Bsys_eval

Ionuţ Cicio

29/08/2025

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

Contents

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 evalation 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 PEtab 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 `TargetPathway`s 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: ε, δ ∈ (0, 1);
input: seed, random seed;

F ← fixed_point(S_T, P_I)
model ← (S_T, S(F), R(F), E(F))
env ← define env for model
V = ∅ // set of virtual patients

while ¬ halt requested do
   v ← parameter assignement for model // virtual patient
   if ¬ v satisfies structural constraints then
     continue;
   if APSG(model, v, env, seed, ε, δ) then
     V ← V ∪ {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_I within the Reactome graph (to be more precise, the closure within the specified bounds, bounds yet to be defined)
  - $S' = S \cup \{s_{\text{avg}} \mid s \in S\}$.
  - $\dot{s} = f(s_1, s_2, s_3, ..., s_n)$
- R is the finite set of reactions
  - $R = R_{\text{fast}} \cup R_{\text{slow}}$
- E is the set edges in the graph (where and 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

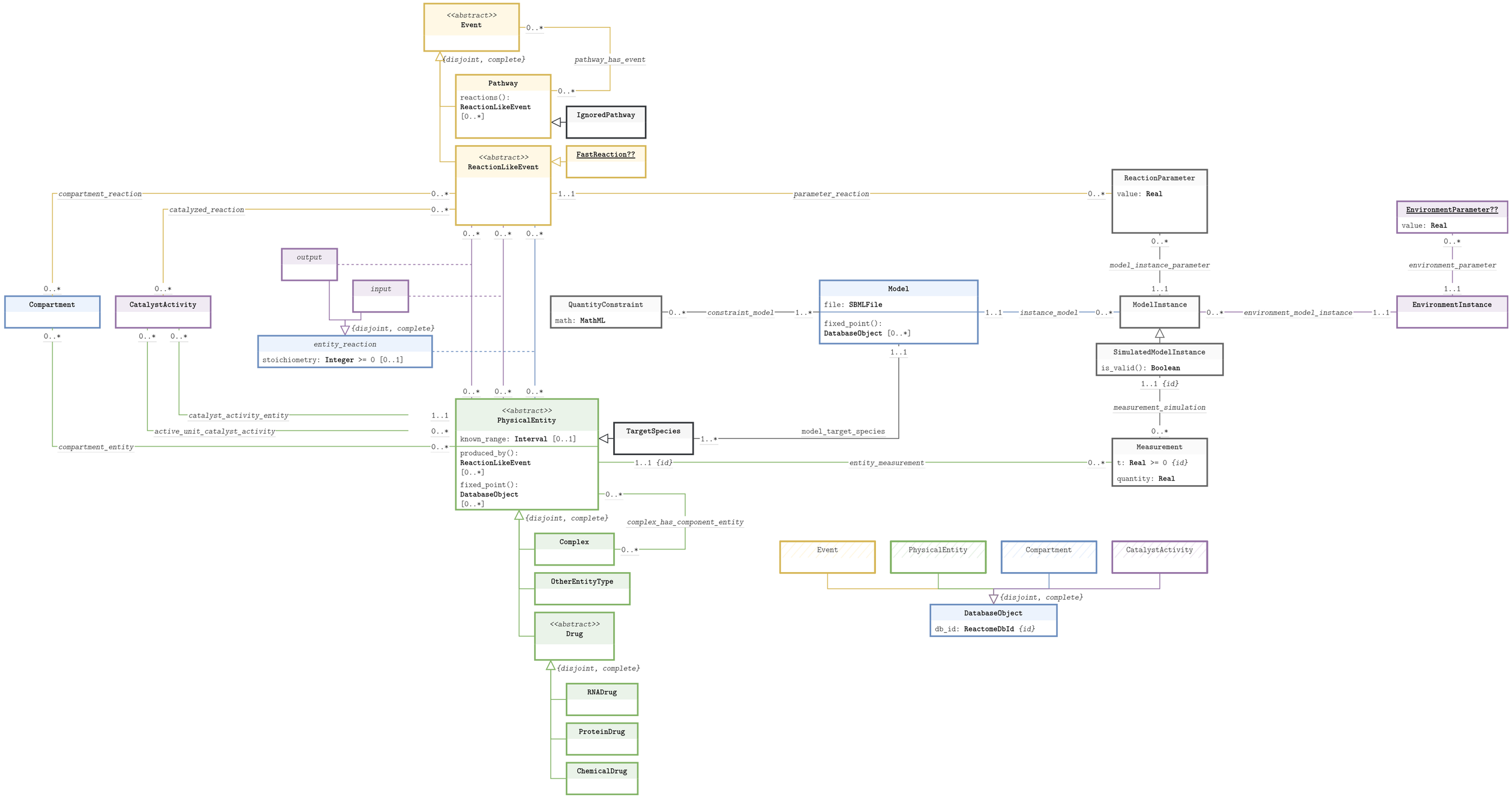
- $S' = S \cup \{S_{\text{avg}} \mid s \in S\}$
- $S' = G(S')$
- $K : R \to \mathbb{R}_+^{|R|} = [10^{-6}, 10^6]^{|R|}$
- find k
- subject to
  - structural constraints
    - partial order on k due to
      - fast/non fast reactions (TODO: as given by Reactome, but how?)

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

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

$$\exists t_0 \ \ \forall t \ \ \forall s$$
$$(t > t_0 \land s \in S_{\text{avg}}) \to s(t) \in [\text{known range}]$$

2 UML class diagram



3 Data types specification

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

```
ReactomeDbId = Integer [1]
StableIdVersion (Reactome) =
   String matching regex /^R-[A-Z]{3}-\d{8}\.\d{2,3}$/ [2]
SId = String matching regex /^[a-zA-Z_]\w*$/ [3, Section 3.1.7]
Interval = (min: Real [0..1], max: Real [0..1])
MathML = String according to https://www.w3.org/1998/Math/MathML/
```

3.1 Interval

The `Interval` type represents a real open interval of the type (min, max).

`C.Interval.min_leq_max`

$$
\begin{aligned}
&\forall\ \text{interval},\ \text{interval\_min},\ \text{interval\_max} \\
&\quad (\\
&\qquad \text{Interval}(\text{interval}) \land \\
&\qquad \text{min}(\text{interval},\ \text{interval\_min}) \land \\
&\qquad \text{max}(\text{interval},\ \text{interval\_max}) \\
&\quad) \rightarrow \\
&\qquad \text{interval\_min} \leq \text{interval\_max}
\end{aligned}
$$

3.2 ReactomeDbId

Other Reactome entities can be identified with a `ReactomeDbId`, but it's pattern does not match the definition of `SId` used to identify objects in SBML. In order to generate a correct SBML Model the `ReactomeDbId` must be converted.

`into(db_id: ReactomeDbId): SId`

POSTCONDITIONS:

. . .

3.3 StableId

The `StableId` type is used to identify a `PhysicalEntity` or an `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 SBML Model the `StableId` must be converted.

`into(st_id: StableId): SId`

POSTCONDITIONS:

. . .

4 Classes specification

4.1 CatalystActivity

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

"If a PhysicalEntity can enable multiple molecular functions, a separate CatalystActivity instance is created for each" [4, 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." [4, 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 Event

4.4 FastReaction

4.5 Model

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

```
  POSTCONDITIONS:
    result = { object | ∃ entity
        initial_entity_model(this, entity) ∧
        fixed_point(entity, object)
    }
```

4.6 ModelInstance

```
C.ModelInstance.every_reaction_has_a_parameter
```

```
C.ModelInstance.reaction_parameters_are_structurally_valid
```

4.7 SimulatedModelInstance

```
is_valid()
```

```
  POSTCONDITIONS:
```

4.8 Pathway

```
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) }
```

4.9 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)
  }
```

TODO: union with `CatalystActivity`

`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)
      }
```

4.10 ReactionLikeEvent

4.11 ReactionParameter??

- it must satisfy structural constraints

Bibliography

[1] [Online]. Available: https://reactome.org/content/schema/DatabaseObject

[2] [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://download.reactome.org/documentation/DataModelGlossary_V90.pdf