Proceedings of the European Control Conference 2007
Kos, Greece, July 2-5, 2007

Interconnection by Sharing Variables

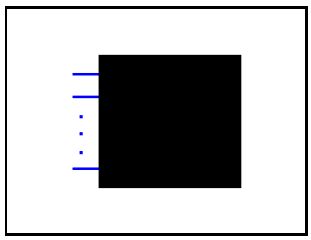
Jan C. Willems
K.U. Leuven
B-3001 Leuven, Belgium
Jan.Willems@esat.kuleuven.be
www.esat.kuleuven.be/∼jwillems

***Abstract*— We present a formalism for system interconnection based on the idea is that systems are interconnected by sharing variables. This leads to a procedure for modeling interconnected systems based on tearing, zooming, and linking. The ingredients of this language are terminals, modules, an interconnection architecture, the modules embedding, and manifest variable assignment.**

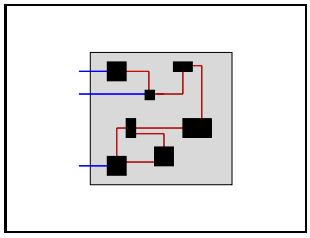
I. Introduction

'Arrows', signal flow graphs, dominate systems theory in engineering. There are many situations, for instance in signal processing, where the signal flow graph structure is eminently suitable. However, the architecture formalized by the signal flow arrows is often viewed as being an essential aspect of describing the interaction of a system with its environment. But, the opposite is actually the case, especially for the description of physical systems and for describing their interconnections. Often, signal flow graphs are unphysical, a figment of the imagination, cumbersome, and unnecessary. Sharing common variables is a much more key idea for system interconnection than input-to-output connection.

A typical modeling task can be viewed as follows. Our aim is to model the dynamic behavior of a number of related variables. This is visualized by means of a blackbox (see the figure below) with a number of terminals. When dealing with interconnections, it is natural to assume (i) that these terminals and their variables are real physical entities, and (ii) that there are usually *many* physical variables collectively and indivisibly associated with *one and the same* terminal.



The blackbox suggests that there is an underlying structure that links the terminal variables and that leads to the (dynamic) laws that govern them. Coming up with the (dynamic) laws that relate the variables of interest requires examining what is inside the blackbox. Most systems consist of interacting components. In order to discover these interactions, we look inside the blackbox, where we find an interconnection architecture of 'smaller' blackboxes that interact through terminals of their own (see the figure below).



Modeling then proceeds by examining the smaller blackboxes and their interactions. This modeling procedure is called *tearing, zooming, and linking*.

1) *Tearing* refers to viewing a system as an interconnection of smaller subsystems.
2) *Zooming* refers to modeling the subsystems.
3) *Linking* refers to modeling the interconnections.

In this paper, an outline is given of a formal procedure for obtaining a model by viewing a system (a blackbox) as an interconnection of subsystems (smaller blackboxes). This formalism uses the notions of a behavior and of latent variables in an effective way. The basic ingredients are: terminals, (parameterized) modules, the interconnection architecture, the module embedding, and the manifest variable assignment.

II. Behavioral systems

The behavioral framework views modeling as follows. Assume that we have a phenomenon that we wish to describe mathematically. Nature (that is, the reality that governs this phenomenon) can produce certain events (also called outcomes). The totality of possible events (*before* we have modelled the phenomenon) forms a set $\mathbb{U}$, called the *universum*. A *mathematical model* of the phenomenon restricts the outcomes that are declared possible to a subset $\mathscr{B}$ of $\mathbb{U}$; $\mathscr{B}$ is called the *behavior* of the model. We refer to $(\mathbb{U}, \mathscr{B})$ (or to $\mathscr{B}$ by itself, since $\mathbb{U}$ usually follows from the context) as a mathematical model.

In the study of dynamical systems we are more specifically interested in situations where the events are signals, trajectories, i.e. maps from a set of *independent variables* (time, in

ISBN: 978-960-89028-5-5

the present paper) to a set of *dependent variables* (the values taken on by the signals). In this case the universum is the collection of all maps from the set of independent variables to the set of dependent variables. A (dynamical) *system* is defined as a triple

$$\Sigma = (\mathbb{T}, \mathbb{W}, \mathscr{B})$$

with $\mathscr{B}$, *the behavior,* a subset of $\mathbb{W}^{\mathbb{T}}$ ($\mathbb{W}^{\mathbb{T}}$ is standard mathematical notation for the set of all maps from $\mathbb{T}$ to $\mathbb{W}$). The behavior is the central object in this definition. It formalizes which signals $w : \mathbb{T} \to \mathbb{W}$ are possible, according to the model: those in $\mathscr{B}$, and which are not: those not in $\mathscr{B}$. The behavioral framework treats a model for what it is: an exclusion law. Of course, in applications, the behavior $\mathscr{B}$ must be specified somehow, and it is here that differential equations (and difference equations for discrete-time systems) enter the scene.

In the equations describing systems, very often other variables appear in addition to those whose behavior the model aims at describing. The origin of these auxiliary variables varies from case to case. They may be state variables (as in flows, automata, and input/state/output systems); they may be potentials (as in the well-known expressions for the solutions of Maxwell's equations); most frequently, they are interconnection variables. It is important to incorporate these variables in our modeling language *ab initio*, and to distinguish clearly between the variables whose behavior the model aims at, and the auxiliary variables introduced in the modeling process. The former are called *manifest* variables, and the latter *latent* variables.

A *mathematical model with latent variables* is defined as a triple $(\mathbb{U}, \mathbb{L}, \mathscr{B}_{\text{full}})$ with $\mathbb{U}$ the universum of manifest variables, $\mathbb{L}$ the universum of latent variables, and $\mathscr{B}_{\text{full}} \subseteq \mathbb{U} \times \mathbb{L}$ the *full behavior*. It induces (or *represents*) the *manifest model* $(\mathbb{U}, \mathscr{B})$, with $\mathscr{B} = \{w \in \mathbb{U} \mid \text{there exists } \ell \in \mathbb{L} \text{ such that } (w, \ell) \in \mathscr{B}_{\text{full}}\}$. A (dynamical) *system with latent variables* is defined completely analogously as

$$\Sigma_{\text{full}} = (\mathbb{T}, \mathbb{W}, \mathbb{L}, \mathscr{B}_{\text{full}})$$

with $\mathscr{B}_{\text{full}} \subseteq (\mathbb{W} \times \mathbb{L})^{\mathbb{T}}$. The notion of a system with latent variables is the natural end-point of a modeling process and hence a very natural starting point for the analysis and synthesis of systems. Latent variables enter also very forcefully in representation questions.

The procedure of modeling by *tearing, zooming, and linking* is an excellent illustration of the appropriateness of the behavioral approach as the supporting mathematical language. We assume throughout finiteness, i.e., that we interconnect a finite number of modules (subsystems), each with a finite number of terminals, etc. Of course, there are many interconnections that do not fit this 'terminal' paradigm: actions at a distance (as gravity), rolling and sliding, mixing, components that are interconnected through distributed surfaces, etc.

III. Tearing, zooming, and linking

In this section, an outline is given of a formal procedure for obtaining a model by viewing a system (a blackbox) as an interconnection of subsystems (smaller blackboxes). This formalism uses the notions of a behavior and of latent variables in an effective way. The basic ingredients are:

(i) terminals,
(ii) (parameterized) modules,
(iii) the interconnection architecture,
(iv) the module embedding, and
(v) the manifest variable assignment.

A *terminal* is specified by its *type*. Giving the type of a terminal identifies the kind of a physical terminal that we are dealing with. The type of terminal implies a universum of *terminal variables*. These variables are physical quantities that characterize the possible 'signal states' on the terminal, it specifies how the module interacts with the environment through this terminal. Some examples of terminals are given in the table below.

Type of terminal	Variables	Universum
electrical	(voltage, current)	$\mathbb{R} \times \mathbb{R}$
1-D mechanical	(force, position)	$\mathbb{R} \times \mathbb{R}$
2-D mechanical	(position, attitude, force, torque)	$\mathbb{R}^2 \times [0, 2\pi) \times \mathbb{R}^2 \times \mathbb{R}$
thermal	(temperature, heat flow)	$\mathbb{R}_+ \times \mathbb{R}$
fluidic	(pressure, flow)	$\mathbb{R} \times \mathbb{R}$
$\mathtt{m}$-dim. input	$(u_1, u_2, \ldots, u_{\mathtt{m}})$	$\mathbb{R}^{\mathtt{m}}$
$\mathtt{p}$-dim. output	$(y_1, y_2, \ldots, y_{\mathtt{p}})$	$\mathbb{R}^{\mathtt{p}}$
etc.	etc.	etc.

A *module* is specified by its *type*, and its *behavior*. Giving the type of a module identifies the kind of a physical system that we are dealing with. Giving a *behavior specification* of a module implies giving a *representation* and the values of the associated *parameters* a representation. Combined these specify the behavior of the variables on the terminals of the module. The type of a module implies an ordered set of terminals. Since each of the terminals comes equipped with a universum of terminal variables, we thus obtain an ordered set of variables associated with that module. The module behavior then specifies what time trajectories are possible for these variables. Thus a module defines a dynamical system $(\mathbb{R}, \mathbb{W}, \mathscr{B})$ with $\mathbb{W}$ the Cartesian product over the terminals of the universa of the terminal variables. However, there are very many ways to specify a behavior (for example, as the solution set of a differential equation, as the image of a differential operator, through a latent variable model, through a transfer function, etc.). The behavioral representation picks out one of these. These representations will then contain unspecified parameters (for example, the coefficients of the differential equation, or the polynomials in a transfer function). Giving the parameter values specifies their numerical values, and completes the specification of the behavior of the signals that are possible on the terminals of a module.

Some general examples of modules with their terminals are given below.

Type of module	Terminals	Type of terminals
resistor	(terminal1, terminal2)	(electrical, electrical)
transistor	(collector, emitter, base)	(electrical, idem, idem)
mass, 2 applicators	(appl1, appl2)	(3-D mechanical, idem)
2-inlet vessel	(inlet1, inlet2)	(fluidic, fluidic)
heat exchanger	(inlet, outlet)	(fluidic-thermal, idem)
signal processor	(input, output)	($\mathbb{m}$-input, p-output)
etc.	etc.	etc.

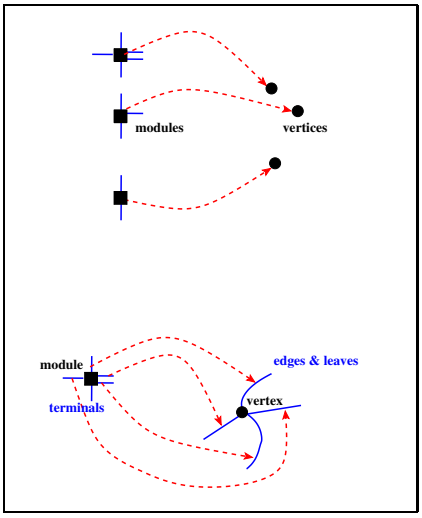
Some examples of behavioral specifications are given below (to give an idea of what we have in mind).

Type of module	Specification	Parameter
resistor	default	R in ohms
n-terminal circuit	transfer impedance	$G \in \mathbb{R}^{\mathbb{n}\times\mathbb{n}}(\xi)$
n-port circuit	i/s/o admittance	(A,B,C,D)
bar, 2 applicators	Lagrangian equations	mass and length
2-inlet vessel	default	geometry
signal processor	kernel representation	$R \in \mathbb{R}^{\bullet\times\bullet}[\xi]$
signal processor	latent variable	(R,M)
etc.	etc.	etc.

Formally, a module Σ of a given type with T terminals yields $\mathbb{W} = \mathbb{W}_1 \times \mathbb{W}_2 \times \cdots \times \mathbb{W}_T$, with $\mathbb{W}_k$ the universum associated with the k-th terminal. The behavioral specification yields the behavior $\mathscr{B} \subseteq \mathbb{W}^{\mathbb{R}}$. If $(w_1, w_2, \ldots, w_T) \in \mathscr{B}$, then we think of $w_k \in (\mathbb{W}_k)^{\mathbb{R}}$ as a signal that can be realized on the k-th terminal.

The next element in the specification of a model is the *interconnection architecture* (or *graph*). This is defined as a graph with leaves. Recall that a *graph* is defined as $\mathscr{G} = (\mathbb{V}, \mathbb{E}, \mathscr{A})$, with $\mathbb{V}$ the set of *vertices*, $\mathbb{E}$ the set of *edges*, and $\mathscr{A}$ the *adjacency map*. $\mathscr{A}$ associates with each edge $e \in \mathbb{E}$ an unordered pair $\mathscr{A}(e) = \{v_1, v_2\}$ with $v_1, v_2 \in \mathbb{V}$, in which case e is said to be *adjacent* to v_1 and v_2. A graph with leaves is a graph in which some of the 'edges' are adjacent to only one vertex. These special 'edges' are called 'leaves'. Formally, a *graph with leaves* is defined as $\mathscr{G} = (\mathbb{V}, \mathbb{E}, \mathbb{L}, \mathscr{A})$, with $\mathbb{V}$ the set of *vertices*, $\mathbb{E}$ the set of *edges*, $\mathbb{L}$ the set of *leaves*, and $\mathscr{A}$ the *adjacency map*. $\mathscr{A}$ associates with each edge $e \in \mathbb{E}$ an unordered pair $\mathscr{A}(e) = \{v_1, v_2\}$ with $v_1, v_2 \in \mathbb{V}$, and with each leaf $\ell \in \mathbb{L}$ an element $\mathscr{A}(\ell) = v \in \mathbb{V}$, in which case e is said to be *adjacent* to v_1 and v_2, and ℓ to v.

The *module embedding* (i) associates with each vertex of the interconnection architecture a module (with its parameter values), and (ii) specifies for every vertex a $1 \leftrightarrow 1$ assignment between the edges and leaves adjacent to the vertex, and the terminals of the module that has been associated with this vertex. This is illustrated in the figure below.



Since each edge is adjacent to two vertices, each edge is associated by the module embedding with 2 terminals. It is assumed that this assignment results in terminals that are of the same type if the type is physical (both electrical, or mechanical, or hydraulic, or thermal, etc.), or of opposite type (one input, one output) if the terminals are of logical type. In other words, if the edge e is adjacent to vertices v_1 and v_2, then the module embedding makes v_1 and v_2 either of the same physical type, or of opposite logical type. In this way, each edge is labelled by a terminal type, and each vertex is labelled as a module.

The edges of the interconnection architecture specifies how terminals of modules are linked. *Assume that there are universal rules that specify relations among the variables on the terminals that are linked.* Pairing of terminals by the edges of the interconnection architecture implies an *interconnection law*. Some examples of interconnection laws are shown below.

Pair of terminal	Variables terminal 1	Variables terminal 2	Interconnection constraints
electrical	(V_1, I_1)	(V_2, I_2)	$V_1 = V_2, I_1 + I_2 = 0$
1-D mechanical	(F_1, q_1)	(F_2, q_2)	$F_1 + F_2 = 0, q_1 = q_2$
thermal	(Q_1, T_1)	(Q_2, T_2)	$Q_1 + Q_2 = 0, T_1 = T_2$
fluidic	$(p_1, f_1$	(p_2, f_2)	$p_1 = p_2, f_1 + f_2 = 0$
information processing	$\mathbb{m}$-input u	$\mathbb{m}$-output y	$u = y$
etc.	etc.	etc.	etc.

In this way, we obtain a set of behavioral equations:
(i) For each vertex of the interconnection architecture, we obtain a behavior relating the variables that 'live' on the terminals of the module that is associated with this vertex. These behavioral equations are called the *module equations*.
(ii) For each edge of the interconnection architecture, we obtain behavioral equations relating the variables that 'live' on the terminals and that are linked by this edge. These behavioral equations are called the *interconnection equations*.

Note that no interconnection equation result from the leaves, but the associated terminal variables do enter in the module equations.

These equations together specify the behavior of all the variables on all the terminals involved. Note that each vertex of the interconnection graph is in the end labelled as a module, and each edge as a terminal type: we have systems

in the vertices, and interconnections in the edges. This stands in contrast to conventional electrical circuit theory, which has the elements (i.e. modules) in the edges, and the interconnections in the vertices. The interconnection equations are usually very simple. Typically they equate certain variables and put the sum of other variables to zero. We therefore think of interconnection as *variable sharing*.

The final step consists of the *manifest variable assignment*. This is a mapping that assigns the manifest variables as a function of the terminal variables. *The terminal variables are henceforth considered as latent variables.*

The behavioral equations (the combination of the module equations and the interconnection equations), combined with the manifest variable assignment define the full behavior of the system that is being modelled. It is the end result of the modeling process based on tearing ($\cong$ the interconnection architecture), zooming ($\cong$ obtaining the module equations and the manifest variable assignment), and linking ($\cong$ setting up the interconnection equations).

This tearing-zooming-linking modeling methodology has many virtues: it is *systematic* and *modular*, it is *adapted to computer assisted modeling* (with module equations in parametric form stored in a database, and with the interconnection equations stored in a database), it is *hierarchical* (once a model of a system has been obtained, it can be used as a subsystem-module on a higher level). A good model library will have items that are *re-useable, extendable, modifiable, flexible*, etc.).

Disadvantages are that it immediately involves many variables. This can be alleviated by the (partial) elimination of variables when possible. For example, the interconnection equations often immediately lead to elimination of half of the variables. There are situations where the special structure of the modules and the interconnections allow a more direct elimination of some of the variables. For example, modeling of mechanical systems using Lagrangians, modeling of electrical circuits using ports instead of terminals, etc.

Note that the philosophy is to keep the interconnections highly standardized and simple, and to deal with complex models in the modules. For example, in the electrical circuit, a multi-terminal connector was viewed as a module, not as a connection. Also, in mechanical systems, joints, hooks and hinges should be viewed as modules, not as connections.

As a caveat, one should note that not all interconnections fit the framework described above. In particular, distributed interconnections were not considered, for example mechanical systems that are interconnected by sharing a surface, or heat conduction along a surface. Also, terminals do not capture interconnections along virtual terminals, for example, action at a distance, as the attraction of masses or electrical charges, etc. Finally, interactions as rolling, sliding, bouncing, etc., also require a different setting.

The resulting graph structure of an interconnected system has the modules in the nodes and the interconnections as the branches. This follows the physics, and should be contrasted with the graph structure pursued in electrical circuit theory, which has the modules in the branches and the elements connections as the nodes. This structures works fine with 2-terminal elements, but is awkward otherwise, and is difficult to generalize to other, non-electrical, domains.

IV. Interconnected behavior

We now formalize the interconnected system. The most effective way to proceed is to specify it as a latent variable system, with as manifest variables the variables associated with the manifest variable assignment, and as latent variables the terminal variables associated with the modules. Its full behavior behavior consists of the behavior as specified by each of the modules, combined by the interconnection laws obtained by the interconnection architecture.

A first principles model of an interconnected system always contains many latent variables. That is one of the main motivations to introduce latent variables in our modeling language *ab initio*. It also underscores the importance of the *elimination theorem* [1], [2], [4].

V. Conclusions

Modeling interconnected via the above method of *tearing, zooming, and linking* provides the prime example of the usefulness of behaviors and the inadequacy of input/output thinking. Even if our system, after interconnection, allows for a natural input/output representation, it is unlikely that this will be the case of the subsystem and of the interconnection architecture. It is only when considering the more detailed signal flow graph structure of a system that input/output thinking becomes useful. Signal flow graphs are useful building blocks for interpreting information processing systems, but physical systems need a more flexible framework.

Acknowledgments

This research is supported by the Research Council KUL project CoE EF/05/006 (OPTEC), Optimization in Engineering, and by the Belgian Federal Science Policy Office: IUAP P6/04 (Dynamical systems, Control and Optimization, 2007-2011).

References

[1] T. Cotroneo, *Algorithms in Behavioral Systems Theory*, Doctoral dissertation, Faculty of Mathematics and Physical Sciences, University of Groningen, 2001.

[2] T. Cotroneo and J.C. Willems, The simulation problem for high order differential equations, *Applied Mathematics and Computation*, volume 145, pages 821-851, 2003.

[3] J.W. Polderman and J.C. Willems, *Introduction to Mathematical Systems Theory: A Behavioral Approach*. Springer-Verlag, 1998.

[4] J.C. Willems, Paradigms and puzzles in the theory of dynamical systems, *IEEE Transactions on Automatic Control,* volume 36, pages 259–294, 1991.