

BSYS_EVAL

Ionuț Cicio

01/09/2025

<https://github.com/CuriousCI/bsys-eval>

Contents

1	BSYS_EVAL	3
1.1	Introduction	3
1.2	Requirements	4
2	Data types specification	6
2.1	Interval	6
2.2	ReactomeDbId	6
2.3	StableIdVersion	7
3	(<i>Reactome</i>) UML class diagram	8
4	Classes specification pt.1	9
4.1	CatalystActivity	9
4.2	Compartment	9
4.3	Pathway	10
4.4	PhysicalEntity	10
5	(<i>Simulation</i>) UML class diagram	12
6	Classes specification pt.2	13
6.1	CompartmentInstance	13
6.2	13
6.3	Model	13
6.4	ModelInstance	13
6.5	SimulatedModelInstance	13
6.6	ReactionParameter??	14
6.7	UnitDefinition	14
	Bibliography	15

1 BSYS_EVAL

1.1 Introduction

BSYS_EVAL is a tool meant to help study the likelihood of a given situation in a biological system.

Given a set of *target species*, a set of constraints on the *target species* (constraints which model a situation that could present, for example, in a disease) and by taking into account all the reactions within a set *target pathways* that lead to the production, both directly and indirectly, of the *target species*, the goal is to find a subset of virtual patients for the described situation.

TODO: find papers in literature that do similar things; what does this method add compared to other approaches? (i.e. using multiple pathways by generating the fixed point, ensemble of SAs etc...)

TODO: add case study, multiple if possible

1.2 Requirements

The basic idea behind the software is to take the description of a model (with *target species*, *target pathways*, constraints on the *target species*, and the parameters $\varepsilon, \delta \in (0, 1)$ for the evaluation of the constraints), to generate a SBML model with

- all the reactions within the *target pathways* that, both directly and indirectly, generate the *target species*
- parameters for the reactions' speeds
- structural constraints on the reactions' speeds (some reactions are faster than others)

TODO: I still haven't figured out how to get that information out of Reactome, maybe I just have to search more

- constraints on the quantities of the entities (for which the model needs to be simulated)

TODO: possibly take a configuration file as input, maybe P_Etab could be good, otherwise JSON should be enough, as everything else is generated automatically from Reactome, the model should work with both StableIDVersion and ReactomeDbId;

The **TargetPathways** should be optional. The ExtraConstraints should be optional. The PreferredCompartmentForSimulation could be specified.

Algorithm 1: eval

```
input:  $S_T$ , set of PhysicalEntity;  
input:  $C_T$ , set of constraints on  $S_T$ ;  
input:  $P_I$ , set of target pathways;  
input:  $\varepsilon, \delta \in (0, 1)$ ;  
input: seed, random seed;  
  
 $F \leftarrow \text{fixed\_point}(S_T, P_I)$   
model  $\leftarrow (S_T, S(F), R(F), E(F))$   
env  $\leftarrow$  define env for model  
 $V = \emptyset$  // set of virtual patients  
  
while  $\neg$  halt requested do  
   $v \leftarrow$  parameter assignement for model // virtual patient  
  if  $\neg v$  satisfies structural constraints then  
    continue;  
  if APSG(model,  $v$ , env, seed,  $\varepsilon$ ,  $\delta$ ) then  
     $V \leftarrow V \cup \{v\}$ ;
```

The idea is to expand a portion of Reactome

Definition 1 (... *Model*). A ... model G is a tuple (S_T, S, R, E) where:

- S_T the set of target species
- S is the finite set of species s.t.
 - $S_T \subseteq S$
 - S is the transitive closure of S_T within the Reactome graph (to be more precise, the closure within the specified bounds, bounds yet to be defined)
 - $S' = S \cup \{s_{\text{avg}} \mid s \in S\}$.
 - $\dot{s} = f(s_1, s_2, s_3, \dots, s_n)$
- R is the finite set of reactions
 - $R = R_{\text{fast}} \cup R_{\text{slow}}$
- E is the set edges in the graph (where an edge goes from a species to a reaction, it also has a stoichiometry)
 - $E \subseteq S \times R \times \mathbb{N}^1$
 - $E = E_{\text{reactant}} \cup E_{\text{product}} \cup E_{\text{modifier}}$
 - TODO: account for order (edges also have an “order” attribute, I have to check how it impacts the simulation and if it’s optional)

Average quantities

- $S' = S \cup \{s_{\text{avg}} \mid s \in S\}$
- $S' = G(S')$
- $K : R \rightarrow \mathbb{R}_+^{|R|} = [10^{-6}, 10^6]^{|R|}$
- find k
- subject to
 - structural constraints
 - partial order on k due to
 - fast/non fast reactions (TODO: as given by Reactome, but how?)

$$\forall r_f, r_s \ (r_f \in R_{\text{fast}} \wedge r_s \in R_{\text{slow}}) \rightarrow r_f > r_s$$

- reaction modifiers (like above?)
- for all dynamics of environment
 - avg concentration of species consistent to knowledge

$$\exists t_0 \ \forall t \ \forall s$$

$$(t > t_0 \wedge s \in S_{\text{avg}}) \rightarrow s(t) \in [\text{known range}]$$

2 Data types specification

- `\d` = `/[0-9]/`
- `\w` = `/[A-Za-z0-9_]/`

Math

`Interval` = (min: `Real` [0..1], max: `Real` [0..1])

`MathML` = `String` matching <https://www.w3.org/1998/Math/MathML/>

`MathMLBoolean` = `String` matching <https://www.w3.org/1998/Math/MathML/>

Reactome

`ReactomeDbId` = `Integer` [1]

`StableIdVersion` =

`String` matching regex `/^R-[A-Z]{3}-\d{8}\.\d{2,3}$/` [2]

SBML

`String1` = `String` matching regex `//`

`SId` = `String` matching regex `/^[a-zA-Z_]\w*$/` [3, Section 3.1.7]

2.1 Interval

The `Interval` type represents an open interval in \mathbb{R} of the type (min,max) s.t.

- when min is not defined, it is interpreted as $-\infty$
- when max is not defined, it is interpreted as $+\infty$

C.`Interval`.min_leq_max

\forall interval, interval_min, interval_max
(
 `Interval`(interval) \wedge
 `min`(interval, interval_min) \wedge
 `max`(interval, interval_max)
) \rightarrow
 interval_min \leq interval_max

2.2 ReactomeDbId

This is required because not all instances of `DatabaseObject` in Reactome have a `StableIdVersion`, which is the one usually displayed in the Reactome Pathway Browser [4]. Instances of `DatabaseObject` in Reactome can be identified with a `ReactomeDbId`, but its pattern does not match the definition of `SId` used to identify objects in SBML.

In order to generate a correct `SBMLModel` the `ReactomeDbId` must be converted into a `SId`.

```
ReactomeDbId_into_SId(db_id: ReactomeDbId): SId
```

POSTCONDITIONS:

. . .

2.3 StableIdVersion

The **StableIdVersion** type is useful because is the one usually displayed in the Reactome Pathway Browser [4]. It is useful to accept it in the description of the models.

The **StableIdVersion** type is used to identify instances of **PhysicalEntity** or **Event** in Reactome, but it's pattern does not match the definition of **SId** used to identify objects in SBML.

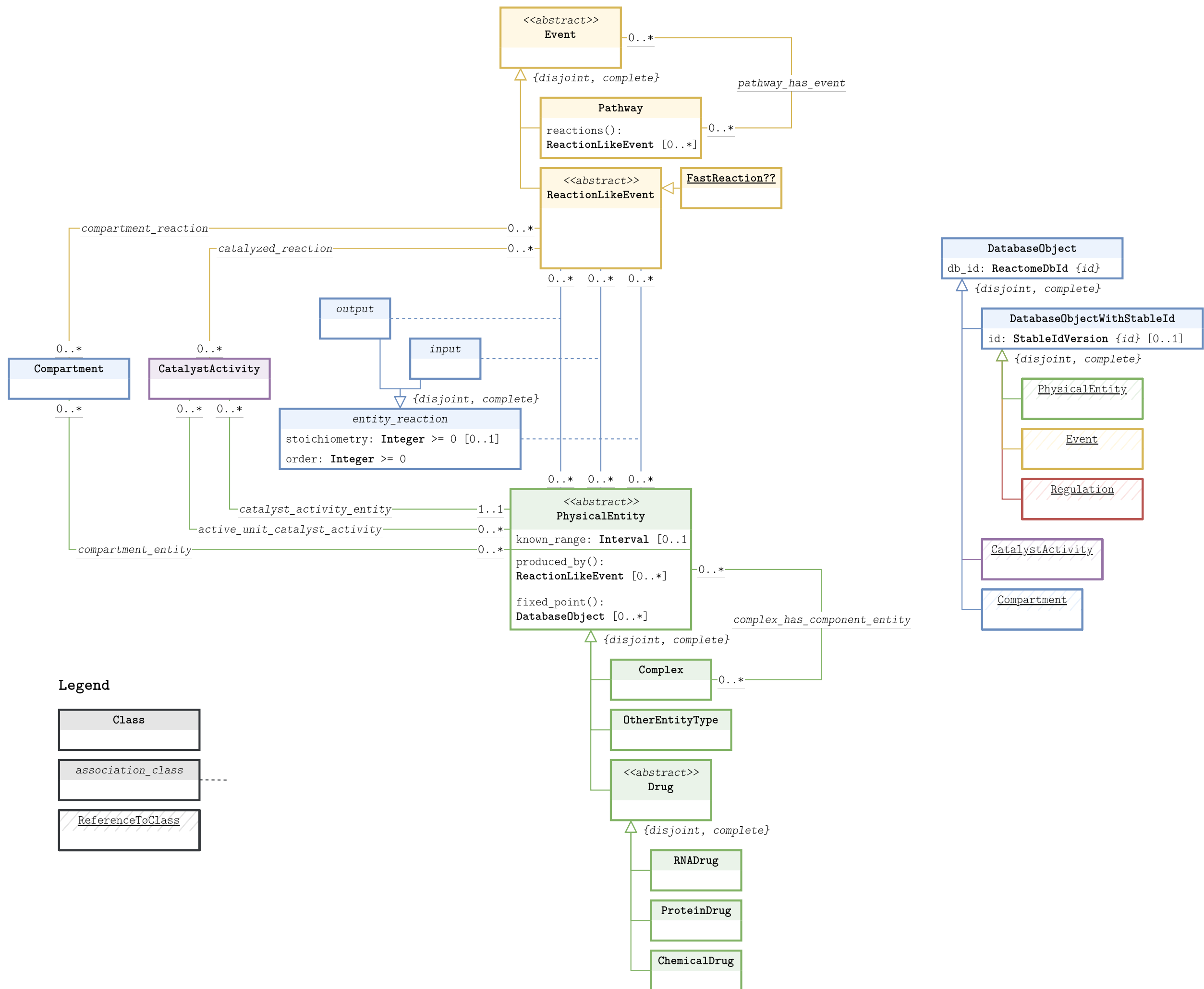
In order to generate a correct **SBMLModel** the **StableId** must be converted.

```
StableIdVersion_into_SId(st_id: StableId): SId
```

POSTCONDITIONS:

. . .

3 (*Reactome*) UML class diagram



4 Classes specification pt.1

4.1 CatalystActivity

The one above is the reason why a `PhysicalEntity`'s role in `catalyst_activity_entity` has multiplicity 0..*.

“If a `PhysicalEntity` can enable multiple molecular functions, a separate `CatalystActivity` instance is created for each” [5, Page 5]

“If the `PhysicalEntity` is a `Complex` and a component of the complex mediates the molecular function, that component should be identified as the active unit of the `CatalystActivity`.” [5, Page 5]

C.`CatalystActivity`.active_unit_is_in_complex

```

  ∀ catalyst_activity, complex, complex_component
    (
      CatalystActivity(catalyst_activity) ∧
      Complex(complex) ∧
      PhysicalEntity(complex_component) ∧
      catalyst_entity(catalyst_activity, complex) ∧
      catalyst_active_unit(catalyst_activity, complex_component)
    ) →
      complex_has_component_entity(complex, complex_component)

```

4.2 Compartment

TODO: move this information to the `compartment_entity` association, or to `PreferredCompartmentForSimulation`

The `Compartment` class has some quirks. In Reactome, the `Compartment`'s role in the `compartment_entity` association has multiplicity 0..*. The problem is that the SBML model requires 1..1 multiplicity for this association to be simulated.

In Reactome there are currently (TODO: version??) 19 physical entities which don't have a compartment (see queries/helper.cypher), so this can be easily solved by just adding a **default compartment** to the SBML model to which these entities map to.

On the other hand there are 14046 entities which have multiple compartments (TODO: how many compartments has each exactly?), so the easiest choice right now is to just pick any of them. For this reason the

4.3 Pathway

The instances of **Pathway** are organized hierarchically, i.e. the signaling pathways are collected under the Signal Transduction **Pathway** (**StableIdVersion** R-HSA-162582.13). This allows to easily cut a subset of reactions by specifying the *target pathways* in a model and taking into consideration only the reactions which are both directly or indirectly under that pathway (see the **reactions()** operation).

Ignoring the *inferred_to* association there are about 34 root pathways.

TODO: handle *inferred_to*

reactions(): **ReactionLikeEvent** [0..*]

POSTCONDITIONS:

```
result =
  { reaction |
    ReactionLikeEvent(reaction) ∧
    pathway_has_event(this, reaction) }
  ∪
  { reaction | ∃ pathway
    Pathway(pathway) ∧
    pathway_has_event(this, pathway) ∧
    reactions(pathway, reaction) }
```

4.4 PhysicalEntity

TODO: how should I handle complexes here?

produced_by(): **ReactionLikeEvent** [0..*]

POSTCONDITIONS:

```
result = { reaction |
  ReactionLikeEvent(reaction) ∧
  output(this, reaction) ∧
  ¬ ∃ pathway
    IgnoredPathway(pathway) ∧ reactions(pathway, reaction)
}
```

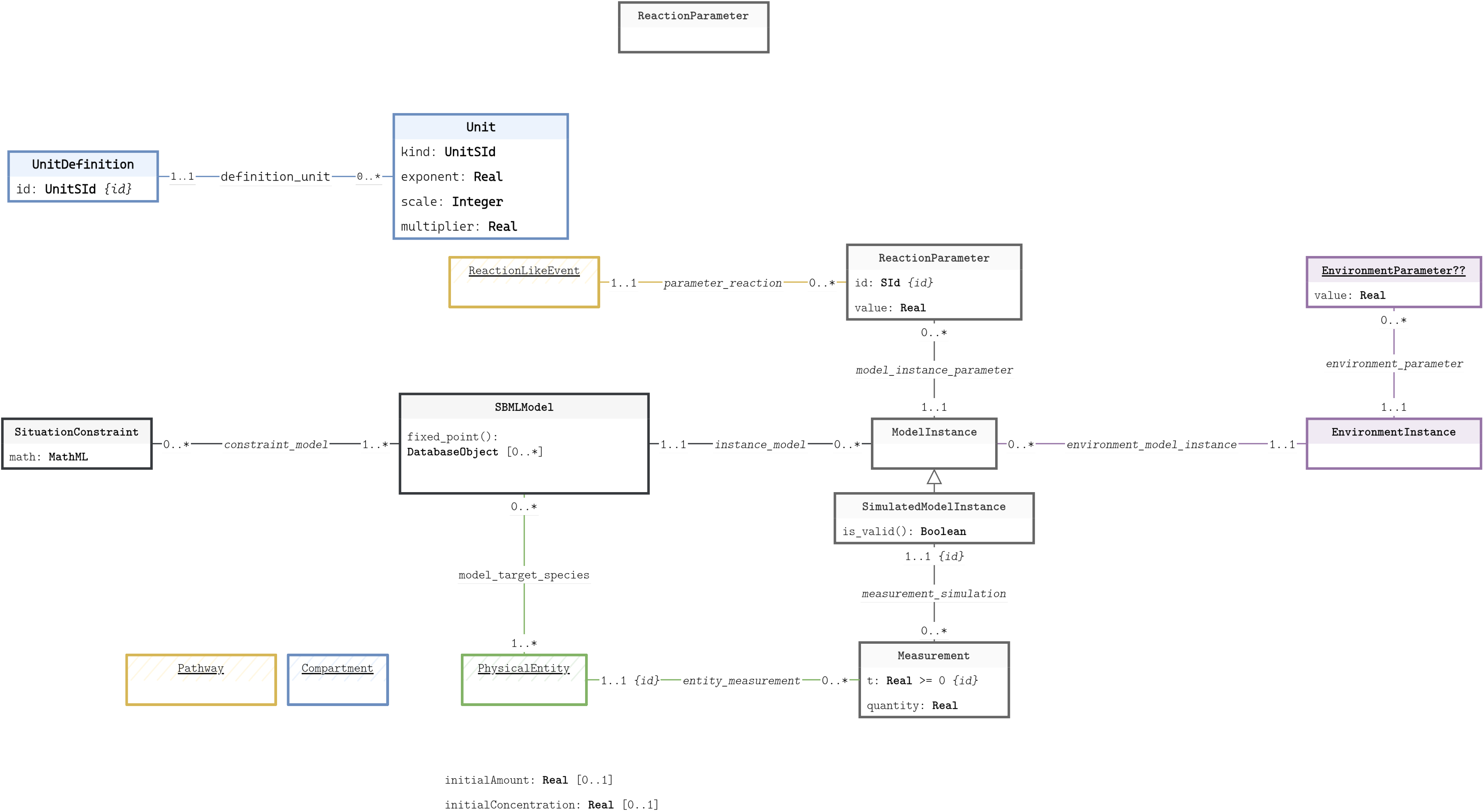
`fixed_point(): DatabaseObject [0..*]`

POSTCONDITIONS:

```
result =
  { this } ∪
  produced_by(this) ∪
  { object | ∃ reaction, reaction_input
    produced_by(this, reaction) ∧
    (
      input(reaction, reaction_input) ∨
      (∃ catalyst_activity
        CatalystActivity(catalyst_activity) ∧
        catalyzed_event(catalyst_activity, reaction)) ∧
        catalyst_entity(
          catalyst_activity,
          reaction_input
        )
      )
    } ∧
    fixed_point(reaction_input, object)
  }
```

TODO: rename `fixed_point()` to `backwards_reachable_objects()`

5 (Simulation) UML class diagram



6 Classes specification pt.2

6.1 CompartmentInstance

C.[CompartmentInstance](#).entities_have_compartment_listed

```
∀ compartment_instance, compartment, species, physical_entity
(
    CompartmentInstance(compartment_instance) ∧
    Compartment(compartment) ∧
    SpeciesInstance(species) ∧
    PhysicalEntity(physical_entity) ∧
    compartment\_instance(compartment, compartment_instance) ∧
    compartment\_species(compartment_instance, species) ∧
    physical\_entity\_species(physical_entity, species)
) →
    compartment\_entity(compartment, physical_entity)
```

TODO: what happens if it a PhysicalEntity has some compartments?

6.2

6.3 Model

[fixed_point](#)() : [DatabaseObject](#) [1..*]

```
POSTCONDITIONS:
    result = { object | ∃ entity
        initial\_entity\_model(this, entity) ∧
        fixed\_point(entity, object)
    }
```

6.4 ModelInstance

C.[ModelInstance](#).every_reaction_has_a_parameter

C.[ModelInstance](#).reaction_parameters_are_structurally_valid

6.5 SimulatedModelInstance

[is_valid](#)()

```
POSTCONDITIONS:
```

6.6 ReactionParameter??

- it must satisfy structural constraints

6.7 UnitDefinition

Basically a unit definition is a product of the single units inside (m/s , m/s^2 etc...), easy as that.

Bibliography

- [1] [Online]. Available: <https://reactome.org/content/schema/DatabaseObject>
- [2] “Reactome.” [Online]. Available: <https://reactome.org/documentation/faq/37-general-website/201-identifiers>
- [3] [Online]. Available: <https://raw.githubusercontent.com/combine-org/combine-specifications/main/specifications/files/sbml.level-3.version-2.core.release-2.pdf>
- [4] [Online]. Available: <https://reactome.org/PathwayBrowser/>
- [5] [Online]. Available: https://download.reactome.org/documentation/DataModelGlossary_V90.pdf