Graph-based Mathematical Modelling – Concepts

Arne Tobias Elve*[1] and Heinz A Preisig[1]

[1]Department of Chemical Engineering, Norwegian University of Science and Technology, Trondheim, Norway

Abstract

The mathematical models representing the dynamic behaviour of any combination of physical, chemical and biological processing plants, requires a large number of equations even if the process is of relative low complexity. Constructing and maintaining these models require a considerable effort. Close inspection though reveals that many of the equations are structurally the same and appear mostly in duplications. Avoiding all duplications and exploiting the aspects of ontologies for knowledge representation and utilising graphs to capture the structure of the plant, both the construction as well as maintenance becomes much more straightforward by handling the complexity in the form of indexing instead of writing a large number of equations. This paper presents the design decisions of an ontology-based modelling tool that produce compact and executable code.

Keywords

Process modelling, computer-aided modelling, multi-scale modelling, automatic code generation, process systems engineering, ontology, analytics

Introduction

Modelling is a key activity in modern chemical engineering, and models are used in many of the core activities such as design, control, optimisation and simulation. A complete and proper process model is able to predict the process behaviour, which means that models can replace expensive experiments. However, the model has to be of reliable quality. Whilst this implies savings, the modelling also requires resources. The construction of the models is often done by a modelling specialist and intricately designed for a special purpose. Also, the model equations are commonly located in different parts of the overall code and difficult to locate. Most models are not well documented, making the interpretation of results and maintenance of a model a troublesome task. We are constructing a generic modelling tool that generates executable model code for essentially any target language. The model code is generated with emphasis on being well-documented, flexible and reusable.

This contribution builds on a generic approach to modelling, with the objectives of (i) generating internally consistent models (Preisig, 2010), (ii) generate complete models: all necessary elements are included: integration, differential, transport, transposition and closure equations, (iii) canonical representation involving a minimum of structural elements and a simple language for representation of equations and (iv) proved parser and compiler. The latter, to be easily configurable for any output language. These objectives are met by formulating networks that again consists of networks that consist of small building blocks. The building blocks, which we refer to as primary models, are complete and compact models containing attributes and equations to describe a selected part of a process. Primary models can be connected to other primary models describing other parts of a process. This composes networks consisting of primary models and networks, which are the models we generate using our tools.

This approach to modelling is in alignment with the suggestions for modelling tools presented by (Foss et al., 1998), which emphasises on a structured approach to modelling. Our modelling tool is analogous to but more fundamental than, the work by (Fedorova et al., 2015) and MOSAIC (Kraus et al., 2014), which formulate model templates, while our approach extract primary models from an ontology, hence it is more related to the work by (Brandt et al., 2008).

* *arne.t.elve@ntnu.no*

Network view

The development of an ontology-based modelling tool requires a proper structuring of process models. A key aspect is thereby that the complete description of a process model is organised under three specified meta sections, namely, *structure, behaviour* and *typing* (Preisig and Elve, 2016). The first section, *structure*, defines structural entities and their relations. The *behaviour* section defines the role of each entity, in other words, it describes how an entity behaves and how it is affected by or the effect it has on other entities. The last section is *typing*, which defines the separation of quantities based on the captured attributes.

The modelling methodology used in the modelling tool involves capturing the behaviour of a set of entities that interact with each other in a structure. We, therefore take the view of considering any model to be a network represented by a directed graph in which the nodes represent capacities, and the arcs represent interactions of quantities that characterise these capacities. The characterising quantities are termed *states* and the interactions are quantities that affect the states. The directionality of the arcs provides the reference coordinate for the respective interaction. The state is the attribute that we want to model. We derive the states from defining tokens. The term token is taken from the analogy of Petri-nets, where tokens are what moves around in the graph, allowing for an abstraction beyond physical systems. The network with its tokens lives in a frame, which in the case of dynamic systems is time and space. Networks may represent different types of systems and are thus specialised accordingly. For example, facilitate for physical and control systems.

In order to capture the behaviour of complex models, we generate networks of primary models with a set of selected attributes described by the *typing* section of the ontology. These networks communicate, hence we get a network of networks interacting on every layer. The ontology uses hierarchical trees for network definitions, meaning that we first formulate a shared ontology for all networks, and then start refining the shared ontology after a set of rules for the different networks using the meta sections *structure, behaviour* and *typing*. The result of this refinement is that the knowledge extracted from the ontology is tailored to a specific domain for a particular network, but does still have shared attributes that open for interaction with other types of networks. This implements an inheritance mechanism.

For the domain specific tailoring of the ontology, the first refinement is the separation of a network for control and a network the physical components. These networks may again be further refined, such as for physical systems that consist of different phases, each of which again can be subdivided into further refined networks. The rule for a network definition is that within each network a specific set of tokens is captured and modelled. To handle the interaction between two networks we introduce a connection network, which serves the purpose of converting a set of tokens from one network to a set of tokens represented in the other network. Refinement of networks within specific phases open for multi-scale modelling by allowing for properties of a phase to be described on a lower level with a refined set of tokens. For a controlled physical process, the process model typically comprises of three types of networks, namely, the physical network, the control network and the connection network that handle interactions between two networks.

The physical network

The physical network represents the process itself with its internal chemistry and biology. The nature of the physical network is described in (Preisig, 2010) and illustrated in fig. 1. The construction of the physical network starts at the integrator, in which the differential state is integrated over time for ODEs and time and space for PDEs. The intensive properties are functions of the states given by the thermodynamic relations. These are thus state variable transformations, which are illustrated by the red box of fig. 1. Some of these intensive properties are responsible for driving the transport between connected pairs of capacities, which primarily are the temperature, the pressure and the chemical potential for which the Bond-graph theory introduced the term "effort variables"(Breedveld et al., 1991).

Following the route in fig. 1, the quantities labelled, y, are the effort variables and other quantities required to compute the transport and the transposition, which are modelled in the two green boxes in the feedback loop of the physical network. The transport box uses the effort variables and provides equations for the transportation of tokens. Tokens may also be converted into other tokens and those are modelled in the lower green box named transposition, which represents the internal changes in capacities, such as phase changes and chemical reactions. The yellow box connects the transport and the transposition equations together and yields the differential balance equations.

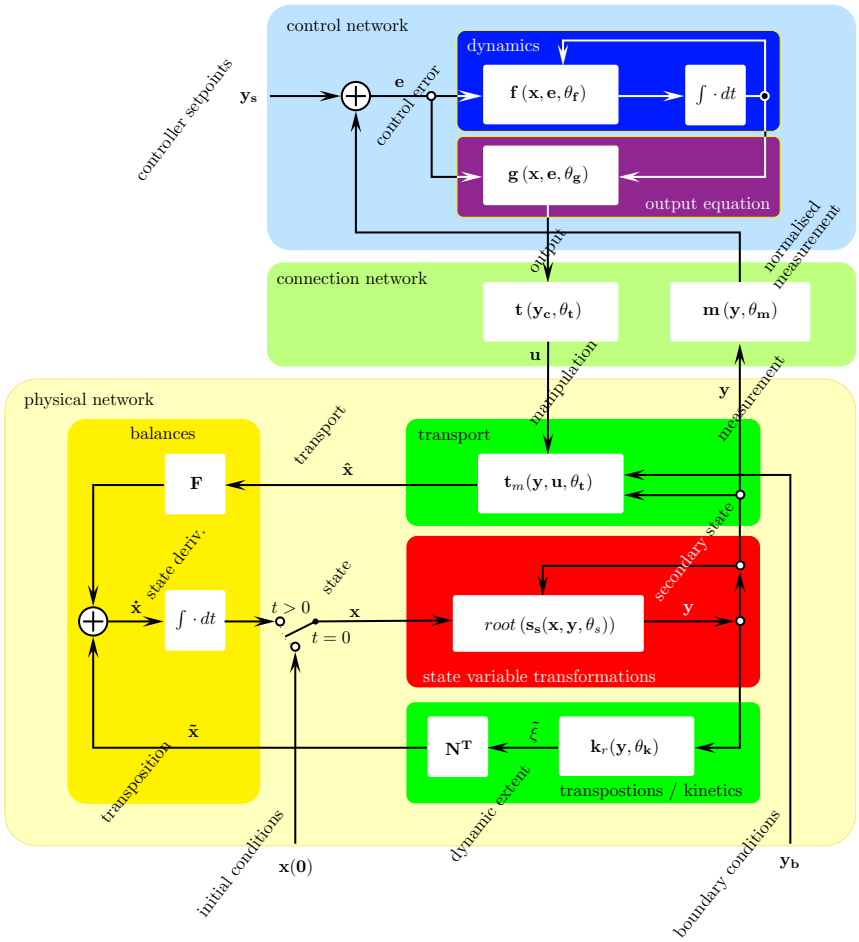

Figure 1. Illustration of a physical network connected to a control network. The measurement in the physical network is converted into a signal comprehensible in the control network as an input to a control structure. The output from the controller is converted into a valve position in the transport equations in the physical network.

Control network

Control is implemented as a signal processing network.The feedback control system illustrated in fig. 1, which could be replaced by any other scheme such as model-predictive control systems, receives a set of measurements from the physical network as signals, which are compared to a defined set-points. Based on the comparison the control action is determined, which is a signal representing valve manipulation.

Connection network

The connection network has access to the properties of the two connected networks and converts tokens of one network to corresponding tokens of the other network. In fig. 1 the connection network converts the token value of the quantity being measured to a token signal, and back again converts the controller output to a variable representing the valve position in the transport laws. The connection network thus converts and broadcasts a variable from one network to another and make the variable understandable for the other network.

Variables and equations

Once the structure of the model is established, the behaviour for the primary models can be given. The behaviour provides the link from the network representation to the variables and equations assigned to the various boxes in fig. 1. The assignment to the boxes determines where the variable can be used. For example, a variable describing the mass transport is assigned to the transport box. Equations describing transport are indirect functions of the states of the two connected capacities. The intermediate variable associated with the indirect function is the result of a state variable transformation that belongs to the red box in fig. 1. The information required for definition of the behaviour is captured by a small meta-language made for representing variables and equations.

Index set operations

To ensure correctness of the model, and at the same time keep the number of calculations to a minimum, we

equip each variable with indexing sets. For example, all variables are indexed with the network they belong to. Variables may, in addition, be associated with an arc or a node and within the node or arc, they can be assigned to a specific token. The mathematical representation is a state-space model of the tokens in the nodes. Consequently, the state is indexed with network and tokens in the nodes, and the transport is indexed with network and tokens in the arcs. Different mechanisms of transportation generate a subset of arcs for each mechanism. For mass systems some of the nodes might allow for chemical reactions, while some might not. Therefore, the transposition variables are indexed with reactive nodes, hence they will only be calculated for nodes with reactions. This leads to compact vectors and matrices, thereby reducing computational and storage costs. Using index sets require mapping procedures for defining subsets and special set operators for handling the mathematical operations required to formulate the modelling equations.

The mapping keep a reference from the global set to the defined subset given as attribute in primary models. For example, if a model consist of total five arcs described by the global arc set, $\mathcal{A}$, then a subset of the arc set only containing two arcs that involve volumetric flow, $\mathcal{A}^{\mathcal{V}}$, can be formulated by assigning a subset of the global set as described in eq. (1).

$$\mathcal{A} = \{a, b, c, d, e\}, \qquad \mathcal{A}^{\mathcal{V}} = \{a, e\} \subset \mathcal{A} \tag{1}$$

This definition generates a reference of the subset into the superset, hence it reduces the space for which an equation using that subset has to be calculated. A variable indexed with $\mathcal{A}^{\mathcal{V}}$ will only be calculated for the two arcs in that set, while a variable indexed with $\mathcal{A}$ will be calculated for the five arcs in the global set. Combinations of index sets are formulated as the Kronecker-product of two sets. For example, the combination of arcs and species, $\mathcal{AS}$ represents the species transported by each arc with the directionality included. A variable with more than one index set, such as matrices and tensors, separate the different index sets with a comma. The index sets are reflected into every variable.

The mathematical operators that handle the indexed objects are equivalent to matrix operations but with an additional twist. We define in total three "set-sensitive" multiplication operators. All of them process the physical units like a generic multiplication, the difference is in the index handling, which is handled analogously to Einstein notation for matrices.

- *expand* – the space is expanded to the maximal space given the two sets of indices. The symbol for the expansion product is a dot: .
- *reduce* – one of the components of the space is eliminated, the space shrinks by the indicated dimension. The symbol for the reduction is: $\overset{\mathcal{S}}{\star}$, with the $\mathcal{S}$ above the star determine over which dimension the reduction takes place.
- *block expansion* – The block operations expand to the maximal space defined by block operations. This product is often referred to as the Khatri-Rao product. The symbol for the block expansion is: $\odot$

These operators are, in addition to the rest of the index handling, included in the meta-language used for equations. A variable defined using the meta-language inherit the index sets through the calculations thus ensure structural correctness.

Units

Quantities are differentiated by two attributes that together span the essential parameters needed to formalise the structures of quantities, namely the *kind* and the *magnitude*. The kind attribute of a quantity identifies the observable property quantified, for example, length, force, frequency, etc., while the magnitude of the quantity expresses its relative size compared to other quantities of the same kind. For the modelling tool, we use the seven basic SI units to describe the kind and we represent them only by integer exponents for units, which are also the quantities being stored. The units of variables are calculated when defining the equations.

Equations

We look at a set of model equations as a circuit of interconnected process variables and equations that fit into the scheme presented in fig. 1. For the physical network, the decomposition starts with the states, which then are used to define the state variable transformations. The state variable transformations primarily provide the effort variables that are used in the transport equations, which in turn provide the flow used in balance equations, which again are used to describe state equations. This decomposition is formulated a bipartite graph consisting of two types of vertices, namely variables and equations that are connected with edges. Figure 2 illustrates the behavioural decomposition of a diffusive mass transport equation described by eq. (2).

$$\underline{\hat{\mathbf{n}}}_{\mathcal{N}^d\mathcal{S}} := \underline{\mathbf{B}}^d{}_{\mathcal{A}^d} \odot \underline{\mathbf{K}}^d{}_{\mathcal{A}^d\mathcal{S}} \cdot \underline{\underline{\mathbf{F}}}^{\mu}{}_{\mathcal{N}^d\mathcal{S},\mathcal{A}^d\mathcal{S}} \overset{\mathcal{N}^d\mathcal{S}}{\star} \underline{\mu}_{\mathcal{N}^d\mathcal{S}}, \tag{2}$$

$\underline{\hat{\mathbf{n}}}_{\mathcal{A}^d\mathcal{S}}$ represents diffusive transport rate of moles, and $\underline{\mathbf{B}}^d{}_{\mathcal{A}^d}$ is the surface area of the boundary in the arc

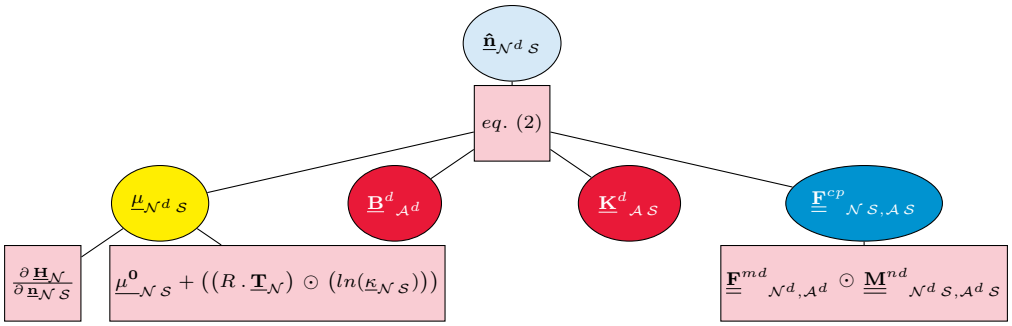

Figure 2. The equation structure given in eq. (2)*. The circular nodes represent variables and the square pink nodes represent the equations. The variable calculated by an equation is placed above the equation and the variables used in the equation are placed below. The colours of the variables assign different behaviour.*

representing the diffusive transport. $\underline{\underline{\mathbf{F}}}^{\mu}_{\mathcal{N}^d\mathcal{S},\mathcal{A}^d\mathcal{S}}$ is the incidence matrix of the species in nodes with diffusion and species transported in arcs with diffusion and $\underline{\mu}_{\mathcal{N}^d\,\mathcal{S}}$ is the chemical potential for every species in the respective nodes. The transport equation is composed of process variables describing the basic phenomena and the mathematical set operators. Each process variable may be defined by more than one equation, which again is represented using different process variables. For example the chemical potential in fig. 2, can either be calculated as a state derivative of enthalpy and amount, or it can be calculated from a reference value and the change of temperature and composition. The outcome is two alternative paths of equations for calculating the same property. Which of the alternatives is finally used in the model has to be decided by the user.

This described behavioural decomposition is recursively repeated until all variables are defined and fit into the structure illustrated in fig. 1. By giving initial conditions and selecting alternative equation paths the equation circuit is trimmed down from a cyclic graph to a tree, which is included into the primary models. The resulting equation tree together with the structural knowledge represented by the networks of the nodes and the arcs will ensure that the model is well defined and complete.

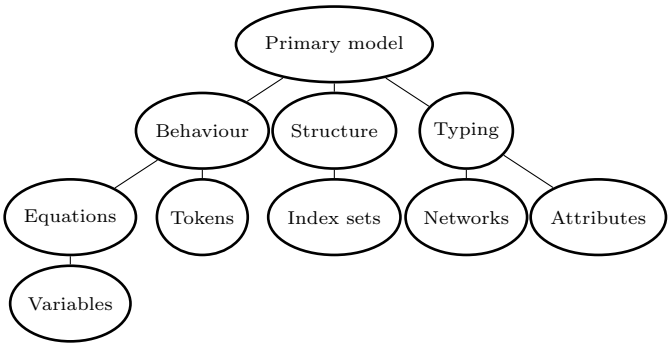

Figure 3. The specialised contents of primary models

Primary model definition

The primary models are the building blocks used for generating the complete model. They contain the detailed knowledge of process quantities and the equations. An illustration of the contents of a primary model is illustrated in fig. 3.

The primary model requires behaviour knowledge for the tokens and what state, transport, transposition or closure equations applies, which also exposes the variables being used in the equation. The relevant index sets and the mappings are stored as structural knowledge, and the typing is stored to describe the attributes that determine how the modelling object will be handled mathematically.

Case study – model of a dynamic flash tank

The following highlights the application of primary models in the modelling tool by the means of a case study of a controlled dynamic flash tank. The tank has an inlet and two outlets, each with a valve. There are two phases in the tank: a liquid that is kept at a constant level and a gas phase that is kept at a constant pressure. The feed into the tank is a mixture of two liquid components. The valve in the drain is used to control the liquid level in the tank by manipulating the liquid flow whilst the valve in the outlet controls the pressure by manipulating the gas flow.

Model decomposition

The model for the flash involves two physical networks representing the liquid and the gas phase, a control network for representing both the liquid controller and the pressure controller. In addition, there are three connection networks, one for the conversion of species from the liquid to the gas phase, one to handle the measurement conversion from the gas phase over to the con-

trol network and a connection network for doing the token conversion from the liquid phase over to the control network and back again. This network information is illustrated in Figure 4. The model for the flash involves

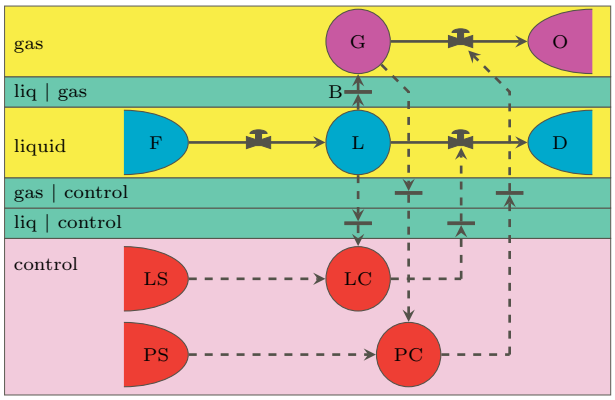

Figure 4. The network of networks for the dynamic flash

three volumetric flows connected to reservoirs (F,D,O). The mass flow between the liquid (L) and the gas phase (G) is diffusive. The level controller receives a set-point from a source (LS) and measures the level in the liquid phase and returns a valve position in the volumetric flow to the drain (D). The pressure controller (PC) measures the pressure in the gas phase, compare the measurement with a set-point it receives from a source (PS) and returns a valve position to limit the volumetric flow from the gas phase to the outlet.

Results

The complete model describes a controlled dynamic flash tank. The implementation of this model is summarised in Table 1. Notice how few equations are being used compared to the number of index mappings. For more complex systems consisting of dynamic flashes, the number of equations will not increase. It is only the index sets that grow in size, but not in complexity.

Table 1. Summary of the implementation of the dynamic flash tank in the modelling tool

Description	**Number**
Networks:	6
Nodes:	14
Arcs:	15
Shared variables:	2
Physical variables:	71
Control variables:	10
Index mappings:	22
Model equations:	23
Control equation:	3

Conclusions

This paper presents a multi-network approach for the representation of generic models applied to the representation of controlled physical-chemical-biological processes. The approach uses a directed graph for representing the structural elements of the model and uses a bipartite equation/variable graph for the representation of the mathematical relations. The equations are formulated starting with the states and adding successively the closure, the transport and transposition and definition of the state differentials to ensure correctness in both structural elements of the index sets and the units. Both units and index sets are generated as part of the equation definition for each variable derived from the state variables. The structure and equations are put together in primary models that can be handled individually. This generic modelling procedure was applied to a case study of a controlled dynamic flash. The result from the case study involves a rather large number of variables and mappings of sets, but the total number of model equations is only 23. With dynamic flashes being the dominating component of all liquid/gas processes, it is only those 23 equations that capture the heart of the description. All the structural complexity is mapped into the index sets.

References

Brandt, S. C., Morbach, J., Miatidis, M., Theißen, M., Jarke, M., and Marquardt, W. (2008). An ontology-based approach to knowledge management in design processes. *Comput Chem Eng*, 32(12):320–342.

Breedveld, P., Rosenberg, R., and Zhou, T. (1991). Bibliography of bond graph theory and application. *J Frankl Inst*, 328(5-6):1067–1109.

Fedorova, M., Sin, G., and Gani, R. (2015). Computer-aided modelling template: Concept and application. *Comput Chem Eng*, 83:232–247.

Foss, B., Lohmann, B., and Marquardt, W. (1998). A field study of the industrial modeling process. *J Process Contr*, 8(5-6):325–338.

Kraus, R., Fillinger, S., Tolksdorf, G., Minh, D. H., Merchan-Restrepo, V. A., and Wozny, G. (2014). Improving Model and Data Integration Using MOSAIC as Central Data Management Platform. *Chem-Ing-Tech*, 86(7):1130–1136.

Preisig, H. A. (2010). Constructing and maintaining proper process models. *Comput Chem Eng*, 34(9):1543–1555.

Preisig, H. A. and Elve, A. T. (2016). Ontology construction for multi-network models. In *Computer Aided Chemical Engineering*, volume 38, pages 1087–1092.