Explicit Composition Constructs in DSLs

The case of the epidemiological language KENDRICK

Thi Mai Anh Bui
IFI, MSI, VNU, Hanoi, Vietnam
IRD, UMI 209, UMMISCO, IRD France Nord, F-93143, Bondy
Sorbonne Universités UPMC, Univ. Paris 6, UMI 209, UMMISCO, F-75005, Paris, France
thi-mai-anh.bui@etu.upmc.fr

Nick Papoulias
IRD, UMI 209, UMMISCO, IRD France Nord, F-93143, Bondy
Sorbonne Universités UPMC, Univ. Paris 6, UMI 209, UMMISCO, F-75005, Paris, France
npapoylias@gmail.com

Mikal Ziane
Sorbonne Paris Cité, Université Paris Descartes, UMR 7606, LIP6, Paris, France
mikal.ziane@lip6.fr

Serge Stinckwich
IRD, UMI 209, UMMISCO, IRD France Nord, F-93143, Bondy
Sorbonne Universités UPMC, Univ. Paris 6, UMI 209, UMMISCO, F-75005, Paris, France
Université de Caen Normandie, Caen, France
serge.stinckwich@ird.fr

Abstract

Domain Specific Languages (DSL) are designed to be syntactically and semantically easier to use than general-purpose languages performing the same task. This is generally achieved by tailoring programming notions and constructs to the domain tasks at hand. Yet there are examples of domain specific problems that demand elaborate constructs (for e.g. aspects in Aspect DSLs) which need to be carefully introduced as to avoid complicating the language. We present such an example in Pharo for the epidemiological language KENDRICK. KENDRICK simplifies the programming of epidemiological simulations by decomposing highly-coupled monolithic models into modular concerns. This decomposition though, is based on a mathematical model that introduces non-trivial composition semantics into the language that need to be carefully integrated. In this work, we address this problem by extending KENDRICK's DSL with explicit composition semantics, entities and operators.

Keywords DSL, Epidemiology, Explicit Composition

CONF 'yy Month d–d, 20yy, City, ST, Country
Copyright © 2016 held by owner/author(s). Publication rights licensed to ACM.
ACM 978-1-nnnn-nnnn-n/yy/mm. . . $15.00
DOI: http://dx.doi.org/10.1145/nnnnnnn.nnnnnnn

1. Introduction

Retrospective analysis [Bro15] on modeling and simulation of environmental threats show us a number of key drawbacks of the modeling and simulation process. For example in the recent Ebola outbreak in West Africa [CP14] initial predictions [RLM+14] were refuted by a large margin and *a constant need for adaptation and evolution of tools, algorithms and models during the outbreak period emerged.*

This need is especially prominent in epidemiology since modelling and simulation (relying on the compartmental framework [AM91, KR08]) are very heavily used: from investigating the mechanisms of disease propagation [KR08, XBG04], to exploring evolutionary dynamics [GMNR01, RH09], informing control strategies [BSG09, LBG+11] etc.

From an engineering point of view this reality makes features such as *decomposition* and *re-usability* crucial for epidemiological languages and software. On the one hand the different concerns (epidemiological, medical, biological, demographical, sociological etc.) of an epidemic belong to different domains of expertise and thus need to be *decomposed* during modelling. On the other hand these same concerns should be *re-usable* if they are to support maintenance, extensibility and evolution of the simulations.

The epidemiological language KENDRICK, which is implemented in Pharo, tries to address these issues by decomposing highly-coupled monolithic epidemiological models into modular concerns. This decomposition though is based on a mathematical model [BZS+16] that introduces non-trivial composition semantics into the language that need to be carefully integrated.

In this work we address this problem by extending KENDRICK's DSL with explicit composition semantics, entities and operators. These additions go beyond normal conflict resolution to express the semantics of the underlying mathematical model. We validate our approach though a series of epidemiological examples that ensure that the new constructs reproduce correctly the domain semantics and do not introduce significant execution overhead for the language.

The main contributions of this paper are:

- An extended Language Model for KENDRICK based on explicit composition,
- A formal specification of our proposed syntax for the KENDRICK DSL in EBNF form,
- A validation of our extensions covering domain semantics and performance overhead.

The rest of this paper is organised as follows: Section 2 presents in more detail our problem statement and gives an overview of the mathematical model of KENDRICK. Then in Section 3 we detail our extensions to KENDRICK's DSL that address all challenges presented in the previous section. Section 4 validates our proposal though a series of epidemiological examples. Section 5 presents related work. Then finally in Section 6 we conclude our paper and present future perspectives.

2. Problem Statement

As we have shown in our previous work [BZS$^+$16] epidemiological modelling can benefit by a decomposition of concerns that are modeled after stochastic automata networks [PS00]. Indeed classical representation of epidemiological models with ODEs (Ordinary Differential Equations) are monolithic and the different concerns are intertwined. These equations mix in the same definition parameters and properties of different domains that are involved in simulations (epidemiological, medical, biological, demographical, sociological etc.). Once translated into executable code for simulation, the result is worse as low-level programming details are also intertwined with the different domain concerns, leading to spaghetti code.

On the contrary the mathematical model introduced by KENDRICK has the advantage of presenting the different concerns as independent automata that can be composed into a larger automaton. As we will see in section 2.1 though, this mathematical model introduces non-trivial composition semantics into the language. This results in a series of challenges (see section 2.2) that this work is trying to address. From the point of view of a domain-language implementor we are trying to answer (in the context of KENDRICK) the following question:

How can advanced composition semantics be incorporated in a DSL without unnecessarily complicating the language?

This is an instance of a larger problem in DSL design, where one needs to incorporate elaborate constructs because of their expressiveness (for e.g. aspects [KLM$^+$97] in ADSLs or entities resembling traits [SDNB03]), without overcomplicating the design.

2.1 Overview of the mathematical model

In KENDRICK an epidemiological model is defined as follows:

Definition 1. $Model = \{P, \mathcal{A}, \mathcal{R}, Prms, Tr\}$

P is a set of individuals: the *population*. Each individual of the population is characterised by several *attributes* such as: species, sex, age,... An attribute is a mapping from P to some domain D_a. $\mathcal{A}$ is the set of attributes of the model. $\mathcal{R}$ is an equivalence relation on P. $\mathcal{C} = P/\mathcal{R}$, is the set of equivalence classes, i.e. the *compartments*, of the population. $Prms$ is a set of parameters (temporal functions).

$Tr \in \mathcal{C} \times \mathcal{C} \rightarrow \mathbb{R}^+$ is the transition rate matrix. Each element in Tr (except the main diagonal elements) represents the rate at which individuals move from one compartment to another one. The diagonal elements are defined as $q_{ii} = -\sum_{j \neq i} q_{ij}$. Therefore, the sum of each row in Tr is 0. For example, suppose that the population is decomposed into two regions R_1 and R_2. The transition rate matrix Tr is:

$$Tr = \begin{pmatrix} -\nu_1 & \nu_1 \\ \nu_2 & -\nu_2 \end{pmatrix}$$

The transition $Tr(C_{R1}, C_{R2}) = \nu_1$ represents the rate at which an individual from region R_1 immigrates to the region R_2 (C_{R_1}, C_{R_2} denote the compartments corresponding to R_1, R_2) and so on.

Separation of Concerns From a model-driven viewpoint, a concern can be seen as a model transformation.

Definition 2. $C = \{\mathcal{A}_C, \mathcal{R}_C, Prms_C, \mathcal{F}_C\}$

A concern may refine the equivalence relation of the model it is applied to, by providing an additional one $\mathcal{R}_C$ which typically uses new attributes that are introduced in $\mathcal{A}_C$. A concern may also introduce a set of parameters $Prms_C$ and a function $\mathcal{F}_C$ to change the transition rate matrix of the model.

Concerns may or may not depend on each other:

Definition 3. A concern C_2 structurally depends on another concern C_1 if the definition of C_2 mentions one or several entities defined by C_1.

A concern is called independent if it has no structural dependency on any other one. The function $\mathcal{F}_C$ provided by each concern to modify the transition rate matrix of a model is generally represented through a set of pre-defined operators.

Combining Concerns A concern can be combined to an another one to generate a new concern. We define the combination of two concerns through a binary operator, noted $\bowtie$.

Definition 4. Given two concerns C_1 and C_2: $C_1 = \{\mathcal{A}_1, \mathcal{R}_1, Prms_1, \mathcal{F}_1\}$ and $C_2 = \{\mathcal{A}_2, \mathcal{R}_2, Prms_2, \mathcal{F}_2\}$

$$C_1 \bowtie C_2 \rightarrow C = \{\mathcal{A}_C, \mathcal{R}_C, Prms_C, \mathcal{F}_C\}$$

where: $\mathcal{R}_C = \mathcal{R}_1 \wedge \mathcal{R}_2$, $Prms_C = Prms_1 \cup Prms_2$, $\mathcal{F}_C = \mathcal{F}_2 \circ \mathcal{F}_1$ and $\mathcal{A}_C = \mathcal{A}_1 \uplus \mathcal{A}_2 = \{a, \mathcal{D}_a^C = \mathcal{D}_a^{C_1} \cup \mathcal{D}_a^{C_2} | a \in \mathcal{A}_1 \wedge a \in \mathcal{A}_2\} \cup \{a | a \in \mathcal{A}_1 \wedge a \notin \mathcal{A}_2\} \cup \{a | a \notin \mathcal{A}_1 \wedge a \in \mathcal{A}_2\}$ with $\mathcal{D}_a^C$ denotes the domain of attribute a in the concern C. $\mathcal{F}_2 \circ \mathcal{F}_1$ is the composition function, given a transition rate matrix Tr, $\mathcal{F}_C(Tr) = \mathcal{F}_2(\mathcal{F}_1(Tr))$.

When C_2 can be defined independently of C_1, $C_1 \bowtie C_2$ can be seen as a product of two automata provided that during a composite transition, only one automaton changes state. The function $\mathcal{F}_C$ typically generates the tensor sum (noted as $\oplus$) of two transition rate matrices.

In the case that C_2 depends structurally on C_1, when combining C_2 to C_1, the former typically transforms the latter by: (a) introducing new states for a given attribute of C_1 (b) removing/updating existing transitions of C_1 (c) adding new transitions. We then can represent $\mathcal{F}_C$ as a set of low-level graph-operations such as *add, edit, remove* on nodes or vertices.

Applying a Concern to a Model A concern can be applied to a model to create a new model. We define the application of a concern to a model through a binary operator, noted $\vdash$. First, we define a *void model* (noted Θ) as:

$$\Theta = \{P, \mathcal{A}_o, \mathcal{R}_o, Prms_o, Tr_o\} \qquad (1)$$

where: $\mathcal{A}_o = \emptyset$, $\mathcal{R}_o = true$, $Prms_o = \{N\}$ with N represents the size of the population P, $Tr_o = 0$.

Definition 5. Given an independent concern $C = \{\mathcal{A}_C, \mathcal{R}_C, Prms_C, \mathcal{F}_C = \oplus\}$ and a *void model* Θ. Applying C to Θ gives a new model M_C:

$$M_C = \Theta \vdash C = \{P_o, \mathcal{A}_M, \mathcal{R}_M, Prms_M, Tr_M\}$$

where: $\mathcal{A}_M = \mathcal{A}_C, \mathcal{R}_M = \mathcal{R}_C, Prms_M = Prms_C \cup \{N\}$, $Tr_M = Tr_C \oplus 0 = Tr_C$

Definition 6. Given a model M and a concern C:

$$M = \{P_M, \mathcal{A}_M, \mathcal{R}_M, Prms_M, Tr_M\}$$

$$C = \{\mathcal{A}_C, \mathcal{R}_C, Prms_C, \mathcal{F}_C\}$$

Applying the concern C to the model M gives:

$$M \vdash C = \Theta \vdash C_M \vdash C = \Theta \vdash (C_M \bowtie C) \qquad (2)$$

where C_M represents the corresponding concern of the model M: $\mathcal{A}_{C_M} = \mathcal{A}_M$, $\mathcal{R}_{C_M} = \mathcal{R}_M$, $Prms_{C_M} = Prms_M, Tr_{C_M} = Tr_M$.

2.2 Main Challenges from a language perspective

Our first direct translation of this mathematical model into Smalltalk presented us with a number of challenges, that our current work is addressing. These challenges are mainly related to the following fact:

Unlike simple inheritance semantics or composition semantics in OO programming languages (for e.g. through traits or aspects[1]*) that usually produce the union of named entities of their parts, the mathematical model of* KENDRICK *produces a set of named entities akin to a cartesian product.*

For example, combining an epidemiological concern with 3 states {*S,I, and R*} with a demographical concern with two states {*Paris, Lyon*} does not produce their union (i.e. {*S, I, R, Paris, Lyon*}) but (in this case) their exact cartesian product {*S_Paris, S_Lyon, I_Paris, I_Lyon, R_Paris, R_Lyon*}.

Consider here that these composite names (like S_Paris, I_Lyon etc.) have not been defined in any of the concerns that are being combined (either in the epidemiological or the demographical concern). They sprang to life through the composition and now depend (i.e. reference) on both the epidemiological and the demographical concerns.

This fact where variables (like the names of the states above) are created at composition time without being able to be previously defined, and carry with them dependencies from multiple entities presents us with a series of challenges. These challenges need to be resolved in order to map the mathematical model correctly in our DSL.

Challenge 1: **Initialisation** Where and when are these composite names initialised? This is a language challenge cause initialising these composite names in the wrong time or place (i.e. in the wrong language entity) might introduce unwanted dependencies that the mathematical model that we are modelling is trying to avoid.

Challenge 2: **Usage** What happens with expressions that use these composite names? These names are used in methods that describe "functional rates" [BZS+16] in the mathematical model and essentially override previously declared parameters. The mathematical model considers functional rates as part of an instantiation phase and we need to model this phase in a re-usable manner taking composite names and their dependencies into account.

Finally, apart from composition of concerns the mathematical model defines extensions of concerns. Here we observed the following challenge:

Challenge 3: **Extension** Dependencies between concerns when extending need to be described through low-level graph operations (add, edit, remove) on nodes or vertices.

[1] we consider here the case where new methods or variables are added through aspects

These operations are out-of sync with the high-level conceptual approach of the rest of the model.

3. Our Solution

3.1 The Language Model

Overview In order to address the issues we described in Section 2 we first introduced syntactically re-usable entities by adopting a more declarative syntax. Consequently we introduced an explicit composition entity (as opposed to implicit composition semantics). This explicit composition construct factors unwanted dependencies (including composite names) out of the rest of the model addressing *Challenges 1 and 2*. Finally we extended the mathematical model for concerns with high-level operators, addressing *Challenge 3* and integrated them into the language.

Abstract Syntax Model Our revised language model for KENDRICK (presented in Figure 1) is organised around a variation of the composite pattern, whose intend is to *"... compose objects into tree-structures to represent part-whole hierarchies"* [ABW98]. The reason for this architectural choice is seen in Figure 2 where we depict the composition semantics of KENDRICK as a tree of interdependencies. Notice here that each sub-tree is re-usable since the only dependencies come from parent nodes to their children. This means that each model or concern (with its children), represented here as syntactical entities, can be re-used in multiple modelling projects, as described by the mathematical model in Section 2.1.

Language Entities and Explicit Composition The following language entities are depicted in Figure 1. The *Model* entity (modelling Definition 1, of Section 2.1) which can be as simple as a void model (defining only the population set) or define its own parameters and attributes (inherited by KendrickModellingComponent). The *Concern* entity (modelling Definitions 2,3,4 of Section 2.1) which can have *structuralDependencies* to other Concerns and defines a more high-level api for doing so (see also subsection 3.2). The *Visualisation and Simulation* entities, which as their name suggests control the algorithms and parameters that the epidemiological simulation and resulting visualisations will use. Then finally we have the *explicit Composition* entity (modelling Definitions 5 and 6) which is in charge of composing concerns with models, while allowing end-users to explicitly add, override, initialise or otherwise change the composition details.

Going back to our Challenges in Section 2.2, we are talking here about *explicit* rather than *implicit* composition since: *(a)* the composition phase is *reified* as an entity inside the DSL itself and *(b)* the user partially controls the details of this composition.

This allowed us to resolve Challenges 1 and 2 by enforcing the use of troublesome composite variables and dependencies (we saw in Section 2.2) only within the reified composition entity.

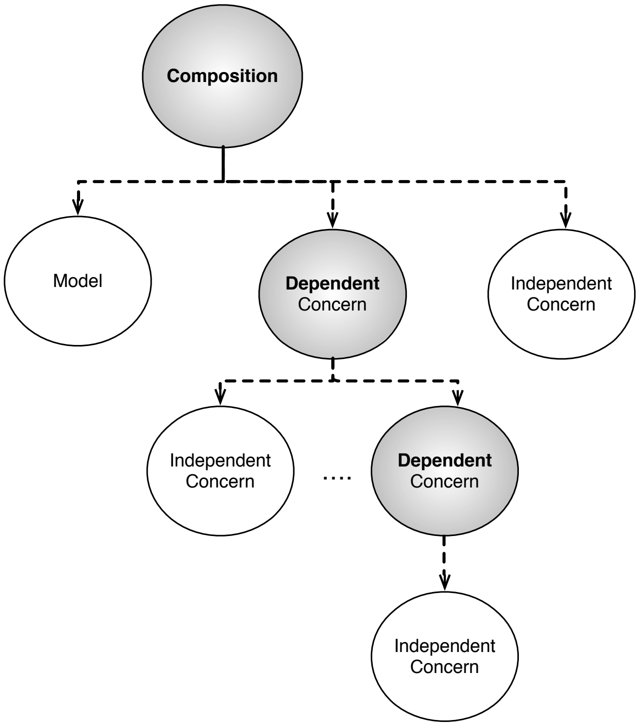

Figure 2. A composition tree in KENDRICK: each sub-tree is re-usable (arrows depict dependencies)

For example in the following code-snippet our Composition entity (line 1) after describing its composition parts (lines 2 to 6) and initialising the populationSize (line 3), handles *(a)* the overriding of parameters that depend on separate concerns (lines 7 to 14) *(b)* functional rates that reference composite names (variables) that did not exist prior to the composition (lines 15 to 16) and finally *(c)* the initialisation of these newly created composite names themselves (lines 17 to 24).

In a nutshell the reification of the composition entity and its clauses factors unwanted dependencies (that are created during composition) out of the rest of the model, while allowing the user to control the composition details (such as initialisation and overriding):

```
Composition MultiSpeciesSpatial                1
  model: 'Infuenza';                           2
  populationSize: 27500;                       3
  concern: 'Demographical';                    4
  concern: 'Biological';                       5
  concern: 'SEIR';                             6
  mu_species: #(0.0000365 0.00137);            7
  sigma_species: #(0.5 0.67);                  8
  gamma_species: #(0.25 0.233);                9
  rho_species: #(0.03 0.1);                    10
  beta_species: #(                             11
```

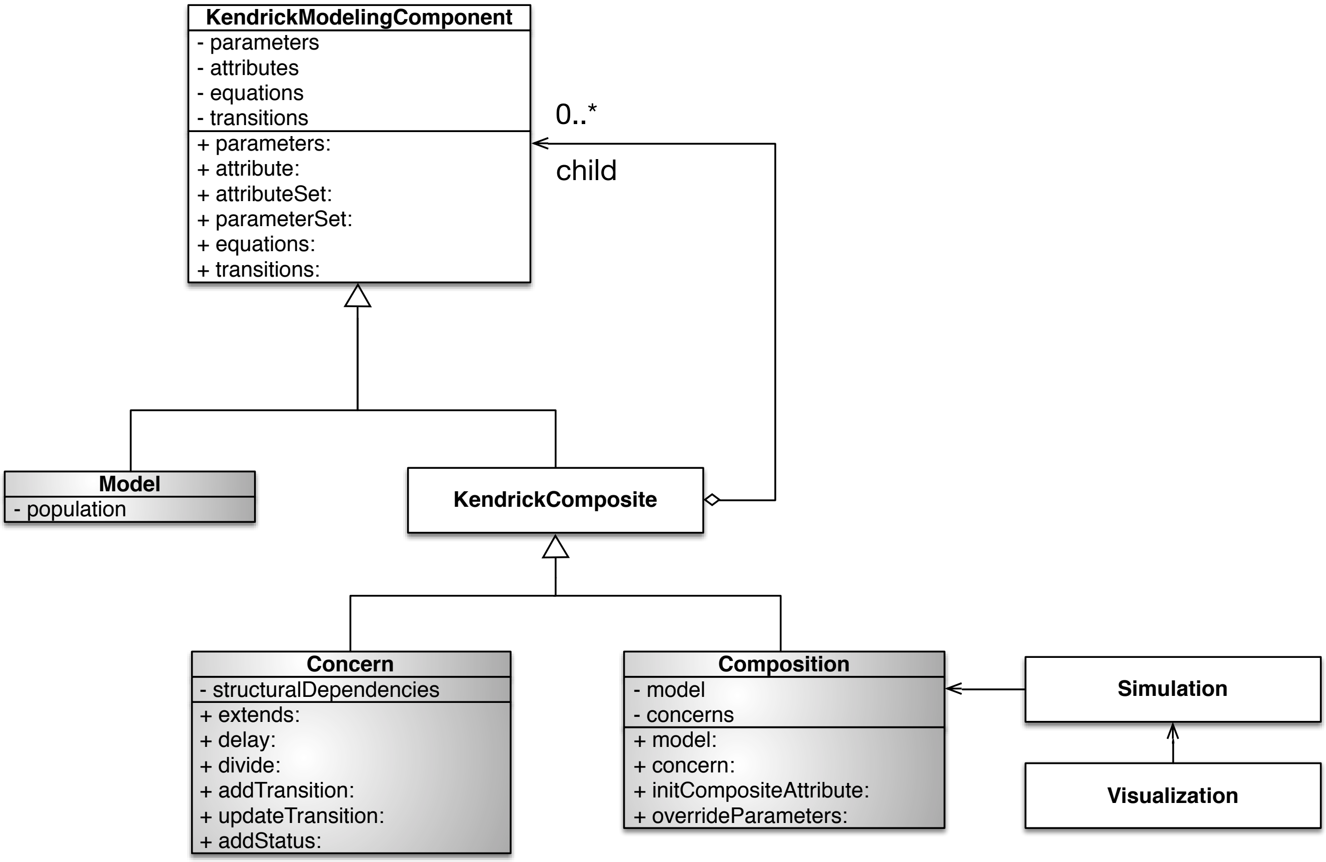

Figure 1. KENDRICK language entities organised around a Composite Pattern

```
    #(0 0.21)
    #(0 0.42)
  );
  lambda: #(beta*I_patch/N sum);
  N: #(patch_species);
  S_species_patch: #(
    #(500 500 500 500 500)
    #(4990 5000 5000 5000 5000)
  );
  I_species_patch: #(
    #(0 0 0 0 0)
    #(10 0 0 0 0)
  ).
```

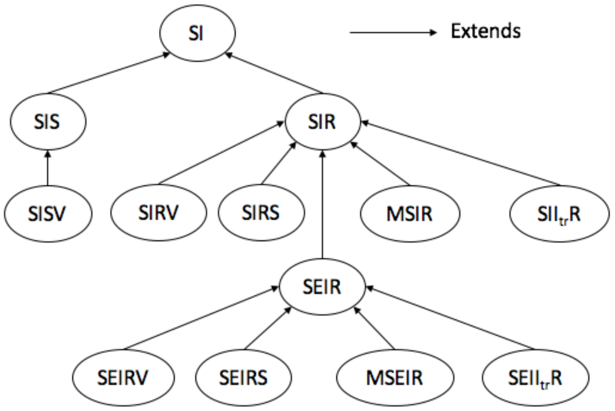

Figure 3. Epidemiological concerns that extends the SI by introducing new status or modifying transitions.

3.2 Higher-level extension API

Finally in order to address *Challenge 3* we extended the mathematical model for concerns with high-level operators and integrated them into the Concern Entity (as seen in the lower part of Figure 1).

As we have mentioned in the section 2.1, when a concern depends structurally on another one, it may introduce new state, update an existing transition or add a new one etc. In epidemiology, every model is expected to have the SI^2 concern. From this initial configuration, an infinity of other status can be added in order to represent different transmission cycles corresponding to each infectious disease, i.e *SIR* - to

[2] Epidemiological models consider individuals who are Susceptible to the pathogen and then can be infected, Infectious to transmit the disease.

represent the immunity state, *SIS* - no immunity, *SIRS* - loss of immunity, *SEIR* - to represent a latent period (infected but not yet infectious), *MSIR* - passive immunity from mothers etc. as can be seen on Figure 3. We then introduce two basic operators to perform these transformations: *add-status* and *add-transition* as follows.

Definition 7. Given a concern $C = \{\mathcal{A}_C, \mathcal{R}_C, Prms_C, \mathcal{F}_C\}$. The operator *add-status* is defined as:

add-status(C, X)

The operator *add-status* first updates the domain of the attribute *status* in $\mathcal{A}_C$ and then modifies the transition rate matrix Tr_C by adding one row of 0 and one column of 0.

Definition 8. Given a concern $C = \{\mathcal{A}_C, \mathcal{R}_C, Prms_C, \mathcal{F}_C\}$. The operator *add-transition* is defined as:

$$\textit{add-transition}(C, source, target, rate)$$

This operator modifies the corresponding element (given by *source, target*) of the transition rate matrix Tr_C by $rate$ and updates the main diagonal element to make sure: $q_{ii} = -\sum q_{ij}$.

Example: The SIR concern extends the SI by adding a new status R and new transition $I \xrightarrow{\gamma} R$

```
Concern SIR
    extends: 'SI';
    parameters: #(gamma);
    addStatus: #(R);
    addTransition: #(I -- gamma --> R.).
```

From these basic operators, we can formulate some other ones which produce some particular transformations and are frequently introduced in epidemiological models. For example, the introduction of an intermediate state to postpone a transmission cycle can be captured by the *delay* operator.

Definition 9. Given a concern $C = \{\mathcal{A}_C, \mathcal{R}_C, Prms_C, \mathcal{F}_C\}$. The operator *delay* is defined as:

delay(C, delay-rate, transition, intermediate-state)

This operator modifies the concern C by introducing a new status (given by *intermediate-status*) then create two new transitions from the old one. Suppose that the transition to be transformed is: $Source \xrightarrow{rate} Target$. The $delay$ operator can be seen as a set of the following operators:

1. *add-status(C, intermediate-state)*
2. *add-transition(C, source, intermedidate-state, rate)*
3. *add-transition(C, intermedidate-state, target, delay-rate)*
4. *add-transition(C, source, target, 0)*

Examples: The concern $SEIR$ introduces the latent period (in which individuals are infected but not yet infectious) to the SIR concern.

```
Concern SEIR
    extends: 'SIR';
    parameters: #(sigma);
    delay: #(sigma , S -- lambda --> I , E);
```

In future work, we will allow users to specify their own operators (like $delay$) provided that *(a)* removing an existing status of a given concern is not allowed; *(b)* the concern after being transformed by such operators remains always a Markov chain.

3.3 Implementation

Our extended language model for KENDRICK[3,4] presented in this section is implemented in Pharo as a mixed-DSL. This means that the DSL itself is mainly internal but we make extensive use of symbolic expressions (for equations, transitions, functional rates etc.) that do have a separate parsing phase.

We found this implementation strategy mandatory in order to avoid using blocks in our language which seemed confusing to non-Smalltalk (or even non-programming) users. Furthermore we made extensive use of proxies in order to naturally capture unquoted variables (through message-sending). This was achieved through classic DNU capturing, where the selector plays the role of the unquoted variable name. Finally we kept all message sends either unary or single-keyword to enhance uniformity.

4. Validation

In order to validate our extensions we made sure that the extended version can – as before – reproduce a series of known epidemiological examples from related bibliography. We used one model for Measles and two for the Infuenza virus (e.g. output shown in Figure 4) ensuring that the simulated timeseries (and key metrics such as peak infected population) are reproduced.

One of these models (Infuenza with two species) showcasing our extended-DSL for KENDRICK is shown below:

Our model is defined in line 1 (as a void model) having all concerns factored-out for re-usability. These include: the Biological Concern (lines 3-4) the Demographical Concern (lines 6-15) and the Epidemiological Concern (lines 18-34). Composition takes place in lines 36-59 which as we saw earlier handles *(a)* the overriding of parameters that depend on separate concerns *(b)* functional rates that reference composite names (variables) that did not exist prior to the composition and finally *(c)* the initialisation of these newly created composite names themselves. Finally in lines 61 through 72 we see the Visualisation and Simulation entities controlling the algorithms (the Runge-Kutta algorithm [GH10] in this case) and parameters that the epidemiological simulation and resulting visualisations will use.

```
KendrickModel Infuenza.                    1
                                           2
```

[3] http://ummisco.github.io/kendrick/

[4] http://ss3.gemtalksystems.com/ss/KendrickExtentions.html

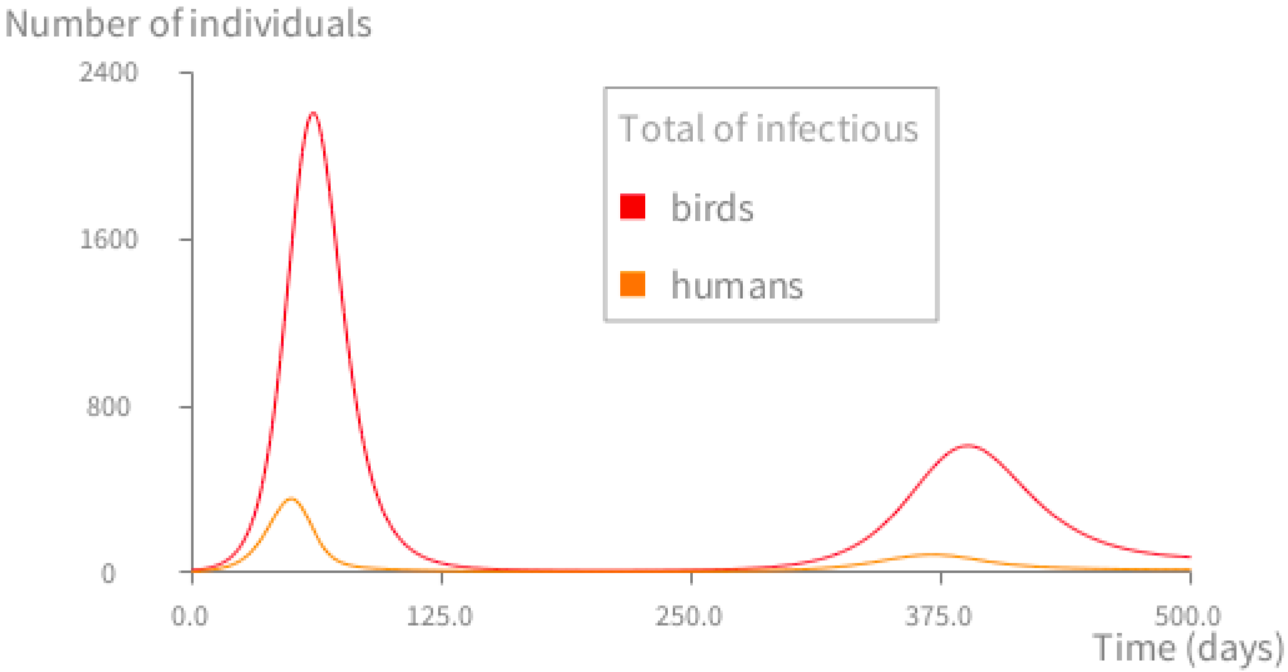

Figure 4. Simulation results showing infection spread over time for Infuenza with two species. The model uses explicit composition of Demographical, Biological and Epidemiological concerns with structural dependencies

Simulation			
Example	Before	After	Slowdown
Measles	$\mu = 39.45$ ms ($\tau = 0.63$)	$\mu = 46.8$ ms ($\tau = 4.40$)	1.18x
Influenza 1SP	$\mu = 1383.05$ ms ($\tau = 121.32$)	$\mu = 1639.59$ ms ($\tau = 87.12$)	1.18x
Influenza 2SP	$\mu = 6308.95$ ms ($\tau = 183.84$)	$\mu = 6327.33$ ms ($\tau = 212.12$)	≈ 1.00x

Table 1. Execution times of simulation examples before and after our extensions

```
Concern Biological
  attribute: #(species -> human bird).

Concern Demographical
  attribute: #(patch -> 1 2 3 4 5);
  parameters: #(rho);
  rho: 0.03;
  transitions: #(
    1 -- rho --> 2.
    2 -- rho --> 3.
    3 -- rho --> 4.
    4 -- rho --> 5.
    5 -- rho --> 1.
  ).

Concern SIR
  attribute: #(status -> S I R);
  parameters: #(beta lambda gamma mu v);
  gamma: 0.25;
  v: 0.00274;
  transitions: #(
    S -- lambda --> I.
    I -- gamma --> R.
    R -- v --> S.
  ).

Concern SEIR
  extends: 'SIR';
  parameters: #(sigma);
  addTransition: #(Empty -- mu --> S.);
  addTransition: #(status -- mu --> Empty.);
  delay: #(sigma , S -- lambda --> I , E).

Composition MultiSpeciesSpatial
  model: 'Infuenza';
  populationSize: 27500;
  concern: 'Demographical';
  concern: 'Biological';
```

```
    concern: 'SEIR';
    mu_species: #(0.0000365 0.00137);
    sigma_species: #(0.5 0.67);
    gamma_species: #(0.25 0.233);
    rho_species: #(0.03 0.1);
    beta_species: #(
        #(0 0.21)
        #(0 0.42)
    );
    lambda: #(beta*I_patch/N sum);
    N: #(patch_species);
    S_species_patch: #(
        #(500 500 500 500 500)
        #(4990 5000 5000 5000 5000)
    );
    I_species_patch: #(
        #(0 0 0 0 0)
        #(10 0 0 0 0)
    ).

Simulation SpatialRungeKutta rungeKutta
    for: 'MultiSpeciesSpatial';
    from: 0;
    to: 500;
    step: 1.

Visualization SpatialDiagramViz diagram
    for: 'SpatialRungeKutta';
    data: #(I_species);
    legendTitle: 'Total of Infectious';
    legends: #('humans' 'birds');
    xLabel: 'Time (days)'.
```

Consequently, as seen in Table 1, we profiled our examples both before and after the introduction of our extensions in order to ensure that we did not introduce any significant overhead to the language. We have run these simulations on Pharo 5.0 (OS X 10.11.1) in a 2,9 GHz Intel Core i7 (using 8 GB of RAM, 1600 MHz DDR3). Each one of the scenarios was run 100 times gathering average time (μ in ms) and the standard deviation (τ). Indeed we found that no significant slowdown is introduced. Results range from a maximum 1.18x slowdown for small (in terms of execution time) simulations (like Measles and Infuenza 1SP) to an almost even 1.002x execution time for longer simulations like Infuenza 2SP.

5. Related Work

Epidemiological modellers have used a variety of tools for constructing models: general programming languages, mathematical modelling languages (Matlab, R, etc.), libraries targeted to epidemiology such as Epipy[5] - a Python tools for epidemiology, GillespieSSA[6] - an R package for generating stochastic simulation using Gillespie's algorithms [Gil77] or dedicated modelling software as GLEAM viz [VdBGG+11], STEM [FFT+13], FRED [GBR+13] etc.

Such tools use different approaches to model the transmission of infectious diseases. However, either they are lower-level programming languages (so that do not focus on the domain of epidemiology) or they are usually closed platforms. They currently lack the ability to provide a level of abstraction to efficiently describe epidemiological models including a variability of domain concerns such as age-structure, social/sexual mixing, multi-species/strains, spatial heterogeneity, etc.

6. Conclusion & Future Work

We have extended Kendrick's DSL with explicit composition semantics, entities and operators in order to properly map its underlying mathematical model. We validated our approach though a series of epidemiological examples that ensure that (a) our extensions reproduce correctly the domain semantics and (b) do not introduce significant execution overhead for the language. For our future work we aim to further extend our DSL to allow users to specify their own operators for Concerns. Furthermore we would like to provide a more quantitive validation for Kendrick's usage patterns in terms of evolution, decomposition and reusability.

References

[ABW98] Sherman R. Alpert, Kyle Brown, and Bobby Woolf. *The Design Patterns Smalltalk Companion*. Addison Wesley, Boston, MA, USA, 1998.

[AM91] R. M. Anderson and R. M. May. *Infectious diseases of humans: Dynamics and control*. Oxford Science Publications, Oxford, 1991.

[Bro15] David Brown. How computer modelers took on the ebola outbreak. *IEEE Spectrum*, 2015.

[BSG09] C. T. Bauch, E Szusz, and L. P. Garrison. Scheduling of measles vaccination in low-income countries: Projections of a dynamic model. *Vaccine*, 27(31):4090–4098, 2009.

[BZS+16] Thi Mai Anh Bui, Mikal Ziane, Serge Stinckwich, Tuong Vinh Ho, Benjamin Roche, and Nick Papoulias. Separation of concerns in epidemiological modelling. In *Companion Proceedings of the 15th International Conference on Modularity*, MODULARITY Companion 2016, pages 196–200, New York, NY, USA, 2016. ACM. URL: `http://doi.acm.org/10.1145/2892664.2892699`, `doi:10.1145/2892664.2892699`.

[CP14] Centers for Disease Control CDC and Prevention. 2014 ebola outbreak in west africa. `http://`

[5] http://cmrivers.github.io/epipy

[6] http://cran.r-project.org/web/packages/GillespieSSA/index.html

www.cdc.gov/vhf/ebola/outbreaks/2014-west-africa/index.html, 2014.

[FFT$^+$13] Alexander Falenski, Matthias Filter, Christian Thöns, Armin A. Weiser, Jan-Frederik Wigger, Matthew Davis, Judith V. Douglas, Stefan Edlund, Kun Hu, James H. Kaufman, et al. A generic open-source software framework supporting scenario simulations in bioterrorist crises. *Biosecurity and bioterrorism: biodefense strategy, practice, and science*, 11(S1):S134–S145, 2013.

[GBR$^+$13] John J. Grefenstette, Shawn T. Brown, Roni Rosenfeld, Jay DePasse, Nathan T.B. Stone, Phillip C. Cooley, William D. Wheaton, Alona Fyshe, David D. Galloway, Anuroop Sriram, et al. Fred (a framework for reconstructing epidemic dynamics): an open-source software system for modeling infectious diseases and control strategies using census-based populations. *BMC public health*, 13(1):940, 2013.

[GH10] David F. Griffiths and Desmond J. Higham. *Numerical methods for ordinary differential equations*. Springer, Springer Undergraduate Mathematics Series, 2010.

[Gil77] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81:2340–2361, 1977.

[GMNR01] S. Gandon, M. J. Mackinnon, S. Nee, and a F Read. Imperfect vaccines and the evolution of pathogen virulence. *Nature*, 414(6865):751–756, 2001.

[KLM$^+$97] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Lopes, Jean-Marc Loingtier, and John Irwin. Aspect-oriented programming. In *ECOOP'97Object-oriented programming*, pages 220–242. Springer, 1997.

[KR08] Matthew J Keeling and P Rohani. *Modeling Infectious Diseases*. Princeton University Press, Princeton, 2008.

[LBG$^+$11] A. Levin, C. Burgess, L. P. Garrison, C. Bauch, J. Babigumira, E. Simons, and A. Dabbagh. Global eradication of measles: an epidemiologic and economic evaluation. *The Journal of infectious diseases*, 204 Suppl (Suppl 1):98–106, 2011.

[PS00] Brigitte Plateau and William J Stewart. Stochastic automata networks. In *Computational Probability*, pages 113–151. Springer, 2000.

[RH09] A. F. Read and S. Huijben. Perspective: Evolutionary biology and the avoidance of antimicrobial resistance. *Evolutionary Applications*, 2(1):40–51, 2009.

[RLM$^+$14] C. M. Rivers, E. T. Lofgren, M. Marathe, S. Eubank, and B. L. Lewis. Modeling the Impact of Interventions on an Epidemic of Ebola in Sierra Leone and Liberia. *ArXiv e-prints*, September 2014. arXiv:1409.4607.

[SDNB03] Nathanael Schärli, Stéphane Ducasse, Oscar Nierstrasz, and Andrew P Black. Traits: Composable units of behaviour. In *ECOOP 2003–Object-Oriented Programming*, pages 248–274. Springer, 2003.

[VdBGG$^+$11] W. Van den Broeck, C. Gioannini, B. Gonçalves, M. Quaggiotto, V. Colizza, and A. Vespignani. The gleamviz computational tool, a publicly available software to explore realistic epidemic spreading scenarios at the global scale. *BMC Infectious Diseases*, 11:37, 2011.

[XBG04] Y. Xia, O. N. Bjornstad, and B. T. Grenfell. Measles metapopulation dynamics: a gravity model for epidemiological coupling and dynamics. *Am Nat*, 164(2):267–281, 2004.

A. The KENDRICK Grammar in EBNF

```
⟨KENDRICK-SCRIPT⟩ ::= ⟨KENDRICK-ENTITIES⟩+

⟨KENDRICK-ENTITIES⟩ ::= ⟨MODEL-DECLARATION⟩
  |  ⟨CONCERN-DECLARATION⟩
  |  ⟨COMPOSITION-DECLARATION⟩
  |  ⟨SIMULATION-DECLARATION⟩
  |  ⟨VISUALIZATION-DECLARATION⟩

⟨IDENTIFIER⟩ ::= ⟨LETTER⟩ (⟨LETTER⟩ | [0-9]+)*

⟨COMPOSITE-IDENTIFIER⟩ ::= ⟨IDENTIFIER⟩ (_ ⟨IDENTIFIER⟩)+

⟨LETTER⟩ ::= [a-zA-Z]

⟨NUMBER⟩ ::= [0-9]+ (.[0-9]+)?

⟨STRING⟩ ::= ' [ˆ ']* '

⟨KEYWORD⟩ ::= ⟨IDENTIFIER⟩ :

⟨VALUE⟩ ::= ⟨NUMBER⟩ | ⟨STRING⟩ | ⟨ARRAY⟩

⟨SHORT-EQ-WITH-OP⟩ ::= ⟨EQUATION-EXPRESSION⟩ ⟨BASIC-OP⟩*

⟨BASIC-OP⟩ ::= sum|sqrt|size|min|max|avg|median|mean

⟨ARRAY⟩ ::= #( (⟨VALUE⟩* | ⟨SHORT-EQ-WITH-OP⟩) )

⟨COMMON-EXPRESSION⟩ ::= ⟨ATTRIBUTE-DEFINITION⟩
  |  ⟨PARAMETERS-DEFINITION⟩
  |  ⟨ASSIGNMENT-CLAUSE⟩
  |  ⟨EQUATIONS-DEFINITION⟩
  |  ⟨TRANSITIONS-DEFINITION⟩

⟨ATTRIBUTE-DEFINITIONS⟩ ::= attribute: ⟨ATTRIBUTE-ARRAY⟩

⟨ATTRIBUTE-ARRAY⟩ ::= #( ⟨IDENTIFIER⟩ -> (⟨IDENTIFIER⟩ | ⟨NUMBER⟩)+ )

⟨ASSIGNMENT-CLAUSE⟩ ::= ⟨KEYWORD⟩ ⟨VALUE⟩

⟨PARAMETER-DEFINITIONS⟩ ::= parameters: ⟨IDENTIFIER-ARRAY⟩

⟨IDENTIFIER-ARRAY⟩ ::= #( ⟨IDENTIFIER⟩+ )

⟨EQUATIONS-CLAUSE⟩ ::= equations: ⟨EQUATIONS-ARRAY⟩

⟨EQUATIONS-ARRAY⟩ ::= #( (⟨EQUATION⟩ .)+ )

⟨EQUATION⟩ ::= ⟨EQUATION-SIGNATURE⟩ = ⟨EQUATION-EXPRESSION⟩

⟨EQUATION-SIGNATURE⟩ ::= ⟨IDENTIFIER⟩ : ⟨IDENTIFIER⟩

⟨EQUATION-EXPRESSION⟩ ::= ⟨TERM⟩ ((+|-) ⟨EQUATION-EXPRESSION⟩)*

⟨TERM⟩ ::= ⟨NUMBER⟩|⟨IDENTIFIER⟩|⟨COMPOSITE-IDENTIFIER⟩|(EQUATION-EXPRESSION)|⟨PRODUCT⟩

⟨PRODUCT⟩ ::= ⟨TERM⟩ * ⟨TERM⟩

⟨TRANSITIONS-DEFINITION⟩ ::= transitions: ⟨TRANSITIONS-ARRAY⟩

⟨TRANSITIONS-ARRAY⟩ ::= #( ⟨TRANSITION⟩+ )

⟨TRANSITION⟩ ::= ⟨IDENTIFIER⟩ -- ⟨IDENTIFIER⟩ --> ⟨IDENTIFIER⟩
```

```
⟨MODEL-DECLARATION⟩ ::= KendrickModel ⟨IDENTIFIER⟩ ⟨MODEL-BODY⟩

⟨MODEL-BODY⟩ ::= (⟨MODEL-EXPRESSION⟩ ;) * ⟨MODEL-EXPRESSION⟩ .

⟨MODEL-EXPRESSION⟩ ::= ⟨POPULATION-CLAUSE⟩ | ⟨COMMON-EXPRESSION⟩

⟨POPULATION-CLAUSE⟩ ::= populationSize: ⟨NUMBER⟩

⟨CONCERN-DECLARATION⟩ ::= Concern ⟨IDENTIFIER⟩ ⟨CONCERN-BODY⟩

⟨CONCERN-BODY⟩ ::= (⟨CONCERN-EXPRESSION⟩ ;) * ⟨CONCERN-EXPRESSION⟩ .

⟨CONCERN-EXPRESSION⟩ ::= ⟨EXTENTION-CLAUSE⟩ | ⟨OPERATOR-CLAUSE⟩ | ⟨COMMON-EXPRESSION⟩

⟨EXTENTION-CLAUSE⟩ ::= extends: ⟨IDENTIFIER⟩

⟨OPERATOR-CLAUSE⟩ ::= ⟨DELAY-CLAUSE⟩
  |  ⟨DIVIDE-CLAUSE⟩
  |  ⟨ADD-CLAUSE⟩
  |  ⟨TRANSITION-CLAUSE⟩

⟨DELAY-CLAUSE⟩ ::= delay: #( ⟨IDENTIFIER⟩ , ⟨TRANSITION⟩ , ⟨IDENTIFIER⟩ )

⟨DIVIDE-CLAUSE⟩ ::= divide: #( ⟨IDENTIFIER⟩, , ⟨IDENTIFIER⟩ , ⟨IDENTIFIER⟩ )

⟨ADD-CLAUSE⟩ ::= addStatus: #( ⟨IDENTIFIER⟩ + ⟨IDENTIFIER⟩ )

⟨LINK-CLAUSE⟩ ::= addTransition: #( ⟨TRANSITION⟩ )

⟨COMPOSITION-DECLARATION⟩ ::= Composition ⟨IDENTIFIER⟩ ⟨COMPOSITION-BODY⟩

⟨COMPOSITION-BODY⟩ ::= (⟨COMPOSITION-EXPRESSION⟩ ;) * ⟨COMPOSITION-EXPRESSION⟩ .

⟨COMPOSITION-EXPRESSION⟩ ::= ⟨MODEL-CLAUSE⟩
  |  ⟨POPULATION-CLAUSE⟩
  |  ⟨CONCERN-CLAUSE⟩
  |  ⟨COMPOSITE-ASSIGNMENT⟩
  |  ⟨COMMON-EXPRESSION⟩

⟨MODEL-CLAUSE⟩ ::= model: ⟨STRING⟩

⟨CONCERN-CLAUSE⟩ ::= concern: ⟨STRING⟩

⟨COMPOSITE-ASSIGNMENT-CLAUSE⟩ ::= ⟨COMPOSITE-IDENTIFIER⟩ : ⟨VALUE⟩

⟨SIMULATION-DECLARATION⟩ ::= Simulation ⟨IDENTIFIER⟩ ⟨SIMULATION-MODIFIER⟩ ⟨SIMULATION-BODY⟩

⟨SIMULATION-MODIFIER⟩ ::= RungeKutta | AB2 | AM3 | BDF4 | Euler | Heun | ImplicitMidPoint | AB4 | BeckwardEuler | BDF3 |
    Midpoint | Trapezoid | AB3 | BDF2 | AM4 | Gillespie | GPUGillespie | TauLeap | IBM

⟨SIMULATION-BODY⟩ ::= (⟨SIMULATION-EXPRESSION⟩ ;) * ⟨SIMULATION-EXPRESSION⟩ .

⟨SIMULATION-EXPRESSION⟩ ::= ((from: | to: | step:) ⟨NUMBER⟩) | for: ⟨IDENTIFIER⟩

⟨VISUALIZATION-DECLARATION⟩ ::= Visualization ⟨IDENTIFIER⟩ ⟨VISUALIAZATION-MODIFIER⟩ ⟨VISUALIZATION-BODY⟩

⟨VISUALIZATION-MODIFIER⟩ ::= diagram | pieChart | barPlot | map

⟨VISUALIZATION-BODY⟩ ::= (⟨VISUALIZATION-EXPRESSION⟩ ;) * ⟨VISUALIZATION-EXPRESSION⟩ .

⟨VISUALIZATION-EXPRESSION⟩ ::= for: ⟨IDENTIFIER⟩
  |  (xlabel: | ylabel: | legendTitle:) ⟨STRING⟩
  |  data: ⟨SHORT-EQ-WITH-OP⟩
  |  legends: ⟨ARRAY⟩
```