Incorporating Qualitative Equations in Process-Based Models

Darko Čerepnalkoski[1], Ljupčo Todorovski[2], Nataša Atanasova[3], Sašo Džeroski[1]

[1] Department of Knowledge Technologies, Jozef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
[2] Faculty of Administration, University of Ljubljana, Gosarjeva ul. 5, 1000 Ljubljana, Slovenia
[3] Faculty of Civil and Geodetic Engineering, University of Ljubljana, Hajdrihova 28, SI-1000 Ljubljana, Slovenia
darko.cerepnalkoski@ijs.si, ljupco.todorovski@fu.uni-lj.si, natanaso@fgg.uni-lj.si, saso.dzeroski@ijs.si

Abstract

This paper explores the possibility of extending Process-based models with Qualitative differential equations. Process-based modeling is a modeling technique that uses two-level approach for modeling dynamical systems. It models systems on a purely qualitative level in terms of entities and processes that involve those entities on one hand, and a quantitative level on which all entities and processes are given a quantitative formulation which is that automatically translated into a set of ordinary differential equations. This paper aims to illustrate that this formalism can be extended with an intermediate level of modeling which consists of qualitative equations.

Introduction

Modeling is an essential part of scientific endeavor, even though different scientific disciplines have different ideas of how modeling is performed. Many different modeling techniques are used across the scientific world, producing as a result many different types of models expressed in a variety of modeling formalisms. Models can be analyzed and classified according to different properties. One important property of models is the level of abstraction. According to this property, models can range from purely qualitative, which focus on the relations between the concepts in the system being modeled, to purely quantitative ones, specified in some precise mathematical notation, most typically as equations. Between these two extremes there is a whole spectrum of modeling approaches with different level of abstraction.

When it comes to modeling dynamical systems, one approach is to use process-based models. Process-based models (Bridewell et al., 2008) use a two-level representation. At the qualitative level, a process-based model consists of entities, which correspond to the main actors of the system and processes, which correspond to relations between entities. At the quantitative level, each entity is described in terms of variables and constants that represent its properties, and each process is represented as a set of equations, algebraic or differential, that quantify the relations between the entities. The equations from all the processes in the model can be compiled to obtain a system of differential equations which is the ultimate quantitative representation of the system.

The key feature of process-based models is that they allow modeling at different levels of abstraction. At the qualitative level, they represent an abstract view of the system being modeled, showing only the key components of the system and the relations between them. At the quantitative level, a detailed view of the system is presented, which is equivalent to a system of ordinary differential equations allowing for further quantitative analysis of the system. This two-level paradigm can be augmented with an additional middle level that will provide an additional level of reasoning about the model.

In this paper, we propose an extension of the process-based formalism with an intermediate level of abstraction. This level of abstraction is modeled using concepts from qualitative reasoning. The principal change is the introduction of qualitative differential equations (QDEs) to the formalism. Every process from the purely qualitative level is described is terms of QDEs. Each QDE, in turn, has a particular quantitative form, when translated to the quantitative level.

The rest of the paper is structured as follows. In Section 2, we present the Process-Based Formalism used to specify process-based models, using an example from aquatic ecosystems. In Section 3, we present a way to extend the formalism using qualitative equations as a middle level between the purely qualitative description and the differential equations. Section 4 concludes the paper and outlines further work.

Process-Based Modeling

For representing process-based models we use a process-based formalism. The formalism is designed for the description of dynamical systems, i.e., systems that change over time. Dynamical systems have a state which is a description of the system at a given point in time and

processes that represent phenomena that occur in the system and cause the state to change over time.

Modeling Components: Entities and Processes

The state of the system is given as a set of entities. Each entity corresponds to one logical object (material or abstract) that appears in the system. If we take, for example, an aquatic environment, such as a lake as a system, then entities would correspond to different nutrients such as phosphorus and nitrogen, different phytoplankton and/or zooplankton species or perhaps fish or other animals. Each entity in turn is described with one or more properties that can be fixed (constants) or can change with time (variables). Phytoplankton, for example, can be specified by giving its concentration (variable), growth rate (variable) and maximal growth rate (constant).

Table 1. A partial model of an ecosystem consisting of entities that appear in the system.

```
entity phyto1 {
   vars: conc{role: state; initial: 10},
         limitation{aggregation:product};
   consts: maxGrowthRate = 0.5, maxLossRate = 0.02,
           sedimentationRate = 0.1;
}
entity phyto2 {
   vars: conc{role: state; initial: 2},
         limitation{aggregation:product};
   consts: maxGrowthRate = 1, maxLossRate = 0.02,
           sedimentationRate = 0.2;
}
entity phosphorus {
   vars: conc{role: state};
   consts: halfSaturation=0.02, alpha=0.1;
}
entity nitrogen {
   vars: conc{role: exogenous};
   consts: halfSaturation=0.2, alpha=0.7;
}
```

Table 1 presents a partial model specification of the hypothetical lake. This partial model specifies the four entities in the system. Two nutrients – phosphorus and nitrogen, and two species of phytoplankton – denoted as phyto1 and phyto2.

Each entity is described by variables and constants that denote its properties that are important in the given context. The specification of the constants is straightforward, by giving their values, and optionally (omitted in this example) their unit of measurement. In Table 1, for example, the constant *halfSaturation* of the entity *nitrogen* is assigned a value of 0.2.

When we specify a variable, we provide several pieces of information. The role of the variable in the system, which can be state or exogenous, gives the information of whether this variable is considered as part of the state of system or as an input/output variable. The aggregation function specifies the method of aggregation of the influences on the variable. The influences come from processes that involve the entity. We can also specify an initial value of the variable. For example, the variable *conc* of *phyto1* in Table 1 has an initial value of 10.

The phenomena that occur in the system are described by the processes of the model. From the name of the formalism: Process-Based, it is apparent that the processes are the key components of a model. Each process in the model corresponds to a phenomenon in the system. In our lake example, processes that occur would be growth and loss of phytoplankton or zooplankton feeding on phytoplankton.

Table 2. Processes involving the entities from Table 1.

```
process limitedGrowthPhyto1(phyto1, [phosphorus, nitrogen]) {
   processes: nutLimitationPs, nutLimitationNs;
   equations :
     td(phyto1.conc) = phyto1.maxGrowthRate * phyto1.conc *
       phyto1.limitation,
     td(phosphorus.conc) = -phosphorus.alpha *
       phyto1.maxGrowthRate * phyto1.conc * phyto1.limitation;
}
process limitedGrowthPhyto2(phyto2, phosphorus) {
   processes: nutLimitationPs2;
   equations :
     td(phyto2.conc) = phyto2.maxGrowthRate * phyto2.conc *
       phyto2.limitation,
     td(phosphorus.conc) = -phosphorus.alpha *
      phyto2.maxGrowthRate * phyto2.conc * phyto2.limitation;
}
process nutLimitationPs(phyto1, phosphorus) {
   equations:
     phyto1.limitation = phosphorus.conc / (phosphorus.conc +
       phosphorus.halfSaturation);
}
process nutLimitationNs(phyto1, nitrogen) {
   equations:
     phyto1.limitation = nitrogen.conc / (nitrogen.conc +
       nitrogen.halfSaturation);
}
process nutLimitationPs2(phyto2, phosphorus) {
   equations:
     phyto2.limitation = phosphorus.conc^2 / (phosphorus.conc^2
       + phosphorus.halfSaturation);
}
```

Table 2 shows the processes from the lake model. These processes involve the entities from Table 1. A process can be thought of as a relation between entities. Every process involves one or more entities from the model. *limitedGrothPhyto1*, for example, involves three entities – *phyto1, phosphorus and nitrogen* because it represents the growth of phytoplankton 1 that is limited by phosphorus and nitrogen.

In addition to being a qualitative relation between entities, a process also provides a quantitative description of that relation as one or more equations. An equation can

contain only variables and constants of the entities that participate in the corresponding process. In the example in Table 2, the equations in *nutLimitationPs* can contain variables and constants only from *phyto1* and *phosphorus* because those are the entities that take part in *nutLimitationPs*.

The equations from all of the processes are combined into a single set of differential equation, which is the purely quantitative model of the system. This model can then be used to perform quantitative analysis. We can vary the values of parameters, perform simulation of the model, perform sensitivity analysis and so forth. For each state variable in the model, we compile one differential equation that will have the derivative of that variable as its left hand side. The equation is compiled by combining all equations in the model that influence (have as left hand side) that variable. We combine the equations with the aggregation function that is given in the specification of that variable. The aggregation function can be summation, multiplication and so on. For example, the variable *phosphorus.conc* (the variable *conc* of the entity *phosphorus*) is a state variable (see Table 1), meaning that the end model will present a differential equation for that variable. This variable is influenced by two equations, those in processes limitedGrowthPhyto1 and limitedGrowthPhyto2. Having in mind that the influences on this variable are combined by the default aggregation function – summation, we obtain the following equation for the rate of change of *phosphorus.conc*:

$$\frac{d}{dt}conc = -0.1 * 0.5 * p1.conc * p1.lim - 0.1 * 1 * p2.conc * p2.lim$$

where *conc* stands for *phosphorus.conc*, *p1* for *phyto1*, *p2* for *phyto2* and *lim* for *limitation*. *p1.lim* and *p2.lim* should also be expanded with their equations given in the model, but for the sake of maintaining simplicity we will omit this here.

Specifying Domain Knowledge: Templates and Instances

Note that some entities and processes presented in Tables 1 and 2 share common properties. If we compare the entities *phyto1* and *phyto2* from Table 1, we can see that they share many similarities with respect to their variables and constants. This is to be expected since they are both phytoplankton species. On the other hand, if we compare the processes *limitedGrowthPhyto1* and *limitedGrowthPhyto2* from Table 2, we can see that they have equations that adhere to the same general pattern, which is logical because they both represent processes of limited growth of phytoplankton. Therefore, it makes sense to try to group such similar properties within some more general concepts. Hence, instead of directly creating/specifying entities and processes, and specifying all their properties, we use two-phase specification.

The knowledge (properties) which holds for more entities or processes is specified in objects which we call templates, in particular, entity templates - for specifying common properties for entities and process templates - for specifying common properties for processes. The idea is that the template captures some general knowledge that holds for many different cases and can be reused when dealing with different specific scenarios. An entity template is an incomplete entity specification. It only contains partial information for an entity. However, this information is general and can be used for more than one entity. A similar statement can be made for process templates – they can be seen as incomplete processes, i.e., processes that only contain some general information and miss specific information. Examples of entity templates and process templates from the lake domain are given in Table 3.

Table 3. Entity templates and process templates for the lake domain

```
template entity EcosystemEntity {
   vars : conc {aggregation:sum; unit:"kg/m^3"; range:<0,inf>};
}
template entity PrimaryProducer : EcosystemEntity {
   vars: limitation{aggregation:product};
   consts:
      maxGrowthRate{ range: <0,inf>; unit:"1/(day)"},
      maxLossRate { range: <0, inf>; unit:"1/(day)"},
      sedimentationRate { range: <0, inf>; unit:"1/(day)"};
}
template entity Nutrient : EcosystemEntity {
   consts:
      halfSaturation {range: <0,inf>; unit:"mg/l"},
      alpha {range: <0,inf>;
            unit: "mgAlgaeBiomass/mgZooBiomass"
      };
}
template process
Growth(pp : PrimaryProducer, ns : Nutrient<1,inf>) {}

template process LimitedGrowth: Growth {
   processes : NutLimitationFunction(pp, <n:ns>);
   equations :
      td(pp.conc) = pp.maxGrowthRate * pp.conc * pp.limitation,
      td(<n:ns>.conc) = -n.alpha * pp.maxGrowthRate * pp.conc
        * pp.limitation;
}
template process
NutLimitationFunction(pp : PrimaryProducer, n : Nutrient) {}
template process LimitationMonod1 : NutLimitationFunction {
   equations: pp.limitation = n.conc / (n.conc + n.halfSaturation);
}
template process LimitationMonod2 : NutLimitationFunction {
   equations: pp.limitation = n.conc * n.conc / (n.conc * n.conc +
      n.halfSaturation);
}
```

Entity templates and process templates are organized in a hierarchy. This enables them to inherit the properties of

their ancestors, and provides for a cleaner design of entity and process templates.

We use templates to create/specify entity or process instances. The instance acquires all the properties which were specified in the template. Additional properties which are characteristic for the particular instance can be specified.

Table 4 presents the equivalent of the model from Tables 1 and 2, specified using the templates from Table 3.

Table 4. Entity instances and process instances from the lake domain

```
entity phyto1 : PrimaryProducer {
   vars: conc{role: state; initial: 10}, limitation;
   consts: maxGrowthRate = 0.5, maxLossRate = 0.02,
           sedimentationRate = 0.1;
}
entity phyto2 : PrimaryProducer {
   vars: conc{role: state; initial: 2}, limitation;
   consts: maxGrowthRate = 1, maxLossRate = 0.02,
           sedimentationRate = 0.2;
}
entity phosphorus : Nutrient {
   vars: conc{role: state};
   consts: halfSaturation=0.02, alpha=0.1;
}
entity nitrogen : Nutrient {
   vars: conc{role: exogenous};
   consts: halfSaturation=0.2, alpha=0.7;
}
process limitedGrowthPhyto1(phyto1, [phosphorus, nitrogen]):
LimitedGrowth{
   processes: nutLimitationPs, nutLimitationNs;
}
process limitedGrowthPhyto2(phyto2, phosphorus):
LimitedGrowth{
   processes: nutLimitationPs2;
}
process nutLimitationPs(phyto1, phosphorus):
LimitationMonod1{}
process nutLimitationNs(phyto1, nitrogen): LimitationMonod1{}
process nutLimitationPs2(phyto2, phosphorus):
LimitationMonod2{}
```

Introducing Qualitative Equations to the Process-Based Formalism

Process-based formalism enables modeling on two levels: purely qualitative and fully quantitative. Between those two levels however, there is a whole spectrum of modeling possibilities with different levels of abstraction. We argue that process-based modeling allows for modeling on various levels of abstraction in excess of those already present in the formalism. This paper proposes an extension of the formalism with the introduction of an intermediate level of abstraction that will lie in between the two existing ones. The addition that we propose is based on QSIM (Kuipers, 1994), i.e., we want to provide for qualitative reasoning by means of qualitative differential equations (QDEs).

QSIM models dynamical systems in terms of qualitative variables and qualitative equations. Therefore our formalism has to be modified to encompass qualitative variables and qualitative equations. The required changes are blended into the two existing fundamental concepts – entities and processes.

Within entities, both variables and constants are adapted. The need for modification arises from the fact that QSIM substitutes the domain of real numbers in regular equations with the concept of landmarks and inter-landmark intervals. The *range* property of both variables and constants within entity templates is substituted with a *domain* property, which consists of a list of allowed landmarks. For example, the *conc* variable of the *EcosystemEntity* entity template from Table 3, instead of having as range the interval $[0, +\infty)$, will have as domain the ordered set {zero, low, medium, high}. On the other hand, every occurrence of a real number in the specification of entities has to be changes to a landmark. In particular, the values of constants and initial values of variables in entity instances have to be specified as landmarks.

When speaking about constants another issue arises. QSIM is agnostic of any quantitative relations, so constants are needed as long as they influence the qualitative behavior of the system. As a result, the user has the opportunity to keep constants and substitute landmark values for the real numbers or to completely disregard constants if they do not affect the qualitative behavior of the system. In our example, constants do not influence the qualitative behavior of the system and therefore we omit them from the model. Table 5 presents the entity templates and entity instances from our lake example.

Table 5. Entity templates and entity instances for the aquatic ecosystem

```
// Entity templates
template entity EcosystemEntity {
   vars : conc {aggregation:sum; unit:"kg/m^3"; range: [zero,
     low, medium, high]};
}
template entity PrimaryProducer : EcosystemEntity {
   vars: limitation{aggregation:product; range:[zero, low,
     medium, high]};
}
template entity Nutrient : EcosystemEntity {}
// Entity instances
entity phyto1 : PrimaryProducer {
   vars: conc{role: state; initial: high}, limitation;
}
entity phyto2 : PrimaryProducer {
   vars: conc{role: state; initial: medium}, limitation;
}
entity phosphorus : Nutrient {vars: conc{role: state};}
entity nitrogen : Nutrient {vars: conc{role: exogenous};}
```

Suitable changes are also introduced to processes. The key modification is the substitution of the quantitative equations with qualitative constraints. Each equation translates into one or more qualitative constraints. The most important constraint is the one that regards monotonicity. The QSIM formalism uses the predicates M+ and M- to specify a term that is monotonically increasing or monotonically decreasing with respect to another term. Each quantitative equation translates to one of this predicates and possibly several other helper predicates. These auxiliary predicates are artifacts of the QSIM formalism and are necessary in order to translate larger and more complex equations. These include DERIV for specifying derivatives of variables, MULT for specifying multiplication and ADD for specifying addition. All these changes are introduced in the process templates. Process instances remain unchanged. Table 6 lists the process templates and process instances for the aquatic ecosystem.

Table 6. Process templates and process instances for the aquatic ecosystem.

```
template process
Growth(pp : PrimaryProducer, ns : Nutrient<1,inf>) {}

template process LimitedGrowth: Growth {
   processes : NutLimitationFunction(pp, <n:ns>);
   constraints :
      DERIV(pp.conc, pp_dt),
      MULT(pp.conc, pp.limitation, X),
      M+(X, pp_dt),
      DERIV(<n:ns>.conc, n_dt),
      M-(X, n_dt);
}
template process
NutLimitationFunction(pp : PrimaryProducer, n : Nutrient) {}
template process LimitationMonod1 : NutLimitationFunction {
   constraints: M+(pp.limitation, n.conc);
}
template process LimitationMonod2 : NutLimitationFunction {
   constraints: M+(pp.limitation, n.conc);
}

// Process instances
process limitedGrowthPhyto1(phyto1, [phosphorus, nitrogen]):
LimitedGrowth{
   processes: nutLimitationPs, nutLimitationNs;
}
process limitedGrowthPhyto2(phyto2, phosphorus):
LimitedGrowth{
   processes: nutLimitationPs2;
}
process nutLimitationPs(phyto1, phosphorus):
LimitationMonod1{}
process nutLimitationNs(phyto1, nitrogen): LimitationMonod1{}
process nutLimitationPs2(phyto2, phosphorus):
LimitationMonod2{}
```

The last note is on combining process-based models to obtain the final model. For quantitative models, the final model was a system of differential equations, whereas here, the final model is simply a list of constraints. The final model is obtained by concatenating the constraints from all of the processes. For each state variable a constraint is added that provides the aggregation of the constraints that influence that variable.

Related Work

The formalism for modeling dynamical systems, presented here, builds on previous work in the areas of equation discovery (Todorovski and Džeroski, 2007) and inductive process modeling (Bridewell et al., 2008). The work on equation discovery uses the formalism of ordinary differential equations to represent models and grammars to represent the space of potential equation-based models for a given modeling task. Human experts have to transform knowledge about modeling dynamical systems in the domain at hand to an appropriate grammar for equation discovery. Algorithms for searching the space of candidate models and fitting constant model parameters against observed system behavior are then combined to find an optimal model that closely fit the observations. Inductive process modeling approach unifies the formalisms for representing models and knowledge into processes, entities, and templates thereof, in a way we outlined in the first part of the paper. This paper extends the process-based modeling formalism towards qualitative models represented in terms of QSIM constraints (Kuipers, 1994).

The work presented in this paper is also related to papers on the topic of integrating qualitative and quantitative reasoning methods to address the task of automated modeling of dynamical systems. Bradley et al. (2001) system PRET is a method inducing equation-based models from observed system behavior. PRET integrates cross-domain knowledge in the process of induction, which is supported by a variety of reasoning methods, ranging from qualitative reasoning and simulation to numerical simulations and parameter fitting methods. While PRET focus on implicit constraints that help checking the validity of a candidate model, our formalism make explicit constraints about how model is composed out of domain-specific entities and processes. These are also used to represent the final equation-based model, which greatly improves its comprehensibility to human experts in the particular domain of use. Furthermore, QOPH system (Garret et al., 2007) induces qualitative models from numeric observations of the system behavior. Similarly to our formalism, QOPH uses modeling knowledge casted in terms of model components corresponding to process templates in our formalism. In contrast to the work presented here, the QOPH focus is limited to qualitative models; it uses discretization of numeric data to obtain qualitative observations that are in turn used to induce qualitative models.

Conclusion and Further Work

This paper presented Process-based modeling trough an example from aquatic ecosystems. Process-based modeling is a paradigm with two distinct levels of abstraction – one purely qualitative and one fully quantitative. Between those levels there is a continuum of different levels of abstraction. In this paper we have presented one way of extending Process-based modeling with an intermediate level of abstraction. QSIM formalism was used for describing concepts on this intermediate level.

This paper serves as a proof of principle that process-based modeling can be extended to include intermediate qualitative levels. In the future, this proof of principle should be further developed into a fully fledged system for modeling using QSIM formalism. Other features should include automatic or assisted translation of equations from the quantitative level of the formalism to qualitative equations and assisted translation of qualitative equations into quantitative ones. Other directions for further work can be implementing other levels of abstraction according to other existing formalism that lie in between the two levels offered by process-based models.

References

Bradley, E., Easley, M., and Stolle, R. (2001) Reasoning about nonlinear system identification. *Artificial Intelligence*, 133, 139—188.

Bridewell, W., Langley, P., Todorovski, L., & Džeroski, S. (2008) Inductive process modeling. *Machine Learning*, 71, 1—32.

Garret S.M., Coghill, G.M., Shrinivasan, A., and King, R.D. (2007) Learning qualitative models of physical and biological systems. In *Computational Discovery of Scientific Knowledge* (pp. 248—272), Springer, Heidelberg, Germany.

Kuipers, B. (1994) Qualitative Reasoning. MIT Press, Cambridbge, MA.

Todorovski, L. and Džeroski S. (2007) Integrating domain knowledge in equation discovery. In *Computational Discovery of Scientific Knowledge* (pp. 69—97), Springer, Heidelberg, Germany.