\_corresp\_Set} \in \text{DataSet} \rightarrow \text{Set}
\text{inv1}_{102}: \text{AttributeMaplet\_corresp\_Constant} \in \text{AttributeMaplet} \rightarrow \text{Constant}
\text{inv1}_{103}: \text{LogicFormula\_involves\_SetItems} \in \text{LogicFormula} \rightarrow (\mathbb{N} \rightarrow \text{SetItem})
\text{inv1}_{104}: \text{EnumeratedDataSet\_corresp\_EnumeratedSet} \subseteq \text{DataSet\_corresp\_Set}
\text{inv1}_{105}: \text{CustomDataSet\_corresp\_AbstractSet} \subseteq \text{DataSet\_corresp\_Set}

\text{EVENTS}
\text{Event} \text{initialize_default_dataSets} \text{ (ordinary)} \cong \text{any}
\text{DM}
\text{o}_{\text{DM}}
\text{where}
\text{grd0}: \text{dom}(\text{DomainModel\_corresp\_Component}) \setminus \text{dom}(\text{DomainModel\_parent\_DomainModel}) \neq \emptyset
\text{grd1}: \text{DefaultDataSet} = \emptyset
\text{grd2}: \text{DM} \in \text{dom}(\text{DomainModel\_corresp\_Component})
\text{grd3}: \text{DM} \notin \text{dom}(\text{DomainModel\_parent\_DomainModel})
\text{grd4}: \text{AbstractSet} \cap \{B_{\text{NATURAL}}, B_{\text{INTEGER}}, B_{\text{FLOAT}}, B_{\text{BOOL}}, B_{\text{STRING}}\} = \emptyset
Then

\[ \text{act1: } \text{DefaultDataSet} := \{ \text{NATURAL}, \text{INTEGER}, \text{FLOAT}, \text{BOOL}, \text{STRING} \} \]

\[ \text{act2: } \text{DataSet} := \text{DataSet} \cup \{ \text{NATURAL}, \text{INTEGER}, \text{FLOAT}, \text{BOOL}, \text{STRING} \} \]

\[ \text{act3: } \text{DataSet} \text{definedInDomainModel} := \text{DataSet} \text{definedInDomainModel} \cup \{ (x \mapsto y) \mid x \in \{ \text{NATURAL}, \text{INTEGER}, \text{FLOAT}, \text{BOOL}, \text{STRING} \} \} \]

\[ \text{act4: } \text{AbstractSet} := \text{AbstractSet} \cup \{ \text{B_NATURAL}, \text{B_INTEGER}, \text{B_FLOAT}, \text{B_BOOL}, \text{B_STRING} \} \]

\[ \text{act5: } \text{Set} := \text{Set} \cup \{ \text{B_NATURAL}, \text{B_INTEGER}, \text{B_FLOAT}, \text{B_BOOL}, \text{B_STRING} \} \]

\[ \text{act6: } \text{DefaultDataSet.correspAbstractSet} := \{ \text{NATURAL} \mapsto \text{B_NATURAL}, \text{INTEGER} \mapsto \text{B_INTEGER}, \text{FLOAT} \mapsto \text{B_FLOAT}, \text{BOOL} \mapsto \text{B_BOOL}, \text{STRING} \mapsto \text{B_STRING} \} \]

\[ \text{act7: } \text{DataSet.correspSet} := \text{DataSet.correspSet} \setminus \{ \text{NATURAL} \mapsto \text{B_NATURAL}, \text{INTEGER} \mapsto \text{B_INTEGER}, \text{FLOAT} \mapsto \text{B_FLOAT}, \text{BOOL} \mapsto \text{B_BOOL}, \text{STRING} \mapsto \text{B_STRING} \} \]

\[ \text{end} \]

\[ \text{END} \]

### 4.2 From Domain Models to B System Specifications

#### B System Components

**Rule 1: Domain model without parent**

**MACHINE** Ontologies.BSystem.specs_translation

**SEES** BSystem_Metamodel_Context, Domain_Metamodel_Context

**Event** rule_1 (ordinary) ⇒ correspondence of a domain model not associated to a parent domain model

**any**

- DM
- o_DM

**where**

\[ \text{grd0: } \text{DomainModel} \setminus (\text{dom(DomainModel.correspComponent)} \cup \text{dom(DomainModel.parentDomainModel)}) \neq \emptyset \]

\[ \text{grd1: } DM \in \text{DomainModel} \]

\[ \text{grd2: } DM \notin \text{dom(DomainModel.correspComponent)} \]

\[ \text{grd3: } DM \notin \text{dom(DomainModel.parentDomainModel)} \]

\[ \text{grd4: } \text{Component.Set} \setminus \text{Component} \neq \emptyset \]

\[ \text{grd5: } o_DM \in \text{Component.Set} \setminus \text{Component} \]

**then**

\[ \text{act1: } \text{System} := \text{System} \cup \{ o_DM \} \]

\[ \text{act2: } \text{Component} := \text{Component} \cup \{ o_DM \} \]

\[ \text{act3: } \text{DomainModel.correspComponent(DM)} := o_DM \]

**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** Ontologies.BSystem.specs_translation

**SEES** BSystem_Metamodel_Context, Domain_Metamodel_Context

**Event** rule_2 (ordinary) ⇒ correspondence of a domain model associated to a parent domain model

**any**

- DM
- PDM
- o_DM
where

\text{grd0:} \quad \text{dom(DomainModel\_parent\_DomainModel) \setminus \text{dom(DomainModel\_corresp\_Component)} \neq \emptyset}

\text{grd1:} \quad DM \in \text{dom(DomainModel\_parent\_DomainModel)}

\text{grd2:} \quad DM \notin \text{dom(DomainModel\_corresp\_Component)}

\text{grd3:} \quad \text{dom(DomainModel\_corresp\_Component)} \neq \emptyset

\text{grd4:} \quad PDM \in \text{dom(DomainModel\_corresp\_Component)}

\text{grd5:} \quad \text{DomainModel\_parent\_DomainModel}(DM) = PDM

\text{grd6:} \quad \text{Component\_Set} \setminus \text{Component} \neq \emptyset

\text{grd7:} \quad a_{DM} \in \text{Component\_Set} \setminus \text{Component}

\text{then}

\text{act1:} \quad \text{Refinement} := \text{Refinement} \cup \{a_{DM}\}

\text{act2:} \quad \text{Component} := \text{Component} \cup \{a_{DM}\}

\text{act3:} \quad \text{Refinement\_refines\_Component}(a_{DM}) := \text{DomainModel\_corresp\_Component}(PDM)

\text{act4:} \quad \text{DomainModel\_corresp\_Component}(DM) := a_{DM}

\text{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 \text{lg\_system\_ref\_1} refining \text{lg\_system\_ref\_0}. 
B System Sets

**Rule 3: Concept without parent**

MACHINE Ontologies_BSystem_specs_translation_ref_1
REFINES Ontologies_BSystem_specs_translation
SEES BSystem_Metamodel_Context,Doman_Metamodel_Context
Event rule_3 (ordinary) ⇔ correspondence of a concept not associated to a parent concept

any

CO

o_CO

where

grd0: Concept \ (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 \emptyset

grd6: \emptyset \subseteq Set_Set \subseteq 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 a B System abstract set. **Example:** in Figure 13 the abstract set
LandingGear appears because of Concept instance LandingGear.

**Rule 4: Enumerated data set**

MACHINE Ontologies_BSystem_specs_translation_ref_1
REFINES Ontologies_BSystem_specs_translation
SEES BSystem_Metamodel_Context,Doman_Metamodel_Context
Event rule_4 (ordinary) ⇔ correspondence of an instance of EnumeratedDataSet

any

EDS

o_EDS

elements

o_elements

mapping_elements \subseteq 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 \emptyset

grd6: \emptyset \subseteq Set_Set \subseteq Set

grd8: \emptyset \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 \subseteq elements \leftrightarrow o_elements

END
then

\[
\text{act1: EnumeratedSet} := \text{EnumeratedSet} \cup \{o_{\text{EDS}}\}
\]
\[
\text{act2: } Set := \text{Set} \cup \{o_{\text{EDS}}\}
\]
\[
\text{act3: EnumeratedDataSet_corresp_EnumeratedSet(EDS)} := o_{\text{EDS}}
\]
\[
\text{act4: } \text{Set\_definedIn\_Component}(o_{\text{EDS}}) := \text{DomainModel\_corresp\_Component(}
\text{DataSet\_definedIn\_DomainModel(EDS))}
\]
\[
\text{elements}
\]
\[
\text{act5: } \text{SetItem} := \text{SetItem} \cup o_{\text{elements}}
\]
\[
\text{act6: } \text{SetItem\_itemOf\_EnumeratedSet} := \text{SetItem\_itemOf\_EnumeratedSet} \cup \{(o_{\text{elements}} \times \{o_{\text{EDS}}\})\}
\]
\[
\text{act7: } \text{Data\_value_corresp\_SetItem} := \text{Data\_value_corresp\_SetItem} \cup \text{mapping\_elements\_o\_elements}
\]
\[
\text{act8: } \text{DataSet\_corresp\_Set} := \text{DataSet\_corresp\_Set} \leftarrow \{\text{EDS} \mapsto o_{\text{EDS}}\}
\]
\[
\end\]

END

**Rule 5** : Custom data set not defined through an enumeration

**MACHINE** Ontologies_BSystem\_specs\_translation\_ref\_1

**REFINES** Ontologies_BSystem\_specs\_translation

**SEES** BSystem\_Metamodel\_Context, Domain\_Metamodel\_Context

**Event** rule\_5 (ordinary) ≡

- correspondence of an instance of CustomData\_Set which is not an instance of EnumeratedData\_Set

any

\[
\text{CS}
\]
\[
\text{o}_{\text{CS}}
\]

**where**

\[
\text{grd0: CustomData\_Set} \setminus (\text{EnumeratedData\_Set} \cup \text{dom(DataSet\_corresp\_Set)}) \neq \emptyset
\]
\[
\text{grd1: } \text{CS} \in \text{CustomData\_Set}
\]
\[
\text{grd2: } \text{CS} \notin \text{EnumeratedData\_Set}
\]
\[
\text{grd3: } \text{CS} \notin \text{dom(DataSet\_corresp\_Set)}
\]
\[
\text{grd4: } \text{DataSet\_definedIn\_DomainModel(CS)} \in \text{dom(DomainModel\_corresp\_Component)}
\]
\[
\text{grd5: } \text{Set\_Set} \setminus \text{Set} \neq \emptyset
\]
\[
\text{grd6: } \text{o}_{\text{CS}} \in \text{Set\_Set} \setminus \text{Set}
\]

then

\[
\text{act1: AbstractSet} := \text{AbstractSet} \cup \{\text{o}_{\text{CS}}\}
\]
\[
\text{act2: } \text{Set} := \text{Set} \cup \{\text{o}_{\text{CS}}\}
\]
\[
\text{act3: CustomData\_Set_corresp\_AbstractSet(CS)} := \text{o}_{\text{CS}}
\]
\[
\text{act4: } \text{Set\_definedIn\_Component}(\text{o}_{\text{CS}}) := \text{DomainModel\_corresp\_Component(}
\text{DataSet\_definedIn\_DomainModel(CS))}
\]
\[
\text{act5: } \text{DataSet\_corresp\_Set} := \text{DataSet\_corresp\_Set} \leftarrow \{\text{CS} \mapsto \text{o}_{\text{CS}}\}
\]
\[
\end\]

END

Any instance of CustomData\_Set, defined through an enumeration, gives rise to a \textit{B System} enumerated set. Example : in Figure [7], the data set \{"lg\_extended", "lg\_retracted"\}, defined in domain model represented in Figure (Fig. [7]), gives rise to the enumerated set \textit{DataSet\_1} = \{\text{lg\_extended}, \text{lg\_retracted}\}.

Any instance of DefaultData\_Set is mapped directly to a \textit{B System} default data set ($\text{NATURAL}$, $\text{INTEGER}$, $\text{FLOAT}$, $\text{STRING}$ or $\text{BOOL}$) following the \textbf{initialize_default_datasets} event.

**B System Constants**

**Rule 6** : Concept with parent

**MACHINE** Ontologies_BSystem\_specs\_translation\_ref\_1

**REFINES** Ontologies_BSystem\_specs\_translation

**SEES** BSystem\_Metamodel\_Context, Domain\_Metamodel\_Context

**Event** rule\_6\_1 (ordinary) ≡

- correspondence of a concept associated to a parent concept (where the parent concept corresponds to an abstract set)
Event \text{rule\_6\_2} (ordinary) \equiv 
correspondence of a concept associated to a parent concept (where the parent concept corresponds to a constant)

any
\begin{align*}
&\text{CO} \\
&\text{o}_\text{CO} \\
&\text{PCO} \\
&\text{o}_\text{lg} \\
&\text{o}_\text{PCO}
\end{align*}

where
\begin{align*}
grd0: & \text{dom(Concept\_parentConcept\_Concept)} \setminus \text{dom(Concept\_corresp\_Constant)} \neq \emptyset \\
grd1: & \text{CO} \in \text{dom(Concept\_parentConcept\_Concept)} \setminus \text{dom(Concept\_corresp\_Constant)} \\
grd2: & \text{dom(Concept\_corresp\_AbstractSet)} \neq \emptyset \\
grd3: & \text{PCO} \in \text{dom(Concept\_corresp\_AbstractSet)} \\
grd4: & \text{Concept\_parentConcept\_Concept(CO)} = \text{PCO} \\
grd5: & \text{Concept\_definedIn\_DomainModel(CO)} \in \text{dom(DomainModel\_corresp\_Component)} \\
grd6: & \text{Constant\_Set} \setminus \text{Constant} \neq \emptyset \\
grd7: & \text{a}_\text{CO} \in \text{Constant\_Set} \setminus \text{Constant} \\
grd8: & \text{LogicFormula\_Set} \setminus \text{LogicFormula} \neq \emptyset \\
grd9: & \text{o}_\text{lg} \in \text{LogicFormula\_Set} \setminus \text{LogicFormula} \\
grd10: & \text{o}_\text{PCO} \in \text{AbstractSet} \\
grd11: & \text{o}_\text{PCO} = \text{Concept\_corresp\_AbstractSet(PCO)}
\end{align*}

then
\begin{align*}
&\text{act1:} \text{Constant} \triangleright= \text{Constant} \cup \{\text{o}_\text{CO}\} \\
&\text{act2:} \text{Concept\_corresp\_Constant(CO)} \triangleright= \text{o}_\text{CO} \\
&\text{act3:} \text{Concept\_definedIn\_Component}(\text{o}_\text{CO}) \triangleright= \text{DomainModel\_corresp\_Component}(
\text{Concept\_definedIn\_DomainModel(CO)}) \\
&\text{act4:} \text{Property} \triangleright= \text{Property} \cup \{\text{o}_\text{lg}\} \\
&\text{act5:} \text{LogicFormula} \triangleright= \text{LogicFormula} \cup \{\text{o}_\text{lg}\} \\
&\text{act6:} \text{LogicFormula\_uses\_Operators}(\text{o}_\text{lg}) \triangleright= \{1 \mapsto \text{Inclusion\_OP}\} \\
&\text{act7:} \text{Constant\_isInvolvedIn\_LogicFormulas}(\text{o}_\text{lg}) \triangleright= \{1 \mapsto \text{o}_\text{lg}\} \\
&\text{act8:} \text{LogicFormula\_involves\_Sets}(\text{o}_\text{lg}) \triangleright= \{2 \mapsto \text{o}_\text{PCO}\} \\
&\text{act9:} \text{LogicFormula\_definedIn\_Component}(\text{o}_\text{lg}) \triangleright= \text{DomainModel\_corresp\_Component}(
\text{Concept\_definedIn\_DomainModel(CO)}) \\
&\text{act10:} \text{Constant\_typing\_Property}(\text{o}_\text{CO}) \triangleright= \text{o}_\text{lg}
\end{align*}

Event \text{rule\_6\_2} (ordinary) \equiv 
correspondence of a concept associated to a parent concept (where the parent concept corresponds to a constant)
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 B System 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 Ontologies_BSystem_specs_translation_ref_1
REFINES Ontologies_BSystem_specs_translation
SEES BSystem_Metamodel_Context, Domain_Metamodel_Context
Event rule_7_1 (ordinary) ≡
correspondence of an instance of Individual (where the concept corresponds to an abstract set)
any
  ind
  o_ind
  CO
  o_la
  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_AbstractSet) ≠ ∅
  grd3 : CO ∈ dom(Concept_corresp_AbstractSet)
  grd4 : Individual_individualOf_Concept(ind) = CO
  grd5 : Constant_definedIn_DomainModel(CO) ∈ dom(DomainModel_corresp_Component)
  grd6 : Constant_Set \ Constant ≠ ∅
  grd7 : o_ind ∈ Constant_Set \ Constant
  grd8 : LogicFormula_Set \ LogicFormula ≠ ∅
  grd9 : o_la ∈ LogicFormula_Set \ LogicFormula
  grd10 : o_CO ∈ AbstractSet
  grd11 : o_CO = Concept_corresp_AbstractSet(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_la}
  act5 : LogicFormula := LogicFormula ∪ {o_la}
  act6 : LogicFormula_used_Operators(o_la) := {1 \ Belonging_OP}
  act7 : Constant_isInvolvedIn_LanguageFormulas(o_ind) := {1 \ o_la}
act8: LogicFormula_involves_Sets(oJg) := \{2 \mapsto o\_CO\}
act9: LogicFormula_definedIn_Component(oJg) := DomainModel_corresp_Component(Concept_definedIn_DomainModel(CO))
act10: Constant_typing_Property(o\_ind) := oJg

END

Event rule7_2 (ordinary)  \equiv

correspondence of an instance of Individual (where the concept corresponds to a constant)

any

ind
o\_ind
CO
oJg
o\_CO

where

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

then

act1: Constant := Constant \cup \{o\_ind\}
act2: Individual\_corresp\_Constant(ind) := o\_ind
act3: Concept\_definedIn\_Component(o\_ind) := DomainModel_corresp\_Component(Concept\_definedIn\_DomainModel(CO))
act4: Property := Property \cup \{oJg\}
act5: LogicFormula := LogicFormula \cup \{oJg\}
act6: LogicFormula\_uses\_Operators(oJg) := \{1 \mapsto Belonging\_OP\}
act7: Constant\_is\_Involved\_In\_LogicFormulas := Constant\_is\_Involved\_In\_LogicFormulas \leftarrow \{(o\_ind \mapsto \{1 \mapsto \{oJg\}\}), o\_CO \mapsto Constant\_is\_Involved\_In\_LogicFormulas(o\_CO) \cup \{2 \mapsto oJg\}\}
act8: LogicFormula\_involves\_Sets(oJg) := \emptyset
act9: LogicFormula\_definedIn\_Component(oJg) := DomainModel_corresp\_Component(Concept\_definedIn\_DomainModel(CO))
act10: Constant\_typing\_Property(o\_ind) := oJg

END

Rule 8 : Data value

MACHINE Ontologies\_BSystem\_specs\_translation\_ref_1
REFINES Ontologies\_BSystem\_specs\_translation
SEES BSystem\_Metamodel\_Context, Domain\_Metamodel\_Context
Event rule8_1 (ordinary)  \equiv

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
oJg
o\_DS
where

grd0: $\text{dom(DataValue_valueOf_DataSet)} \setminus \text{dom(DataValue_corresp_Constant)} \neq \emptyset$
grd1: $dva \in \text{dom(DataValue_valueOf_DataSet)} \setminus \text{dom(DataValue_corresp_Constant)}$
grd2: $\text{dom(CustomDataSet_corresp_AbstractSet)} \neq \emptyset$
grd3: $DS \in \text{dom(CustomDataSet_corresp_AbstractSet)}$
grd4: $\text{DataValue_valueOf_DataSet}(dva) = DS$
grd5: $\text{DataSet_definedIn_DomainModel}(DS) \in \text{dom(DomainModel_corresp_Component)}$
grd6: $\text{Constant_Set} \setminus \text{Constant} \neq \emptyset$
grd7: $o_dva \in \text{Constant_Set} \setminus \text{Constant}$
grd8: $\text{LogicFormula_Set} \setminus \text{LogicFormula} \neq \emptyset$
grd9: $o_{lg} \in \text{LogicFormula_Set} \setminus \text{LogicFormula}$
grd10: $o_{DS} \in \text{AbstractSet}$
grd11: $o_{DS} = \text{CustomDataSet_corresp_AbstractSet}(DS)$

then

act1: $\text{Constant} := \text{Constant} \cup \{o_{dva}\}$
act2: $\text{DataValue_corresp_Constant}(dva) := o_{dva}$
act3: $\text{Constant_definedIn_Component}(o_{dva}) := \text{DomainModel_corresp_Component}(\text{DataSet_definedIn_DomainModel}(DS))$
act4: $\text{Property} := \text{Property} \cup \{o_{lg}\}$
act5: $\text{LogicFormula} := \text{LogicFormula} \cup \{o_{lg}\}$
act6: $\text{LogicFormula_uses_Operators}(o_{lg}) := \{1 \mapsto \text{Belonging_OP}\}$
act7: $\text{Constant_isInvolvedIn_LogicFormulas}(o_{lg}) := \{1 \mapsto o_{lg}\}$
act8: $\text{LogicFormula_involves_Sets}(o_{lg}) := \{2 \mapsto o_{DS}\}$
act9: $\text{LogicFormula_definedIn_Component}(o_{lg}) := \text{DomainModel_corresp_Component}(\text{DataSet_definedIn_DomainModel}(DS))$
act10: $\text{Constant_typing_Property}(o_{dva}) := o_{lg}$

end

END

Rule 10: Constant relation

MACHINE Ontologies_BSystem_specs_translation_ref_1
REFINES Ontologies_BSystem_specs_translation
SEES BSystem_Metamodel_Context,Domain_Metamodel_Context
Event rule_10.1 (ordinary) ≜
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: $\text{Relation_isVariable}^{-1}[[\text{FALSE}]] \setminus \text{dom(Relation_Type)} \neq \emptyset$
grd1: $RE \in \text{Relation_isVariable}^{-1}[[\text{FALSE}]] \setminus \text{dom(Relation_Type)}$
grd2: $\text{dom(Concept_corresp_AbstractSet)} \neq \emptyset$
grd3: $CO1 = \text{Relation_domain_Concept}(RE)$
grd4: $CO2 = \text{Relation_range_Concept}(RE)$
grd5: $\{CO1, CO2\} \subseteq \text{dom(Concept_corresp_AbstractSet)}$
grd6: $\text{Relation_definedIn_DomainModel}(RE) \in \text{dom(DomainModel_corresp_Component)}$
grd7: $\text{Constant_Set} \setminus \text{Constant} \neq \emptyset$
\[\text{Event rule}\_10\_2\ (\text{ordinary}) \triangleq\]

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

\begin{align*}
&\text{RE} \\
&T_{\text{RE}} \\
&o_{\text{RE}} \\
&\text{CO1} \\
&o_{\text{CO1}} \\
&\text{CO2} \\
&o_{\text{CO2}} \\
&o_{\text{lg1}} \\
&o_{\text{lg2}} \\
&\text{DM} \\
\end{align*}

where

\begin{align*}
&\text{grd0: } \text{Relation} \_\text{isVariable}^{-1}[\{\text{FALSE}\}] \setminus \text{dom(Relation}\_\text{Type}) \neq \emptyset \\
&\text{grd1: } \text{RE} \in \text{Relation} \_\text{isVariable}^{-1}[\{\text{FALSE}\}] \setminus \text{dom(Relation}\_\text{Type}) \\
&\text{grd2: } \text{dom(Concept}\_\text{corresp}\_\text{AbstractSet}) \neq \emptyset \\
&\text{grd3: } \text{CO1} = \text{Relation}\_\text{domain}\_\text{Concept(RE)} \\
&\text{grd4: } \text{CO1} \in \text{dom(Concept}\_\text{corresp}\_\text{AbstractSet}) \\
&\text{grd5: } \text{dom(Concept}\_\text{corresp}\_\text{Constant}) \neq \emptyset \\
&\text{grd6: } \text{CO2} = \text{Relation}\_\text{range}\_\text{Concept(RE)} \\
&\text{grd7: } \text{CO2} \in \text{dom(Concept}\_\text{corresp}\_\text{Constant}) \\
&\text{grd8: } \text{Relation}\_\text{definedIn}\_\text{DomainModel(RE)} \in \text{dom(DomainModel}\_\text{corresp}\_\text{Component}) \\
&\text{grd9: } \text{Constant}\_\text{Set} \setminus \text{Constant} \neq \emptyset \\
&\text{grd10: } \{T_{\text{RE}}, o_{\text{RE}}\} \subseteq \text{Constant}\_\text{Set} \setminus \text{Constant} \\
&\text{grd11: } \text{LogicFormula}\_\text{Set} \setminus \text{LogicFormula} \neq \emptyset \\
&\text{grd12: } \{o_{\text{lg1}}, o_{\text{lg2}}\} \subseteq \text{LogicFormula}\_\text{Set} \setminus \text{LogicFormula}
\end{align*}
then

\[\begin{align*}
act1: & \text{Constant} := \text{Constant} \cup \{T_{\text{RE}}, o_{\text{RE}}\} \\
act2: & \text{Relation}_{\text{Type}}(RE) := T_{\text{RE}} \\
act3: & \text{Relation}_{\text{corresp\_Constant}}(RE) := o_{\text{RE}} \\
act4: & \text{Constant}_{\text{defined\_In\_Component}} := \text{Constant}_{\text{defined\_In\_Component}} \cup \{o_{\text{RE}} \mapsto \text{DomainModel}_{\text{corresp\_Component}}(DM), T_{\text{RE}} \mapsto \text{DomainModel}_{\text{corresp\_Component}}(DM)\}
\end{align*}\]

\[\begin{align*}
act5: & \text{Property} := \text{Property} \cup \{o_{\text{lg}1}, o_{\text{lg}2}\} \\
act6: & \text{LogicFormula} := \text{LogicFormula} \cup \{o_{\text{lg}1}, o_{\text{lg}2}\} \\
act7: & \text{Constant}_{\text{typing\_Property}} := \text{Constant}_{\text{typing\_Property}} \cup \{T_{\text{RE}} \mapsto o_{\text{lg}1}, o_{\text{RE}} \mapsto o_{\text{lg}2}\} \\
act8: & \text{Constant}_{\text{isInvolved\_In\_LogicFormulas}} := \text{Constant}_{\text{isInvolved\_In\_LogicFormulas}} \cup \{T_{\text{RE}} \mapsto \{1 \mapsto o_{\text{lg}1}, 2 \mapsto o_{\text{lg}2}\}, o_{\text{RE}} \mapsto \{1 \mapsto o_{\text{lg}2}\}, o_{\text{CO}2} \mapsto \{3 \mapsto o_{\text{lg}1}\} \cup \text{Constant}_{\text{isInvolved\_In\_LogicFormulas}}(o_{\text{CO}2})\}
\end{align*}\]

\[\begin{align*}
act9: & \text{LogicFormula}_{\text{uses\_Operators}} := \text{LogicFormula}_{\text{uses\_Operators}} \cup \{o_{\text{lg}1} \mapsto \{1 \mapsto \text{RelationSet}_{\text{OP}}, o_{\text{lg}2} \mapsto \{2 \mapsto \text{Belonging}_{\text{OP}}\}\} \\
act10: & \text{LogicFormula}_{\text{involves\_Sets}} := \text{LogicFormula}_{\text{involves\_Sets}} \cup \{o_{\text{lg}1} \mapsto \{2 \mapsto o_{\text{CO}1}\}, o_{\text{lg}2} \mapsto \emptyset\} \\
act11: & \text{LogicFormula}_{\text{defined\_In\_Component}} := \text{LogicFormula}_{\text{defined\_In\_Component}} \cup \{o_{\text{lg}1} \mapsto \text{DomainModel}_{\text{corresp\_Component}}(DM), o_{\text{lg}2} \mapsto \text{DomainModel}_{\text{corresp\_Component}}(DM)\}
\end{align*}\]

**Event** \(\text{rule}\_10\_3\) (ordinary) \(\triangleq\)

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**

\(\text{RE}\)

\(\text{T}_{\text{RE}}\)

\(o_{\text{RE}}\)

\(\text{CO}1\)

\(o_{\text{CO}1}\)

\(\text{CO}2\)

\(o_{\text{CO}2}\)

\(o_{\text{lg}1}\)

\(o_{\text{lg}2}\)

\(\text{DM}\)

**where**

\[\begin{align*}
grd0: & \text{Relation}_{\text{isVariable}}^{-1}(\{\text{FALSE}\}) \setminus \text{dom}(\text{Relation}_{\text{Type}}) \neq \emptyset \\
grd1: & \text{RE} \in \text{Relation}_{\text{isVariable}}^{-1}(\{\text{FALSE}\}) \setminus \text{dom}(\text{Relation}_{\text{Type}}) \\
grd2: & \text{dom}(\text{Concept}_{\text{corresp\_Constant}}) \neq \emptyset \\
grd3: & \text{CO}1 = \text{Relation}_{\text{domain\_Concept}}(\text{RE}) \\
grd4: & \text{CO}1 \in \text{dom}(\text{Concept}_{\text{corresp\_Constant}}) \\
grd5: & \text{dom}(\text{Concept}_{\text{corresp\_AbstractSet}}) \neq \emptyset \\
grd6: & \text{CO}2 = \text{Relation}_{\text{range\_Concept}}(\text{RE}) \\
grd7: & \text{CO}2 \in \text{dom}(\text{Concept}_{\text{corresp\_AbstractSet}}) \\
grd8: & \text{Relation}_{\text{defined\_In\_DomainModel}}(\text{RE}) \in \text{dom}(\text{DomainModel}_{\text{corresp\_Component}}) \\
grd9: & \text{Constant}_{\text{Set}} \setminus \text{Constant} \neq \emptyset \\
grd10: & \{\text{T}_{\text{RE}}, o_{\text{RE}}\} \subseteq \text{Constant}_{\text{Set}} \setminus \text{Constant} \\
grd11: & \text{LogicFormula}_{\text{Set}} \setminus \text{LogicFormula} \neq \emptyset \\
grd12: & \{o_{\text{lg}1}, o_{\text{lg}2}\} \subseteq \text{LogicFormula}_{\text{Set}} \setminus \text{LogicFormula} \\
grd13: & o_{\text{CO}2} = \text{Concept}_{\text{corresp\_AbstractSet}}(\text{CO}2) \\
grd14: & o_{\text{CO}1} = \text{Concept}_{\text{corresp\_Constant}}(\text{CO}1)
\end{align*}\]
\[ DM = \text{DefinedIn DomainModel(RE)} \]
\[ T\_RE \neq o\_RE \]
\[ o\_lg1 \neq o\_lg2 \]

then

\[ \text{Event} \text{ rule}\_10\_4 \text{ (ordinary)} \equiv \]

\[ \text{correspondence of an instance of Relation having its isVariable property set to false (case where domain and range correspond to constants)} \]

any

RE
T\_RE
o\_RE
CO1
o\_CO1
CO2
o\_CO2
o\_lg1
o\_lg2
DM

where

\[ \text{relationVariable}^{-1}[\{\text{FALSE}\}] \setminus \text{domain(\text{RelationType})} \neq \emptyset \]
\[ \text{rel} \in \text{relationVariable}^{-1}[\{\text{FALSE}\}] \setminus \text{domain(\text{RelationType})} \]
\[ \text{domain(Concept_correspConstant)} \neq \emptyset \]
\[ \text{CO1} = \text{Relation\_domain\_Concept(RE)} \]
\[ \text{CO2} = \text{Relation\_range\_Concept(RE)} \]
\[ \{\text{CO1}, \text{CO2}\} \subseteq \text{domain(Concept_correspConstant)} \]
\[ \text{Relation\_definedIn\_DomainModel(RE)} \in \text{domain(DomainModel_corresp\_Component)} \]
\[ \text{Constant\_Set} \setminus \text{Constant} \neq \emptyset \]
\[ \text{LogicFormula\_Set} \setminus \text{LogicFormula} \neq \emptyset \]
\[ \{o\_lg1, o\_lg2\} \subseteq \text{LogicFormula\_Set} \setminus \text{LogicFormula} \]
\[ o\_CO1 = \text{Concept_correspConstant}(CO1) \]
\[ o\_CO2 = \text{Concept_correspConstant}(CO2) \]
\[ \text{DM} = \text{Relation\_definedIn\_DomainModel(RE)} \]
\[ T\_RE \neq o\_RE \]
\[ o\_lg1 \neq o\_lg2 \]
\[ o\_CO1 \neq o\_CO2 \]
then

\[
\text{act1: } \text{Constant} := \text{Constant} \cup \{T_{RE}, a_{RE}\} \\
\text{act2: } \text{Relation\_Type}(RE) := T_{RE} \\
\text{act3: } \text{Relation\_corresp\_Constant}(RE) := a_{RE} \\
\text{act4: } \text{Constant\_defined\_In\_Component} := \text{Constant\_defined\_In\_Component} \cup \{a_{RE} \mapsto \text{DomainModel\_corresp\_Component}(DM), T_{RE} \mapsto \text{DomainModel\_corresp\_Component}(DM)\} \\
\]

\[
\text{act5: } \text{Property} := \text{Property} \cup \{o_{Ig1}, o_{Ig2}\} \\
\text{act6: } \text{LogicFormula} := \text{LogicFormula} \cup \{o_{Ig1}, o_{Ig2}\} \\
\text{act7: } \text{Constant\_typing\_Property} := \text{Constant\_typing\_Property} \cup \{T_{RE} \mapsto o_{Ig1}, o_{RE} \mapsto o_{Ig2}\} \\
\text{act8: } \text{Constant\_is\_Involved\_In\_Logic\_Formulas} := \text{Constant\_is\_Involved\_In\_Logic\_Formulas} \mapsto \{T_{RE} \mapsto \{1 \mapsto o_{Ig1}, 2 \mapsto o_{Ig2}, a_{RE} \mapsto \{1 \mapsto o_{Ig2}, 2 \mapsto o_{Ig1}, a_{CO1} \mapsto \{2 \mapsto o_{Ig1}\} \cup \text{Constant\_is\_Involved\_In\_Logic\_Formulas}(a_{CO2}), a_{CO2} \mapsto \{3 \mapsto o_{Ig1}\} \cup \text{Constant\_is\_Involved\_In\_Logic\_Formulas}(a_{CO2})\} \\
\text{act9: } \text{LogicFormula\_uses\_Operators} := \text{LogicFormula\_uses\_Operators} \cup \{o_{Ig1} \mapsto \{1 \mapsto \text{Relation\_Set\_OP}, o_{Ig2} \mapsto \{1 \mapsto \text{Belonging\_OP}\}\} \\
\text{act10: } \text{LogicFormula\_involves\_Sets} := \text{LogicFormula\_involves\_Sets} \cup \{o_{Ig1} \mapsto \emptyset, o_{Ig2} \mapsto \emptyset\} \\
\text{act11: } \text{LogicFormula\_defined\_In\_Component} := \text{LogicFormula\_defined\_In\_Component} \cup \{o_{Ig1} \mapsto \text{DomainModel\_corresp\_Component}(DM), o_{Ig2} \mapsto \text{DomainModel\_corresp\_Component}(DM)\} \\
\]

\end
knowing that the maplets give rise to variables in case of variable relation and constants
in case of constant relationship

then

act1: Constant := Constant ∪ \{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 ∪ \{o_Jg\}
act5: LogicFormula := LogicFormula ∪ \{o_Jg\}
act6: LogicFormula Uses_Operators(o_Jg) := \{1 → Maplet_OP\}
act7: Constant_isInvolvedIn_LoogicFormulas := Constant_isInvolvedIn_LoogicFormulas \{o_remap ↦
  \{1 → o_Jg\}, o_antecedent ↦ \{2 → o_Jg\} \cup Constant_isInvolvedIn_LoogicFormulas(o_antecedent),
  o_image ↦ \{3 → o_Jg\} \cup Constant_isInvolvedIn_LoogicFormulas(o_image)\}
act8: LogicFormula_involves_Sets(o_Jg) := \emptyset
act9: LogicFormula_definedIn_Component(o_Jg) := DomainModel_corresp_Component(
  Relation_definedIn_DomainModel(RE))
act10: Constant_typing_Property(o_remap) := o_Jg

end

Event rule_11_2_1 (ordinary) \(\equiv\)
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_Jg
o_antecedent
o_image

where

grd0: AttributeMaplet \dom(AttributeMaplet_corresp_Constant) \neq \emptyset
grd1: atmap ∈ AttributeMaplet \dom(AttributeMaplet_corresp_Constant)
grd2: \dom(Attribute_corresp_Constant) \cup \dom(Attribute_corresp_Variable) \neq \emptyset
grd3: AttributeMaplet_mapletOf_Attribute(atmap) = AT
grd4: AT ∈ \dom(Attribute_corresp_Constant) \cup \dom(Attribute_corresp_Variable)
grd5: Attribute_definedIn_DomainModel(AT) ∈ \dom(DomainModel_corresp_Component)
grd6: Constant_Set \ Constant \neq \emptyset
grd7: o_atmap ∈ Constant_Set \ Constant
grd8: LogicFormula_Set \ LogicFormula \neq \emptyset
grd9: o_Jg ∈ 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_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 ∪ \{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_Jg\}
act5: LogicFormula := LogicFormula ∪ \{o_Jg\}
act6: LogicFormula_uses_Operators(o_Jg) := \{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), o_image ←
{3 → o/lg} ∪ Constant_isInvolvedIn.LogicFormulas(o_image)}

act8: LogicFormula_involves_Sets(o/lg) := ∅
act8: 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

Each relation gives rise to a constant representing the type of its associated B System element and
defined as the set of relations between the B System element corresponding to the relation domain and the
Rule 14: Constant attribute

MACHINE Ontologies_BSyste_m_specs_translation_ref_1
REFINES Ontologies_BSyste_m_specs_translation
SEES BSyste_m_Metamodel_Context, Domain_Metamodel_Context
Event rule_14_1 (ordinary) ≜ 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) ≠ ∅
grd1: AT ∈ Attribute_isVariable⁻¹[\{FALSE\}] \setminus dom(Attribute_Type)
grd2: dom(Concept_corresp_AbstractSet) ≠ ∅
grd3: CO = Attribute_domain_Concept(AT)
grd4: CO ∈ dom(Concept_corresp_AbstractSet)
grd5: dom(DataSet_corresp_Set) ≠ ∅
grd6: DS = Attribute_range_DataSet(AT)
grd7: DS ∈ dom(DataSet_corresp_Set)
grd8: Attribute_defined_In_DomainModel(AT) ∈ dom(DomainModel_corresp_Component)
grd9: Constant_Set \setminus Constant ≠ ∅
grd10: \{T_AT, o_AT\} ⊆ Constant_Set \setminus Constant
grd11: LogicFormula_Set \setminus LogicFormula ≠ ∅
grd12: \{o_lg1, o_lg2\} ⊆ LogicFormula_Set \setminus LogicFormula
grd13: o_CO = Concept_corresp_AbstractSet(CO)
grd14: o_DS = DataSet_corresp_Set(DS)
grd15: DM = Attribute_defined_In_DomainModel(AT)
grd16: T_AT ≠ o_AT
grd17: o_lg1 ≠ o_lg2
grd18: AT ∈ Attribute_isFunctional⁻¹[\{FALSE\}]

then

act1: Constant := Constant ∪ \{T_AT, o_AT\}
act2: Attribute_Type(AT) := T_AT
act3: Attribute_corresp_Constant(AT) := o_AT
act4: Constant_defined_In_Component := Constant_defined_In_Component ∪ \{o_AT ↦ DomainModel_corresp_Component(DM), T_AT ↦ DomainModel_corresp_Component(DM)\}
act5: Property := Property ∪ \{o_lg1, o_lg2\}
act6: LogicFormula := LogicFormula ∪ \{o_lg1, o_lg2\}
act7: Constant_typing_Property := Constant_typing_Property ∪ \{T_AT ↦ o_lg1, o_AT ↦ o_lg2\}
act8: Constant_isInvolved_In_LogicFormulas := Constant_isInvolved_In_LogicFormulas ∪ \{T_AT ↦ \{1 ⇒ o_lg1, 2 ⇒ o_lg2\}, o_AT ↦ \{1 ⇒ o_lg2\}\}

35
**Event** rule_14_2 (ordinary) ≡

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
- \( \text{grd0: } \text{Attribute_isVariable}^{-1}([\text{FALSE}]) \cap \text{dom(} \text{Attribute_Type} \text{)} \neq \emptyset \)
- \( \text{grd1: } \text{AT} \in \text{Attribute_isVariable}^{-1}([\text{FALSE}]) \cap \text{dom(} \text{Attribute_Type} \text{)} \)
- \( \text{grd2: } \text{dom(} \text{Concept_corresp_Constant} \text{)} \neq \emptyset \)
- \( \text{grd3: } \text{CO} = \text{Attribute_domain_Concept(AT)} \)
- \( \text{grd4: } \text{CO} \in \text{dom(} \text{Concept_corresp_Constant} \text{)} \)
- \( \text{grd5: } \text{dom(} \text{DataSet_corresp_Set} \text{)} \neq \emptyset \)
- \( \text{grd6: } \text{DS} = \text{Attribute_range_DataSet(AT)} \)
- \( \text{grd7: } \text{DS} \in \text{dom(} \text{DataSet_corresp_Set} \text{)} \)
- \( \text{grd8: } \text{Attribute_definedIn_DomainModel(} \text{AT} \text{)} \in \text{dom(} \text{DomainModel_corresp_Component} \text{)} \)
- \( \text{grd9: } \text{Constant_Set} \setminus \text{Constant} \neq \emptyset \)
- \( \text{grd10: } \{ \text{T_AT, o_AT} \} \subseteq \text{Constant_Set} \setminus \text{Constant} \)
- \( \text{grd11: } \text{LogicFormula_Set} \setminus \text{LogicFormula} \neq \emptyset \)
- \( \text{grd12: } \{ \text{o_lg1, o_lg2} \} \subseteq \text{LogicFormula_Set} \setminus \text{LogicFormula} \)
- \( \text{grd13: } \text{o_CO} = \text{Concept_corresp_Constant(CO)} \)
- \( \text{grd14: } \text{o_DS} = \text{DataSet_corresp_Set(DS)} \)
- \( \text{grd15: } \text{DM} = \text{Attribute_definedIn_DomainModel(} \text{AT} \text{)} \)
- \( \text{grd16: } \text{T_AT} \neq \text{o_AT} \)
- \( \text{grd17: } \text{o_lg1} \neq \text{o_lg2} \)
- \( \text{grd18: } \text{AT} \in \text{Attribute_isFunctional}^{-1}([\text{FALSE}]) \)

then

- \( \text{act1: } \text{Constant} := \text{Constant} \cup \{ \text{T_AT, o_AT} \} \)
- \( \text{act2: } \text{Attribute_Type(} \text{AT} \text{)} := \text{T_AT} \)
- \( \text{act3: } \text{Attribute_corresp_Constant(} \text{AT} \text{)} := \text{o_AT} \)
- \( \text{act4: } \text{Constant_definedIn_Component} := \text{Constant_definedIn_Component} \cup \{ \text{o_AT} \mapsto \text{DomainModel_corresp_Component(DM)}, \text{T_AT} \mapsto \text{DomainModel_corresp_Component(DM)} \} \)

- \( \text{act5: } \text{Property} := \text{Property} \cup \{ \text{o_lg1, o_lg2} \} \)
- \( \text{act6: } \text{LogicFormula} := \text{LogicFormula} \cup \{ \text{o_lg1, o_lg2} \} \)
- \( \text{act7: } \text{Constant_typing_Property} := \text{Constant_typing_Property} \cup \{ \text{T_AT} \mapsto \text{o_lg1, o_AT} \mapsto \text{o_lg2} \} \)
- \( \text{act8: } \text{Constant_isInvolvedIn_LogicFormulas} := \text{Constant_isInvolvedIn_LogicFormulas} \cup \{ \text{T_AT} \mapsto \{1 \mapsto \text{o_lg1, 2 \mapsto \text{o_lg2}}, \text{o_AT} \mapsto \{1 \mapsto \text{o_lg2}, \text{o_CO} \mapsto \{2 \mapsto \text{o_lg1}\} \cup \text{Constant_isInvolvedIn_LogicFormulas(o_CO)} \} \)

36
shortcuts

\[\text{act9: LogicFormula\_uses\_Operators} \coloneqq \text{LogicFormula\_uses\_Operators} \cup \{o_{lg1} \mapsto \{1 \mapsto \text{RelationSet\_OP}\}, o_{lg2} \mapsto \{1 \mapsto \text{Belonging\_OP}\}\] 

\[\text{act10: LogicFormula\_involves\_Sets} \coloneqq \text{LogicFormula\_involves\_Sets} \cup \{o_{lg1} \mapsto \{3 \mapsto o_{DS}\}, o_{lg2} \mapsto \emptyset\} \]

\[\text{act11: LogicFormula\_definedIn\_Component} \coloneqq \text{LogicFormula\_definedIn\_Component} \cup \{
\quad o_{lg1} \mapsto \text{DomainModel\_corresp\_Component}(DM), o_{lg2} \mapsto \text{DomainModel\_corresp\_Component}(DM)\}\]

end

Event rule14_3 (ordinary) \(\equiv\)
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 )

\begin{itemize}
  \item any
  \begin{itemize}
    \item AT
    \item T\_AT
    \item o\_AT
    \item CO
    \item o\_CO
    \item DS
    \item o\_DS
    \item o\_lg1
    \item o\_lg2
    \item DM
  \end{itemize}
\end{itemize}

where

\begin{itemize}
  \item [grd0: Attribute\_isVariable\(^{-1}\)[\{FALSE\}] \setminus \text{dom(Attribute\_Type)} \neq \emptyset ]
  \item [grd1: AT \in \text{Attribute\_isVariable\(^{-1}\)[\{FALSE\}] \setminus \text{dom(Attribute\_Type)} ]
  \item [grd2: \text{dom(Concept\_corresp\_AbstractSet)} \neq \emptyset ]
  \item [grd3: CO = \text{Attribute\_domain\_Concept}(AT) ]
  \item [grd4: CO \in \text{dom(Concept\_corresp\_AbstractSet)} ]
  \item [grd5: \text{dom(DataSet\_corresp\_Set)} \neq \emptyset ]
  \item [grd6: DS = \text{Attribute\_range\_DataSet}(AT) ]
  \item [grd7: DS \in \text{dom(DataSet\_corresp\_Set)} ]
  \item [grd8: \text{Attribute\_definedIn\_DomainModel}(AT) \in \text{dom(DomainModel\_corresp\_Component)} ]
  \item [grd9: \text{Constant\_Set}\setminus\text{Constant} \neq \emptyset ]
  \item [grd10: \{T\_AT, o\_AT\} \subseteq \text{Constant\_Set}\setminus\text{Constant} ]
  \item [grd11: \text{LogicFormula\_Set}\setminus\text{LogicFormula} \neq \emptyset ]
  \item [grd12: \{o_{lg1}, o_{lg2}\} \subseteq \text{LogicFormula\_Set}\setminus\text{LogicFormula} ]
  \item [grd13: o_{CO} = \text{Concept\_corresp\_AbstractSet}(CO) ]
  \item [grd14: o_{DS} = \text{DataSet\_corresp\_Set}(DS) ]
  \item [grd15: DM = \text{Attribute\_definedIn\_DomainModel}(AT) ]
  \item [grd16: T\_AT \neq o\_AT ]
  \item [grd17: o_{lg1} \neq o_{lg2} ]
  \item [grd18: AT \in \text{Attribute\_isFunctional\(^{-1}\)[\{TRUE\}] }]
\end{itemize}

then

\begin{itemize}
  \item [act1: \text{Constant} := \text{Constant} \cup \{T\_AT, o\_AT\} ]
  \item [act2: \text{Attribute\_Type}(AT) := T\_AT ]
  \item [act3: \text{Attribute\_corresp\_Constant}(AT) := o\_AT ]
  \item [act4: \text{Constant\_definedIn\_Component} := \text{Constant\_definedIn\_Component} \cup \{
\quad o\_AT \mapsto \text{DomainModel\_corresp\_Component}(DM), T\_AT \mapsto \text{DomainModel\_corresp\_Component}(DM)\}\]
\end{itemize}

\begin{itemize}
  \item [act5: \text{Property} := \text{Property} \cup \{o_{lg1}, o_{lg2}\} ]
  \item [act6: \text{LogicFormula} := \text{LogicFormula} \cup \{o_{lg1}, o_{lg2}\} ]
  \item [act7: \text{Constant\_typing\_Property} := \text{Constant\_typing\_Property} \cup \{T\_AT \mapsto o_{lg1}, o\_AT \mapsto o_{lg2}\} ]
  \item [act8: \text{Constant\_isInvolved\_In\_LogicFormulas} := \text{Constant\_isInvolved\_In\_LogicFormulas} \cup \{T\_AT \mapsto \{
\quad 1 \mapsto o_{lg1}, 2 \mapsto o_{lg2}, o\_AT \mapsto \{1 \mapsto o_{lg2}\}\}\]}
  \item [act9: \text{LogicFormula\_uses\_Operators} := \text{LogicFormula\_uses\_Operators} \cup \{o_{lg1} \mapsto \{1 \mapsto \text{Function\_Set\_OP}\}, o_{lg2} \mapsto \{1 \mapsto \text{Belonging\_OP}\}\} ]
\end{itemize}
\textbf{act10}: \textit{LogicFormula\_involves\_Sets} := \textit{LogicFormula\_involves\_Sets} \cup \{ \textit{o} \_lg1 \mapsto \{ 2 \mapsto \textit{o} \_CO, 3 \mapsto \textit{o} \_DS \}, \textit{o} \_lg2 \mapsto \emptyset \}

\textbf{act11}: \textit{LogicFormula\_defined\_In\_Component} := \textit{LogicFormula\_defined\_In\_Component} \cup \{ \\
\textit{o} \_lg1 \mapsto \textit{DomainModel\_corresp\_Component}(\textit{DM}), \textit{o} \_lg2 \mapsto \textit{DomainModel\_corresp\_Component}(\textit{DM}) \}

\textbf{end Event rule\_14\_4} (ordinary) \equiv

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 )

\textbf{any}

\begin{itemize}
    \item AT
    \item T\_AT
    \item o\_AT
    \item CO
    \item o\_CO
    \item DS
    \item o\_DS
    \item o\_lg1
    \item o\_lg2
    \item DM
\end{itemize}

\textbf{where}

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

\textbf{then}

\begin{itemize}
    \item act1: \textit{Constant} := \text{Constant} \cup \{ T\_AT, o\_AT \}
    \item act2: \textit{Attribute\_Type}(\textit{AT}) := T\_AT
    \item act3: \textit{Attribute\_corresp\_Constant}(\textit{AT}) := o\_AT
    \item act4: \textit{Constant\_defined\_In\_Component} := \textit{Constant\_defined\_In\_Component} \cup \{ o\_AT \mapsto \textit{DomainModel\_corresp\_Component}(\textit{DM}), T\_AT \mapsto \textit{DomainModel\_corresp\_Component}(\textit{DM}) \}
    \item act5: \textit{Property} := \textit{Property} \cup \{ o\_lg1, o\_lg2 \}
    \item act6: \textit{LogicFormula} := \textit{LogicFormula} \cup \{ o\_lg1, o\_lg2 \}
    \item act7: \textit{Constant\_typing\_Property} := \textit{Constant\_typing\_Property} \cup \{ T\_AT \mapsto o\_lg1, o\_AT \mapsto o\_lg2 \}
    \item act8: \textit{Constant\_is\_Involved\_In\_Logic\_Formulas} := \textit{Constant\_is\_Involved\_In\_Logic\_Formulas} \cup \{ 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 \} \}
    \item act9: \textit{Logic\_Formula\_uses\_Operators} := \textit{Logic\_Formula\_uses\_Operators} \cup \{ o\_lg1 \mapsto \{ 1 \mapsto \text{Function\_Set\_OP} \}, o\_lg2 \mapsto \{ 1 \mapsto \text{Belonging\_OP} \} \}
\end{itemize}
Similarly to relations, each attribute gives rise to a constant representing the type of its associated B System element and, in the case when isVariable is set to false, to another constant having its name. However, when the isFunctional attribute is set to true, the constant representing the type is defined as the set of functions between the B System element corresponding to the attribute domain and the one corresponding to the attribute range. The B System 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}).

B System Variables

Rule 9: Variable concept

MACHINE Ontologies_BSystem_specs_translation_ref_1
REFINES Ontologies_BSystem_specs_translation
SEES BSystem_Metamodel_Context, Domain_Metamodel_Context
Event rule_9_1 (ordinary) ⇔
  handling the variability of a concept and initializing the associated variable (when the concept corresponds to an abstract set)

any
  CO
  x_CO
  o_lg
  o_CO
  o_ia
  o_inds
  bij_o_inds

where
  grd0: (dom(Concept_corresp_AbstractSet) ∩ Concept_isVariable⁻¹([TRUE])) \ dom(Concept_corresp_Variable) ≠ ∅
  grd1: CO ∈ (dom(Concept_corresp_AbstractSet) ∩ Concept_isVariable⁻¹([TRUE])) \ dom(Concept_corresp_Variable)
  grd2: Concept_definedIn_DomainModel(CO) ∈ dom(DomainModel_corresp_Component)
  grd3: Individual_individualOf_CONCEPT⁻¹([CO]) ⊆ dom(Individual_corresp_Constant)
  grd4: Variable_Set \ Variable ≠ ∅
  grd5: x_CO ∈ Variable_Set \ Variable
  grd6: LogicFormula_Set \ LogicFormula ≠ ∅
  grd7: o_lg ∈ LogicFormula_Set \ LogicFormula
  grd8: o_CO ∈ AbstractSet
  grd9: o_CO = Concept_corresp_AbstractSet(CO)
  grd10: InitialisationAction_Set \ InitialisationAction ≠ ∅
  grd11: o_ia ∈ InitialisationAction_Set \ InitialisationAction
  grd12: o_inds = Individual_corresp_Constant[Individual_individualOf_CONCEPT⁻¹([CO])]
  grd13: finite(o_inds)
  grd14: bij_o_inds ∈ 1..card(o_inds) ↦ o_inds

then
  act1: Variable := Variable \ {x_CO}
  act2: Concept_corresp_Variable(CO) := x_CO
  act3: Variable_definedIn_Variable(CO) := DomainModel_corresp_Component(CO)
  act4: Invariant := Invariant \ {o_lg}
Variable inds definedIn inds Component definedIn Component Constants o CO OP 9 CO ia Variables CO Concept ia CO Concept corresp Variables Component Component Variable involves isInvolvedIn DomainModel Set corresp uses ia corresp CO Set isInvolvedIn inds CO Component corresp Concept Sets involves CO ia InitialisationAction inds corresp Operators CO DomainModel inds Component inds Component inds Constant Component corresp corresp CO isVariable LogicFormulas corresp involves definedIn CO o inds Operators individualOf Set involves definedIn ia Constant individualOf involves CO corresp corresp CO variable LogicFormulas CO CO corresp CO isVariable OP isVariable Covariant LogicFormulas CO CO corresp CO constant ( handling the variability of a concept and initializing the associated variable (when the concept corresponds to a constant)

any
CO x_CO o_Ig o_CO o_ia o_inds bij_o_inds

where
grd0: (dom(Concept_corresp_Constant) \ Concept_isVariable^{-1}([TRUE])) \ dom(Concept_corresp_Variable) \neq \emptyset
grd1: CO \in (dom(Concept_corresp_Constant) \ Concept_isVariable^{-1}([TRUE])) \ 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 \ Variable \neq \emptyset
grd5: x_CO \in Variable_Set \ Variable
grd6: LogicFormula_Set \ LogicFormula \neq \emptyset
grd7: o_Ig \in LogicFormula_Set \ LogicFormula
grd8: o_CO \in Constant
grd9: o_CO = Concept_corresp_Constant(CO)
grd10: InitialisationAction_Set \ InitialisationAction \neq \emptyset
grd11: o_ia \in InitialisationAction_Set \ InitialisationAction
grd12: o_inds = Individual_corresp_Constant[Individual_individualOf_Concept^{-1}([CO])]
grd13: finite(o_inds)
grd14: bij_o_inds \in 1..\text{card}(o_inds) \rightarrow 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_Ig\}
act5: LogicFormula := LogicFormula \cup \{o_Ig\}
act6: LogicFormula_used_Operators(o_Ig) := \{1 \rightarrow \text{Inclusion}_OP\}
act7: Variable_used_Variables(o_Ig) := \{1 \rightarrow x_CO\}
act8: Concept_isInvolvedIn_LogicsFormulas(o_CO) := Concept_isInvolvedIn_LogicsFormulas(o_CO) \cup \{2 \rightarrow o_Ig\}
act9: LogicFormula_used_Sets(o_Ig) := \emptyset
act10: LogicFormula_definedIn_Component(o_Ig) := DomainModel_corresp_Component(Concept_definedIn_DomainModel(CO))
act11: InitialisationAction := InitialisationAction \cup \{o_ia\}
act12: \textit{InitialisationAction\_uses\_Operators}(o\_ia) := \{1 \mapsto \textit{BecomeEqual2SetOf\_OP}\}
act13: \textit{Variable\_init\_InitialisationAction}(x\_CO) := o\_ia
act14: \textit{InitialisationAction\_involves\_Constants}(o\_ia) := bij\_o\_inds
act15: \textit{Variable\_typing\_Invariant}(x\_CO) := o\_lg

\textbf{Rule 13} : variable relation

\begin{itemize}
\item \textbf{MACHINE} Ontologies\_BSystem\_specs\_translation\_ref\_1
\item \textbf{REFINES} Ontologies\_BSystem\_specs\_translation
\item \textbf{SEES} BSystem\_Metamodel\_Context, Domain\_Metamodel\_Context
\item \textbf{Event} rule\_13\_1 \textup{(ordinary)} \equiv 
\end{itemize}

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)

\begin{itemize}
\item any
\item RE
\item T\_RE
\item o\_RE
\item CO1
\item o\_CO1
\item CO2
\item o\_CO2
\item o\_lg1
\item o\_lg2
\item DM
\item o\_ia
\end{itemize}

\textbf{where}

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

\textbf{then}

\begin{itemize}
\item act1: Constant := Constant \cup \{T\_RE\}
\item act2: Variable := Variable \cup \{o\_RE\}
\item act3: Relation\_Type(RE) := T\_RE
\item act4: Relation\_corresp\_Variable(RE) := o\_RE
\item act5: Constant\_definedIn\_Component(T\_RE) := DomainModel\_corresp\_Component(DM)
\item act6: Variable\_definedIn\_Component(o\_RE) := DomainModel\_corresp\_Component(DM)
\item act7: Property := Property \cup \{o\_lg1\}
\item act8: Invariant := Invariant \cup \{o\_lg2\}
\end{itemize}
In this section, we are interested in the correspondences between the domain model and the B System model that are likely to give rise to invariants, properties or initialization clauses.

Rule 12: closure property or action raised by relation maplets

MACHINE Ontologies_BSystem_specs_translation_ref_1
REFINES Ontologies_BSystem_specs_translation
SEES BSystem_Metamodel_Context, Domain_Metamodel_Context
Event rule_12_1 (ordinary) ≡
closure property for constant relations
any
RE
o_lg
o_RE
maplets
ο_maplets

where
grd0: \( \text{dom}(\text{Relation_corresp}\_\text{Constant}) \neq \emptyset \)
grd1: \( \text{RE} \in \text{dom}(\text{Relation_corresp}\_\text{Constant}) \)
grd2: \( o_{RE} = \text{Relation_corresp}\_\text{Constant}(RE) \)
grd3: \( \text{LogicFormula}\_\text{uses}\_\text{Operators}^{-1}([1 \mapsto \text{Equal2SetOf_OP}\})] \cap \\
\text{ran(\text{Constant_isInvolvedIn_Lo}gicFormulas(o_{RE})]) = \emptyset} \)
grd4: \( \text{RelationMaplet_mapletOf_Relation}^{-1}([\text{RE}]]) = \text{maplets} \)
grd5: \( \text{maplets} \subseteq \text{dom}(\text{RelationMaplet_corresp}\_\text{Constant}) \)
grd6: \( o_{maplets} = \text{RelationMaplet_corresp}\_\text{Constant}[\text{maplets}] \)
grd7: \( \text{Relation_definedIn_DomainModel(RE)} \in \text{dom}(\text{DomainModel_corresp}\_\text{Component}) \)
grd8: \( \text{LogicFormula_Set} \setminus \text{LogicFormula} \neq \emptyset \)
grd9: \( o_{lg} \in \text{LogicFormula_Set} \setminus \text{LogicFormula} \)
grd10: \( o_{RE} \notin o_{maplets} \)

then
act1: Property := Property \cup \{o_{lg}\}
Rule 15: closure property or action raised by relation maplets

MACHINE Ontologies_BSystem_specs_translation_ref_1
REFINES Ontologies_BSystem_specs_translation
SEES BSystem_Metamodel_Context, Domain_Metamodel_Context
Event rule_15_1 (ordinary) \(\equiv\) closure property for constant attribute
any
AT
\(o_{\text{lg}}\)
\(o_{\text{AT}}\)
maplets
  o_maplets

where
grd0:  dom(AttributecorrespConstant) ≠ ∅
grd1:  AT ∈ dom(AttributecorrespConstant)
grd2:  oAT = AttributecorrespConstant(AT)
grd3:  LogicFormula Uses Operators⁻¹[{{1 → Equal2SetOfOP}}] ∩
      ran(AttributecorrespConstantInLogicFormulas(oAT)) = ∅
grd4:  AttributeMaplet_mapletOfAttribute⁻¹[AT] = maplets
grd5:  maplets ⊆ dom(AttributeMaplet_correspConstant)
grd6:  o_maplets = AttributeMaplet_correspConstant[maplets]
grd7:  AttributeDefinedInDomainModel(AT) ∈ dom(DomainModel_correspComponent)
grd8:  LogicsFormula_Set \ LogicFormula ≠ ∅
grd9:  oJl ∈ LogicsFormula_Set \ LogicFormula
grd10:  o_AT ≠ o_maplets

then

act1:  Property := Property ∪ {oJl}
act2:  LogicFormula := LogicFormula ∪ {oJl}
act3:  LogicFormulaUsesOperators(oJl) := {1 → Equal2SetOfOP}
act4:  ConstantIsInvolvesInLogicFormulas := ConstantIsInvolvesInLogicFormulas \-
      ((oAT ⇒ ({1 ⇒ oJl} ∪ ConstantIsInvolvesInLogicFormulas(oAT))) ∪
       {co ⇒ lgs | co ∈ o_maplets ∧ lgs =
        {2 ⇒ oJl} ∪ ConstantIsInvolvesInLogicFormulas(co)})
      appearance order does not matter
act5:  LogicFormulaInvolvesSets(oJl) := ∅
act6:  LogicFormulaDefinedInComponent(oJl) := DomainModel_correspComponent(
      AttributeDefinedInDomainModel(AT))

END

Rule 16 : optional characteristics of relations

MACHINE Ontologies_BSystem_specs_translation_ref_1
REFINES Ontologies_BSystem_specs_translation
SEES BSysten_Metamodel_Context, Domain_Metamodel_Context
Event rule_16_1 (ordinary) ≜
  handling the transitivity of a constant relation
any
  RE
  oJl1
  oJl2
  o_RE

composition

where
grd0:  (dom(AttributecorrespConstant) ∩ RelationIsTransitive⁻¹[TRUE]) ≠ ∅
grd1:  RE ∈ (dom(AttributecorrespConstant) ∩ RelationIsTransitive⁻¹[TRUE])
grd2:  (RE ⇒ isTransitive) ∉ dom(AttributecorrespLogicFormula)
grd3:  oRE = Relation_correspConstant(RE)
grd4:  RelationDefinedInDomainModel(RE) ∈ dom(DomainModel_correspComponent)
grd5:  LogicFormula_Set \ LogicFormula ≠ ∅
grd6:  {oJl1, oJl2} ⊆ LogicFormula_Set \ LogicFormula
grd7:  partition({oJl1, oJl2}, {oJl1}, {oJl2})
grd8:  Constant_Set \ Constant ≠ ∅
grd9:  composition ∈ Constant_Set \ Constant

then

act0:  Constant := Constant ∪ {composition}
act1:  Property := Property ∪ {oJl1, oJl2}
act2:  LogicFormula := LogicFormula ∪ {oJl1, oJl2}
act3:  Constanttyping_Property(composition) := oJl1
We were not interested in validating the transformation rules of predicates expressed using the SysML/KAOS Domain Modeling formalism to B System logical formulas because either uses first-order logic for predicate expression. As a result, the predicates expressed in one of the formalisms are integrally replicated, without additional transformations in the other. When the predicate is an instance of \textit{GluingInvariant}, the assertion raised is an Event-B gluing invariant. For example, in Figure 14 assertion (1.21) is a gluing invariant.
4.3 Handling Updates on B System Specifications within SysML/KAOS Domain Models

Here, we are interested in handling modifications on B System 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 Ontologies_BSystem_specs_translation_ref_1
REFINES Ontologies_BSystem_specs_translation
SEES BSystem_Metamodel_Context,Domain_Metamodel_Context
Event rule_101 (ordinary) ≡
  handling the addition of a new abstract set (correspondence to a concept)
  any
  CO
  o_CO
  where
  grd0: AbstractSet \ (ran(Concept_corresp_AbstractSet) \cup (DataSet_corresp_Set)) ≠ ∅
  grd1: o_CO ∈ AbstractSet \ (ran(Concept_corresp_AbstractSet) \cup (DataSet_corresp_Set))
  grd2: Set_definedIn_Component(o_CO) ∈ ran(DomainModel_corresp_Component)
  grd3: Concept_Set \ Concept ≠ ∅
  grd4: CO ∈ Concept_Set \ 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) ≡
  handling the addition of a new abstract set (correspondence to a custom data set)
  any
  DS
  o_DS
  where
  grd0: AbstractSet \ (ran(Concept_corresp_AbstractSet) \cup (DataSet_corresp_Set)) ≠ ∅
  grd1: o_DS ∈ AbstractSet \ (ran(Concept_corresp_AbstractSet) \cup (DataSet_corresp_Set))
  grd2: Set_definedIn_Component(o_DS) ∈ ran(DomainModel_corresp_Component)
  grd3: DataSet_Set \ DataSet ≠ ∅
  grd4: DS ∈ DataSet_Set \ DataSet
  grd5: DS ∉ {_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 Ontologies_BSystem_specs_translation_ref_1
REFINES Ontologies_BSystem_specs_translation
SEES BSystem_Metamodel_Context, Domain_Metamodel_Context
Event rule_103 (ordinary) =
handling the addition of an enumerated set
any
EDS
o_EDS
elements
o_elements
mapping_elements_o_elements

where

grd0: EnumeratedSet \ ran(DataSet_corresp_Set) \= \emptyset
grd1: o_EDS \in EnumeratedSet \ ran(DataSet_corresp_Set)
grd2: Set_definedIn_Component(o_EDS) \in ran(DomainModel_corresp_Component)
grd3: DataSet_Set \ DataSet \= \emptyset
grd4: EDS \in DataSet_Set \ DataSet
grd5: DataValue_Set \ DataValue \= \emptyset
grd6: elements \subseteq DataValue_Set \ DataValue
grd7: o_elements = SetItem_itemOf_EnumeratedSet^{-1}(o_EDS)
grd8: card(o_elements) = card(elements)
grd9: mapping_elements_o_elements \subseteq elements \mapsto o_elements
grd10: ran(DataValue_corresp_SetItem) \cap o_elements = \emptyset
grd11: EDS \notin \{_NATURAL, _INTEGER, _FLOAT, _BOOL, _STRING\}

then

act1: EnumeratedDataSet := EnumeratedDataSet \cup \{EDS\}
act2: DataSet := DataSet \cup \{EDS\}
act3: EnumeratedDataSet_corresp_EnumeratedSet(EDS) := o_EDS
act4: DataSet_definedIn_DomainModel(EDS) := DomainModel_corresp_Component^{-1}(Set_definedIn_Component(o_EDS))
act5: DataValue := DataValue \cup elements
act6: DataValue_elements_EnumeratedDataSet := DataValue_elements_EnumeratedDataSet \cup \{(xx \mapsto yy) | xx \in elements \land yy = EDS\}
act7: DataValue_corresp_SetItem := DataValue_corresp_SetItem \cup mapping_elements_o_elements
act8: DataSet_corresp_Set := DataSet_corresp_Set \leftarrow \{EDS \mapsto o_EDS\}
act9: DataValue_valueOf_DataSet := DataValue_valueOf_DataSet \cup \{(xx \mapsto yy) | xx \in elements \land yy = EDS\}
act10: CustomDataSet := CustomDataSet \cup \{EDS\}

end
END

Addition of Non-Existing Set Items or Constants

Rule 104: addition of a set item

MACHINE Ontologies_BSystem_specs_translation_ref_1
REFINES Ontologies_BSystem_specs_translation
SEES BSystem_Metamodel_Context, Domain_Metamodel_Context
Event rule_104 (ordinary) =
handling the addition of a new element in an existing enumerated set
any
EDS
o_EDS
element
o_element

where

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

MACHINE Ontologies_BSystem_specs_translation_ref_1
REFINES Ontologies_BSystem_specs_translation
SEES BSystem_Metamodel_Context,Domain_Metamodel_Context
Event rule_105_1 (ordinary) \(\cong\)
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

then

act1: Concept := Concept \cup \{CO\}
act2: Concept_corresp_Constant(CO) := o_CO
act3: Concept_definedIn_DomainModel(CO) := DomainModel_corresp_Component^\{-1\}(Concept_definedIn_Component(o_CO))
act4: Concept_parentConcept_Concept(CO) := PCO
act5: Concept_isVariable(CO) := FALSE

END

Event rule_105_2 (ordinary) \(\cong\)
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

48
where

\[ \text{grd0: } \text{dom}(\text{Constant\_typing\_Property}) \setminus \text{ran}(\text{Concept\_corresp\_Constant}) \neq \emptyset \]
\[ \text{grd1: } a_{\text{CO}} \in \text{dom}(\text{Constant\_typing\_Property}) \setminus \text{ran}(\text{Concept\_corresp\_Constant}) \]
\[ \text{grd2: } a_{\text{lg}} = \text{Constant\_typing\_Property}(a_{\text{CO}}) \]
\[ \text{grd3: } \text{LogicFormula\_uses\_Operators}(a_{\text{lg}}) = \{1 \mapsto \text{Inclusion\_OP}\} \]
\[ \text{grd4: } \text{LogicFormula\_involves\_Sets}(a_{\text{lg}}) = \emptyset \]
\[ \text{grd5: } a_{\text{PCO}} \in \text{dom}(\text{Constant\_isInvolvedIn\_LogicFormulas}) \]
\[ \text{grd6: } (2 \mapsto a_{\text{lg}}) \in \text{Constant\_isInvolvedIn\_LogicFormulas}(a_{\text{PCO}}) \]
\[ \text{grd7: } a_{\text{PCO}} \in \text{ran}(\text{Concept\_corresp\_Constant}) \]
\[ \text{grd8: } \text{PCO} = \text{Concept\_corresp\_Constant}^{-1}(a_{\text{PCO}}) \]
\[ \text{grd9: } \text{Concept\_Set} \setminus \text{Concept} \neq \emptyset \]
\[ \text{grd10: } \text{CO} \in \text{Concept\_Set} \setminus \text{Concept} \]
\[ \text{grd11: } \text{Constant\_definedIn\_Component}(a_{\text{CO}}) \in \text{ran}(\text{Domain\_Model\_corresp\_Component}) \]

\[ \text{then} \]
\[ \text{act1: } \text{Concept} := \text{Concept} \cup \{\text{CO}\} \]
\[ \text{act2: } \text{Concept\_corresp\_Constant}(\text{CO}) := a_{\text{CO}} \]
\[ \text{act3: } \text{Concept\_definedIn\_Domain\_Model}(\text{CO}) := \text{Domain\_Model\_corresp\_Component}^{-1}(\]
\[ \quad \text{Constant\_definedIn\_Component}(a_{\text{CO}})) \]
\[ \text{act4: } \text{Concept\_parentConcept\_Concept}(\text{CO}) := \text{PCO} \]
\[ \text{act5: } \text{Concept\_isVariable}(\text{CO}) := \text{FALSE} \]

\[ \text{END} \]

**Rule 106 : addition of an individual**

**MACHINE** Ontologies\_BSys\_specs\_translation\_ref\_1
**REFINES** Ontologies\_BSys\_specs\_translation
**SEES** BSystem\_Metamodel\_Context, Domain\_Metamodel\_Context

**Event** rule\_106\_1 (ordinary) \[\equiv\]
handling the addition of an individual (case where the concept corresponds to an abstract set)

any
ind
\text{o\_ind}
\text{CO}
\text{o\_lg}
\text{o\_CO}

where

\[ \text{grd0: } \text{dom}(\text{Constant\_typing\_Property}) \setminus \text{ran}(\text{Individual\_corresp\_Constant}) \neq \emptyset \]
\[ \text{grd1: } a_{\text{ind}} \in \text{dom}(\text{Constant\_typing\_Property}) \setminus \text{ran}(\text{Individual\_corresp\_Constant}) \]
\[ \text{grd2: } a_{\text{lg}} = \text{Constant\_typing\_Property}(a_{\text{ind}}) \]
\[ \text{grd3: } \text{LogicFormula\_uses\_Operators}(a_{\text{lg}}) = \{1 \mapsto \text{Belonging\_OP}\} \]
\[ \text{grd4: } \text{LogicFormula\_involves\_Sets}(a_{\text{lg}}) \neq \emptyset \]
\[ \text{grd5: } (2 \mapsto a_{\text{CO}}) \in \text{LogicFormula\_involves\_Sets}(a_{\text{lg}}) \]
\[ \text{grd6: } a_{\text{CO}} \in \text{ran}(\text{Concept\_corresp\_Abstract\_Set}) \]
\[ \text{grd7: } \text{CO} = \text{Concept\_corresp\_Abstract\_Set}^{-1}(a_{\text{CO}}) \]
\[ \text{grd8: } \text{Individual\_Set} \setminus \text{Individual} \neq \emptyset \]
\[ \text{grd9: } \text{ind} \in \text{Individual\_Set} \setminus \text{Individual} \]

\[ \text{then} \]
\[ \text{act1: } \text{Individual} := \text{Individual} \cup \{\text{ind}\} \]
\[ \text{act2: } \text{Individual\_individualOf\_Concept}(\text{ind}) := \text{CO} \]
\[ \text{act3: } \text{Individual\_corresp\_Constant}(\text{ind}) := a_{\text{ind}} \]

**Event** rule\_106\_2 (ordinary) \[\equiv\]
handling the addition of an individual (case where the concept corresponds to a constant)

any
ind
\text{o\_ind}
\text{CO}
\begin{verbatim}

where

grd0: dom(Constant_typing_Property) \ ran(Individual_corresp_Constant) \= \emptyset
grd1: o_ind \in dom(Constant_typing_Property) \ ran(Individual_corresp_Constant)
grd2: o_lg = Constant_typing_Property(o_ind)
grd3: LogicFormula\_uses\_Operators(o_lg) = \{1 \map \text{Belonging\_OP}\}
grd4: LogicFormula\_involves\_Sets(o_lg) = \emptyset
grd5: o_CO \in dom(Constant_isInvolvedIn\_LogicFormulas)
grd6: (2 \map 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 \= \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 Ontologies\_BSystem\_specs\_translation\_ref\_1
REFINES Ontologies\_BSystem\_specs\_translation
SEES BSystem\_Metamodel\_Context,Domain\_Metamodel\_Context
Event rule\_107 (ordinary) \(=\)

handling the addition of a data value

any
dva
o_dva
DS
o_lg
o_DS

where

grd0: dom(Constant_typing_Property) \ ran(DataValue_corresp_Constant) \= \emptyset
grd1: o_dva \in dom(Constant_typing_Property) \ ran(DataValue_corresp_Constant)
grd2: o_lg = Constant_typing_Property(o_dva)
grd3: LogicFormula\_uses\_Operators(o_lg) = \{1 \map \text{Belonging\_OP}\}
grd4: LogicFormula\_involves\_Sets(o_lg) \neq \emptyset
grd5: (2 \map 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 \= \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

Rule 109 : addition of a constant, defined as a maplet

MACHINE Ontologies\_BSystem\_specs\_translation\_ref\_1
REFINES Ontologies\_BSystem\_specs\_translation
SEES BSystem\_Metamodel\_Context,Domain\_Metamodel\_Context
Event rule\_109\_1 (ordinary) \(=\)

handling the addition of a constant, defined as a maplet, element of a relation (case where the relation corresponds to a constant relation)

\end{verbatim}
Addition of Non-Existing Variables

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

MACHINE Ontologies_BSystem_specs_translation_ref_1 
REFINES Ontologies_BSystem_specs_translation 
SEES BSystem_Metamodel_Context,Domain_Metamodel_Context 
Event rule_108_1 (ordinary) ≜ 
  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
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 B System 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 B System 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 [17] and at evaluating the impact of updates on domain models on B System specifications.

Acknowledgment

This work is carried out within the framework of the FORMOSE project [4] funded by the French National Research Agency (ANR).
Fig. 15. Preview of the SysML/KAOS Domain Model Parser Tool

References

1. Abrial, J.: Modeling in Event-B - System and Software Engineering. Cambridge University Press (2010)
2. Alkhammash, E., Butler, M.J., Fathabadi, A.S., Cîrstea, C.: Building traceable Event-B models from requirements. Sci. Comput. Program. 111, 318–338 (2015)
3. Alkhammash, Eman H.: Derivation of Event-B Models from OWL Ontologies. MATEC Web Conf. 76, 04008 (2016)
4. ANR-14-CE28-0009: Formose ANR project (2017)
5. Bjørner, D., Eir, A.: Compositionality: Ontology and mereology of domains. Essays in Honor of Willem-Paul de Roever, LNCS, vol. 5930, pp. 22–59. Springer (2010)
6. Boniol, F., Wiels, V.: The landing gear system case study. pp. 1–18. ABZ, Springer (2014)
7. ClearSy: Atelier B: B System (2014), http://clearsy.com/
8. Doberkat, E.: The Object-Z specification language. Softwaretechnik-Trends 21(1) (2001)
9. Dong, J.S., Sun, J., Wang, H.H.: Z approach to semantic web. In: Formal Methods and Software Engineering - ICFEM, LNCS. vol. 2495, pp. 156–167. Springer (2002)
10. Gnaho, C., Semmak, F.: Une extension SysML pour l’ingénierie des exigences dirigée par les buts. In: 28e Congrès INFORSID, France. pp. 277–292 (2010)
11. van Harmelen, F., Patel-Schneider, P.F., Horrocks, I.: Reference description of the DAML+ OIL ontology markup language (2001)
12. JetBrains: JetBrains MPS (2017), https://www.jetbrains.com/mps/
13. Laleau, R., Mammar, A.: An overview of a method and its support tool for generating B specifications from UML notations. In: The Fifteenth IEEE International Conference on Automated Software Engineering, ASE 2000, Grenoble, France, September 11-15, 2000. pp. 269–272. IEEE Computer Society (2000), 1109/ASE.2000.873675
14. van Lamsweerde, A.: Requirements Engineering - From System Goals to UML Models to Software Specifications. Wiley (2009)
15. Mammar, A., Laleau, R.: On the use of domain and system knowledge modeling in goal-based Event-B specifications. In: ISoLA 2016, LNCS. pp. 325–339. Springer (2016)
Fig. 16. Preview of B System Specifications Generated by the SysML/KAOS Domain Model Parser Tool for the Landing Gear System Case Study

16. Matoussi, A., Gervais, F., Laleau, R.: A goal-based approach to guide the design of an abstract Event-B specification. In: ICECCS 2011. pp. 139–148. IEEE Computer Society (2011)
17. Openflexo: Openflexo project (2015), http://www.openflexo.org
18. 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)
19. Sengupta, K., Hitzler, P.: Web ontology language (OWL). In: Encyclopedia of Social Network Analysis and Mining, pp. 2374–2378 (2014)
20. Snook, C., Butler, M.: UML-B: Formal Modeling and Design Aided by UML. ACM Trans. Softw. Eng. Methodol. 15(1), 92–122 (Jan 2006)
21. Tueno, S.: SysML/KAOS Domain Model Parser (2017), https://github.com/stuenofotso/SysML_KAOS_Domain_Model_Parser
22. Tueno, S., Laleau, R., Mammar, A., Frappier, M.: Towards Using Ontologies for Domain Modeling within the SysML/KAOS Approach. IEEE proceedings of MoDRE workshop, 25th IEEE International Requirements Engineering Conference (2017)
23. UL, I.: Owlgred home (2017), http://owlged.lumii.lv/
24. 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)